Metrics of computing trends in NHEP

Jim Pivarski

Princeton University – IRIS-HEP

March 21, 2022

This is a talk about measuring *physicists*: what they talk about and what they do for computing.

This is a talk about measuring *physicists*: what they talk about and what they do for computing.

Measuring people, such as advertising click-throughs, seemed odd to me when I first went from physics to data science, since the events are *not* independent and it would be hard to quantify errors.

This is a talk about measuring *physicists*: what they talk about and what they do for computing.

Measuring people, such as advertising click-throughs, seemed odd to me when I first went from physics to data science, since the events are *not* independent and it would be hard to quantify errors.

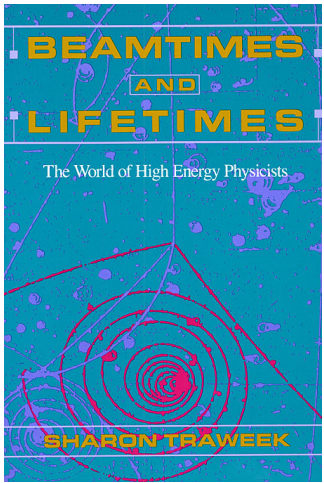
However, it can be a meaningful thing to do, taking all the caveats seriously, and certainly better than *guessing* or *assuming* we understand the community.

This is a talk about measuring *physicists*: what they talk about and what they do for computing.

Measuring people, such as advertising click-throughs, seemed odd to me when I first went from physics to data science, since the events are *not* independent and it would be hard to quantify errors.

However, it can be a meaningful thing to do, taking all the caveats seriously, and certainly better than *guessing* or *assuming* we understand the community.

Inspiration: read Sharon Traweek's anthropological study of physicists at SLAC and KEK in the 1970's. Physicists can be data points!



User needs are very different from my expectations, 6 years ago

Big Data

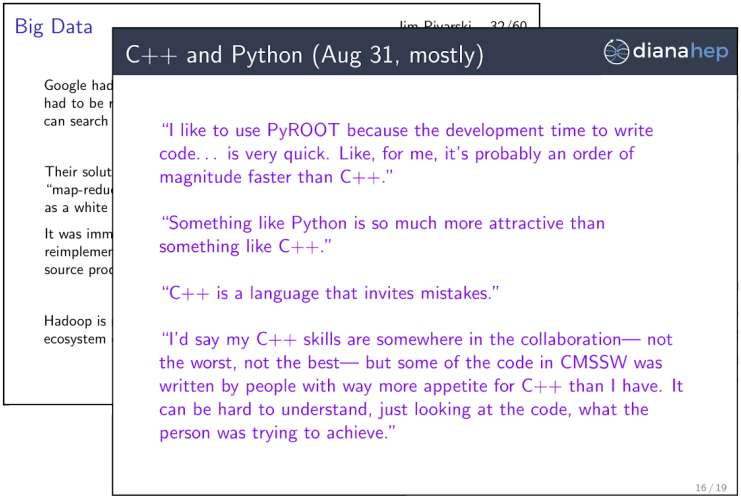
Google had an re-indexing problem: a set of webpages containing words had to be re-indexed as a set of words pointing to webpages, so that you can search for pages by keyword.

Their solution, called "map-reduce," was published as a white paper in 2004.

It was immediately reimplemented as an open source product, Apache Hadoop.

Hadoop is now almost synonymous with Big Data, and it has spawned an ecosystem of tools that interoperate with it, much like ROOT in HEP.

User needs are very different from my expectations, 6 years ago



Big Data

Jim Pivarski — 32/60

Google had
had to be r
can search

Their solut
"map-reduc
as a white

It was imm
reimplemer
source proc

Hadoop is
ecosystem

C++ and Python (Aug 31, mostly)

dianahep

"I like to use PyROOT because the development time to write code... is very quick. Like, for me, it's probably an order of magnitude faster than C++."

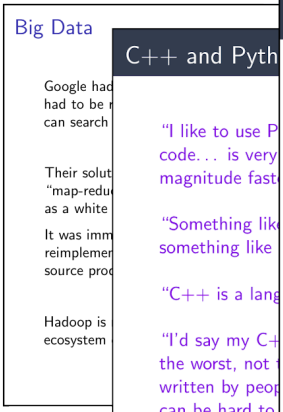
"Something like Python is so much more attractive than something like C++."

"C++ is a language that invites mistakes."

"I'd say my C++ skills are somewhere in the collaboration— not the worst, not the best— but some of the code in CMSSW was written by people with way more appetite for C++ than I have. It can be hard to understand, just looking at the code, what the person was trying to achieve."

16 / 19

User needs are very different from my expectations, 6 years ago



Half-hour interviews with physicists about array syntax

1 grad student, 2 postdocs (beginning & advanced), and 1 advanced researcher

Everyone had most experience in C++ (5 years to decades), less in Python, which was primarily PyROOT (6 months to 3–4 years), very little in Numpy (2 to 5 months).

Some found it easier, some more difficult.

- "Way, way much easier than applying cuts with for loops."
- "Surprised by how conceptually different you have to think about selections, combining objects." but "Not good or bad, just surprising that it has a learning curve."
- "Individual problems have been much more difficult than expected." and "Translating 'if' statements is where I get hung up." but "Not inherently harder; just harder now for those of us used to the 'for' loop version."

18 / 20

User needs are very different from my expectations, 6 years ago

Big Data

Google had
had to be r
can search

Their solut
"map-reduc
as a white

It was imm
reimplemer
source proc

Hadoop is
ecosystem

C++ and Pyth

"I like to use P
code. . . is very
magnitude fast

"Something like
something like

"C++ is a lang

"I'd say my C+
the worst, not
written by peop
can be hard to
person was trying to achieve."

Half-hour interviews with physicists about array syntax

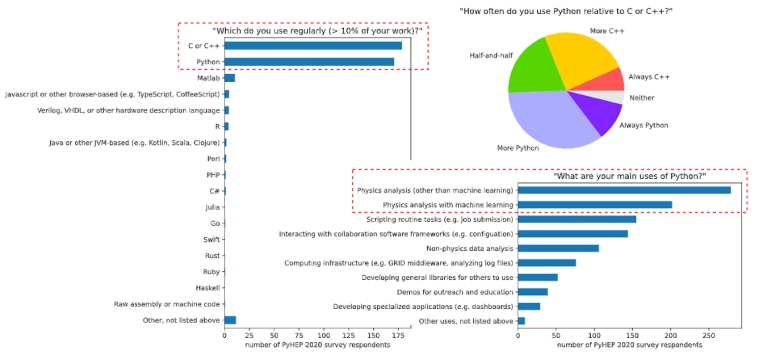
1 grad stu

Everyone
primarily

Some four

- "Way
- "Surp
object
- "Indiv
"Tran
"Not

Consistent with survey results (PyHEP 2020 participants)



