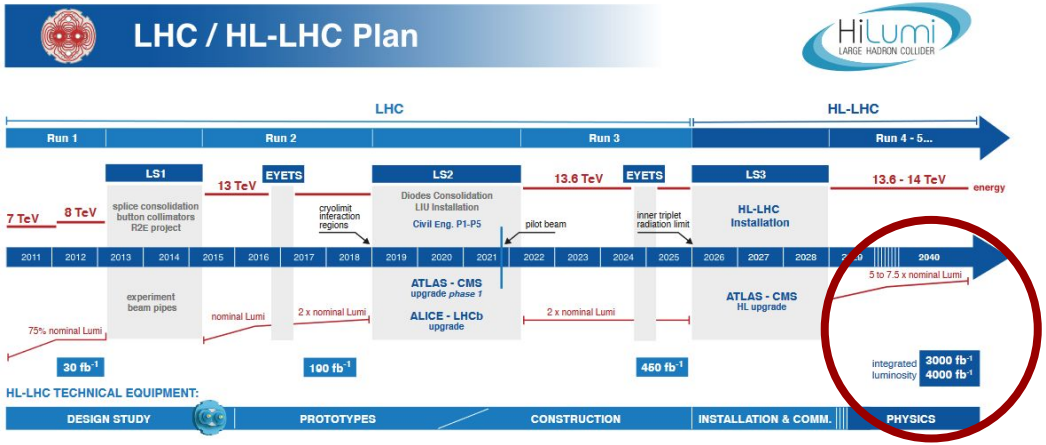
HEP Software Foundation and the OSSR

Graeme A Stewart, for HSF Coordination and Conveners

ESCAPE OSSR Workshop, 2022-12-02

HEP Software Foundation in a Nutshell



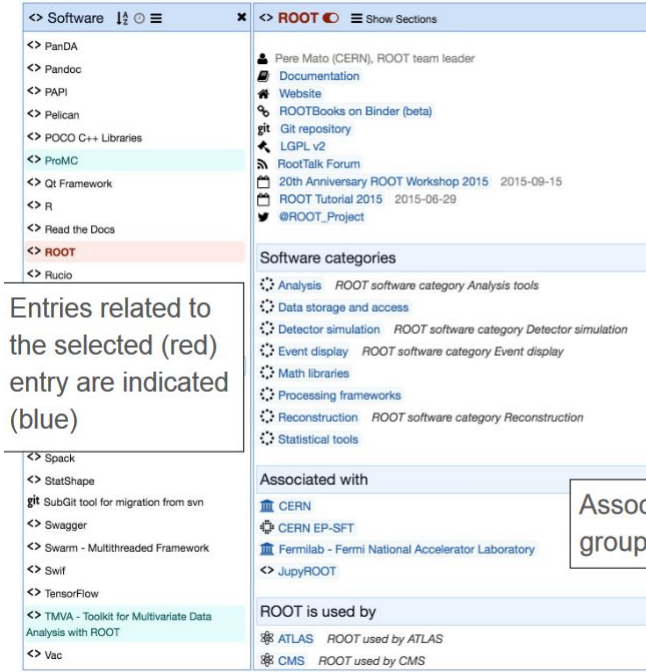
- Why a software foundation focussing on high-energy physics?
  - Upgrade to high-luminosity LHC (HL-LHC) brings opportunities and challenges
  - 5-7.5x higher luminosity than the original LHC plan
  - In total, more than 10x more data than we have taken with the LHC so far (target 3000fb^{-1})
    - ~7.3EB for ATLAS alone by 2036
- We realised that as well as the detector upgrades we also needed a *software upgrade*, tackling the problems of **data rate** and **data volume**
  - Evolution and revolution of a massive code base (~50M+ lines of C++)
    - With many 1000s of developers, of uneven skill level
  - Happening alongside major architecture evolution, parallelism and heterogeneous computing
  - Large process of community consultation that resulted in the HSF Community White Paper in 2017-2018

HSF Activities Today

- After 5 years the HSF has firmly established itself as part of the landscape in HEP
  - We are building strong bridges with the nuclear physics community as well (EIC)
- We have working groups focusing on different areas that are important for the field
  - **Event generation**, **Simulation**, **Reconstruction**, **Analysis**, **PyHEP** (fairly domain specific)
- We also have a dedicated working groups looking at **Tools and Packaging** and **Software Frameworks**
  - These are very *developer* and *developer tools* focused
- Plus a working group on **Training**, which is a critical piece of the puzzle
  - Educate the next generation
    - Basic training in Python, git, etc.
    - C++ training, covering basic and advanced topics
  - Software quality does not fall from the sky - people need to be taught these skills
  - **Many people** are involved from **different experiment communities**, a common problem for all experiments
- As well as this we organise workshops and other events and advocate for the community at high levels

