pip install ROOT

Experiences making a complex multi-language package accessible for Python users

Vincenzo Eduardo Padulano[1], *Jonas Rembser*[1]

[1] CERN, EP-SFT

21.10.2024, CHEP ‘24, Kraków, Poland

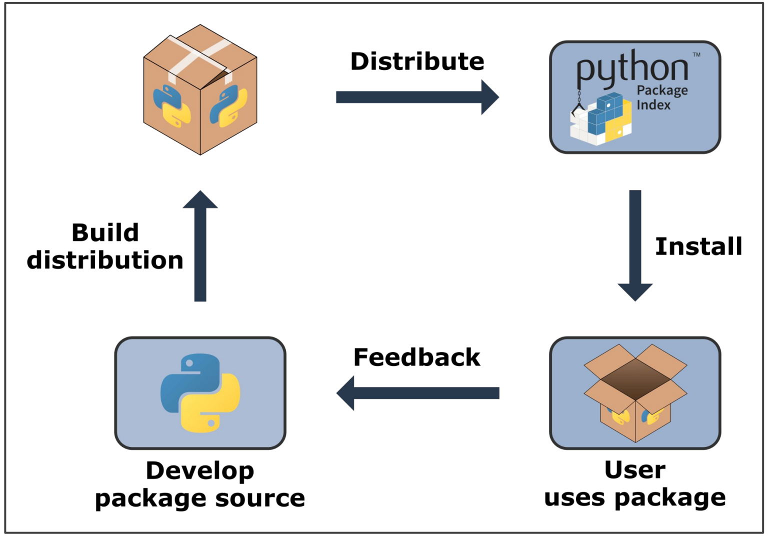
Outline

- Python packaging ecosystem
- `pip install ROOT`

Python packaging ecosystem

Python packaging

- Python Packaging Authority (PyPA)
  - Core working group for projects concerned with Python packaging
- Python packaging is a vast subject
  - PyPA's overview



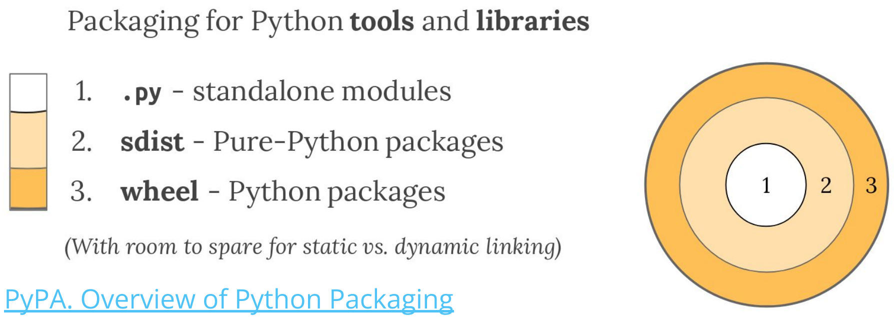
Python Packages, 2023. Tomas Beuzen, Tiffany Timbers

Python packaging

Three categories of distribution:

- Share one (or more) Python scripts
- sdist: **source code** installable via a backend
- wheel: **binary** compressed **archive**

> *"In fact, Python's package installer, pip, always prefers wheels because installation is always faster, so even pure-Python packages work better with wheels."*
> [PyPA. Overview of Python Packaging](#)



Packaging for Python **tools** and **libraries**

1. **.py** – standalone modules
2. **sdist** – Pure-Python packages
3. **wheel** – Python packages

(With room to spare for static vs. dynamic linking)

[PyPA. Overview of Python Packaging](#)

Python package build backends

Generic

Support package building for many programming languages

Examples: Anaconda, Spack

Full flexibility in dependency management

Enable building complex software stacks in one environment

The same software provides package manager and build backend

Python-focused

Specialised for building Python packages

Examples: setuptools, poetry, hatchling

Some emphasise integration with C, C++ code in the package

Examples: scikit-build-core, py-build-cmake, meson-python

Build backends are separate from the package manager (pip)

```
pip install ROOT
```