5 / 14

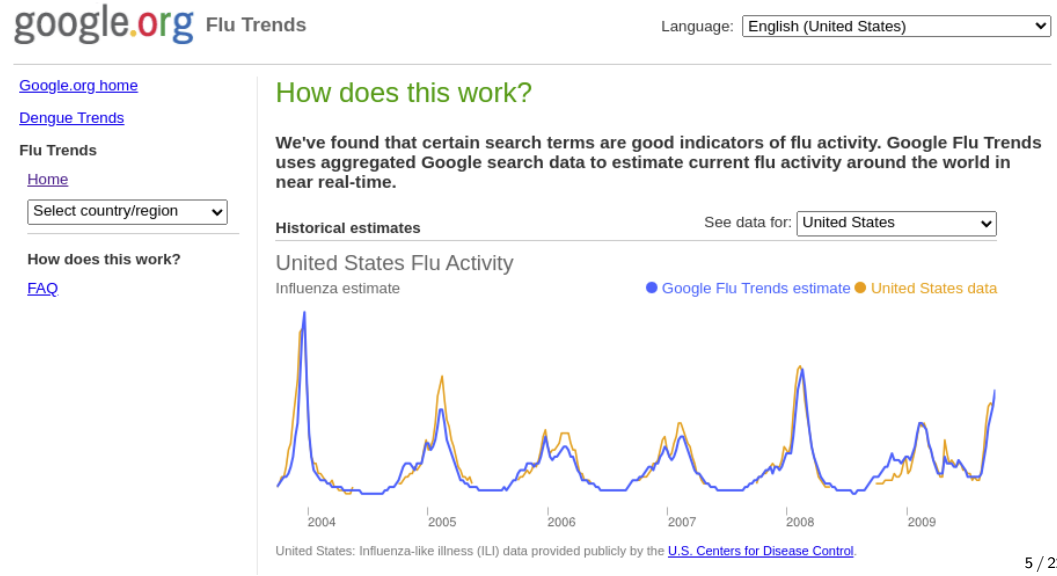
16 / 19

Ways to study humans

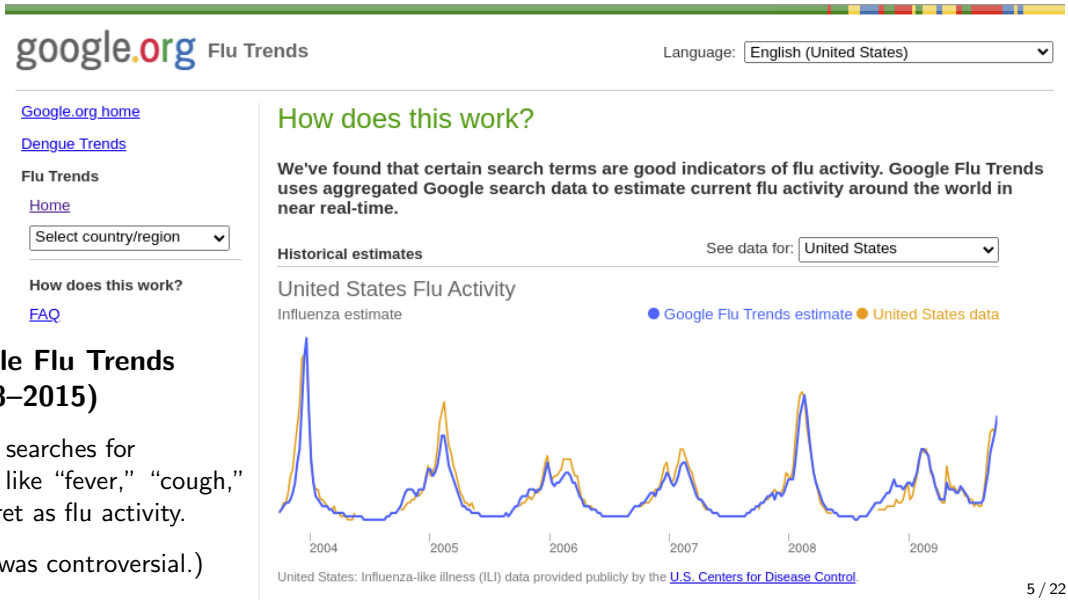
(Important note: I am not an expert. Below is what I learned from college friends who went into social sciences.)

- Qualitative:
  - Focus groups: most open to unexpected ideas. Want to keep the group size and mix such that participants are willing to speak up. Goal is to discover new *dimensions* of the vector space, not just points within it.
  - One-on-one interviews: can be deeper but less broad than focus groups. Lacks the multiplying effect of responding to each other's opinions.
  - History/documents: observational, rather than experimental, but this method can reach further into the past.
- Quantitative:
  - Surveys: can get large, statistically meaningful datasets, at the cost of losing flexibility/openness to new ideas. Now you *are* filling in a vector space.
  - Proxy metrics: can measure what people *do*, rather than what they *say*.

Proxy metrics: high statistics, cautious interpretation



Proxy metrics: high statistics, cautious interpretation



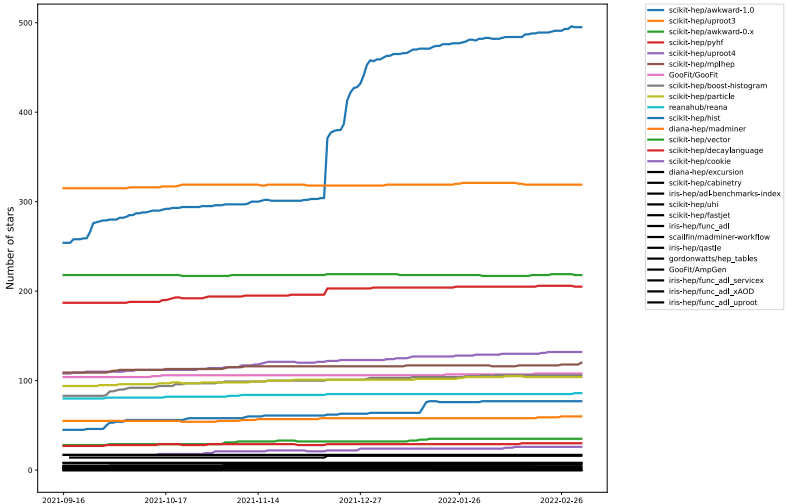
Google Flu Trends (2008–2015)

Count searches for things like "fever," "cough," interpret as flu activity.

(This was controversial.)

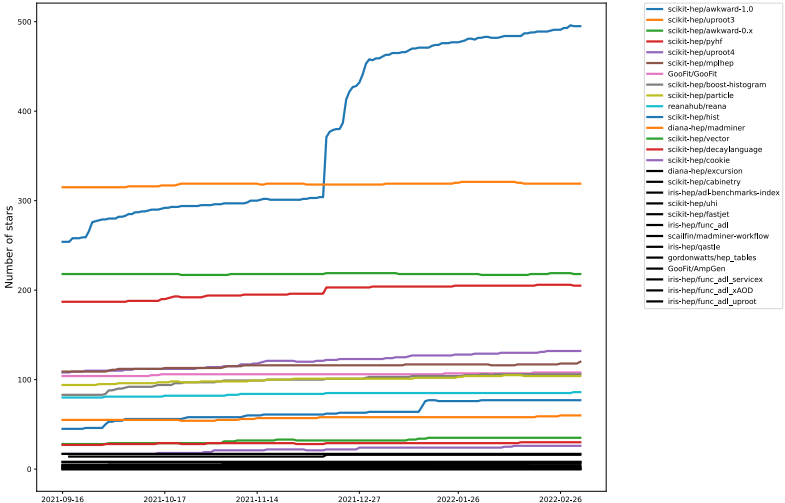
Example: what happened here?

GitHub stars versus time in IRIS-HEP projects



Example: what happened here?

