Getting evt2root started - AMS style!

K. Ostdiek

March 25, 2014

1 Getting It and the Libraries

Have Karen copy the evt2root directory from her AMS user space to where you'd like to work with it (your home directory or /ams/user directory). She'll need you to enter your password for this. This directory should have several files and folders in it. Those include: analysis.c, cmake, CMakeLists.txt, doc, ExpEvent, include, NOTES, README, and src. The README file is from Karl and includes how to build and install all of this. The NOTES file is Karen's notes on evt2root and ROOT in general (may contain gibberish).
Navigate to the directory ExpEvent (cd ExpEvent). Here you will find another README file from Karl. You will also find several other files including eventScaler.h, eventScaler.cxx, eventData.h, and eventData.cxx.

- eventScaler.h -> header for this Class, timing and scaler information.
- eventScaler.cxx -> defines the functions from the header file. Both this and the header file, I have not changed from the original.
- eventData.h -> header file for this Class, variables and functions needed later are called here.
- eventData.cxx -> defines the slot # and the channel # for the event info such as the dE's, Anodes, Scalers, Time, etc). This also defines Pos_X from LL, LR, RL, RR (Calibrate function).

This should all be set for a standard AMS run using the PGAC and IC. But if something looks off on your spectrum or if you need to add something, this would be the place to go.

1.1 Build it And They will come

Once the libraries have been set (eventData/Scaler.h/.cxx) then we'll need to install this.
Navigate back up to the evt2root folder. Then:

Type This	Comment
mkdir build	Makes folder called "build"
cd build	Navigates to folder
module load cmake	Load cmake module
cmake ../	Some stuff happens on screen, looks for ROOT, configures, writes build files, etc.
make install	Does more stuff, builds target evt2root and evtDump, installs.

Now in the build folder you should have:

- CMakeCache.txt
- cmake_install.cmake
- evtDump (colored green)
- install_manifest.txt
- Makefile
- CMakeFiles (colored blue)
- evt2root (colored green)
- ExpEvent (colored blue)
- libExpEvent.so (colored green)
- src (colored blue)

This build directory is where all the magic will happen. If you have issues, check out the README files in the evt2root main folder or in the ExpEvent folder. You should now move the analysis.c file and the .evt and .root example into the build directory as well so that ROOT knows what you are talking about later.
You may notice in my NOTES file, that I have something on updating evt2root with something called "git." **Currently you can ignore that.** By following the steps there, you will update to the newest version of ev2root from the web, however recent changes implimented for other groups will make our version **explode!** The version we have now is just fine. I'll try to figure out the update at a later date.

2 "But I like you the way you are!" - Change Event to Root

Navigate into the build folder. If you type only the words "evt2root" or "evtDump" you will see how to use it. For evtDump, which is good place to start to make sure everything is working as it should, you will only need the .evt file. This will spit everything to the screen, event by event. This is helpful for troubleshooting. It should end with the scaler information and the words "Run Ended." For evt2root, you need the input file name (whatever.evt) and the output file (whatever.root). For example:
evt2root run378-4096.evt run378.root
You can navigate to the input file but the output file will be made in the build directory. For example:
evt2root 60Fe/2011/September/run378-4096.evt run378.root
Let's say you have 300 files that you want to convert from .evt to .root. Move them into the build folder (they can even be in a new separate folder under build). Then wherever the runs are, type bash in the command line. Then as bash scripting, type the following (hit enter between lines):

Type This	Comment
for file in *.evt	For every file in this folder that has .evt as the file extension
do	Do the following
./evt2root "$file" "$file.root"	Calls the file.evt and converts to file.root
done	Finishes and starts doing what you told it to do
exit	After it's done convert, type this to get out of bash

Wait... now my files are whatever.evt.root. Do this, again in bash:

Type this	Comment
for i in {first#..last#}	Works for (example) 200..300 run #, harder for such as 1..100
do	Do the following
mv run$i-4096.evt.root run$i.root	Renames run#-4096.evt.root to run#.root
done	Ends

This shouldn't take long. Type "exit" to get out of bash.

3 Getting to the ROOT of the problem

So lets open up root. Type "module load root" while in the build directory. Then type "root". Note, if you type "root -l" then it won't add all of those extra lines at the beginning about "Welcome to ROOT"). I've assumed PuTTY for all of this, fyi. So if root doesn't open, then you may not have the X-11 forwarding enabled which is a quick fix.

Exit out of PuTTY, and open up X-ming (that's what I use for X-11). Then open up the PuTTY configuration. Under "SSH", you'll find "X11." There you will find at the top a box to check for "Enable X11 forwarding." Make sure this checked. Then open PuTTY. Good to go.

