CAF

Common Analysis Files

Dominick Rocco

University of Minnesota

April 1, 2014

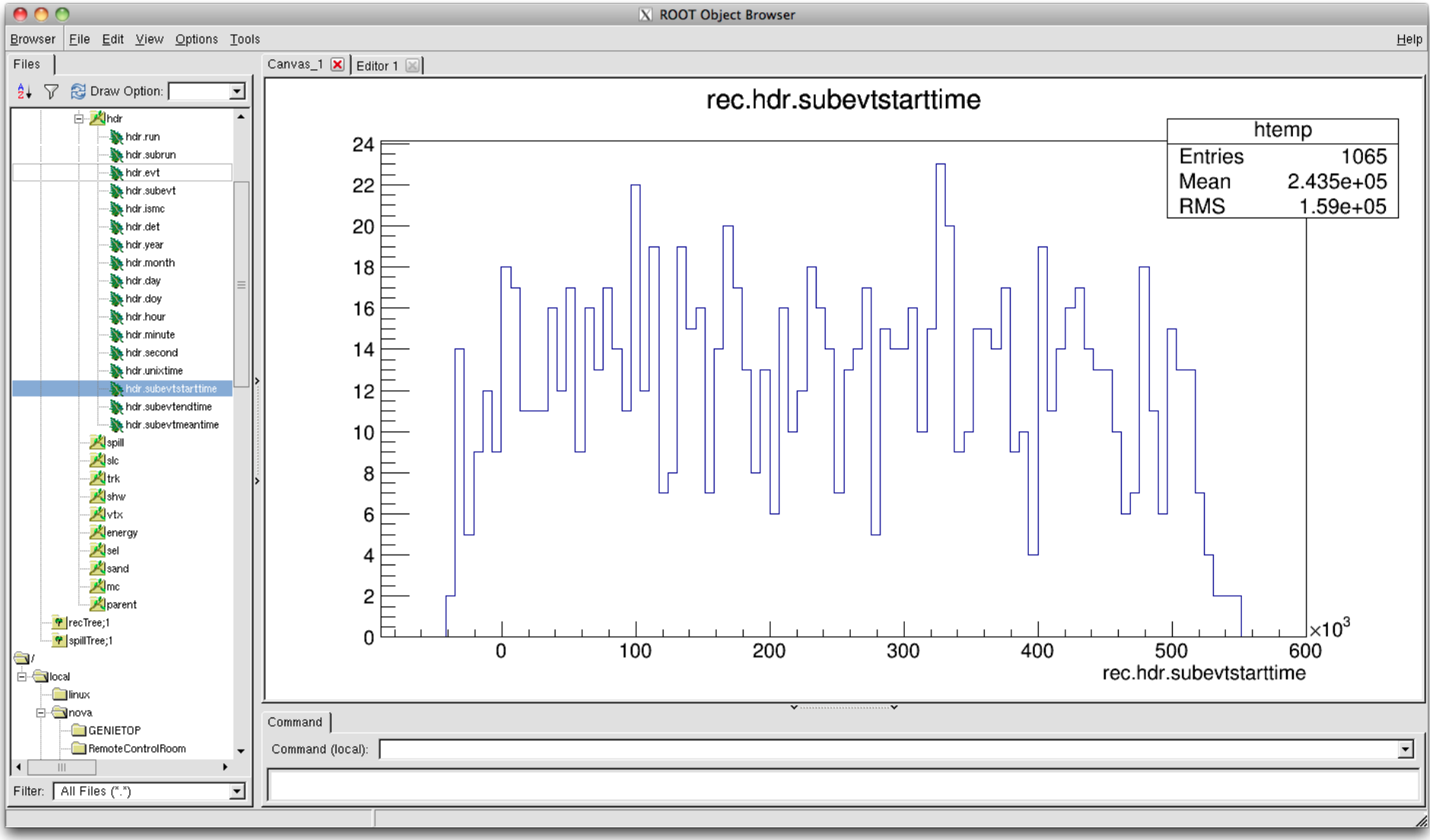
Introduction

- CAF stands for Common Analysis File
- These files contain a summary of the offline reconstruction
- The summary includes:
  - Header information, run, subrun, date, etc
  - Spill POT information
  - Reco information: tracks, prongs, showers, vertices, etc.
  - PID information
  - Energy estimates
  - MC information
- The summary does **not** include cell hit information

