Imperial College London

Tracker Software: How to

A. Dobbs, CM39, 25th June 2014

Contents

Tracker Software Overview

- a very brief review

Tracker Software Overview

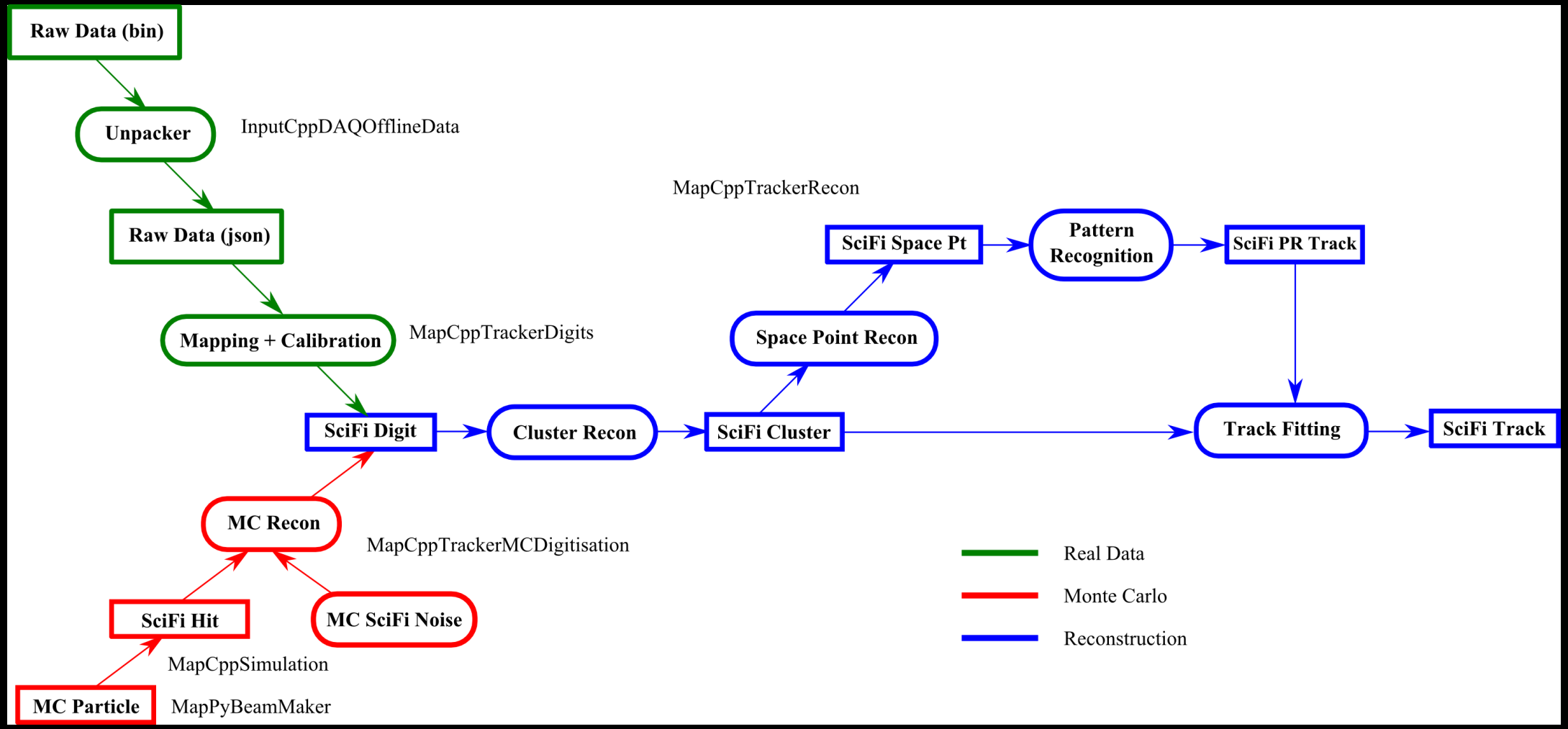
What It Does

- Digitise real data from DAQ output
- Digitise MC data
- Provide MC geometry and noise
- Reconstruct data from digits through to final Kalman tracks
- Display output for online use and offline convenience