The ROOT (Python) package

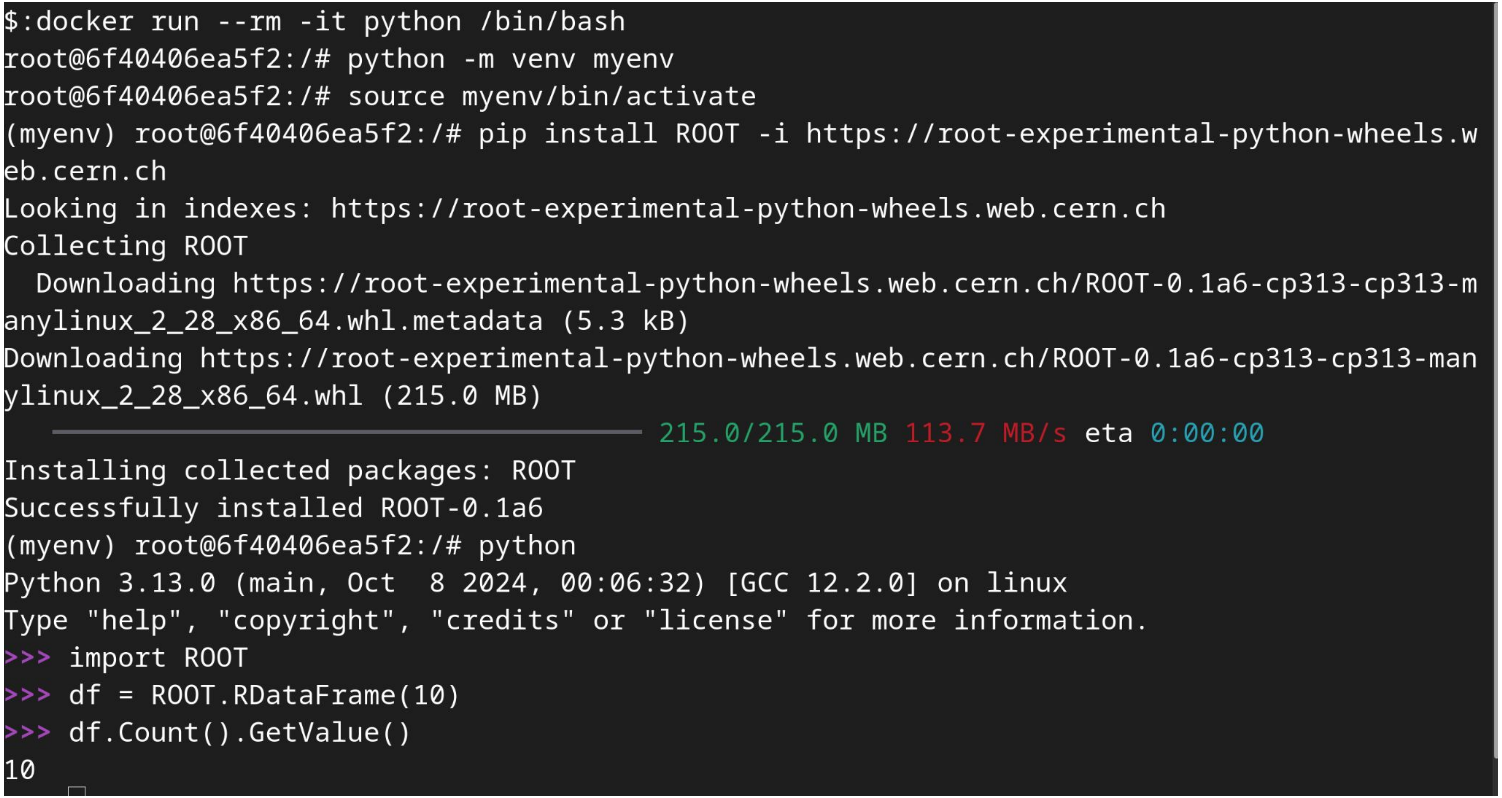
- **ROOT**: storage, I/O, processing, scientific analysis of **structured** data
- **EBs** data stored in **ROOT** format
- Many distribution channels
  - CVMFS, conda, system package managers (Linux, MacOS), official docker images, prebuilt binaries (Linux, MacOS, Windows), Snap.
- **Missing** distribution via **pip**
  - Makes the package more **easily obtainable** for Python users
  - Makes integration with downstream Python packages smoother

Seeing it in action

Working ROOT installation with pip (1.5x video speedup)