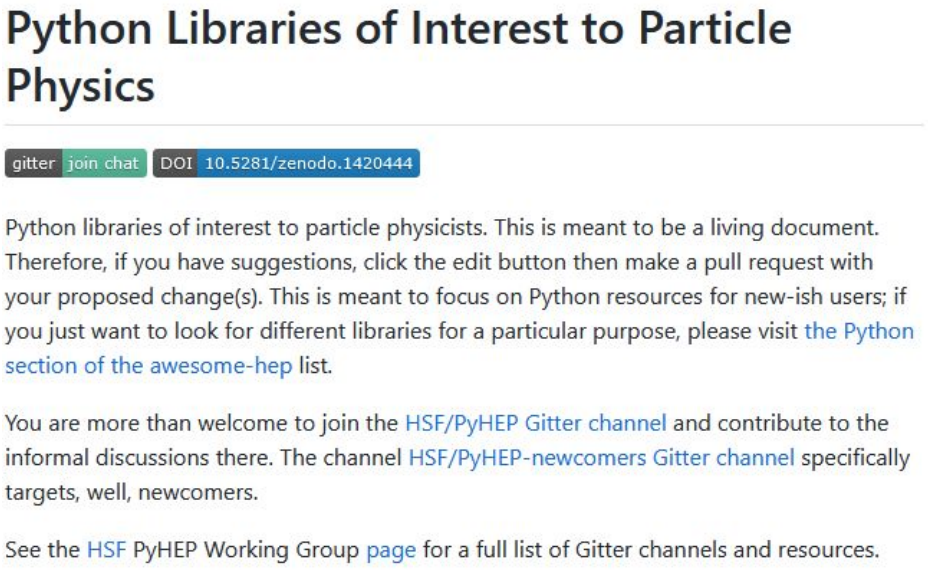
HEP (Software) Knowledge Base

- One of our first HSF projects was an ambitious knowledge base
  - Much more than a software catalogue
  - Aimed to incorporate knowledge of experiments, organisations, events, etc. as well
  - Then all cross-referenceable, e.g.,
    - Can ask what software is the ATLAS experiment is using?
    - Can ask which experiments use ROOT?
  - Allow knowledge to be accessed for new people
- It was a great idea, but…
  - In the end, *the community didn't buy into it*
  - It wasn't maintainable on the few cycles of effort volunteers could put in
  - Once it starts to decay, it goes downhill quickly
    - Eventually we just decommissioned it



Another Approach - Curated Lists

- A more focused approach has worked better
  - Small and beautiful? Well, at least far easier to maintain and curate, which is sustainable
  - Also maintenance and contributions were easy - GitHub PRs in both these examples



Python Libraries of Interest to Particle Physics

gitter join chat | DOI 10.5281/zenodo.1420444

Python libraries of interest to particle physicists. This is meant to be a living document. Therefore, if you have suggestions, click the edit button then make a pull request with your proposed change(s). This is meant to focus on Python resources for new-ish users; if you just want to look for different libraries for a particular purpose, please visit the Python section of the awesome-hep list.

You are more than welcome to join the HSF/PyHEP Gitter channel and contribute to the informal discussions there. The channel HSF/PyHEP-newcomers Gitter channel specifically targets, well, newcomers.

See the HSF PyHEP Working Group page for a full list of Gitter channels and resources.

HSF Software Training Center

Idea

Training in software and computing are essential ingredients for the success of any HEP experiment. As most experiments have similar basic prerequisites (Unix shell, Python, C++, ...) we want to join our efforts and create one introductory software training curriculum that serves HEP newcomers the software skills needed as they enter the field, and in parallel, instill best practices for writing software.

The curriculum is comprised of a set of standardized *modules*, so that students can focus on what is most relevant to them.

The modules

Basics

The UNIX Shell
A guide through the basics of the file systems and the shell.
Start learning now!
Contribute!

Version controlling with git
Track code changes, undo mistakes, collaborate. This module is a must.
Start learning now!
Contribute!

Programming with python
Get started with an incredibly popular programming language.
Start learning now!
Contribute!

SSH
Introduction to the Secure Shell (SSH)
Status: Early development
Start learning now!
Contribute!

Machine learning
Get behind the buzzword and teach machines to work for you intelligently!
Start learning now!
Watch the videos!
Contribute!

