# Introducing new thesis students to ROOT

A. Iuliano (Università di Napoli and INFN)

ROOT Train The Trainers

12 May 2022

# Introduction and presentation

- About me: a PostDoc from Naples, graduated in 2021
- Main research background: neutrino detection within the ShiP and SND@LHC Collaboration
- Vital contribution provided by Bachelor and Master students: new thesis, original perspectives on the research, more young life to the group
- Here, talking about their ROOT training and research introduction
- My experience: trained 5 Bachelor and 4 Master students

Hello, my name is

Antonio

# Target of the training

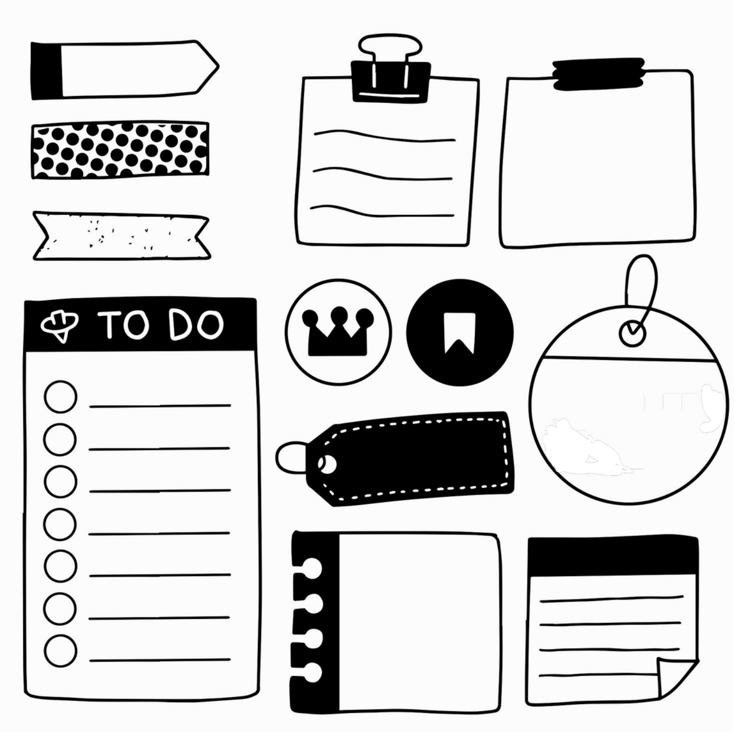
- Two main topologies of students (times here referring to Italian universities)
- Bachelor students (3 years degree):
  - Thesis duration not more than in 3 months, of which 2 in the lab
  - Almost always first time with ROOT
  - Need to focus in a few topics
  - Usually more guided into known coding structures and techniques
- Master students (2 years degree, after Bachelor):
  - Thesis duration about 8 months, of which 7 in the lab
  - May or not may already know ROOT from the Bachelor thesis
  - More freedom in code development and data analysis

# Common goals of the training

- Introduction to the software, focusing on the topics more needed for their analysis of laboratory data
- Starting with general ROOT training: what, why and how to use it
- Then, more focused sessions on data structure and libraries from our libraries
- Duration of the training itself:
  - 1 week for Bachelor students
  - 1-2 weeks for Master students
- After the training, the guidance is generally driven on question/answer basis.

  Learn what you need, when you need it

  (trying not to forget it the day after).



# Used resources

- General resources:
  - ROOT official basic course: https://github.com/root-project/training/tree/master/BasicCourse
  - Tutorial Jupyter/ROOT Notebooks, written as inspiration from the tutorials, but taking samples of our data to show the data format
- Both ROOT C++ and PyROOT are mentioned, then the training is focused towards one language or another, according to student preferences
- Last year students more driven to Python

  (due to a Python course in their courses)