Seeing it in action

`pip install ROOT -i https://root-experimental-python-wheels.web.cern.ch`



```
$:docker run --rm -it python /bin/bash
root@6f40406ea5f2:/# python -m venv myenv
root@6f40406ea5f2:/# source myenv/bin/activate
(myenv) root@6f40406ea5f2:/# pip install ROOT -i https://root-experimental-python-wheels.w
eb.cern.ch
Looking in indexes: https://root-experimental-python-wheels.web.cern.ch
Collecting ROOT
  Downloading https://root-experimental-python-wheels.web.cern.ch/ROOT-0.1a6-cp313-cp313-m
anylinux_2_28_x86_64.whl.metadata (5.3 kB)
Downloading https://root-experimental-python-wheels.web.cern.ch/ROOT-0.1a6-cp313-cp313-man
ylinux_2_28_x86_64.whl (215.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 215.0/215.0 MB 113.7 MB/s eta 0:00:00
Installing collected packages: ROOT
Successfully installed ROOT-0.1a6
(myenv) root@6f40406ea5f2:/# python
Python 3.13.0 (main, Oct  8 2024, 00:06:32) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import ROOT
>>> df = ROOT.RDataFrame(10)
>>> df.Count().GetValue()
10
```

The ROOT wheels

- ROOT is published as a wheel
  - One wheel per supported Python version (3.8+)
  - For now, only support all Linux distributions (x86_64)
- The build backend (for now) is setuptools
- The wheels are built using two excellent tools
  - cibuildwheel
  - manylinux container image

The ROOT wheels

- cibuildwheel
  - CI job orchestration
  - Automatic process to build wheel
  - Includes options to run tests
- manylinux
  - Portable Linux build distributions (PEP513)
  - Defines a minimal image with core set of dependencies (like conda)
    - glibc + few others
  - ROOT wheel complies with `manylinux_2_28_x86_64` (PEP600)

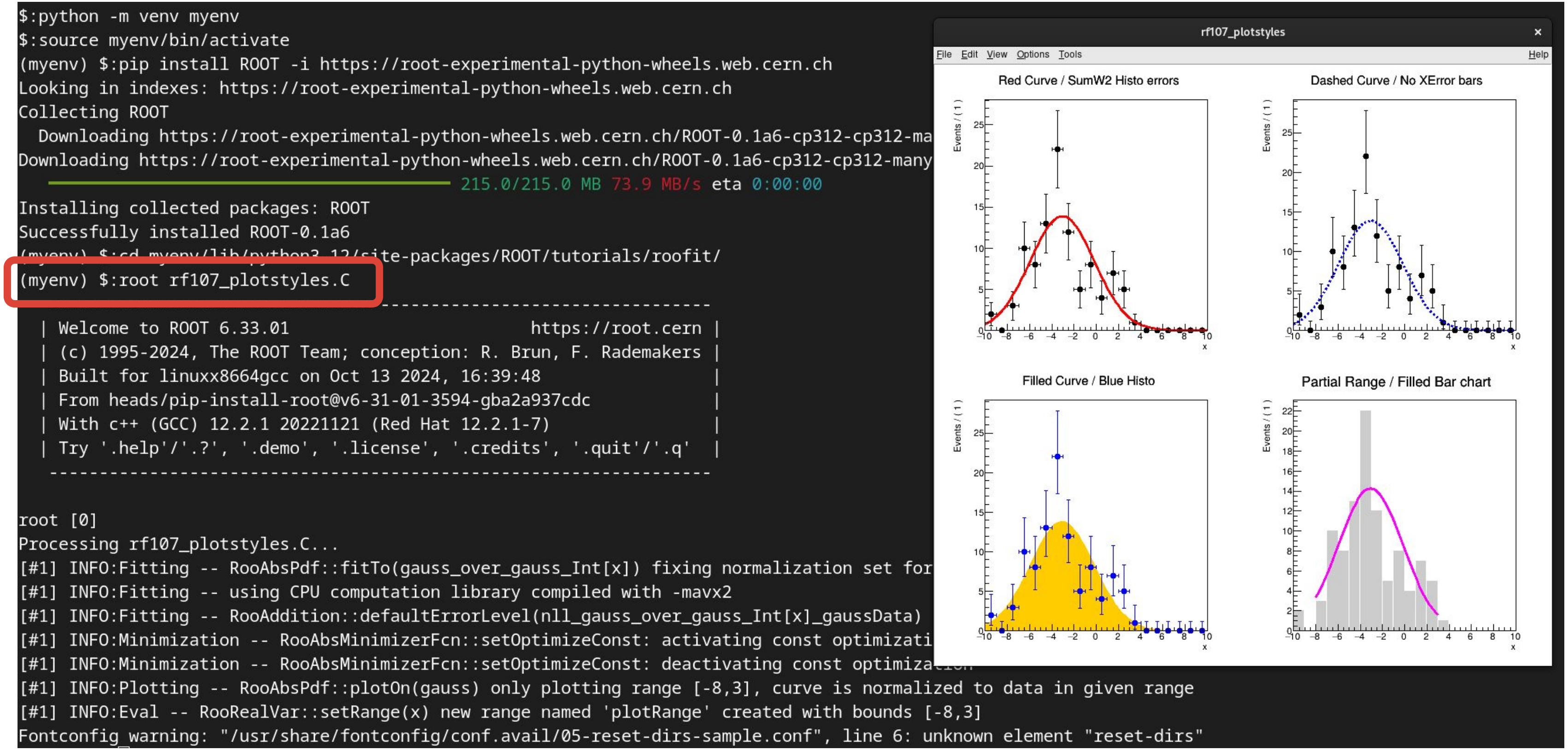
Full compatibility and portability across Linux distributions