3.1 Commands and such

The first thing that you will need to type in ROOT is ".L libExpEvent.so" This will load the libraries needed to define the spectra.
Other commands include (words in italics are replaced with the applicable names or variables and the quotations are necessary):

What	How
Open File	TFile **file-variable*=TFile::Open("*filename.root*")
Make new Canvas	TCanvas **canvas-variable*=
	new TCanvas("*canvas-variable*", "*title*", width, height)
	Note: "width" and "height" are in number of pixels.
Divide Canvas "c1" into pads	c1->Divide(*number of columns*, *number of rows*)
Navigate to different pads on c1	c1->cd(1) will navigate to the first pad.
Navigate to another defined file	*filename*.cd()
Create 2D histogram	TH2F **hist-variable*= new
	TH2F ("*hist-varbiable*", "*title*","xbins,xmin,xmax,ybins,ymin,ymax)
	Note: For most AMS purposes x/y bins = 4096, x/y min = 0,
	Note: x/ymax = 4096. If you forget this, look at the .evt file. ;)
Draw a histogram (example)	evtTree->Draw("dE3:Pos_X", " ", "COLZ")
	Note: evtTree is the root file to call the variables from
	Note: dE3:Pos_X plots de3 (y) vs Pos_X (x), ":" = "vs'.
	Note: 3rd piece will hold the name of any cuts applied.
	Note: COLZ will set the colors for the histogram.

Table 1: Common ROOT command line prompts.

We can also write the Draw as evtTree->Draw("dE3:Pos_X»*hist-variable*(4096, 0, 4096, 4096, 0, 4096)", " ", "COLZ"). This will make a new histogram with those specifications and that variable name.
For cuts, after a cut (or two) has been made (will discuss later), you should put the name of the cut in the 3rd position, in quotations. If you have more than one cut, you will need || or &&. The || is OR and && is AND. For example, evtTree->Draw("dE3:Pos_X", "FeFull||NiFull", "COLZ"). This will draw the events that appear in the FeFull or NiFull cuts.

3.2 Screen Shots of the Above

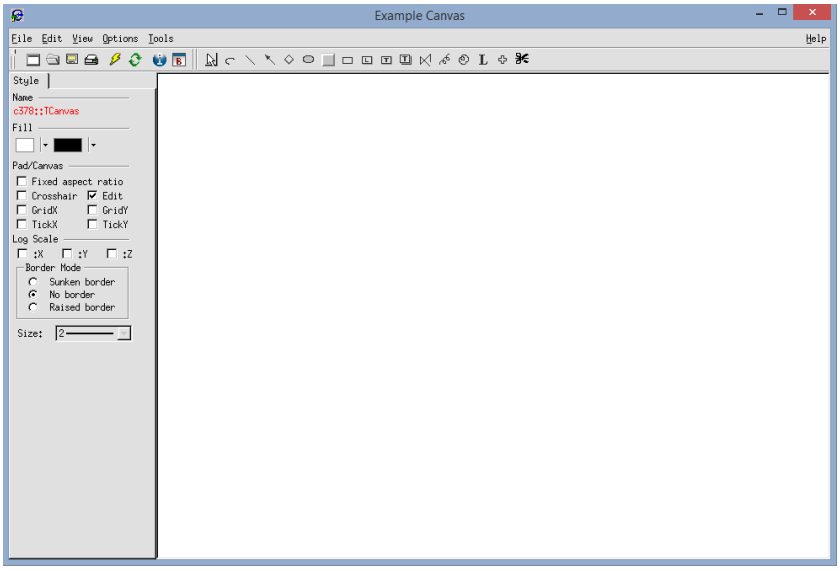

Figure 1: A new Canvas. Under "View" you will find "Editor" (shows on the left) and "Toolbar" (shows on the top) amongst other things. This will be useful later.