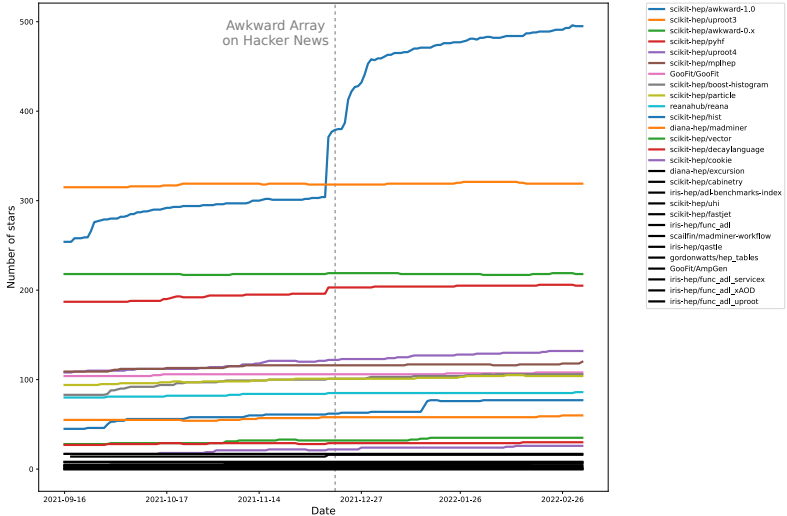
GitHub stars versus time in IRIS-HEP projects



What we're trying to explain is a big, qualitative feature, not the little bumps.

Example: what happened here?

GitHub stars versus time in IRIS-HEP projects

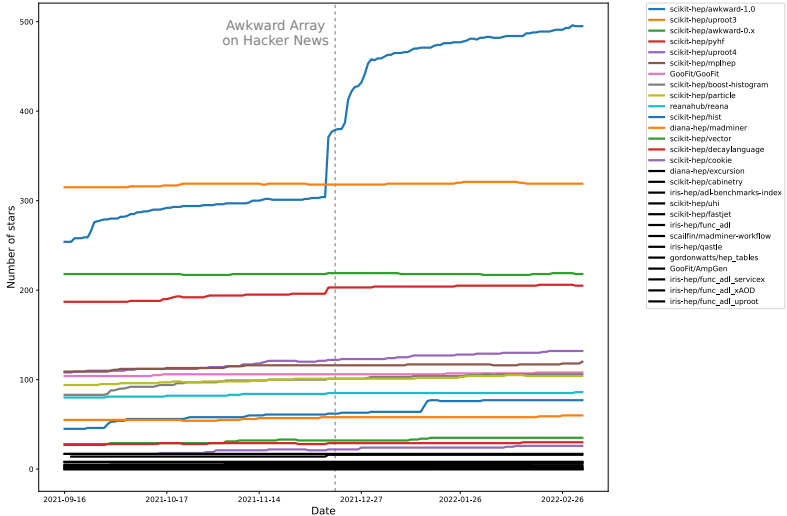


What we're trying to explain is a big, qualitative feature, not the little bumps.

Temporally coincides with this event: https://news.ycombinator.com/item?id=29576323

Example: what happened here?

GitHub stars versus time in IRIS-HEP projects



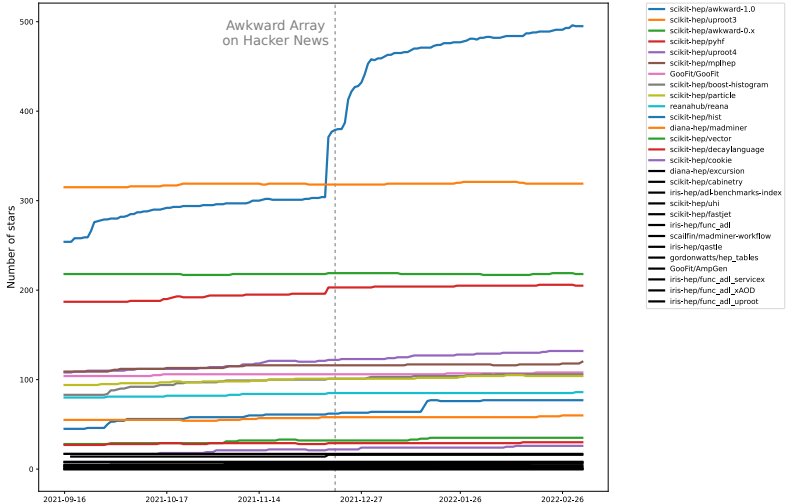
What we're trying to explain is a big, qualitative feature, not the little bumps.

Temporally coincides with this event: https://news.ycombinator.com/item?id=29576323

Is it causal?

Example: what happened here?

GitHub stars versus time in IRIS-HEP projects



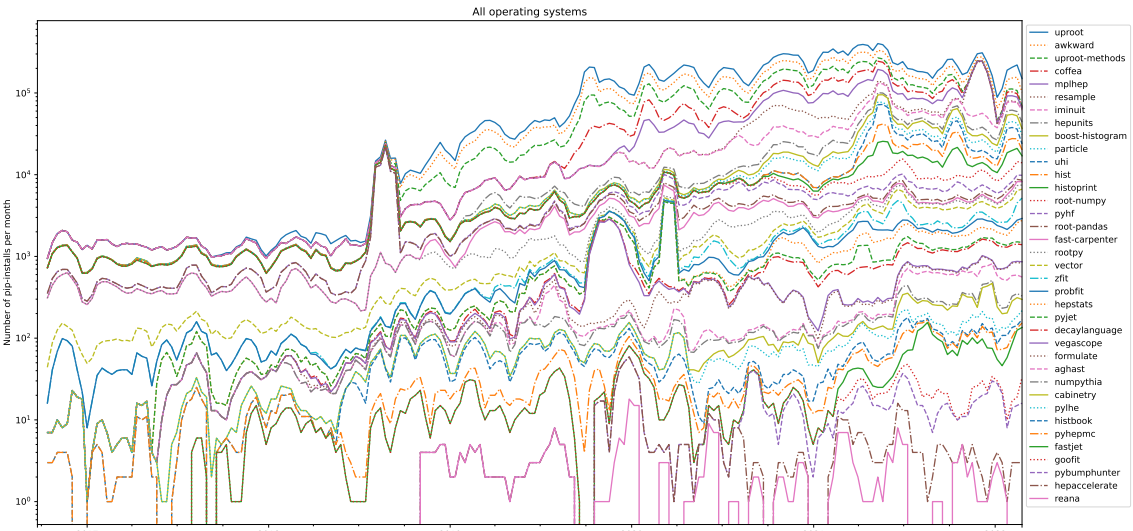
What we're trying to explain is a big, qualitative feature, not the little bumps.

Temporally coincides with this event: https://news.ycombinator.com/item?id=29576323

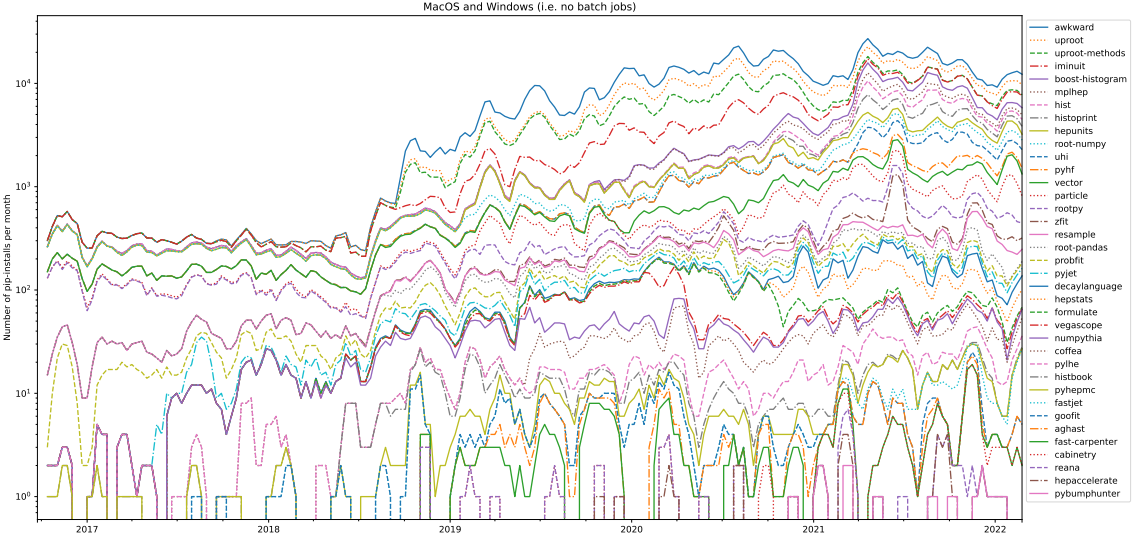
Is it causal?