Wheel work in progress

- WIP at https://github.com/root-project/root/pull/16669
- Few important ingredients
  - **Proper** management of **RPATH** variable **aligned** with “standard” Python `venv` **directory layout**
  - Similarly, install ROOT modules and libraries where they are expected
- Introducing changes that benefit the whole ROOT build system
- **Currently** provide only **few components**, to try things out
  - cling, Core libraries, I/O, RDataFrame, RooFit
- **Showcase CLI** executables with `root`

ROOT cli executables

As a first showcase, provide the `root` executable



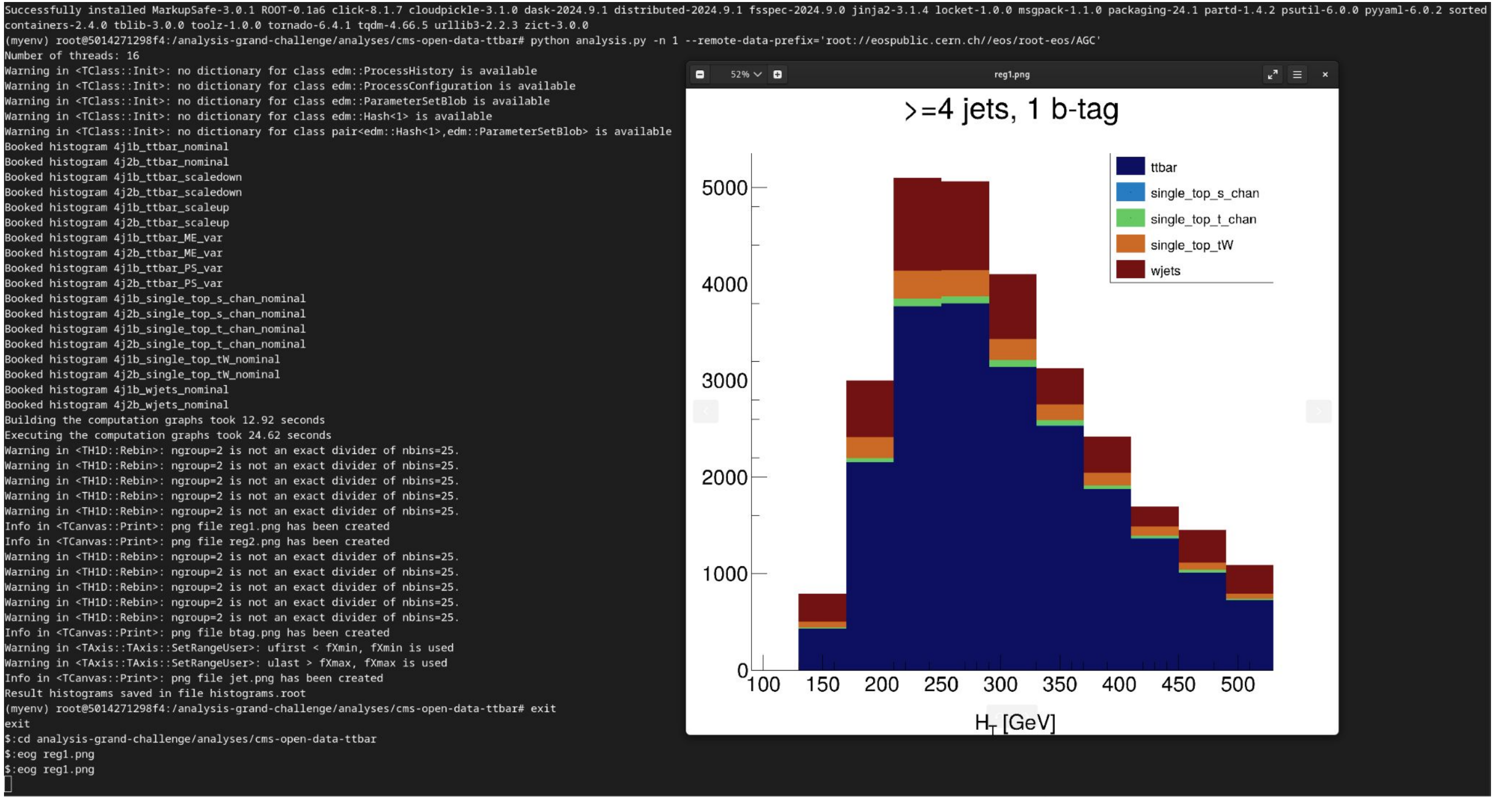
```
$:python -m venv myenv
$:source myenv/bin/activate
(myenv) $:pip install ROOT -i https://root-experimental-python-wheels.web.cern.ch
Looking in indexes: https://root-experimental-python-wheels.web.cern.ch
Collecting ROOT
  Downloading https://root-experimental-python-wheels.web.cern.ch/ROOT-0.1a6-cp312-cp312-ma
Downloading https://root-experimental-python-wheels.web.cern.ch/ROOT-0.1a6-cp312-cp312-many
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 215.0/215.0 MB 73.9 MB/s eta 0:00:00
Installing collected packages: ROOT
Successfully installed ROOT-0.1a6
(myenv) $:cd myenv/lib/python3.12/site-packages/ROOT/tutorials/roofit/
(myenv) $:root rf107_plotstyles.C