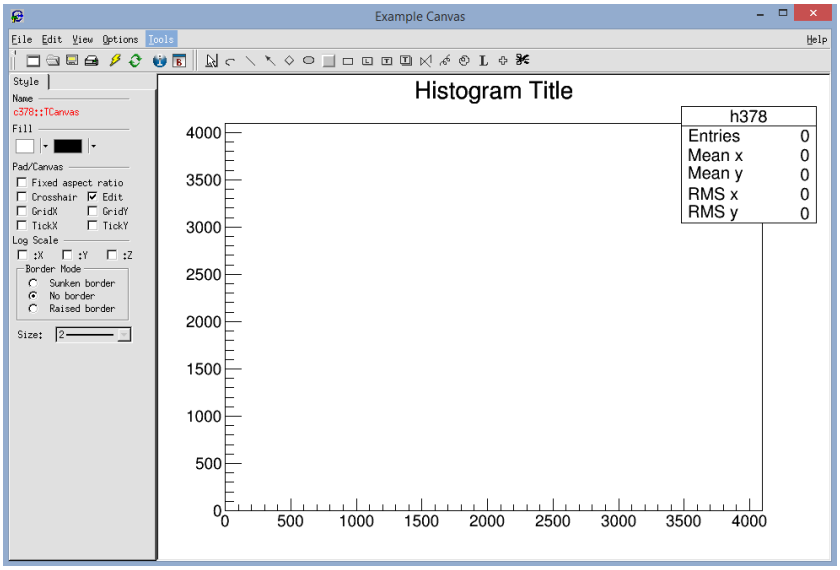

Figure 2: A Blank Histogram drawn on a Canvas. Note the box on the right side with some statisical information. This can be customized.

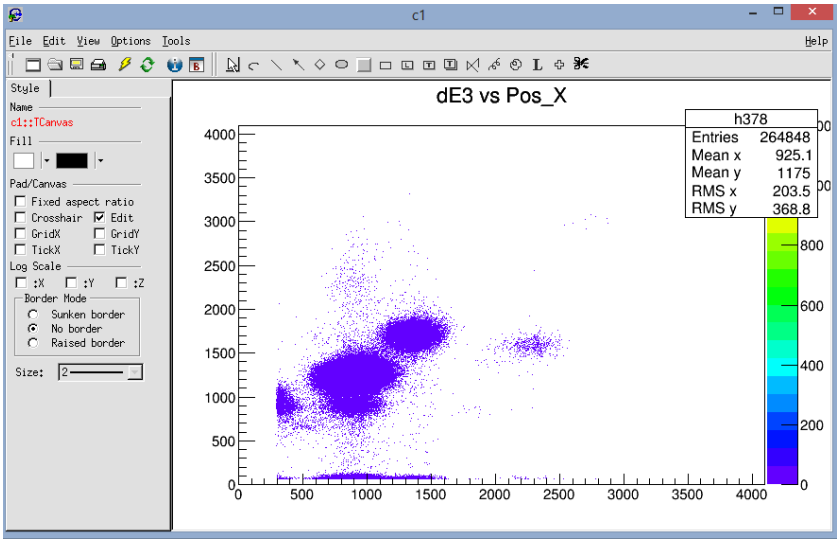

Figure 3: dE3 vs Pos_X histogram. Not zoomed in.

Clicking outside of the graph will give you on the left side the Canvas Option of "Style." If you click on the histogram itself, you will get the Histogram Options of "Style" and "Binning." Note: under "Name" on the left, you can see what you are clicked on.

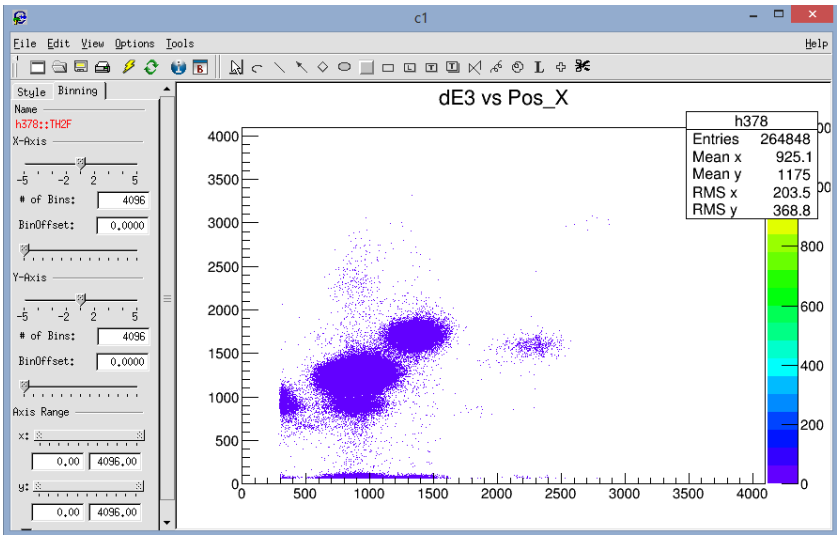

Figure 4: dE3 vs Pos_X histogram with the Binning Option shown on left.

Under Binning, you will find the tools for changing the number of bins and also options that allow you to zoom in with x and y minimums and maximums.