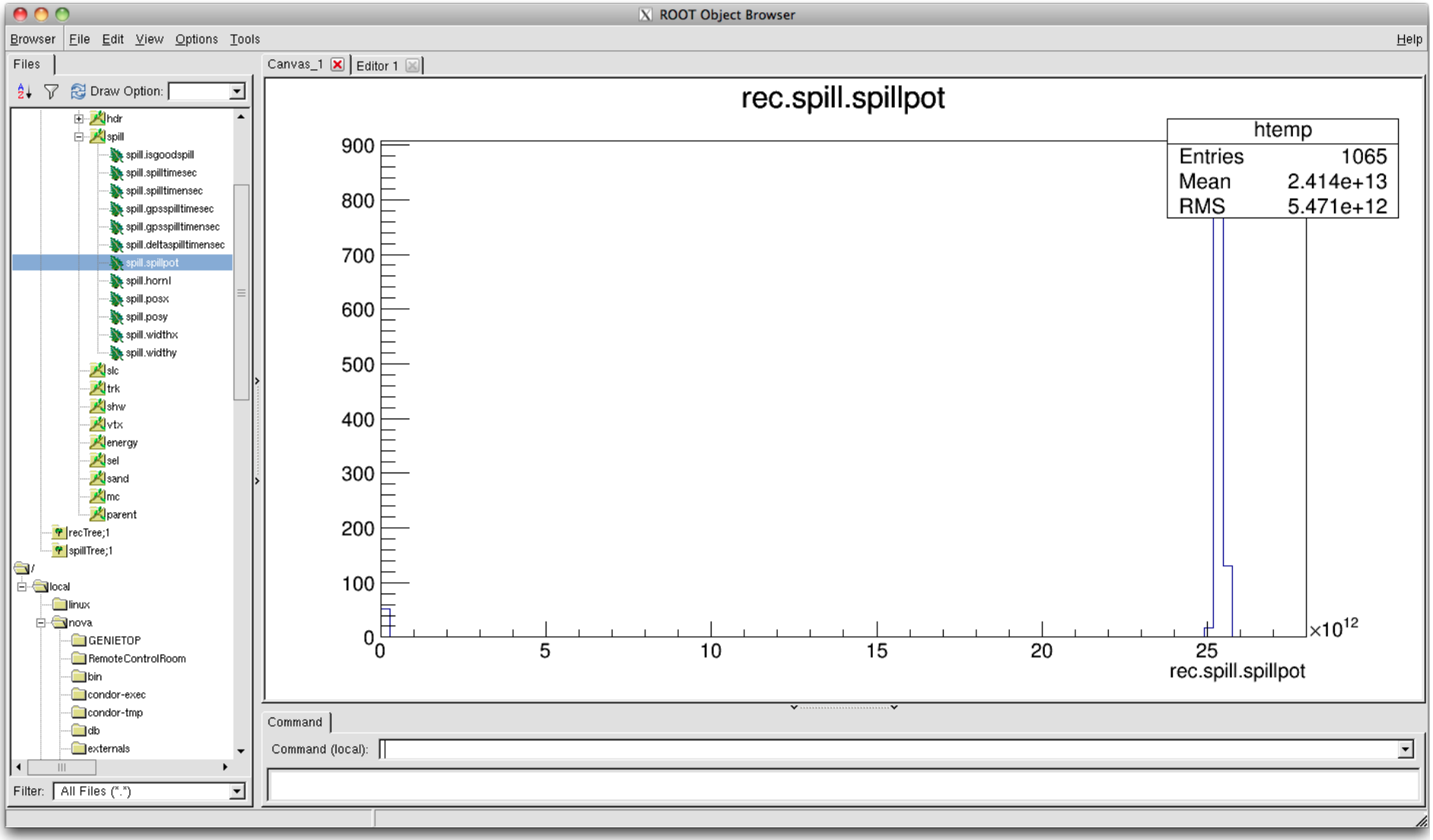
CAF STRUCTURE

- Entries in the CAF tree are intended to represent individual interactions
  - As opposed to art::Events which correspond to trigger windows
  - Interactions are resolved using Slicer4D
- Each entry corresponds to a single slice
- Each entry stores its data in a `StandardRecord` object
- `StandardRecord` objects are containers for the various tree branches