It would be hard to believe it isn't.

Stacked download statistics for Scikit-HEP and related packages



Stacked download statistics for Scikit-HEP and related packages



Linux includes batch jobs, which sometimes `pip install` the same package on thousands of workers.

Linux includes batch jobs, which sometimes `pip install` the same package on thousands of workers.

Selecting only MacOS and Windows removes most batch jobs, but it excludes some individual users (like me), probably with a behavioral bias.

Linux includes batch jobs, which sometimes `pip install` the same package on thousands of workers.

Selecting only MacOS and Windows removes most batch jobs, but it excludes some individual users (like me), probably with a behavioral bias.

Still, there's continuous testing jobs on MacOS and Windows.

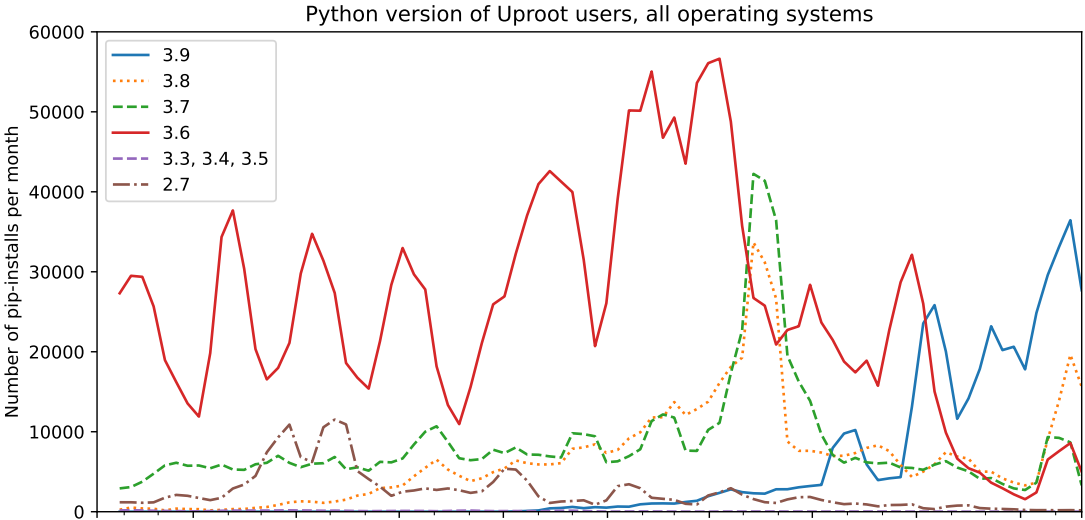
Linux includes batch jobs, which sometimes `pip install` the same package on thousands of workers.

Selecting only MacOS and Windows removes most batch jobs, but it excludes some individual users (like me), probably with a behavioral bias.

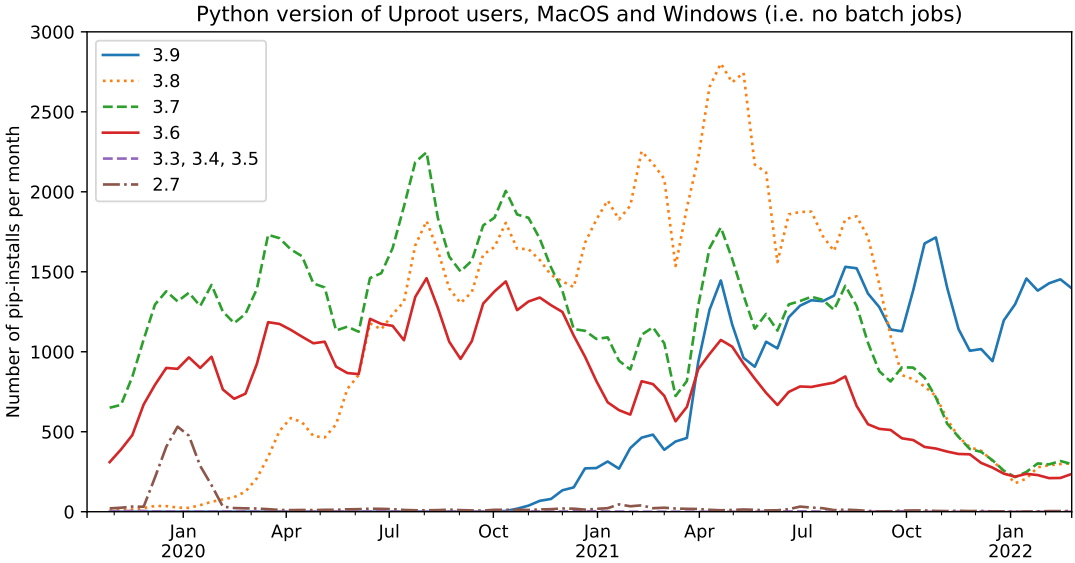
Still, there's continuous testing jobs on MacOS and Windows.

Do we even *want* to exclude these things? What do we *want* the observable to quantify?

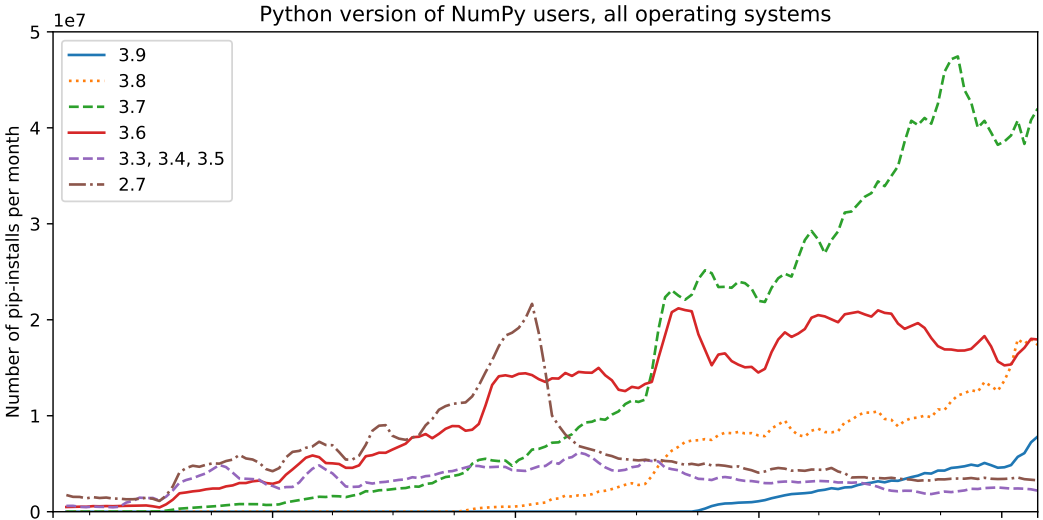
How about this one: Python version in `pip install uproot`