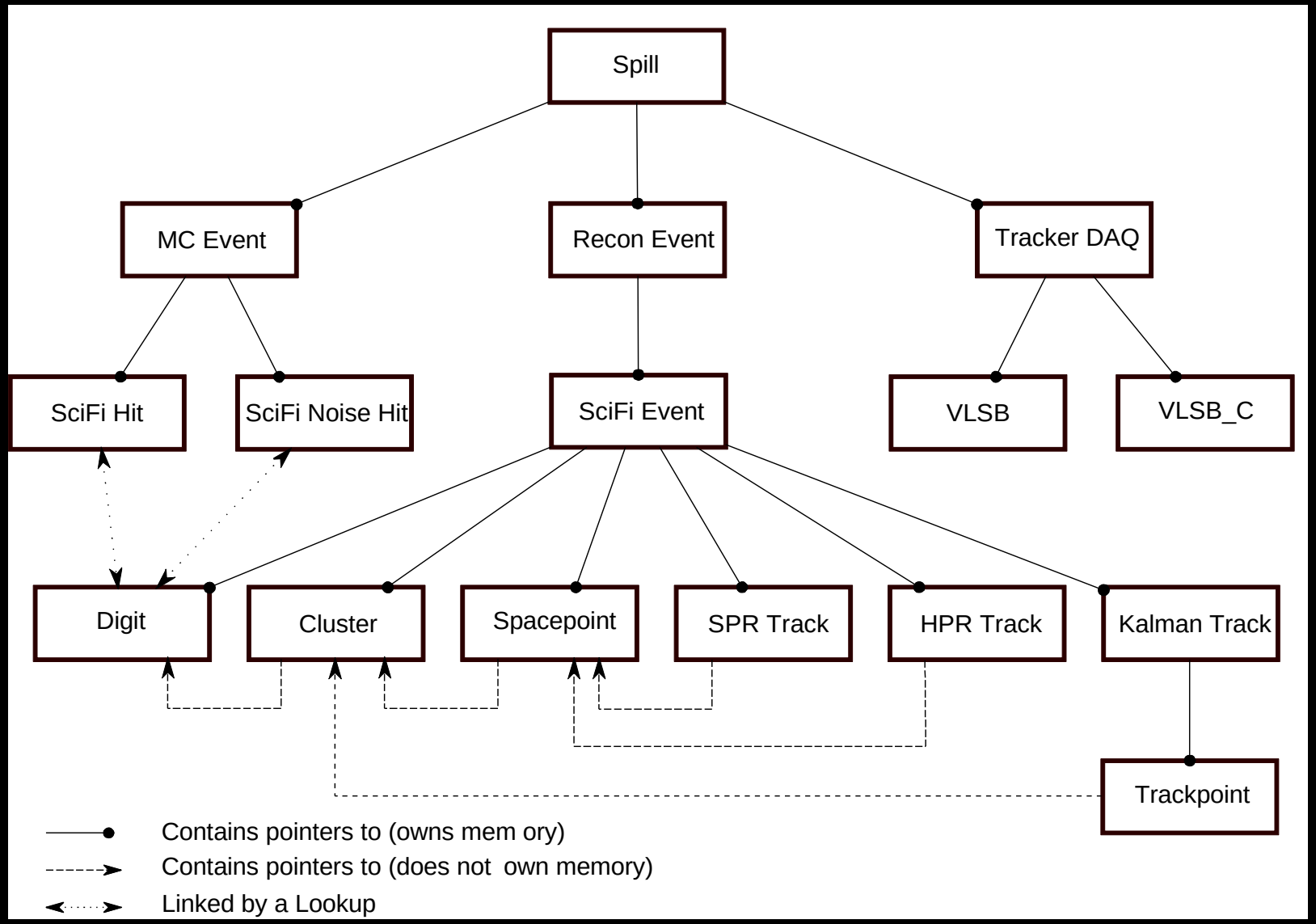
What It Does Not Do

- Unpacking (DAQ)
- Emittance reconstruction (xboa)
- Global fitting (global)
- Analysis (you! though I will provide some freebies…)