   ------------------------------------------------------------------
  | Welcome to ROOT 6.33.01                        https://root.cern |
  | (c) 1995-2024, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Oct 13 2024, 16:39:48                 |
  | From heads/pip-install-root@v6-31-01-3594-gba2a937cdc            |
  | With c++ (GCC) 12.2.1 20221121 (Red Hat 12.2.1-7)                |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
   ------------------------------------------------------------------

root [0]
Processing rf107_plotstyles.C...
[#1] INFO:Fitting -- RooAbsPdf::fitTo(gauss_over_gauss_Int[x]) fixing normalization set for
[#1] INFO:Fitting -- using CPU computation library compiled with -mavx2
[#1] INFO:Fitting -- RooAddition::defaultErrorLevel(nll_gauss_over_gauss_Int[x]_gaussData)
[#1] INFO:Minimization -- RooAbsMinimizerFcn::setOptimizeConst: activating const optimizati
[#1] INFO:Minimization -- RooAbsMinimizerFcn::setOptimizeConst: deactivating const optimiza
[#1] INFO:Plotting -- RooAbsPdf::plotOn(gauss) only plotting range [-8,3], curve is normalized to data in given range
[#1] INFO:Eval -- RooRealVar::setRange(x) new range named 'plotRange' created with bounds [-8,3]
Fontconfig warning: "/usr/share/fontconfig/conf.avail/05-reset-dirs-sample.conf", line 6: unknown element "reset-dirs"
```