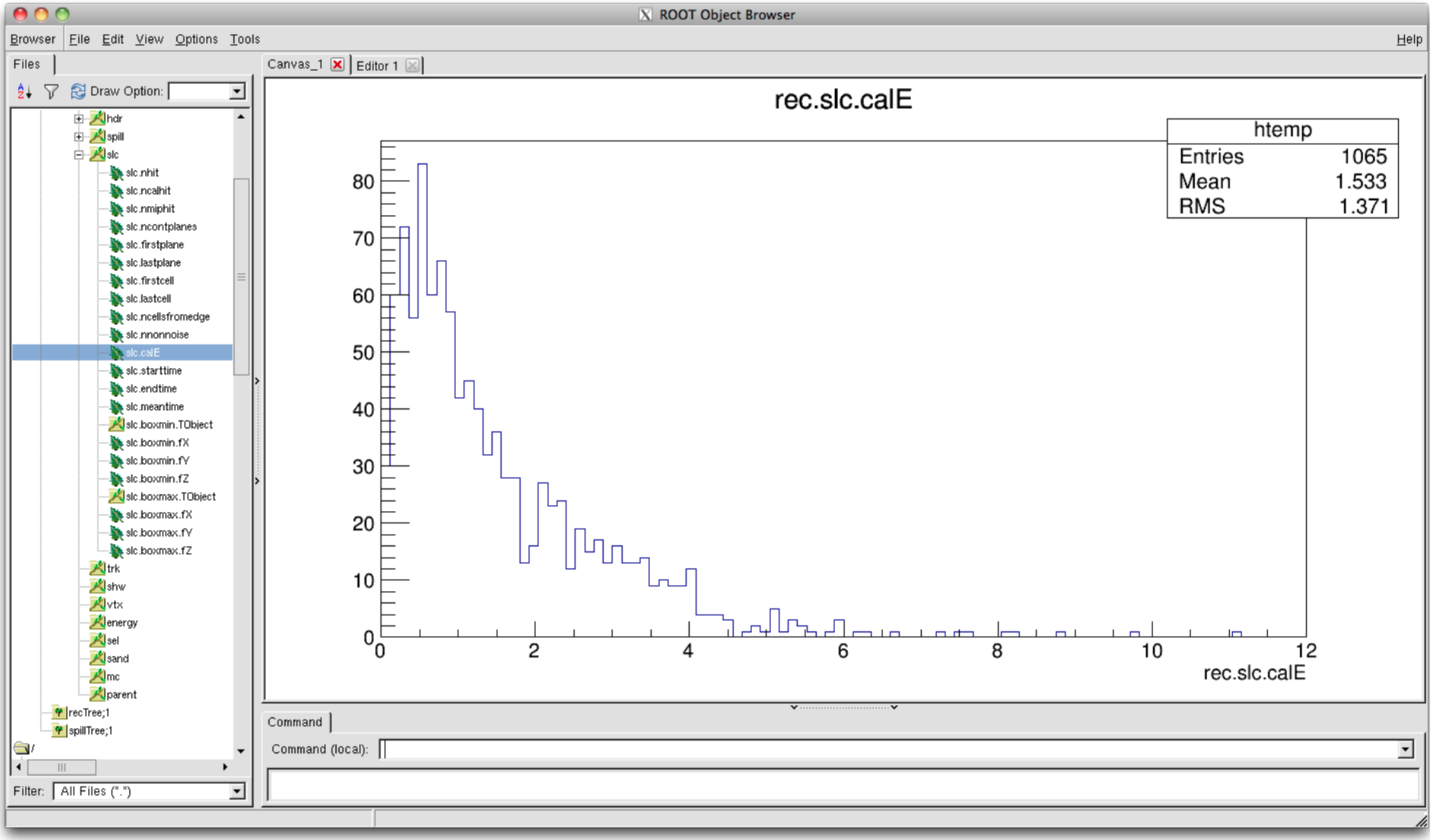
Header Branch



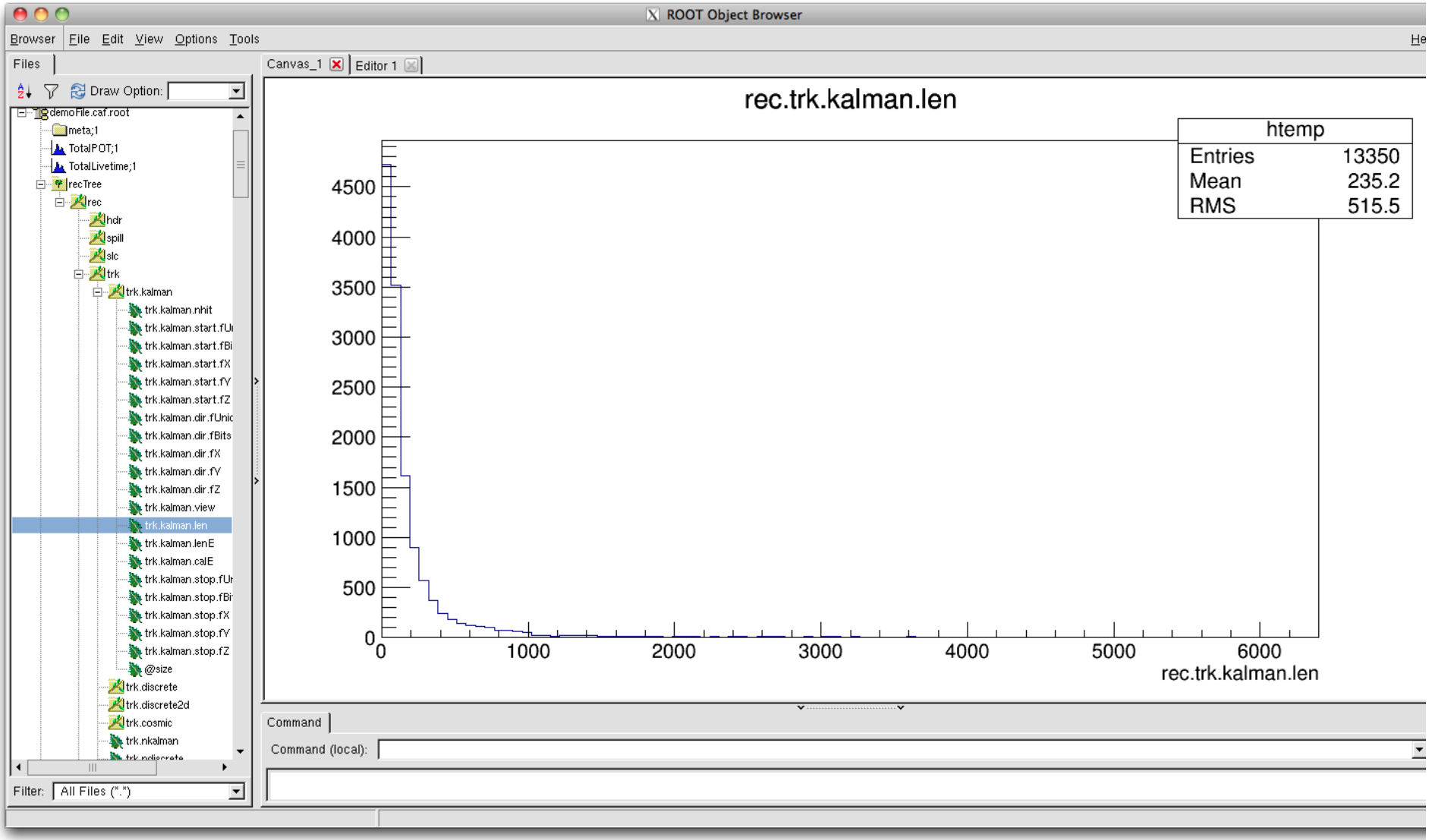
SPILL BRANCH



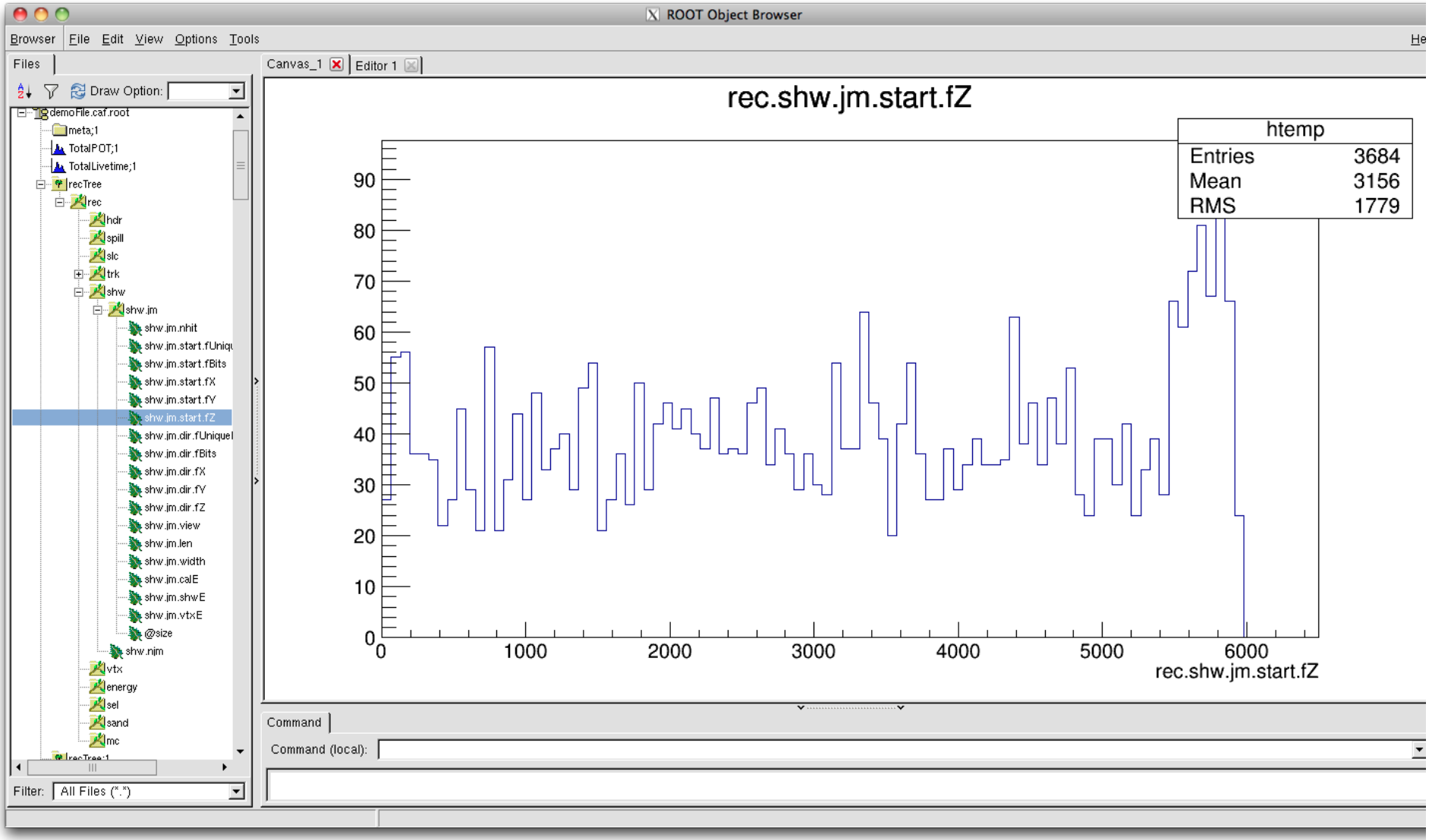
SLICE BRANCH



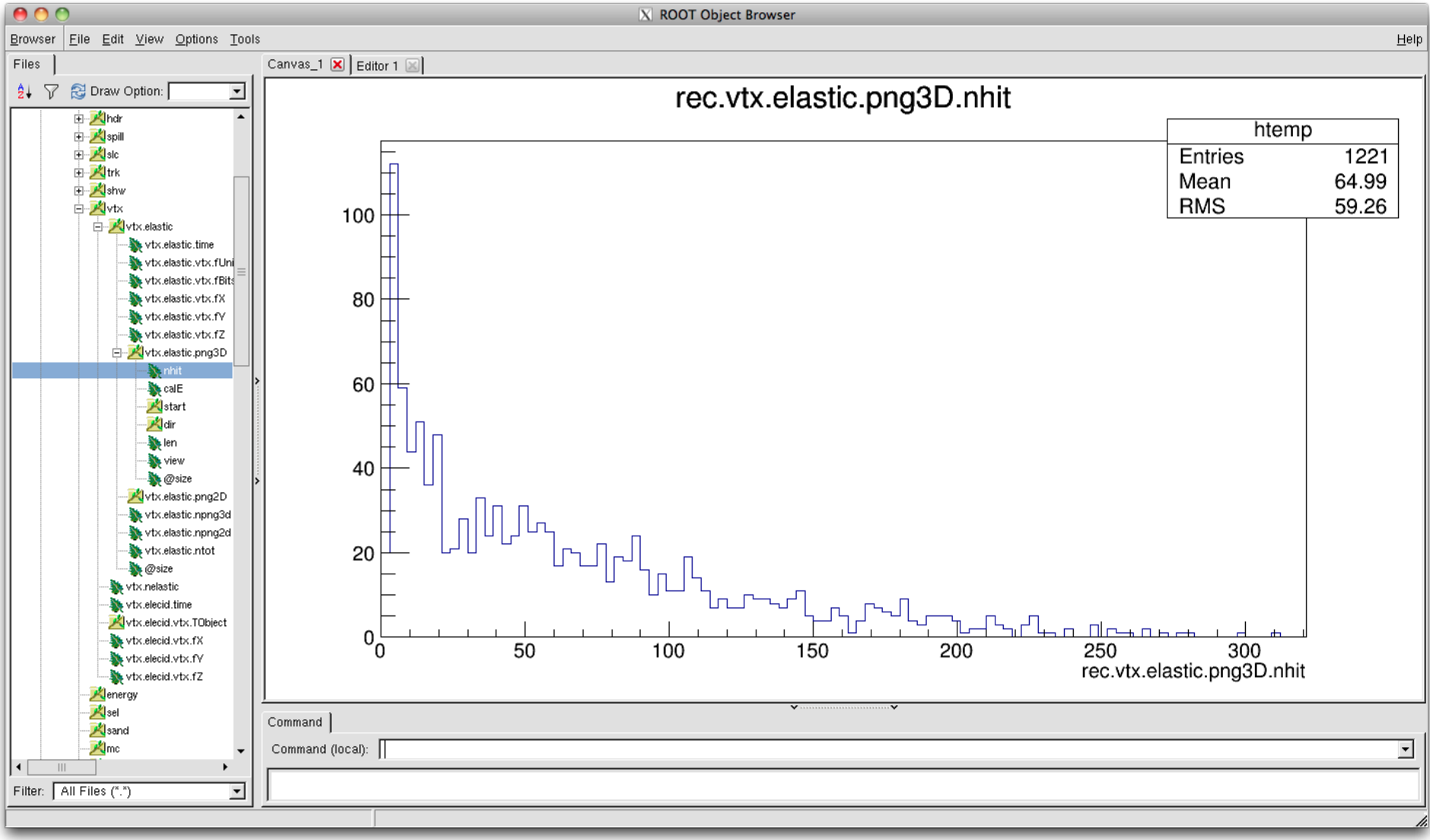
TRACK BRANCH



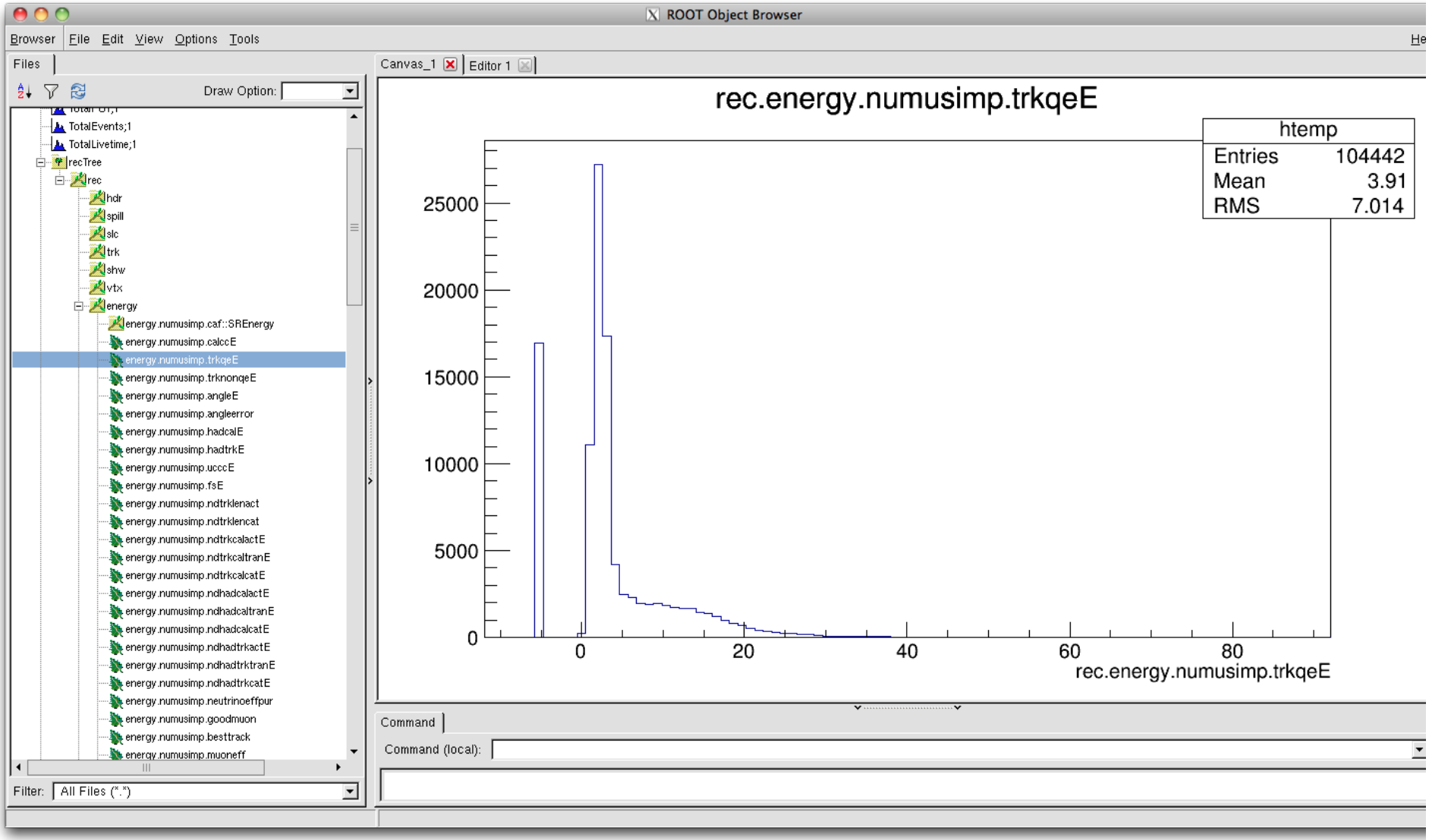
Shower Branch



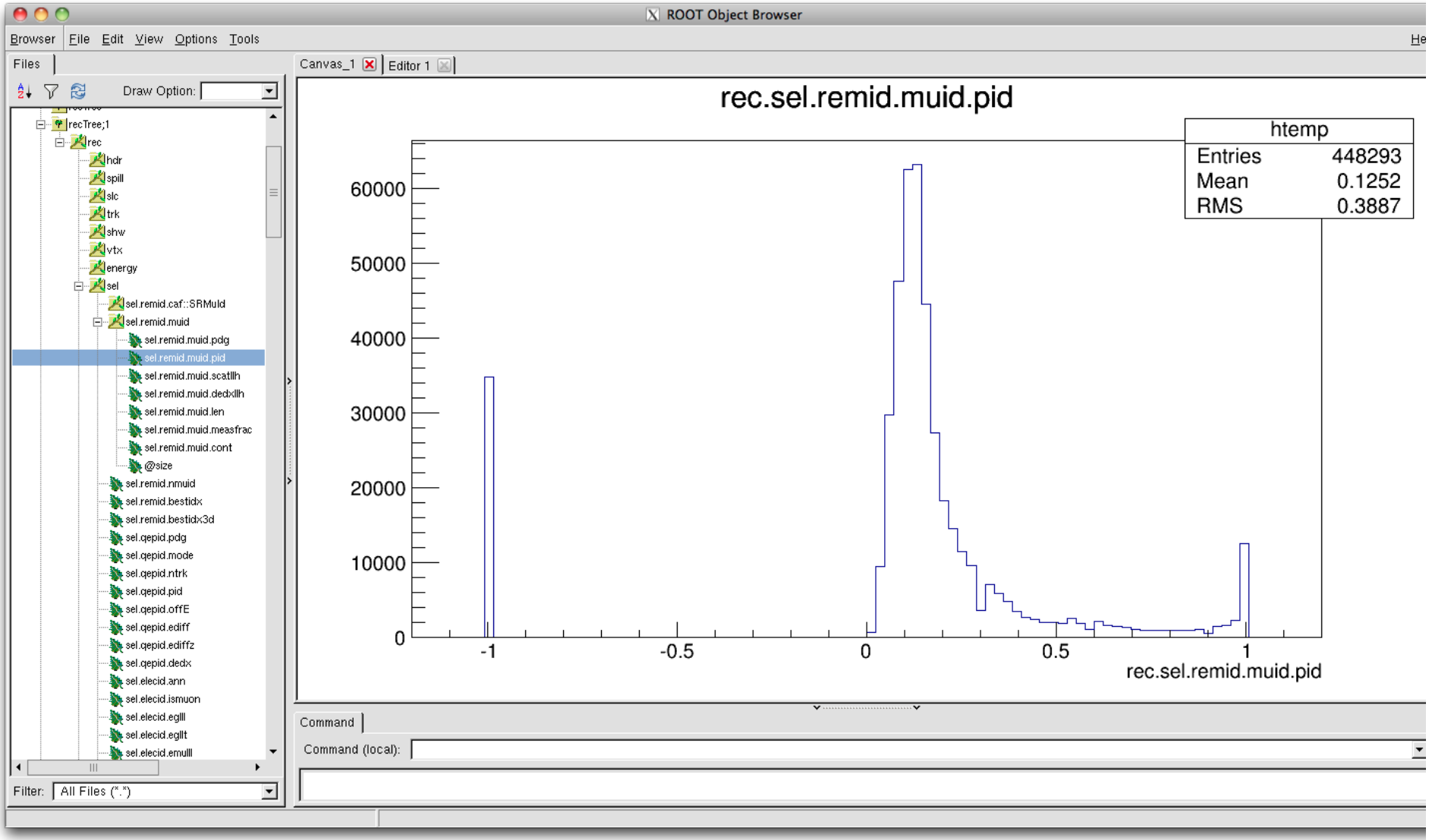
VERTEX BRANCH



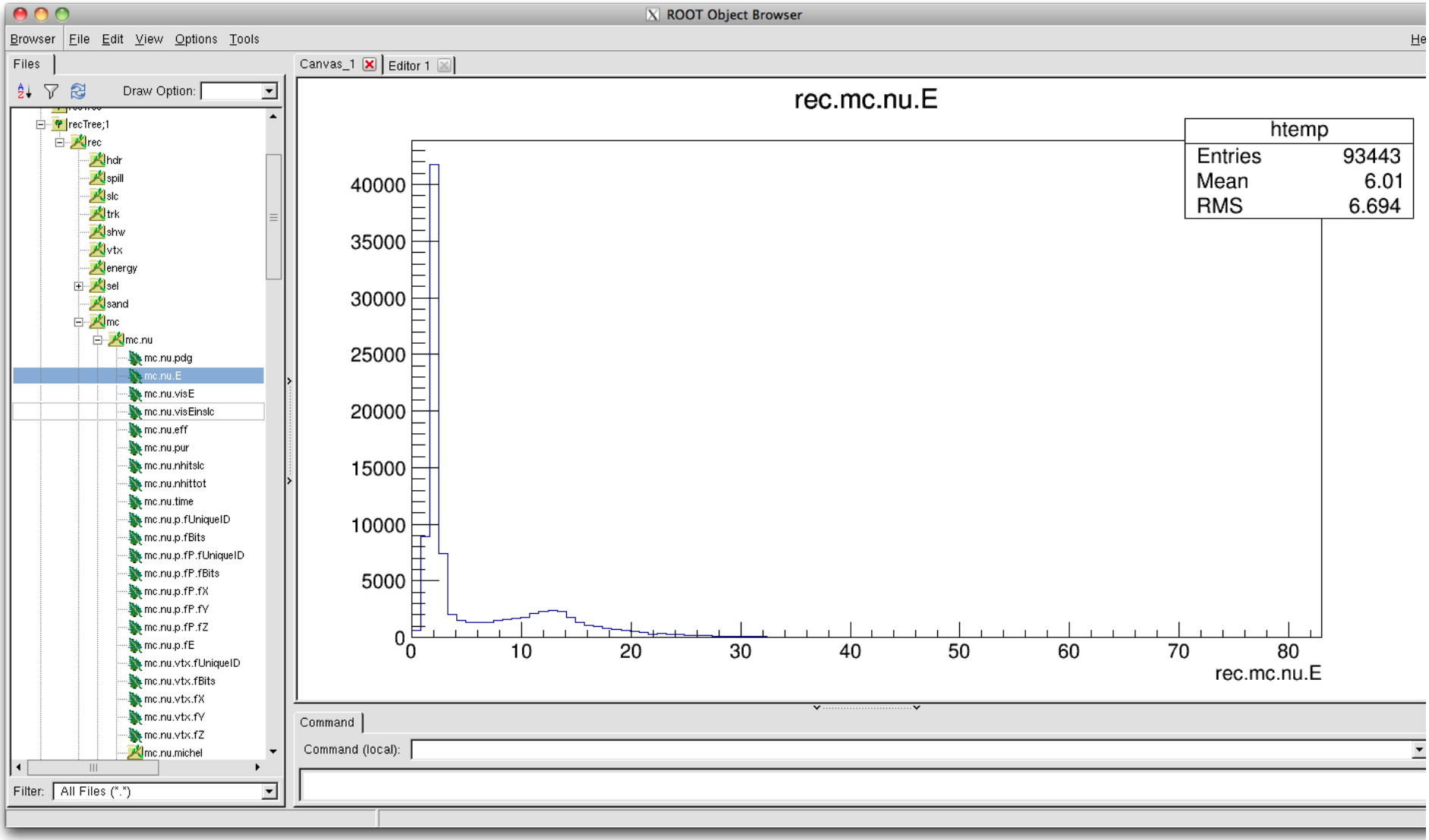
Energy Branch



PID Branch


ROOT Object Browser
Browser File Edit View Options Tools
Files
Draw Option:
recTree;1
rec
hdr
spill
slc
trk
shw
vtx
energy
sel
sel.remid.caf::SRMuId
sel.remid.muid
sel.remid.muid.pdg
sel.remid.muid.pid