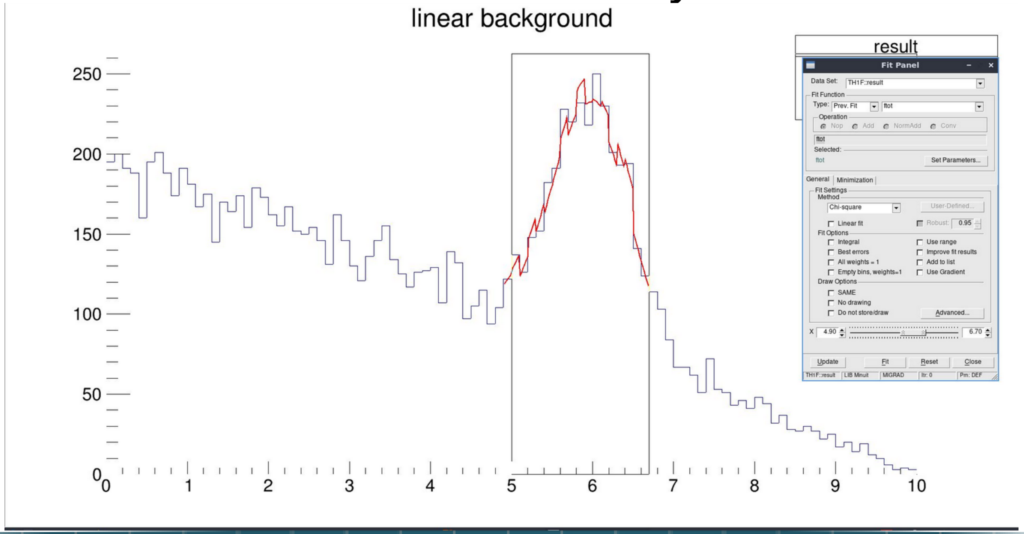
# Bachelor training topics

- Presenting here the topics and feedback received
- The workflow is usually:
  - Introduction to the C++/Python language (usually not required);
  - **Running ROOT with terminal, executing and compiling macros.**
- All bachelor thesis work done in execution mode (root -l mymacro.C). ACLiC compilation mentioned, but rarely used

**HELLO WORLD**

# Bachelor training topics

- The workflow is usually:
  - Introduction to the C++/Python language (usually not required);
  - Running ROOT with terminal, executing and compiling macros;
  - **Presenting data with graphs and histograms, simple fits**
- First showing how to edit histogram and perform fit with the ROOT GUI, then how to replicate the commands in the macro.
- Most of the students prefer to find the “most suitable” fit range and initial parameters with the Fit Panel GUI, then they use them in the macro.



Data Set: TH1F::result

Type: Prev. Fit | ftot

ftot

Selected: ftot

Method: Chi-square

□ Linear fit ☑ Robust: 0.95

□ Integral □ Use range

□ Best errors □ Improve fit results

□ All weights = 1 □ Add to list

□ Empty bins, weights=1 □ Use Gradient

□ SAME

□ No drawing

□ Do not store/draw

X 4.90 6.70

TH1F::result | LIB Minuit | MIGRAD | Itr: 0 | Pm: DEF

# Bachelor training topics

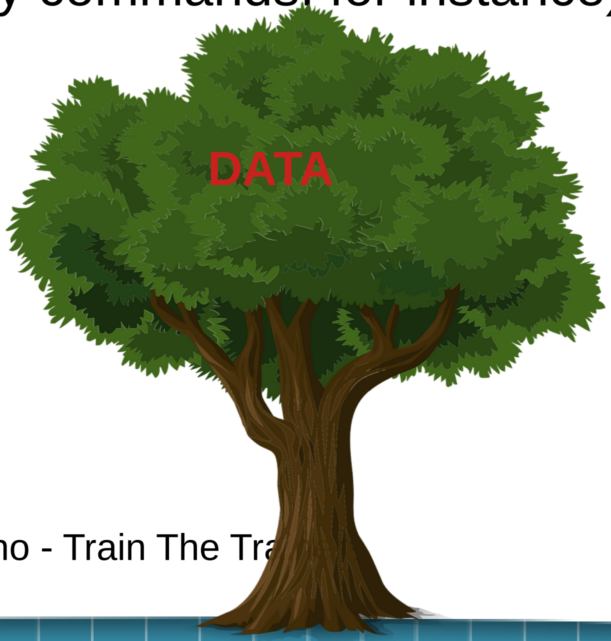
- The workflow is usually:
  - Introduction to the C++/Python language (usually not required)
  - Running ROOT with terminal, executing and compiling macros
  - Presenting data with graphs and histograms, simple fits
  - **TFile and TTree I/O**
- Great appreciation for TBrowser and TTreeViewer

# Bachelor training topics

- The workflow is usually:
  - Introduction to the C++/Python language (usually not required)
  - Running ROOT with terminal, executing and compiling macros
  - Presenting data with graphs and histograms, simple fits
  - **TFile and TTree I/O**
- Great appreciation for TBrowser and TTreeViewer
- Initially, TTree readout is shown in the “old way”, with SetBranchAddress (this applies to C++, for PyROOT is automatic). Main reasons for this approach:
  - More control in the flow: you “see” the variable storing the value;
  - Can be explained along the TTree creation and writiing, with Branch();
  - Easier to generalize to any convoluted data structure with MakeClass (RDataFrames and TClonesArrays do not usually get along well, at least to my experience)