Reconstruction Path Review



Datastructure



Obtaining and Installing the Code

Obtaining the Code

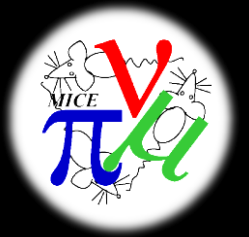
- MAUS available as tar balls from http://micewww.pp.rl.ac.uk/maus/
- MAUS is also stored in Launchpad: https://launchpad.net/maus
  - Software is version controlled by Bazaar (modern version of CVS, Subversion, etc): http://bazaar.canonical.com/en/
  - Access latest stable release with: `bzr branch lp:maus`
  - Access SciFi development branch with: `bzr branch lp:~maus-scifi/maus/devel`
- Once downloaded, install by running the script: `./install_build_test.bash`
- See documentation at http://micewww.pp.rl.ac.uk/projects/maus/wiki

Running a Simulation

Running a Simulation with SciFi

- From the MAUS root directory enable the MAUS environment with: `source env.sh`
- From the MAUS root directory move to `bin/user/scifi`
- Run `./simulate_scifi.py --configuration_file datacard_mc_helical`
- Wait a bit…
- Output stored in `maus_output.root`
- Job done.

Customising the Launch Script




```
import io       # Generic python library for I/O
import gzip     # For compressed output # pylint: disable=W0611
import MAUS     # The main MAUS module

def run():

    my_input = MAUS.InputPySpillGenerator()              # Define the simulation spill structure
    # my_input = MAUS.InputCppRoot()                     # Input from a ROOT file

    my_map = MAUS.MapPyGroup()                           # Define a collection of map modules
    my_map.append(MAUS.MapPyBeamMaker())                 # Beam construction
    my_map.append(MAUS.MapCppSimulation())               # GEANT4 simulation
    # my_map.append(MAUS.MapCppTrackerMCNoise())         # SciFi noise
    my_map.append(MAUS.MapCppTrackerMCDigitization())    # SciFi electronics
    my_map.append(MAUS.MapCppTrackerRecon())             # SciFi recon
    datacards = io.StringIO(u"")                         # Datacard options (empty here)

    reducer = MAUS.ReduceCppPatternRecognition()         # Plot scifi events to screen while running
    # reducer = MAUS.ReducePyDoNothing()                 # Do not plot anything while running

    # my_output = MAUS.OutputPyJSON()                    # Write the output to a JSON file
    my_output = MAUS.OutputCppRoot()                     # Write the output to a ROOT file

    MAUS.Go(my_input, my_map, reducer, my_output, datacards)  # Run the simulatation

if __name__ == '__main__':
    run()
```

Useful Datacard Options

Variable	Value	Description
output_root_file_name	“maus_output.root”	Output ROOT file name
simulation_geometry_filename	“Stage4.dat”	What geometry to use
geant4_visualisation	False	Output GEANT4 track display
spill_generator_number_of_spills	10	Number of spills
binomial_n	1	Particles per spill
binomial_p	1	Probability a particle will become a track
SciFiPRHelicalOn	True	Helical track pattern recognition on
SciFiPRStraightOn	True	Straight track pattern recognition on
SciFiKalmanOn	True	Final Kalman fit on