sel.remid.muid.scatllh
sel.remid.muid.dedxllh
sel.remid.muid.len
sel.remid.muid.measfrac
sel.remid.muid.cont
@size
sel.remid.nmuid
sel.remid.bestidx
sel.remid.bestidx3d
sel.qepid.pdg
sel.qepid.mode
sel.qepid.ntrk
sel.qepid.pid
sel.qepid.offE
sel.qepid.ediff
sel.qepid.ediffz
sel.qepid.dedx
sel.elecid.ann
sel.elecid.ismuon
sel.elecid.egIll
sel.elecid.egIlt
sel.elecid.emulll
Filter: All Files (*.*)
Canvas_1
Editor 1
rec.sel.remid.muid.pid
htemp
Entries 448293
Mean 0.1252
RMS 0.3887
60000
50000
40000
30000
20000
10000
0
-1
-0.5
0
0.5
1
rec.sel.remid.muid.pid
Command
Command (local):

MC Branch


ROOT Object Browser
Browser File Edit View Options Tools
Files
Draw Option:
recTree;1
rec
hdr
spill
slc
trk
shw
vtx
energy
sel
sand
mc
mc.nu
mc.nu.pdg
mc.nu.E
mc.nu.visE
mc.nu.visEinslc
mc.nu.eff
mc.nu.pur
mc.nu.nhitslc
mc.nu.nhittot
mc.nu.time
mc.nu.p.fUniqueID
mc.nu.p.fBits
mc.nu.p.fP.fUniqueID
mc.nu.p.fP.fBits
mc.nu.p.fP.fX
mc.nu.p.fP.fY
mc.nu.p.fP.fZ
mc.nu.p.fE
mc.nu.vtx.fUniqueID
mc.nu.vtx.fBits
mc.nu.vtx.fX
mc.nu.vtx.fY
mc.nu.vtx.fZ
mc.nu.michel
Filter: All Files (*.*)
Canvas_1
Editor 1
rec.mc.nu.E
htemp
Entries 93443
Mean 6.01
RMS 6.694
40000
35000
30000
25000
20000
15000
10000
5000
0
0 10 20 30 40 50 60 70 80