# Master training topics

- If the student did not use ROOT before, a first week with the same topics of the Bachelor. Then, more advanced topics:
  - **Different ways of reading TTrees: TTreeReader,RDataFrame**
- Usually, TTreeReader used for loop-based procedural macros. Very useful alternative to SetBranchAddress, requires less knowledge of data types
- RDataFrame (as long as RVec operations) essential for vector-based computations in C++ codes
- However, in Python codes, export to pandas remains preferable as it gives more options (sort and groupby commands, for instance)



# Master training topics

- If the student did not use ROOT before, a first week with the same topics of the Bachelor. Then, more advanced topics:
  - Different ways of reading TTrees: TTreeReader,RDataFrame,
  - **Creating their own ROOT-based classes**
- Here, ACLiC or MakeClass compilation of C++ classes is used.
- Then the classes can be loaded into ROOT C++ and PyROOT scripts

**ClassDef(myclass,5)**

**ClassImp(myclass)**

# Master training topics

- If the student did not use ROOT before, a first week with the same topics of the Bachelor. Then, more advanced topics:
  - Different ways of reading TTrees: TTreeReader,RDataFrame,
  - Creating their own ROOT-based classes
  - **Advanced ROOT libraries:** TGeometry, TMVA, RooFit (depending on the thesis topic)
- Each of these libraries comes with a nice manual and tutorials
- Usually, at this point students are independent enough to study the references on their own → just provide them, and answer their questions

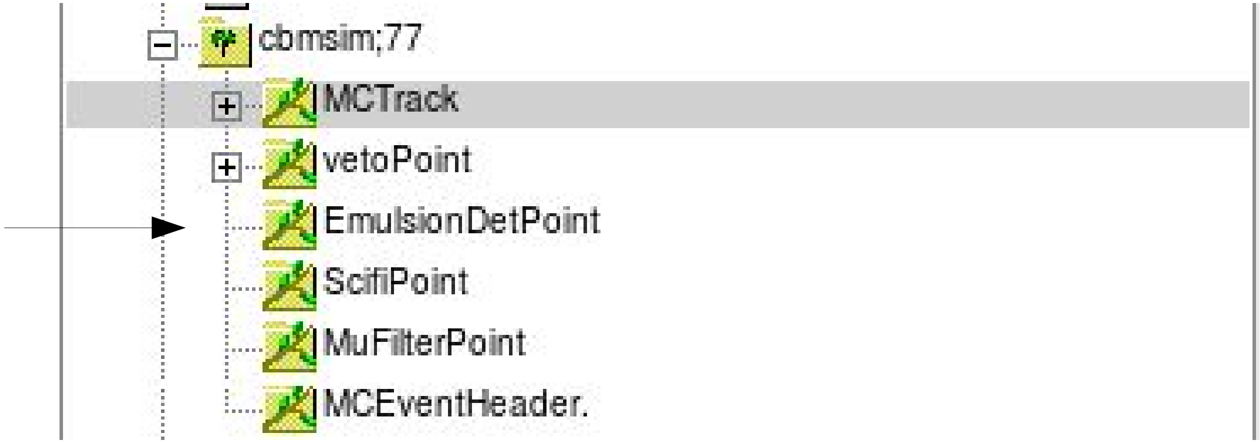
# Final remarks

- Today, HEP research is more and more directed towards scipy and sklearn algorithms and libraries
- Easier and wider interface to commercial and advanced classifiers, especially for machine learning applications
- However, ROOT still considered the main toolkit to introduce new researchers with, for the following reasons:
  - A clear and self-contained toolkit for physicists: in the same Canvas, you see the histogram, statistics, fit results, etc.
  - Comfortable data storage and inspection: TTrees in .root files can be inspected on the fly, and can be much more convoluted than .csv files
- Many thanks to all the ROOT developers for their continuous work!

# TTree Readout

- Initially, TTree readout is shown in the “old way”, with SetBranchAddress (of course in C++, for PyROOT is automatic). Main reasons for this approach:
  - More control in the flow: you “see” the variable storing the value;
  - Can be explained along the TTree creation and writiing, with Branch();
  - Easier to generalize to any convoluted data structure with MakeClass (RDataFrames and TClonesArrays do not usually get along well, at least to my experience)

These are all TClonesArrays of ROOT based objects


cbmsim;77
MCTrack
vetoPoint
EmulsionDetPoint
ScifiPoint
MuFilterPoint
MCEventHeader.