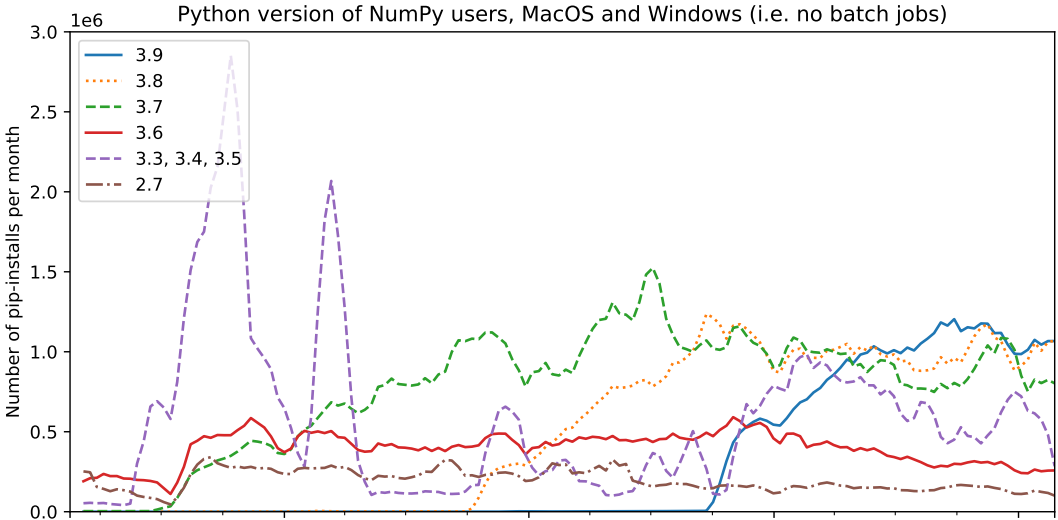
How about this one: Python version in `pip install uproot`



How about this one: Python version in `pip install numpy`

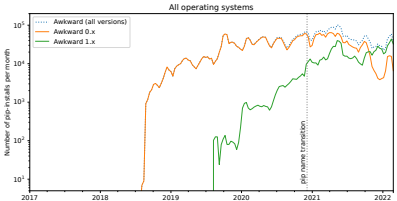


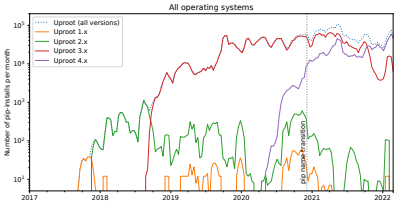
How about this one: Python version in `pip install numpy`

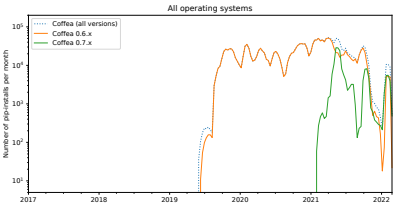


More often useful when *comparing* two things

Transition from “old” Awkward/Uproot/Coffea to “new”



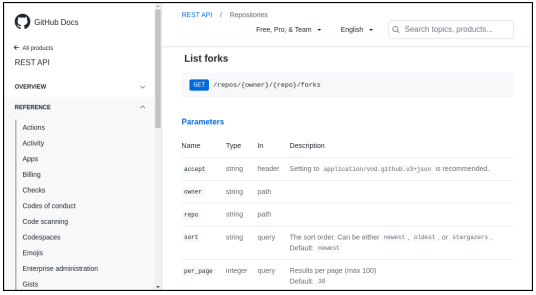




Directed study: how are physicists using C++ and Python?

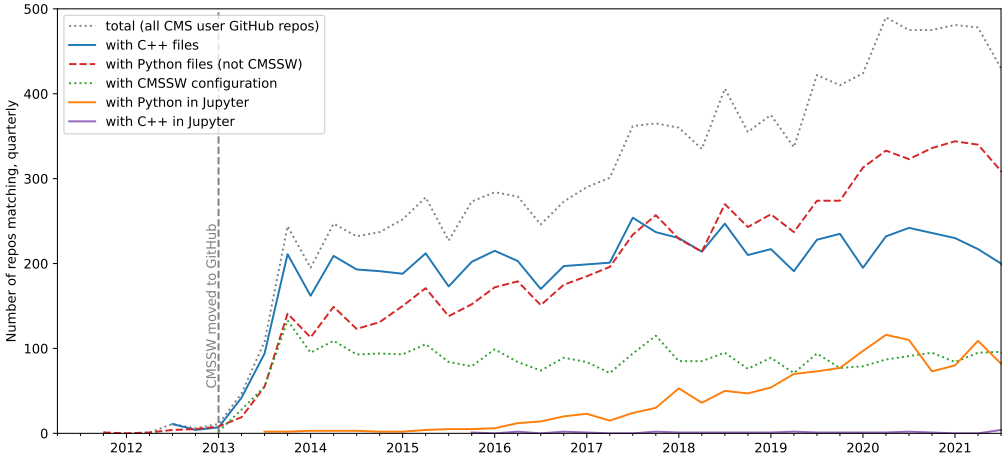
Analyze code in 11 635 GitHub repos written by 2 172 physicists:

1. Ask GitHub which users forked CMSSW and call them "CMS physicists." (CMSSW has been on GitHub for a long enough time to see trends.)
2. Clone all of the physicists' repos (the ones that are not forks of something else).
3. Search the code of these repos and count matches.
4. Take care to exclude CMSSW configuration files, which are also Python.



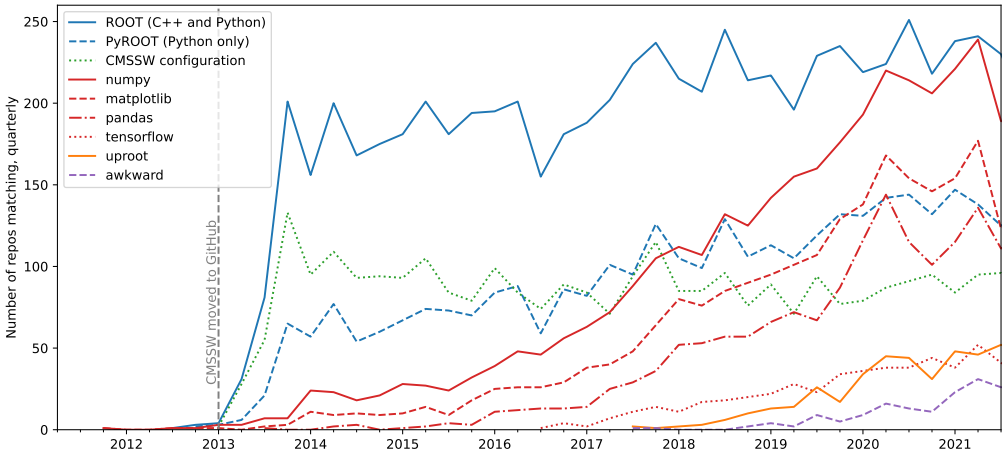
Language use: C++, Python, and Jupyter

Number of non-fork GitHub repos created by CMS physicists (users who forked CMSSW)



Packages: ROOT, Scientific Python, Uproot/Awkward

Same sample, now counting regex matches for `import XYZ`, `from XYZ import`, etc.



This is an update of my earlier GitHub-CMSSW study

1. Inclusively counting "repo that contains a C++ file" or "a Python file," rather than GitHub's exclusive determination of "repo language."
2. Distinguishing between Python files that are CMSSW configurations and other Python files (GitHub doesn't).
3. I've downloaded all the repos, so I can run my own regex searches, rather than relying on GitHub's.

This is an update of my earlier GitHub-CMSSW study

1. Inclusively counting "repo that contains a C++ file" or "a Python file," rather than GitHub's exclusive determination of "repo language."
2. Distinguishing between Python files that are CMSSW configurations and other Python files (GitHub doesn't).
3. I've downloaded all the repos, so I can run my own regex searches, rather than relying on GitHub's.