A more complete example

Analysis grand challenge benchmark with CMS OpenData

A more complete example

Analysis grand challenge benchmark with CMS OpenData



```
Successfully installed MarkupSafe-3.0.1 ROOT-0.1a6 click-8.1.7 cloudpickle-3.1.0 dask-2024.9.1 distributed-2024.9.1 fsspec-2024.9.0 jinja2-3.1.4 locket-1.0.0 msgpack-1.1.0 packaging-24.1 partd-1.4.2 psutil-6.0.0 pyyaml-6.0.2 sorted
containers-2.4.0 tblib-3.0.0 toolz-1.0.0 tornado-6.4.1 tqdm-4.66.5 urllib3-2.2.3 zict-3.0.0
(myenv) root@5014271298f4:/analysis-grand-challenge/analyses/cms-open-data-ttbar# python analysis.py -n 1 --remote-data-prefix='root://eospublic.cern.ch//eos/root-eos/AGC'
Number of threads: 16
Warning in <TClass::Init>: no dictionary for class edm::ProcessHistory is available
Warning in <TClass::Init>: no dictionary for class edm::ProcessConfiguration is available
Warning in <TClass::Init>: no dictionary for class edm::ParameterSetBlob is available
Warning in <TClass::Init>: no dictionary for class edm::Hash<1> is available
Warning in <TClass::Init>: no dictionary for class pair<edm::Hash<1>,edm::ParameterSetBlob> is available
Booked histogram 4j1b_ttbar_nominal
Booked histogram 4j2b_ttbar_nominal
Booked histogram 4j1b_ttbar_scaledown
Booked histogram 4j2b_ttbar_scaledown
Booked histogram 4j1b_ttbar_scaleup
Booked histogram 4j2b_ttbar_scaleup
Booked histogram 4j1b_ttbar_ME_var
Booked histogram 4j2b_ttbar_ME_var
Booked histogram 4j1b_ttbar_PS_var
Booked histogram 4j2b_ttbar_PS_var
Booked histogram 4j1b_single_top_s_chan_nominal
Booked histogram 4j2b_single_top_s_chan_nominal
Booked histogram 4j1b_single_top_t_chan_nominal
Booked histogram 4j2b_single_top_t_chan_nominal
Booked histogram 4j1b_single_top_tW_nominal
Booked histogram 4j2b_single_top_tW_nominal
Booked histogram 4j1b_wjets_nominal
Booked histogram 4j2b_wjets_nominal
Building the computation graphs took 12.92 seconds
Executing the computation graphs took 24.62 seconds
Warning in <TH1D::Rebin>: ngroup=2 is not an exact divider of nbins=25.
Warning in <TH1D::Rebin>: ngroup=2 is not an exact divider of nbins=25.
Warning in <TH1D::Rebin>: ngroup=2 is not an exact divider of nbins=25.
Warning in <TH1D::Rebin>: ngroup=2 is not an exact divider of nbins=25.
Warning in <TH1D::Rebin>: ngroup=2 is not an exact divider of nbins=25.
Info in <TCanvas::Print>: png file reg1.png has been created
Info in <TCanvas::Print>: png file reg2.png has been created
Warning in <TH1D::Rebin>: ngroup=2 is not an exact divider of nbins=25.
Warning in <TH1D::Rebin>: ngroup=2 is not an exact divider of nbins=25.
Warning in <TH1D::Rebin>: ngroup=2 is not an exact divider of nbins=25.
Warning in <TH1D::Rebin>: ngroup=2 is not an exact divider of nbins=25.
Warning in <TH1D::Rebin>: ngroup=2 is not an exact divider of nbins=25.
Info in <TCanvas::Print>: png file btag.png has been created
Warning in <TAxis::TAxis::SetRangeUser>: ufirst < fXmin, fXmin is used
Warning in <TAxis::TAxis::SetRangeUser>: ulast > fXmax, fXmax is used
Info in <TCanvas::Print>: png file jet.png has been created
Result histograms saved in file histograms.root
(myenv) root@5014271298f4:/analysis-grand-challenge/analyses/cms-open-data-ttbar# exit
exit
$:cd analysis-grand-challenge/analyses/cms-open-data-ttbar
$:eog reg1.png
$:eog reg1.png
```

Managing external dependencies

- Scientific packages often have non-trivial dependencies
- For this purpose, ROOT benefits a lot from conda
- `pip` build backends do not support external dependencies
    - This is a known and discussed limitation (PEP725, 1, 2, 3)
- What to do in the meanwhile?