rec.mc.nu.E
Command
Command (local):

Browsing Files Interactively

- You can start up a ROOT prompt with cafe
  - `cafe` is a simple wrapper for ROOT, it loads StandardRecord libraries
  - It is available if you set up the NOvA offline environment

    ```
    cafe /nova/app/users/rocco/CAFDemo/demoFile.caf.root
    ```

  - This loads the file as `_file0`

- If you want to click around, open a TBrowser

  ```
  new TBrowser
  ```

  - You should see the file listed in the left pane

- You could also make a tree object:

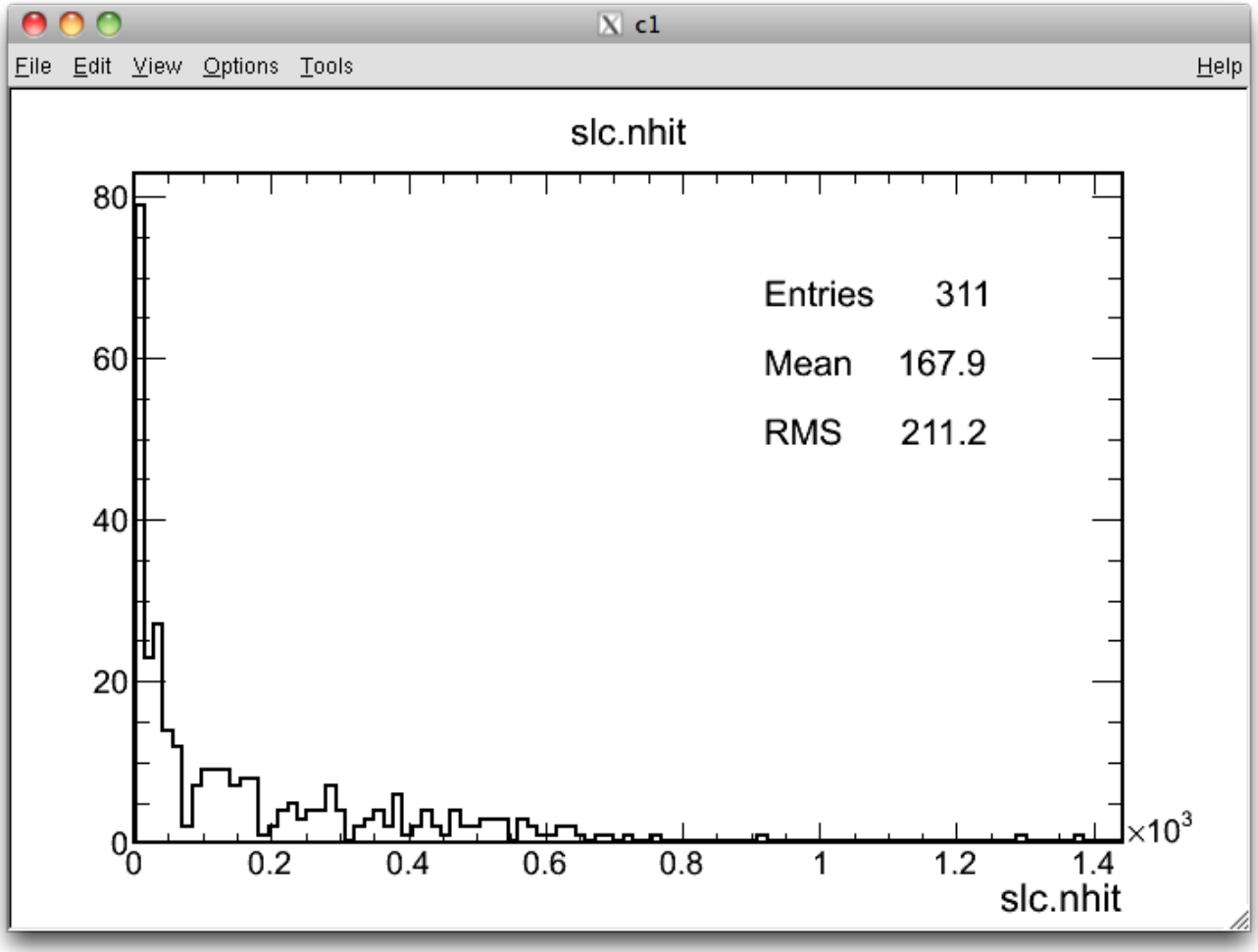
  ```
  tree = (TTree*)_file0->Get("recTree")
  ```