Where Everything Is

What	Where
Executables and datacards	bin/user/scifi
Analysis executables	bin/user/scifi & bin/user/scifi/cpp
Useful utilities	bin/utilities
Datastructure	src/common_cpp/Datastructure
Maps	src/map
Reducers	src/reduce
Reconstruction backend	src/common_cpp/SciFi & src/common_cpp/Kalman
Plotting backend	src/common_cpp/Plotting/SciFi
Geometry	src/legacy/FILES/Models/

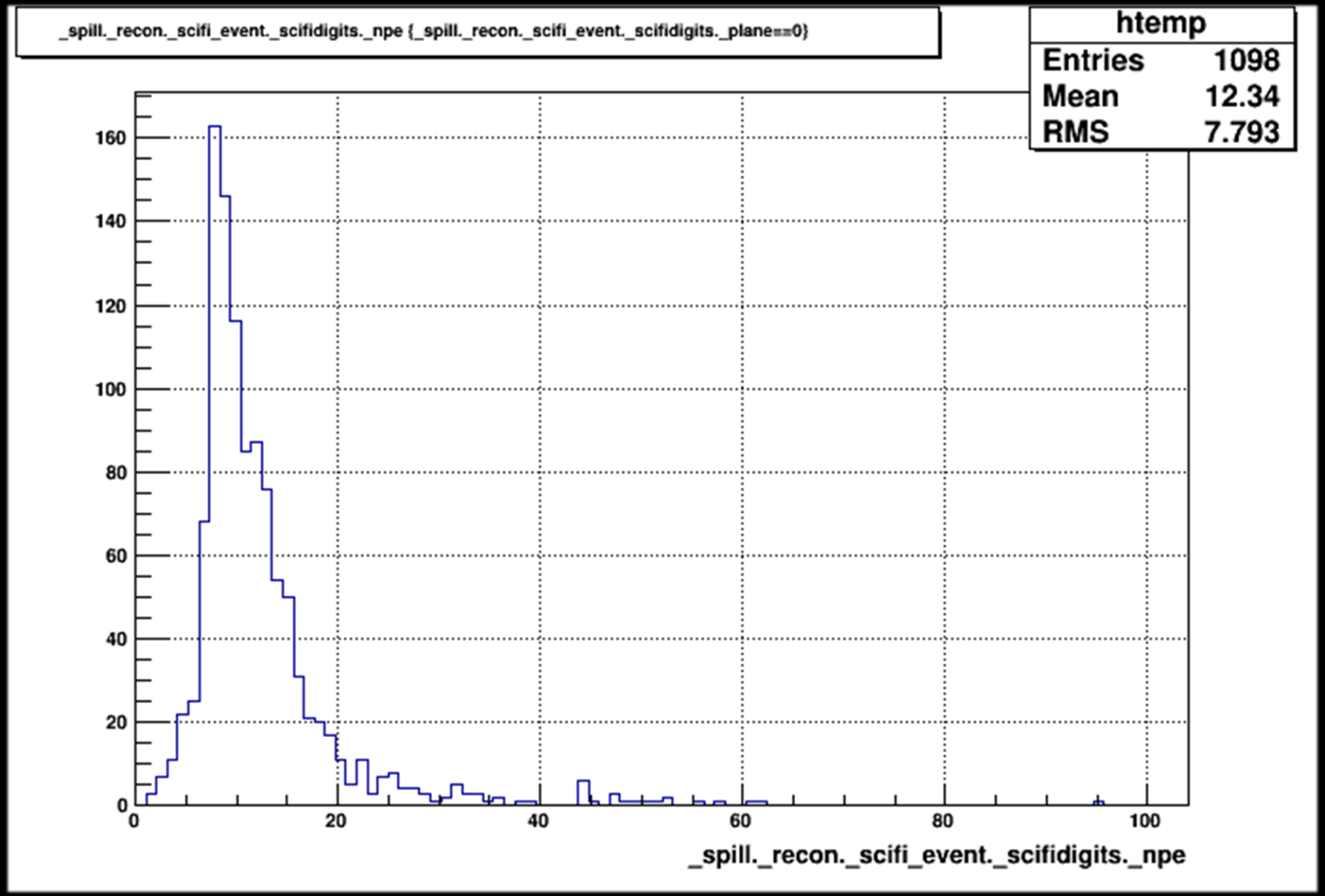
Analysing the Output

ROOT (Interpreted)