Matplotlib for HEP
Make science prettier with beautiful plots!
Status: Beta testing
Start learning now!
Contribute!

ROOT

What we learned so far...

- Large scale catalogues manually maintained by a few people don't work
  - Even if developers like the idea, there's an *out of sight, out of mind* problem
    - Developers focus on code, so better maintain everything in the same repository
  - The might work if they are **automated**
- Smaller scale curated lists do seem to work
  - Smaller groups have more of a sense of community responsibility, and an immediate concrete benefit
    - On the large scale this dilutes fast through diffusion of responsibility
  - *Bottom-up approach* (much more the HSF style)
  - However, there is then the issue of how does knowledge of the resource propagate outside the group...?
- Give it a low barrier to entry...
  - Then try and make it lower!
- Automate if possible
  - The less human cycles required the better

HSF/IRIS-HEP Workshop: Software Citation and Recognition

- Closely related topic to cataloging, we recently had a workshop organised on Software Citation and Recognition
  - Review status of citation in HEP
  - Give credit to software developers and maintainers
  - Provide better and more sustainable software
  - Support for reproducibility
- Key principles developed by Force11 group
  - Importance, Credit and attribution, Unique identification, Persistence, Accessibility, Specicifity
  - Group then had task forces which helped to develop
    - Citation Format File standard (CITATION.cff)
    - CodeMeta
      - Metadata standard for software, a richer description of software
- Workshop report is being prepared now

What do the Experiments do?

- Experiments do have citation guidance for their papers
  - So there is an attempt at standardisation
  - Although not all advice is the same across the experiments
- Citations are quite well settled for event generators
  - 1000s of citations for main MCEGs
- Less consistent for analysis software, statistical methods, machine learning
  - Especially if there is no clear citation advice from the authors
- Sometimes one is citing the software as software, sometimes one is citing the physics results that are then expressed as software…
- Definite interest in having some community standard for these citations (e.g., curated by the HSF)

What to cite?

- An academic paper written about the software
  - This is the traditional approach, currently giving the most academic credit
    - Some feedback from RSEs - at least a subset don't like writing papers
  - There is a serious issue with ancestor papers picking up all citations
    - E.g., the 2003 Geant4 paper gets most citations - even though the code today is almost completely different and all the modern authors are missing
- The software itself
  - E.g., the Zenodo DOI
  - Not well rewarded academically
  - Does it describe why the software exists? The design choices?
- A combination of the two
  - E.g. the Journal of Open Source Software (JOSS)
  - Combining the code, plus a short paper describing the software
  - Code and repository is reviewed as well - has to meet best-practice standards like build instructions, basic tests, and user documentation

INSPIRE

- INSPIRE is the main HEP information platform
  - Tracks papers, citations and other information in the field
  - 1.5M literature entries
    - Including software papers
    - But currently data and software are not supported
  - Plans for 2023
    - Will add entries for software and data
      - Must be automatically harvestable (e.g., from Zenodo INSPIRE HEP, CERN open data)
      - Must adhere to FAIR principles
      - Will start to track citations, giving credit for authors

Best Practice for Developers

- If you want your software properly cited, put the citation everywhere...
  - In the README, in the documentation, on the distribution page (PyPI)
  - And make this a single source of truth!
- Adopt a citation format file
  - CITATION.cff - first version can be easily generated via a webpage
- Make sure you keep things up to date
  - E.g. use tbump
  - Prefer the Zenodo concept/project DOI to a version specific one
- Activate automatic retrieval from Zenodo
- Make the citation available directly, e.g.,
  - `mytool --citation`
  - `import mytool; mytool.utils.citation()`

Will be added to our Best Practices Technical Note

A Straw Poll

- Where are we now?
  - Developers are interested in making sure their software is discoverable, useable and that they get credit for it
  - The means for doing so (best practice) aren't so well known, it seems...
  - This is just a bunch of 9 important repositories I picked ~at random from HEP software and computing

	CITATION.cff	codemeta.json	.zenodo.json
Repos...	5/9	1/9	2/9

Summary

- HSF is the community software organisation in high-energy physics
  - Volunteer and community driven
- We have a huge interest helping making excellent software
  - We don’t write it directly (though most HSF members are also developers)
  - We help with training, tooling and best practices (our Project Best Practices TN)
  - FAIR also applies to software
- We believe that cataloging, metadata and citations can help a lot with this
  - Have to have a low barrier of entry for developers
  - And a clear incentive for them to invest
- And we are always open to working with other sciences and organisations such as ESCAPE to improve things for developers and users