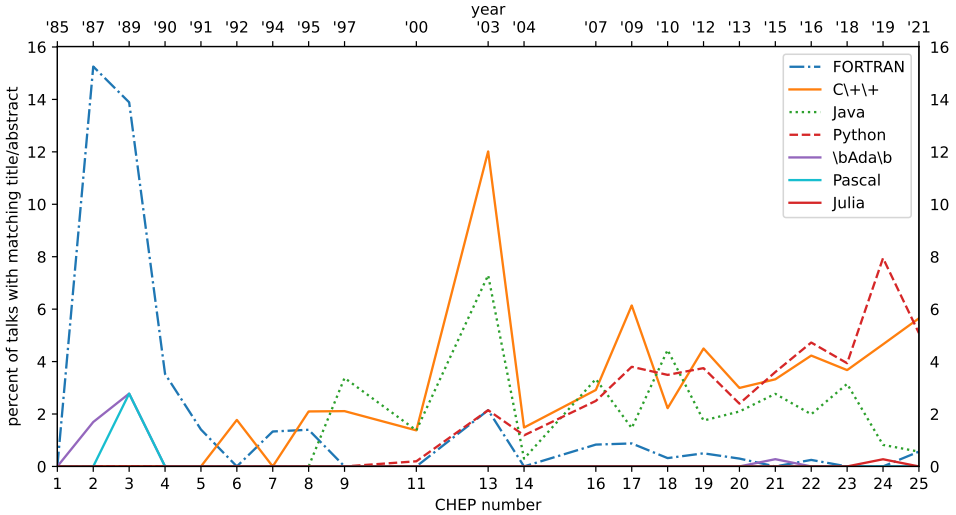
We can do more: run clang-tidy/pylint? Features of libraries used?

https://pivarski-princeton.s3.amazonaws.com/GitHub-CMSSW-user-nonfork-raw-data.tar

(Note: these are all public repos/public data.)

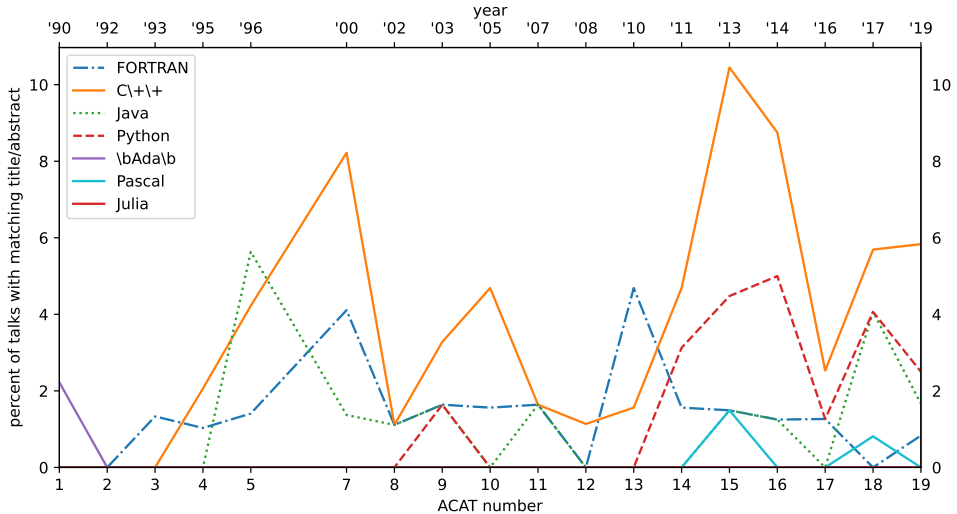
Longer baseline: title/abstract matches in InspireHEP

Programming languages in CHEP papers



Longer baseline: title/abstract matches in InspireHEP

Programming languages in ACAT papers



Emerging Standard ? Python as "Software Glue"

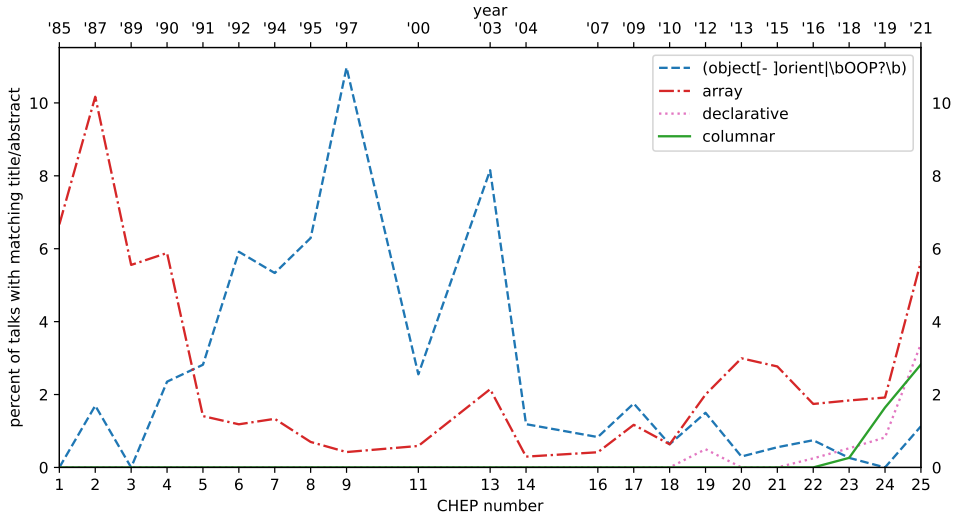
- Clear trend towards Python
  - Used by: ATLAS (Athena),CMS, D0, LHCb (Gaudi), SND,...
  - Used by: Lizard/Anaphe, HippoDraw, JAS (Jython)...
  - Architecturally, scripting is "just another service"
  - ROOT is the exception to the "Python rule"
    - CINT interpreter plays a central role
    - Developers and users seem happy
- Python is popular with developers...
  - Rapid prototyping; gluing together code
  - (Almost) auto-generation of wrappers (SWIG)
- ...but acceptance by users not yet proven
  - Another language to learn, syntax,...

"Summary of Track 2: Data Analysis and Visualis. Lucas Taylor, Northeastern U. CHEP 01, Beijing, 3-7 S.

Note: PyROOT introduced in 2004 (v4.00/04).

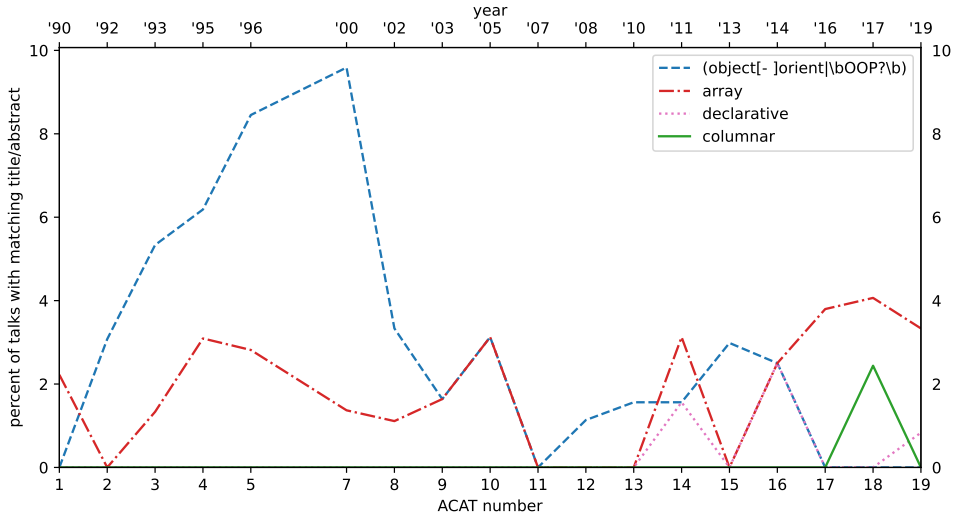
Longer baseline: title/abstract matches in InspireHEP