- Example interactive session (don't forget to `source env.sh` first!) :

```
.L $MAUS_ROOT_DIR/build/libMausCpp.so
TFile f1("maus_output.root")
TBrowser b
.ls
TTree* t = (TTree*)f1.Get("Spill")
t->Print()
t->Draw("_spill._recon._scifi_event._scifitracks._f_chi2")
t->Draw("_spill._recon._scifi_event._scifidigits._npe",
        "_spill._recon._scifi_event._scifidigits._plane==0")
```

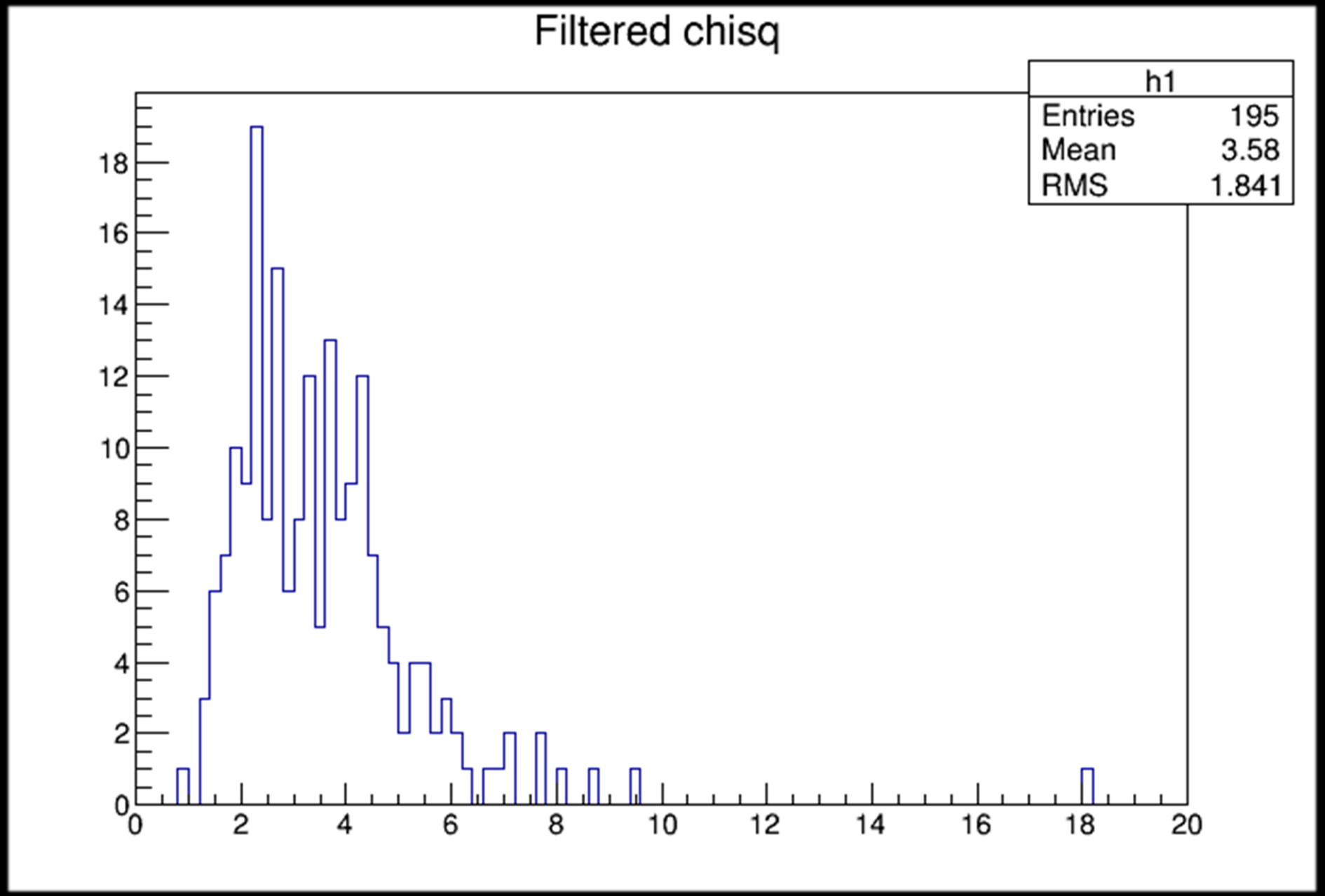
- Note: cannot look at data nested more than 2 nested containers deep (ROOT restriction)
- Can also loop over events (see python and c++ example)