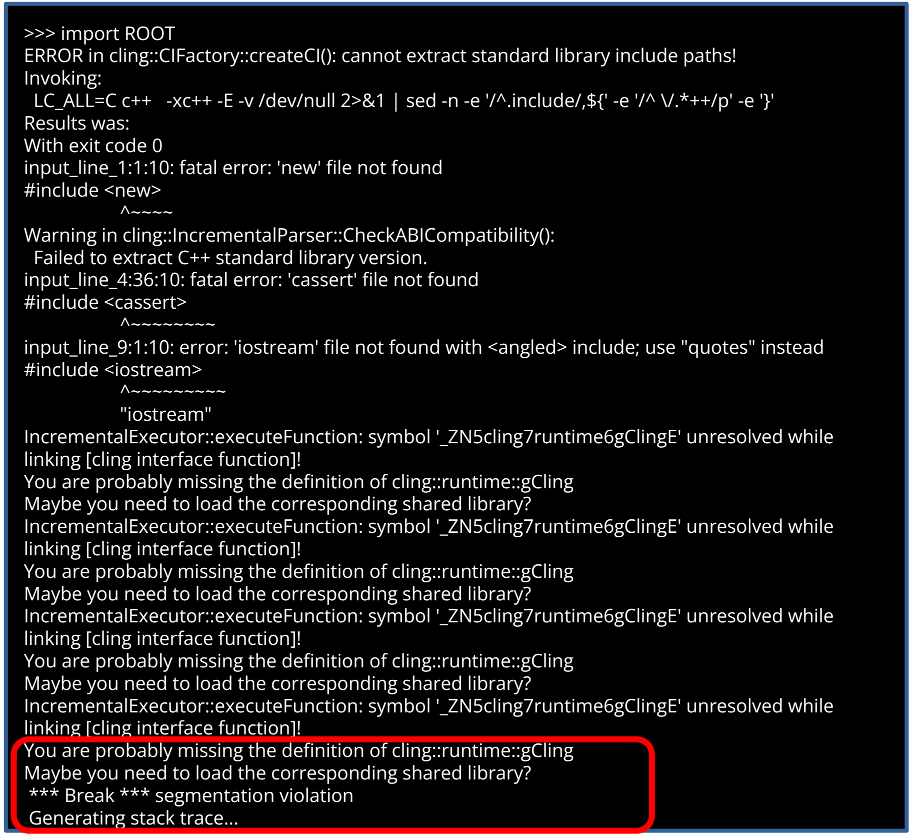
Managing external dependencies

Different strategies can be adopted:

- Bundling external dependency in library (quite common)
- Load libraries from other pip-installable dependencies (e.g. xrootd, tbb) (challenging)
- Expect the dependency in the system, fail graciously otherwise
  - Inviting the user to follow installation instructions