Programming paradigms in CHEP papers



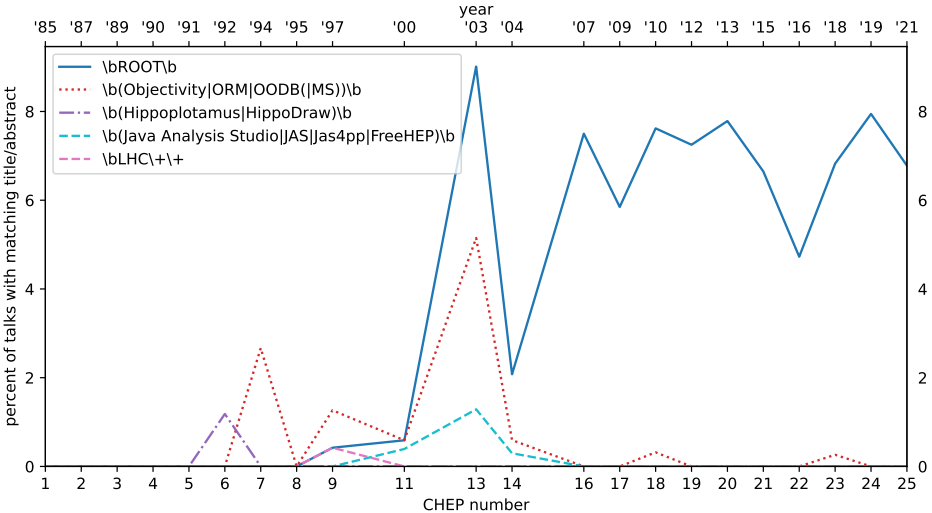
Longer baseline: title/abstract matches in InspireHEP

Programming paradigms in ACAT papers



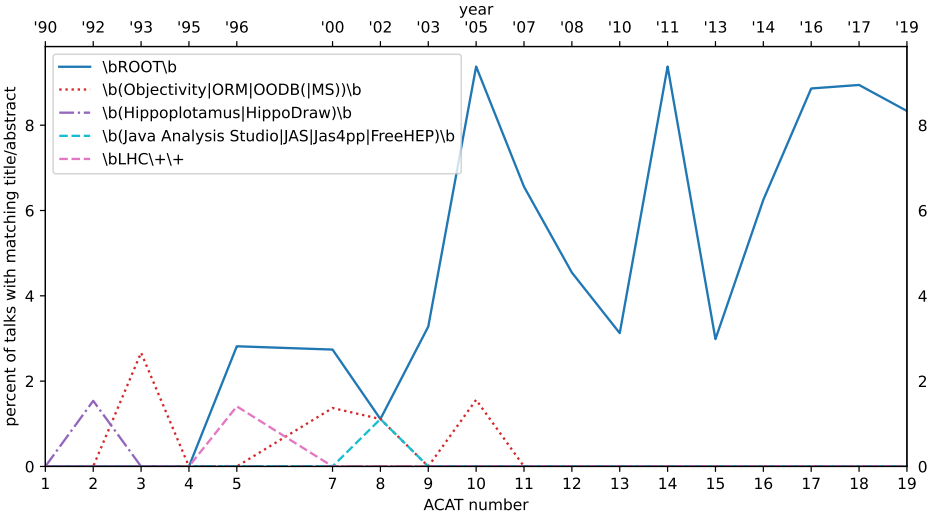
Longer baseline: title/abstract matches in InspireHEP

Software frameworks in CHEP papers



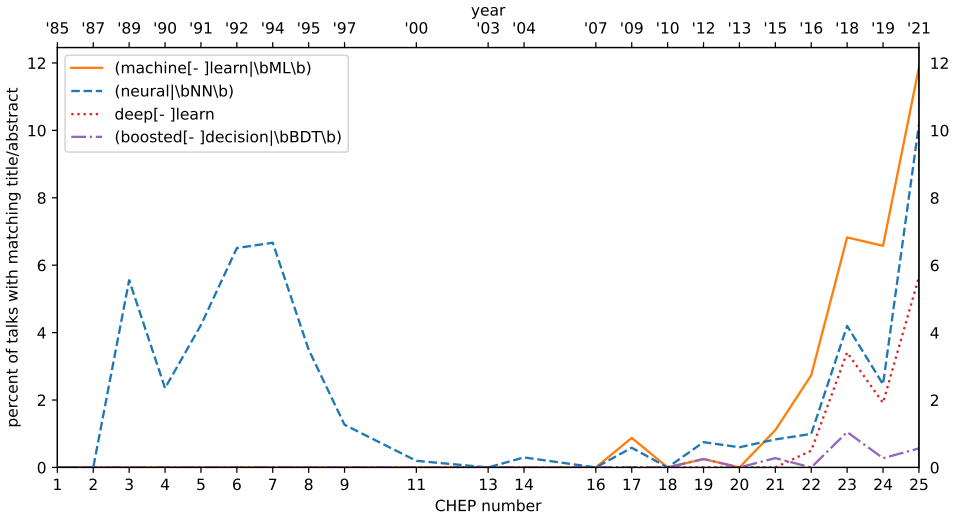
Longer baseline: title/abstract matches in InspireHEP

Software frameworks in ACAT papers



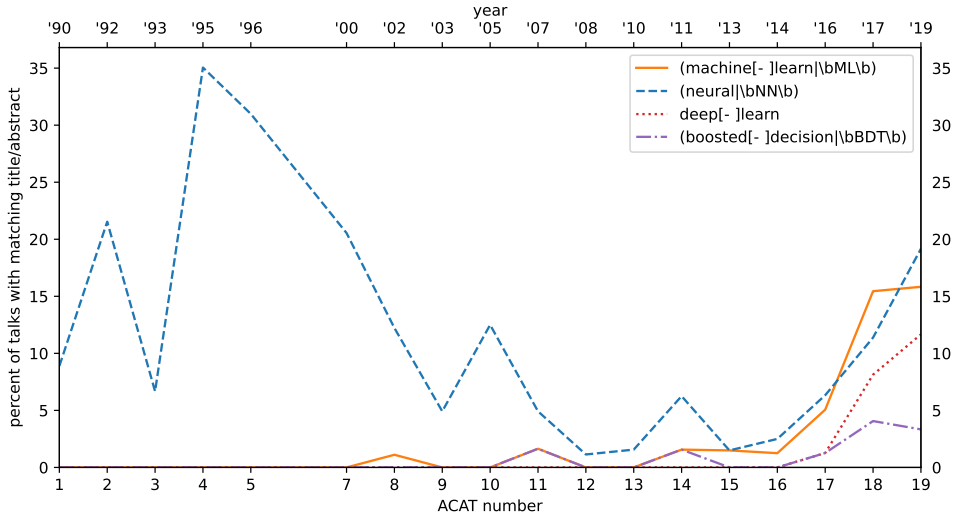
Longer baseline: title/abstract matches in InspireHEP