_spill._recon._scifi_event._scifidigits._npe {_spill._recon._scifi_event._scifidigits._plane==0}
htemp
Entries 1098
Mean 12.34
RMS 7.793
160
140
120
100
80
60
40
20
0
0
20
40
60
80
100
_spill._recon._scifi_event._scifidigits._npe

PyROOT

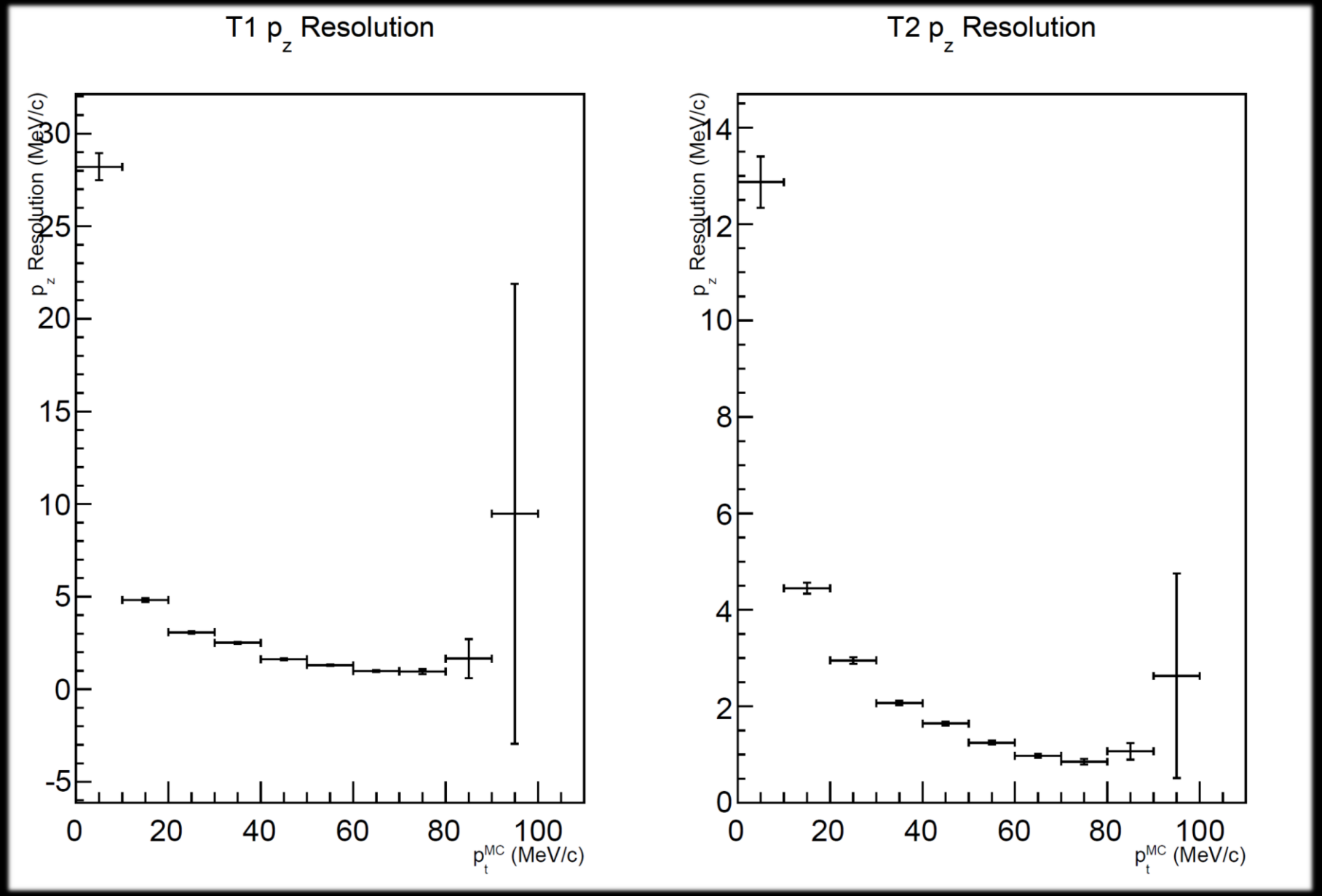
```
#!/usr/bin/env python

import libMausCpp
from ROOT import *
f1 = TFile("maus_output.root")
t1 = f1.Get("Spill")
data = MAUS.Data()
t1.SetBranchAddress("data", data)
h1 = TH1D("h1", "Filtered chisq", 100, 0, 20)
for i in range(1, t1.GetEntries()):
    t1.GetEntry(i)
    spill = data.GetSpill()
    if spill.GetDaqEventType() == "physics_event":
        for j in range(spill.GetReconEvents().size()):
            sfevt = spill.GetReconEvents()[j].GetSciFiEvent()
            for k in range(sfevt.scifitracks().size()):
                trk = sfevt.scifitracks()[k]
                h1.Fill(trk.f_chi2())
h1.Draw()
input("Press Enter to finish...")
```