  - This will allow you to draw histograms with `TTree::Draw()`

PLOTS WITH TTREE::DRAW()

- It's really easy to make basic histograms with `TTree::Draw()`

```
tree->Draw("rec.slc.nhit")
```

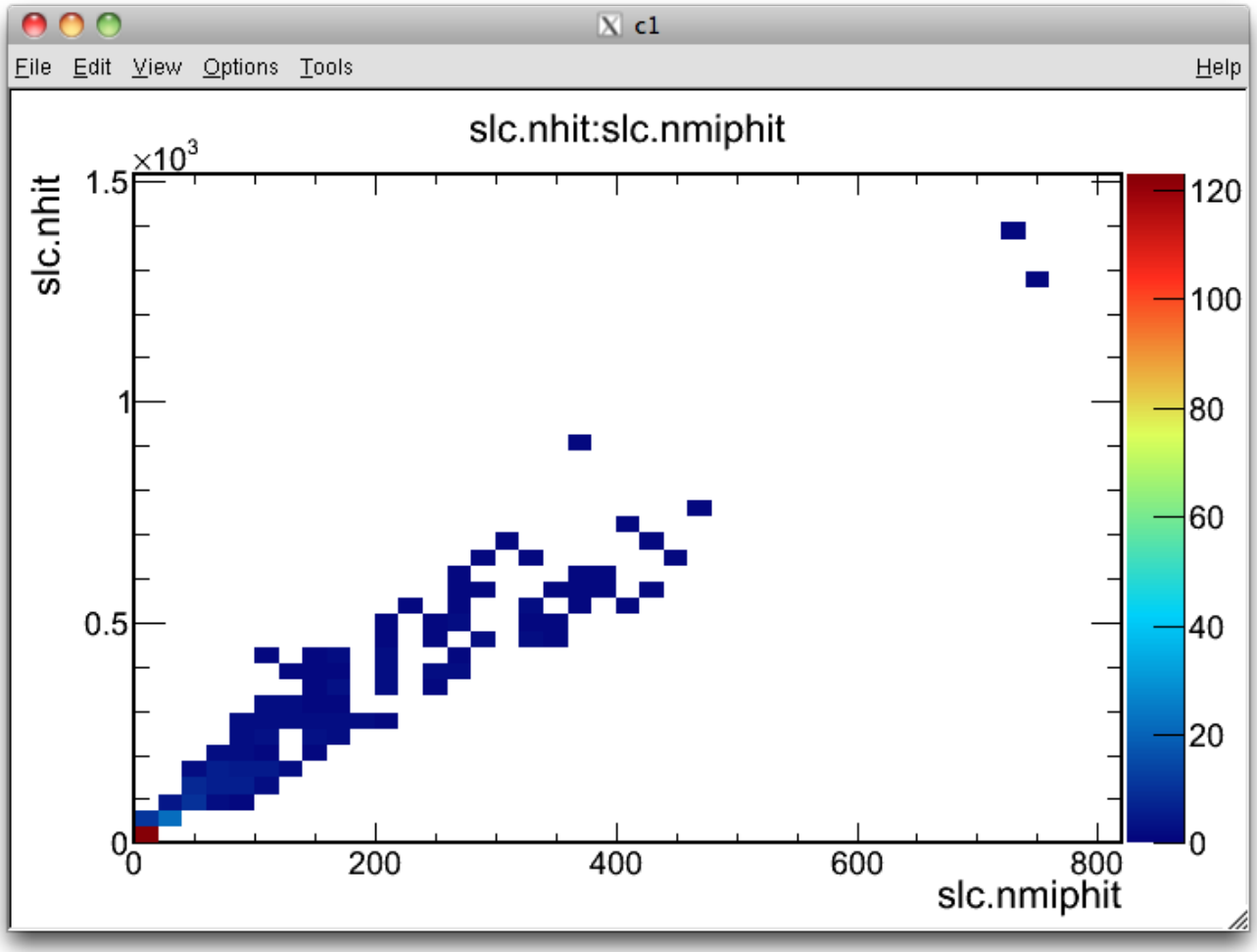


2D Histograms

- 2D histograms are created by specifying two variables separated by a colon (`:`)

```
tree->Draw("rec.slc.nhit:slc.nmiphit", "", "colz")
```

- The second parameter is a cut, we'll talk about those next
- The third parameter is the option, `colz` makes a color plot with a scale