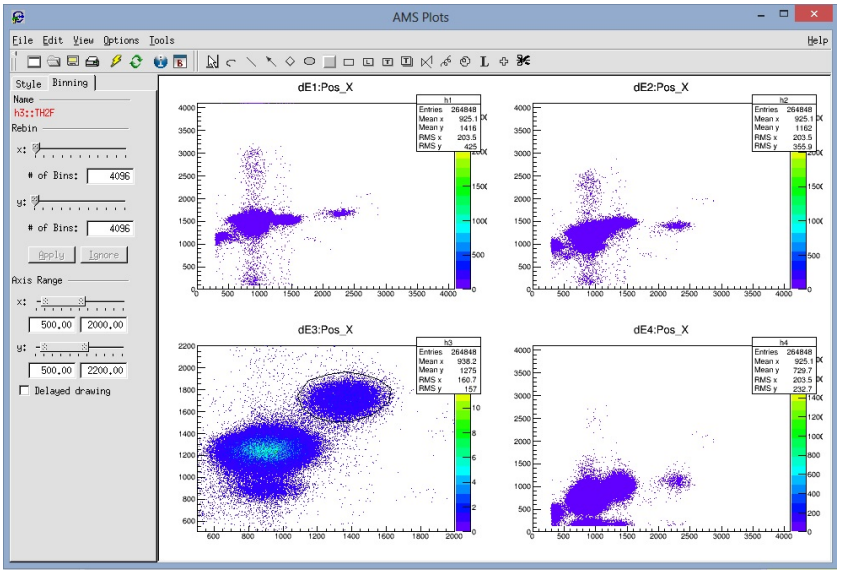

Figure 5: A Canvas, divided into 4 pads. Each pad can be controlled separately by clicking on the one of interest. Again you can see your selection under "Name" on the left panel.

3.3 Making the Cut

Looking at our histogram (with the Toolbar activated under the View menu), you'll see a pair of scissors, the last icon on the toolbar. This is what we'll use to make our Graphical Cut. Click on the scissors when you're ready to make your cut. Then you'll need to click around the region that you want your cut to be. You'll see from Figure 5 on the third pad (dE3 vs Pos_X) that I have drawn an oval shaped Graphical Cut around the 60Fe peak. You are not limited to circles, of course. You will need to end exactly where you started though. Once you have made a cut, you need to rename it. Scroll your mouse over the black line that defines the region. When your pointer turns into a hand, right click. A menu will pop up. Click on "SetName" (See Figure 6). Here you can change the name to whatever you'd like. This object should be a "TCutG." If the SetName pop up has either TCanvas or TH2F then you have clicked on the histogram or the canvas and the name of that is what you will change, not the cut.

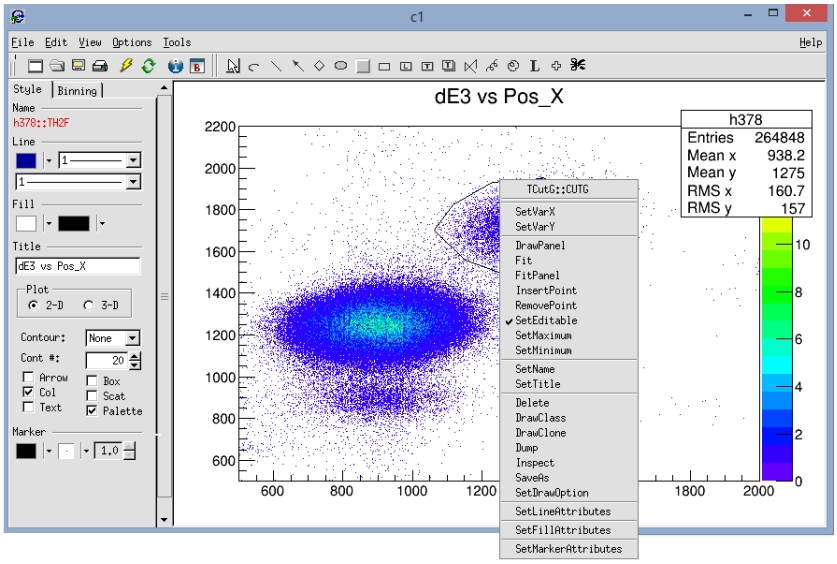

Figure 6: Scoll mouse over Graphical Cut. When pointer turns into hand, right click. Scroll down to SetName. Ensure that you are on the cut and not the histogram.

