Reflexions on HEP software training

Personal perspective on my time in SHiP, SND@LHC, LHCb, DUNE

Oliver Lantwin

2022-05-12

LAPP

Who am I?

- First exposure to ROOT 2014 during my MSci analysing LHCb data
- PhD with SHiP at Imperial College London 2015–2019, there part of core software developers
- Postdoc on LHCb, SHiP and SND@LHC at Zürich 2019–2021, focussing more on analysis, and core software for SND@LHC
- Now, postdoc on DUNE at LAPP in Annecy, mainly prototyping hardware and analysing data from the prototypes
- Still involved with LHCb, SHiP, SND@LHC to varying degrees

Outline

I'll talk about:

- My experience and lessons learnt from teaching ROOT, HEP software and data analysis to various audiences over 4+ years
- How different experiments can work together to share the teaching load (and learn from each other)
- Change of perspective from user to core-software author and back to novice user

Today I'm not speaking on behalf of any collaboration: all opinions and reflections are my own

How I learned ROOT

- Thrown in the deep end with colleagues' examples and online tutorials to guide me
- Started mostly learning RooFit (LHCb analysis), then other parts of ROOT (IO, TGeo, interface with GEANT4, event generators...) as part of my work on SHiP
- Kept up with scientific python, mixing in numpy, rootpy, tensorflow etc. or using them instead of ROOT

How I taught ROOT (and other HEP software)

- Informally startet teaching other graduate students and postdocs during my PhD
- In 2018, Inspired by the LHCb starter kit, we started up our own SHiP version (1-day workshop taught partially with the help of external experts from e.g. the ROOT team)
- Subsequently got into contact with LHCb and ALICE to collaborate on future iterations (2018, 2019), with great success
- 2019 and 2020: Taught data analysis to Uni Zürich students using scientific python
- 2021: Taught data analysis with LHCb Open Data to students from several Moscow universities using ROOT and scikit-hep (as part of the MISiS MegaScience 2021 programme)

Joint starter kits

Example schedule:

Mon	Tue	Wed	Thu	Fri
Bash	Python part II	SHiP	SHiP	Hackathon
Python I	Git	Modern C++	Hackathon	
			Social event	

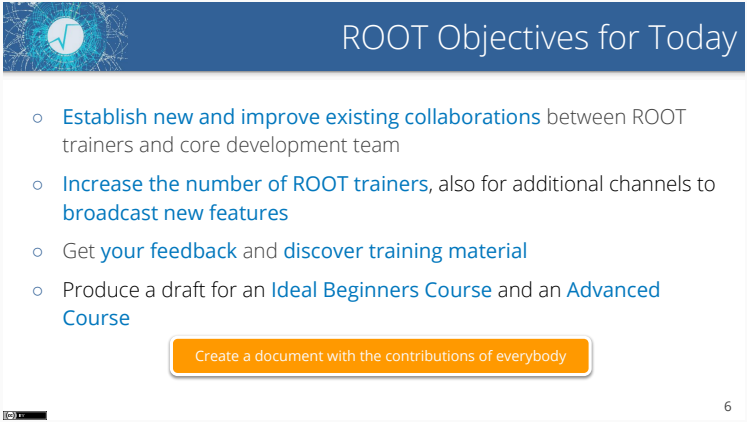
2 days ALICE + LHCb + SHiP (Analysis essentials, now part of HSF)

0.5 days ALICE + SHiP

1 social event (very important)

2.5 days SHiP (SHiP collaboration tools and invited talks by CERN batch team and ROOT team)

About 90 (60+20+10) for the last in-person edition



- Participated in 2019
- Very interesting discussion about the different experiments' approaches
- No chance yet to implement the "Ideal Courses"

Training ROOT: Change over time

Then

- Getting people set up hardest thing about teaching ROOT IMO (especially without a CERN account) (Docker, VMs etc.)
- CINT was really hard to learn and teach: mostly C++, but loads of exceptions
- PyROOT felt like an afterthought, only with rootpy etc. really useable

Now

- Now with conda, setting up ROOT is as easy (arguably easier) than a scientific python based stack
- Cling's use of (nearly) standard, modern C++ makes it a valuable debugging and learning tool
- Python interface has gotten very good!
  - Coupling cling/pyROOT with jupyter creates a very powerful interactive environment for prototyping, teaching and exploratory analysis
  - RDataFrame alone already has profoundly influenced how I and others think about their analyses, looking forward to the rest of ROOT 7

Training ROOT: Different audiences

Undergraduate students:

- Usually very computer-sciency or very little computing background. Little middle-ground

Graduate students:

- Usually very focussed on their individual needs

Big experiments vs. small experiments:

- For small experiments, the boundary between developer and user very unclear: which parts of ROOT need to be taught?

For many tasks ROOT still the easiest “batteries included” solution: often hard to pick right packages to use in scientific python landscape

Training ROOT: External experts or in-house?

External

- Clearly no-one knows more about software than the authors
- Can bring new ideas to the collaboration
- Not all collaborations have experts on all tools that need teaching

Very complementary, ideally combine both!

In-house

- Much more familiar with use case, can focus on most relevant topics
- Knows customisations and details of experiment's framework (sometimes not used as intended by original authors)

Moving between experiments

- Experiments have extremely different software frameworks:
  - foundations (scikit-hep, ROOT) and analysis similar
  - everything inbetween very different in the details, even though the architectural concepts are similar
- Even “software experts” start off completely lost in new experiments
- In some experiments, strong separation between users and developers
- LHCb has a very good tiered teaching system: Starter Kit → Impact Kit, in both cases taught by (expert) peers, frequently participants in the previous year
  - Starter Kit: The essentials
  - Impact Kit: The nitty-gritty details, very useful for people changing between experiments
- With most experiments using more up-to-date ROOT versions now, and efforts like scikit-hep, the ecosystem is getting a bit more standardises, reducing the hurdle

In times of COVID: in person vs. hybrid vs. virtual

LHCb starter kits originally in-person only:

- very easy to determine whether students are lost, engage
- limits participation to those that can easily travel to CERN for a week (somewhat mitigated by very extensive online documentation)

SHiP and ALICE starter kits always hybrid:

- allows participation of people without the time/means to come to CERN

Due to COVID everything had to shift to virtual-only:

- Very hard to check on and engage with students
- Social side of getting to know the other new collaborators completely disappears

While hybrid and virtual-only starter kits are possible and often necessary, in-person only starter kits hard to beat

Some personal takeaways from teaching

- Learning by experimentation very important (for me), so move to jupyter and improvements to cling/pyROOT very welcome!
- Peers make excellent teachers: they know what they can expect from the participants. Also: Teaching is the best way to learn!
- Most important thing: teaching to ask the right questions! Hard to do in person, even harder virtually when you can't determine if somebody is stuck
- Very high turnover of teachers: while it's good to get new students involved in teaching, experienced teachers often can't justify the time commitment or leave the field

Are we teaching software or physics?

- Difference between how to (technical) and why (statistical), particularly when teaching analysis. Impossible to separate!
- Many students (especially with computing backgrounds) are good at learning how to satisfy the compiler/interpreter when learning ROOT, but don't think about the statistical implications of what they are doing. Some ROOT documentation very good at explaining statistical background or giving helpful references, while others aren't
- RDataFrame in some parts of ROOT already allowed to think more about the analysis by removing unecessary technical complexity; can we do the same for statistics?

Fin