Machine learning in CHEP papers



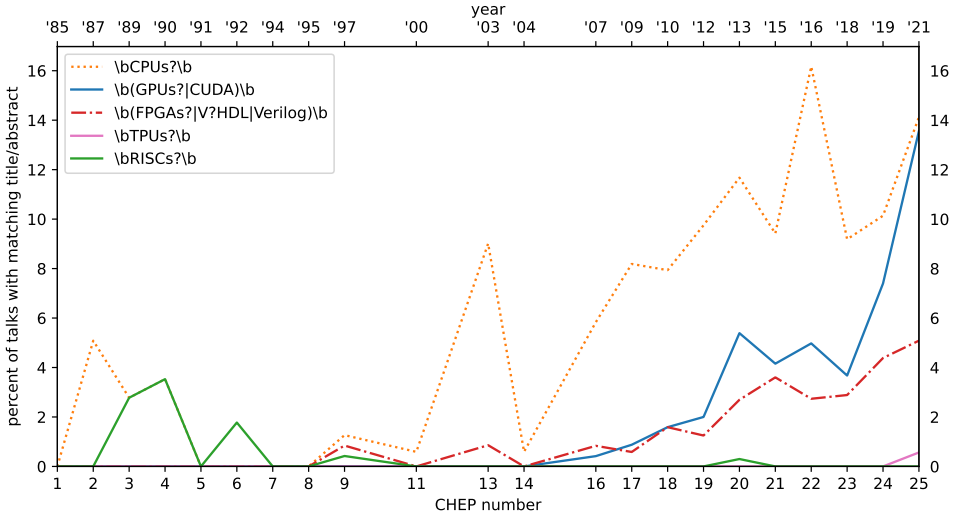
Longer baseline: title/abstract matches in InspireHEP

Machine learning in ACAT papers



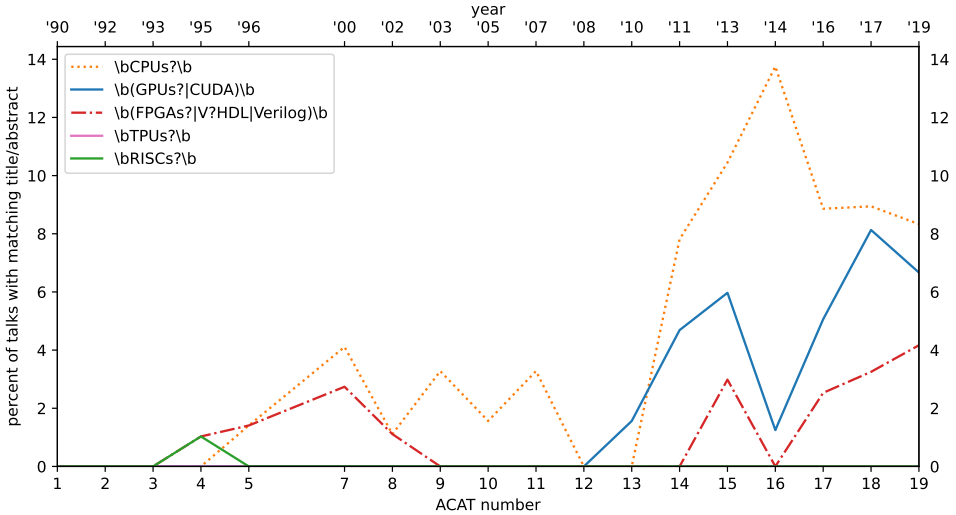
Longer baseline: title/abstract matches in InspireHEP

Hardware accelerators in CHEP papers



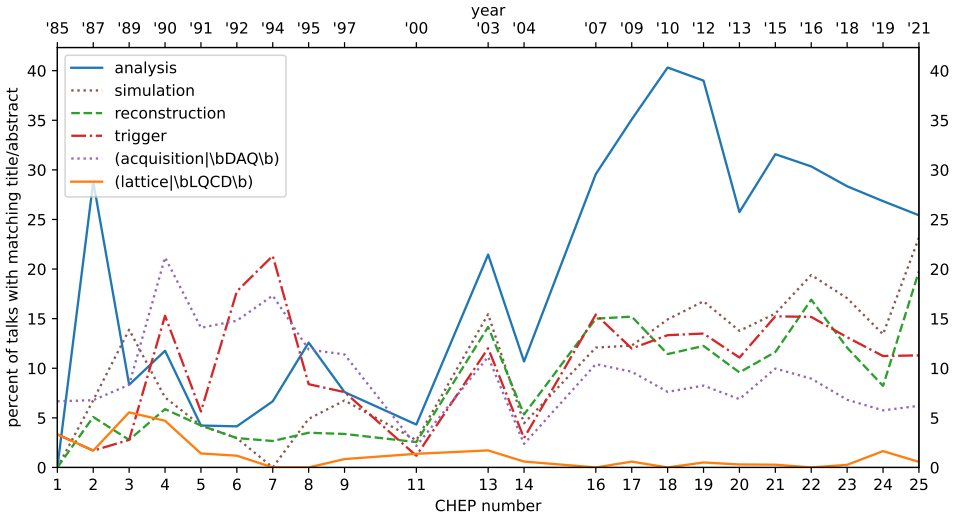
Longer baseline: title/abstract matches in InspireHEP

Hardware accelerators in ACAT papers



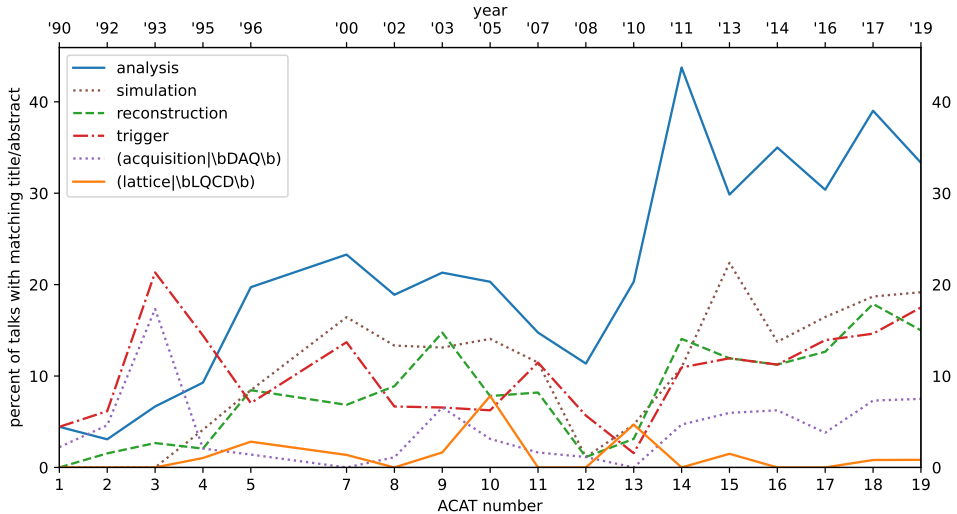
Longer baseline: title/abstract matches in InspireHEP

Kinds of tasks in CHEP papers



Longer baseline: title/abstract matches in InspireHEP

Kinds of tasks in ACAT papers



Conclusions

- Different ways of understanding people, including the NHEP software community: focus groups, interviews, historical documents, surveys, and proxy metrics.
- This talk focused on proxy metrics, which are quantitative, but you have to pay close attention to what they're quantifying.
- Some clear trends and conclusions emerged. Others are muddled.

Conclusions

Relatively clear trends/conclusions.

- Scikit-HEP adoption increased by more than 10× since 2018. Maybe 100×.
- Most physicists use Python 3.9, ahead of the wider (NumPy) community.
- The Awkward/Uproot/Coffea version update is done (more new than old).
- C++ use in CMS is *steady, not decreasing*, while scientific Python (NumPy/Matplotlib/Pandas/Jupyter) is increasing past that level.
- Python (in general, e.g. for configuration files), has been slowly increasing since 2000, at the same time as the abrupt Fortran → C++ switch.
- Declarative/columnar interest is increasing, but nothing like "object oriented" in the 1990's. "Arrays" are coming back (since Fortran).
- Machine learning is a *resurgence*, with more synonyms than in the 1990's. (GPUs/FPGAs on the same timescale.)
- "Analysis" has grown as a preoccupation at CHEP and ACAT.