Once you have made a Graphical Cut, you can use it when you draw a histogram as well. In the third position of the Draw function, you can add the name of your cut. For example:
evtTree->Draw("dE3:Pos_X»hist1", "*Cut-name*", "COLZ")
You can also do "*Cut1*||*Cut2*" for OR or "*Cut1*&&*Cut2*" for AND. This will this look like Figure 7.

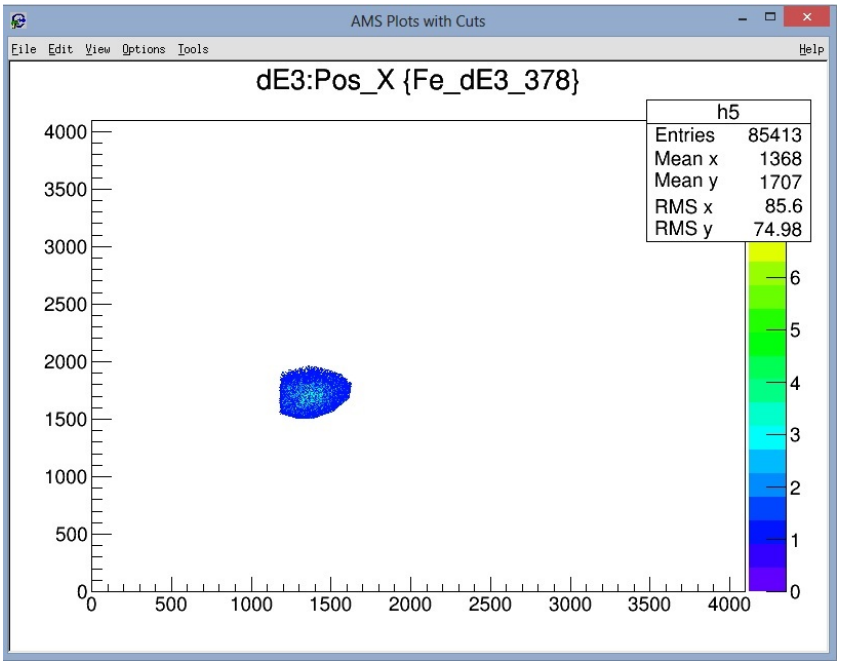

Figure 7: A histogram will only the contents of a graphical cut drawn. The name of the cut is in the title between the curly brackets.

You can also use the cut from the root command line. By typing the following: *my_cut_name*->IntegralHist(*histogram_name*), the number of counts in that cut on the histogram will print to the screen.
If you like to save your cuts to use during different sessions of ROOT, do the following after creating the cut via the Graphical Cut and renaming it:

What to Type in Command Line	Why
TFile *variable-name*("*cut-name*.root", "recreate")	Opens up a file with the same Cut name.
cut-name.Write()	Writes the Cut to that ROOT file.
TCutG * *cut-name*	This makes a Cut called "Cut-name." You shouldn't need this when in ROOT session that you created the Cut.
TFile * *variable-name* = new TFile("*cut-name*.root")	Opens ROOT file where cut is stored.
cut-name = (TCutG*)*variable-name*->Get("*cut-name*")	Open up the Graphical Cut File
cut-name->Draw()	Draws on the currently selected spectra.
cut-name->IntegralHist(*histogram-name*)	Adds up the counts in that cut.

You can also draw the same cut to a different spectra simply by navigating to the pad that you'd like to Draw on (c1->cd(#)). Also you don't need to draw the cut in order to add up the number of counts in the region of the cut.

4 Right Tools for the Job

I have programmed a few useful tools for my own analysis in the file analysis.c in the build folder. To edit or just view it, use your favorite text editor (I use gedit). There are several functions in there. They include:

Function Name	Description
draw_can()	Draws a Canvas named c1
draw_splitcan()	Splits the Canvas c1 into 2 columns and 2 rows.
draw2	Draws a 2d histogram
draw_de1x()	Make and draws a 2d histogram of dE1 vs Pos_X
draw_de2x()	Make and draws a 2d histogram of dE2 vs Pos_X
draw_de3x()	Make and draws a 2d histogram of dE3 vs Pos_X
draw_de4x()	Make and draws a 2d histogram of dE4 vs Pos_X
draw_de14()	Draws the histograms dE1 through dE4 on the split canvas.
draw_FeFull(TH2F **histname*)	Draws the Cut "FeFull" on a Histogram Note: Several similar cuts follow that I used. Note: Previously saved cuts as .root files. Use as examples.
draw_all(TH2F **histname*)	Draws all of the previously mentioned cuts on a Histogram

Table 2: Functions included in analysis.c file.