Gracious failure mechanism

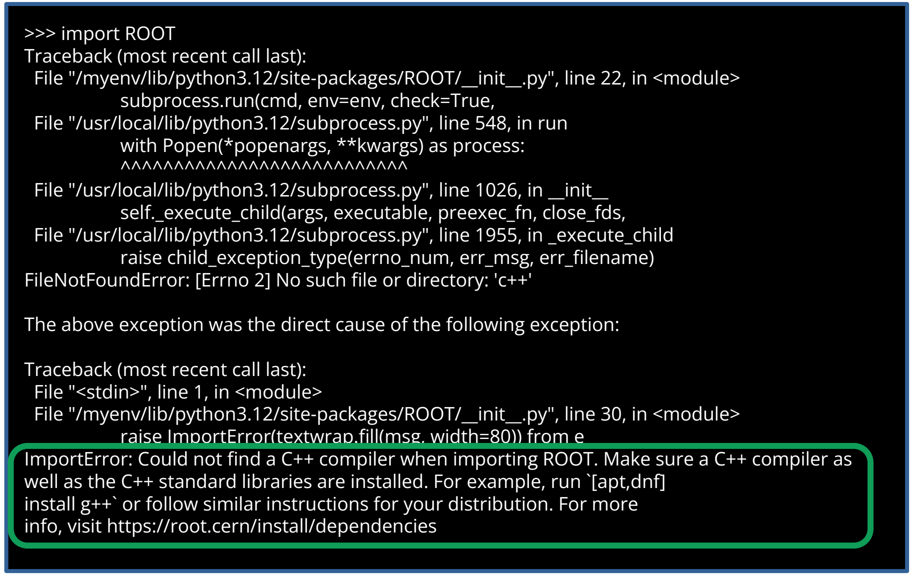
Without



```
>>> import ROOT
ERROR in cling::CIFactory::createCI(): cannot extract standard library include paths!
Invoking:
  LC_ALL=C c++   -xc++ -E -v /dev/null 2>&1 | sed -n -e '/^.include/,${' -e '/^ \/.*++/p' -e '}'
Results was:
With exit code 0
input_line_1:1:10: fatal error: 'new' file not found
#include <new>
         ^~~~~
Warning in cling::IncrementalParser::CheckABICompatibility():
  Failed to extract C++ standard library version.
input_line_4:36:10: fatal error: 'cassert' file not found
#include <cassert>
         ^~~~~~~~~
input_line_9:1:10: error: 'iostream' file not found with <angled> include; use "quotes" instead
#include <iostream>
         ^~~~~~~~~~
         "iostream"
IncrementalExecutor::executeFunction: symbol '_ZN5cling7runtime6gClingE' unresolved while
linking [cling interface function]!
You are probably missing the definition of cling::runtime::gCling
Maybe you need to load the corresponding shared library?
IncrementalExecutor::executeFunction: symbol '_ZN5cling7runtime6gClingE' unresolved while
linking [cling interface function]!
You are probably missing the definition of cling::runtime::gCling
Maybe you need to load the corresponding shared library?
IncrementalExecutor::executeFunction: symbol '_ZN5cling7runtime6gClingE' unresolved while
linking [cling interface function]!
You are probably missing the definition of cling::runtime::gCling
Maybe you need to load the corresponding shared library?
IncrementalExecutor::executeFunction: symbol '_ZN5cling7runtime6gClingE' unresolved while
linking [cling interface function]!
You are probably missing the definition of cling::runtime::gCling
Maybe you need to load the corresponding shared library?
 *** Break *** segmentation violation
 Generating stack trace...
```

`docker run python:3.12-slim` (~45MB)

With



```
>>> import ROOT
Traceback (most recent call last):
  File "/myenv/lib/python3.12/site-packages/ROOT/__init__.py", line 22, in <module>
    subprocess.run(cmd, env=env, check=True,
  File "/usr/local/lib/python3.12/subprocess.py", line 548, in run
    with Popen(*popenargs, **kwargs) as process:
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/subprocess.py", line 1026, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
  File "/usr/local/lib/python3.12/subprocess.py", line 1955, in _execute_child
    raise child_exception_type(errno_num, err_msg, err_filename)
FileNotFoundError: [Errno 2] No such file or directory: 'c++'

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/myenv/lib/python3.12/site-packages/ROOT/__init__.py", line 30, in <module>
    raise ImportError(textwrap.fill(msg, width=80)) from e
ImportError: Could not find a C++ compiler when importing ROOT. Make sure a C++ compiler as
well as the C++ standard libraries are installed. For example, run `[apt,dnf]
install g++` or follow similar instructions for your distribution. For more
info, visit https://root.cern/install/dependencies
```

Conclusions and outlook

Conclusions and outlook

```
pip install ROOT -i https://root-experimental-python-wheels.web.cern.ch
```

- Demonstrated **ROOT** installation via **pip**
- Experimental **wheels** are provided
  - Installable and working on **any Linux distribution** (x86_64)
  - Featuring some ROOT components to try out
- Drastically lowered obtainability barrier for Python users
- Next steps:
  - Run ROOT's **test suite** with **pip** builds as part of ROOT CI
  - Try more **flexible** build **backends** – scikit-build-core?
  - Split in **smaller wheels**, e.g. `root-core`, `root-rdf`, `root-roofit`
  - Make the **pip** installation **robust**, **towards** a first **beta version**

Conclusions and outlook

```
pip install ROOT -i https://root-experimental-python-wheels.web.cern.ch
```

Want to know more? Eager to try it out? Do you have suggestions for improvements? Would you like to contribute?

Meet me around CHEP! And feel free to contact me at

`vincenzo.eduardo.padulano@cern.ch`

And of course also send us feedback on the forum!