Script available from:
http://micewww.pp.rl.ac.uk/documents/89


Filtered chisq
h1
Entries 195
Mean 3.58
RMS 1.841
0
2
4
6
8
10
12
14
16
18
20

Compiled ROOT - C++

- See `bin/user/scifi/cpp/scifi_analysis.cc` together with associated `Makefile` (try it out with `./scifi_analysis maus_output.root`)
- …
- Fastest method for large scale processing
- Allows code to be shared between online reducers and offline analysis code, see for example `bin/user/scifi/cpp/scifi_offline_viewer.cc` which shares a backend with the pattern recognition reducer (try it out with `./scifi_offline_viewer maus_output.root -p`)
- Somewhat harder to develop at the start (but really not that hard!)
- Can adapt the existing code for your own purposes


T1 p_z Resolution
p_z Resolution (MeV/c)
30
25
20
15
10
5
0
-5
0
20
40
60
80
100
p_t^MC (MeV/c)
T2 p_z Resolution
p_z Resolution (MeV/c)
14
12
10
8
6
4
2
0
0
20
40
60
80
100
p_t^MC (MeV/c)

Conclusion

- More material at:
  - Tracker software hints: http://micewww.pp.rl.ac.uk/projects/tracker/wiki/SoftwareHints
  - MAUS wiki: http://micewww.pp.rl.ac.uk/projects/maus/wiki
  - Doxygen (great for figuring out what data you have access to from the various data classes and how to call it): http://micewww.pp.rl.ac.uk/maus/MAUS_latest_version/doc/index.html
  - User guide: http://micewww.pp.rl.ac.uk/maus/MAUS_latest_version/maus_user_guide.pdf
- Report issues and request support using the issue tracker at http://micewww.pp.rl.ac.uk/projects/maus/issues
- Try it out and let me know how you get on! Code is under active development and I am interested in feedback (even if you are wrong…)