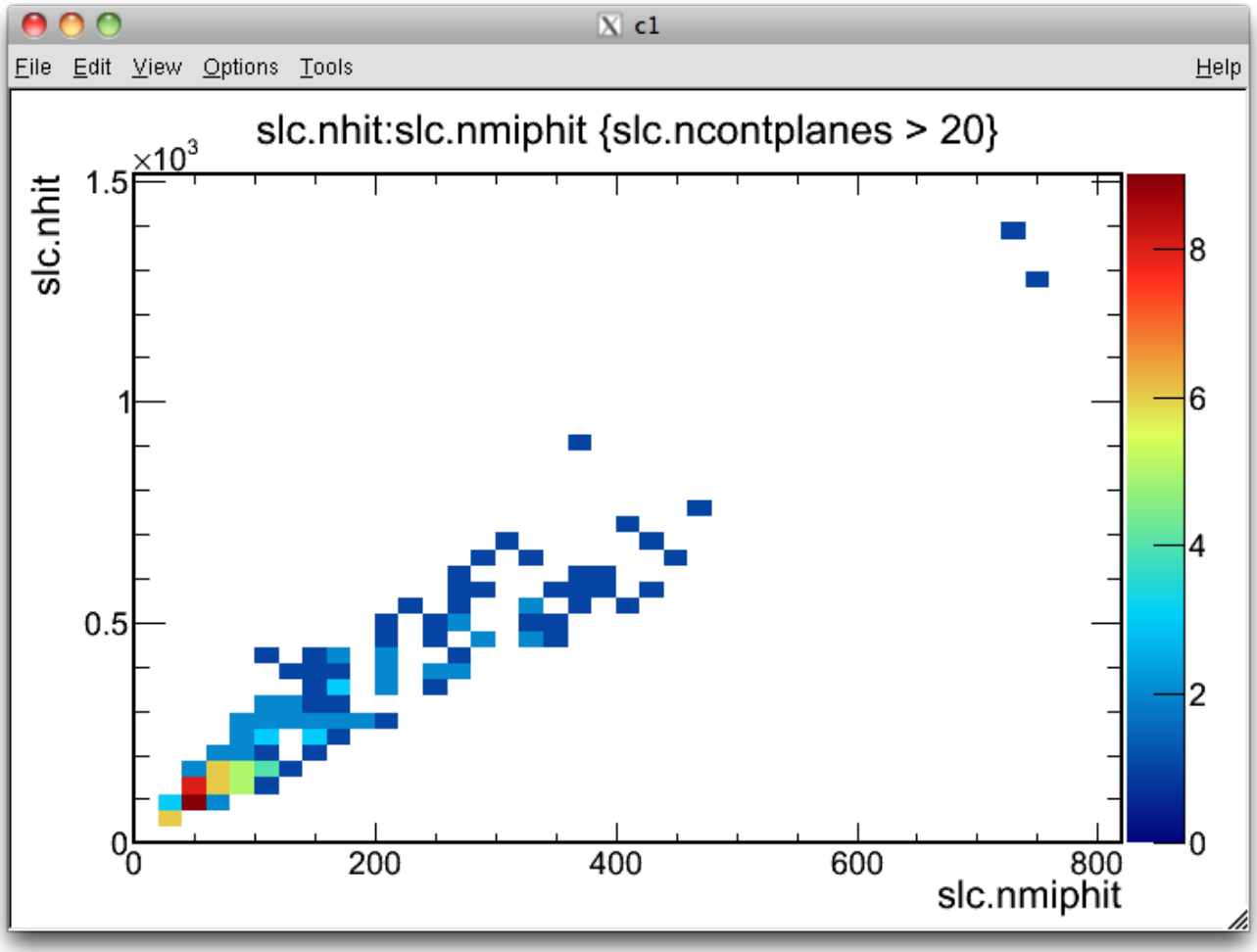


Cuts

- The second parameter is a cut, or really a fill weight

```
tree->Draw("rec.slc.nhit:slc.nmiphit", "slc.ncontplanes > 20", "colz")
```

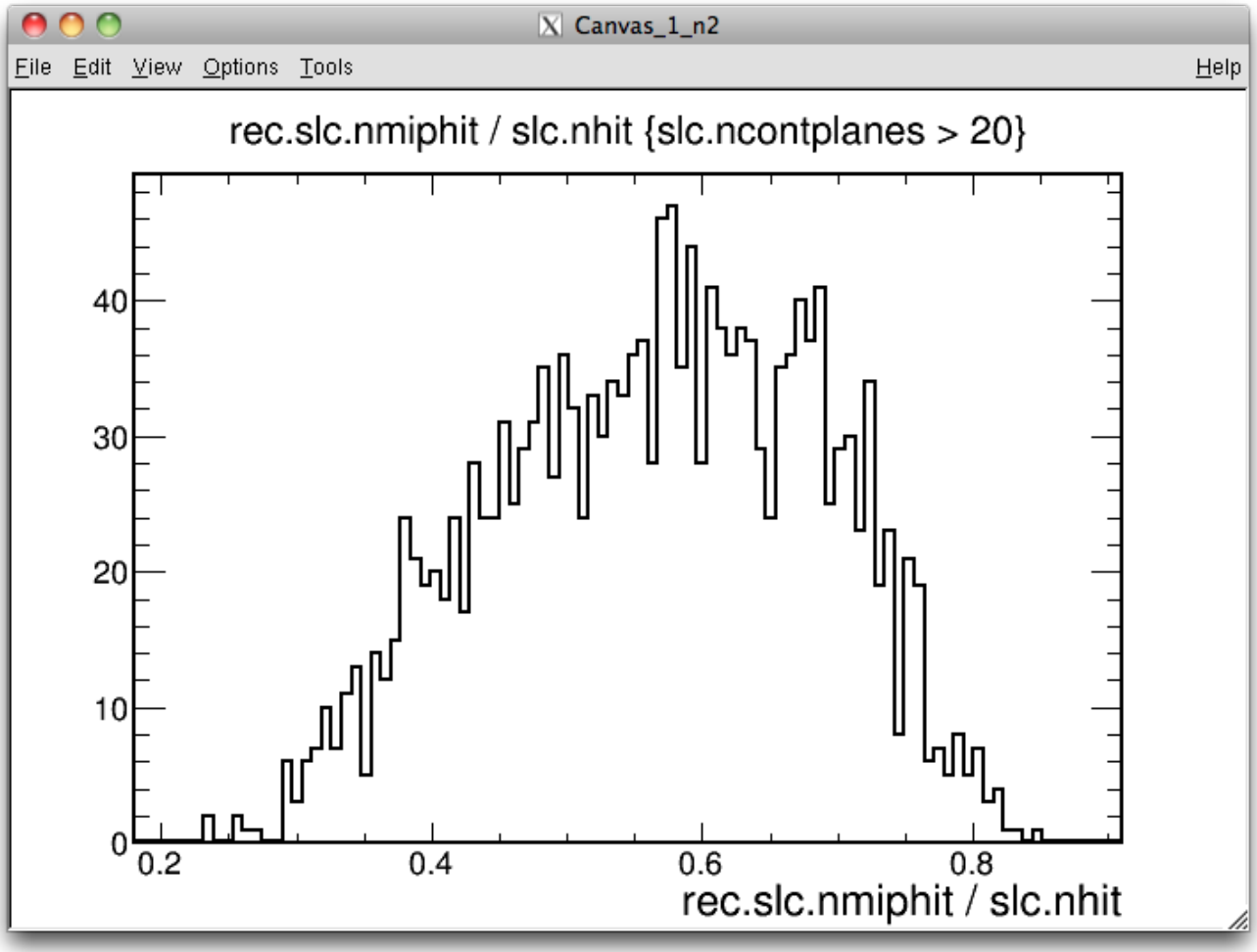
- If the weight is zero, it effectively cuts
- Could use this for a weighted histogram (oscillation probability, perhaps)



ARITHMETIC

- We can do arithmetic in `Tree::Draw()` also

```
tree->Draw("rec.slc.nmiphit / slc.nhit", "slc.ncontplanes > 20")
```



ROOT Macros

- ROOT macros are a good way to do more complicated things
- Here is an example:
  `http://cdcvs.fnal.gov/redmine/projects/novaart/repository/entry/trunk/Demo/tutCAFMacro.C`

Conclusions

- CAF is a good resource for lightweight analysis tasks
- More documentation on techniques here:
  `http://cdcvs.fnal.gov/redmine/projects/novaart/wiki/CAF_ROOT_Macros_and_PyROOT`
- And on tree structure here:
  `http://cdcvs.fnal.gov/redmine/projects/novaart/wiki/CAF_Tree_Structure_and_Variable_Listing`
- Good luck, have fun!