Getting Started with ROOT

GETTING STARTED WITH ROOT **1**

FIRST ROOT SESSION STEP-BY-STEP **2**

STEP 1: START A ROOT SESSION 2
STEP 2: USE THE ROOT OBJECT BROWSER 3
STEP 3: THE ROOT CANVAS 4
STEP 4: THE CONTEXT MENUS 5
STEP 5: SETTING THE AXIS TITLE: 7
STEP 6: THE CLASS AND MEMBERS REFERENCE 9
STEP 7: THE CANVAS EDITOR 9
STEP 8: A CANVAS DISPLAYING MULTIPLE HISTOGRAM 11
STEP 9: THE TREEVIEWER 12
STEP 10: ENDING THE ROOT SESSION 15

ROOT COMMANDS **15**

THE TAB-TAB FEATURE 16
THE COMMAND HISTORYFILE 16
OPEN A ROOT FILE AND DISPLAY THE BROWSER 17
DRAWING HISTOGRAMS 17
ADDING TEXT 18
AXIS TITLE 18
ADDING CUTS 19
CHANGING STATISTICS 19
FITS AND FUNCTIONS 19
MULTI-HISTOGRAM DISPLAY 20
GLOBAL VARIABLES 21
TWO DIMENSIONAL DISPLAY 21
CREATING A ROOT FILE 22

HELPFUL HINTS **24**

CONVERTING A PAW NTUPLE FILE INTO A ROOT FILE 24
READING AN ASCII FILE AND MAKING A ROOT NTUPLE 24
SAVING THE ROOT CANVAS AS A MACRO 24

FURTHER DOCUMENTATION ABOUT ROOT **25**

SETTING UP ROOT AT FEMILAB **26**

APPENDIX A: ROOT COMMAND REFERENCE **27**

APPENDIX B: THE CANVAS MENUS **29**

Welcome to Getting Started with ROOT. Physicists are involved in the business of getting data into files, analyzing it, and then producing histogram plots and fits. This tutorial is aimed to the novice ROOT user as the tool for doing this. Hopefully it will get you into using ROOT quickly even if you have little to no knowledge of C++, the language of ROOT commands.

To begin, we will provide you with a sample root file to use with the ROOT graphical user interface that will take you step-by-step through your first ROOT session. You will analyze the data by using the ROOT Object Browser, where just by clicking you can display histograms, save them, and print them. You can interact with the canvas to modify objects that are in the display and also create new graphical objects. Using only the mouse you are able to set up different views of histograms and fit them to various functions.

Although there are many things to do in ROOT without even typing in a command, once you've had some fun and experience with ROOT's GUI, we hope you will be interested in going to the next step - using root commands as well as the GUI during ROOT interactive sessions. Part 2 - ROOT COMMANDS deals with this. These commands are equivalent to what you will have done using the GUI in the step-by-step session with additional commands to provide more flexibility for drawing histograms with cuts and using functions. You'll want to use ROOT with ntuple files that you are familiar with, so if you were working with ntuples using PAW we'll show you how to convert those files into ROOT and if you have an ASCII file, we'll show you how to read the file and make an ntuple from it.

After you are acquainted with root commands, the next step is writing ROOT macros. We will point you towards documentation for the more advanced user and examples of macros that you can use as templates.

Now let's get started.

First ROOT Session Step-by-Step

What are we going to do? Start an interactive ROOT session. If you need to set up ROOT, click here. Although we've stated that this part is ROOT command free, we will give you one command that you just have to copy. The command brings up the ROOT Object Browser on your screen. Then you will use the Browser and your mouse to see some of the fun things that you can do. These include graphical manipulations such as color, font, text, etc and fitting and drawing procedures for histograms (lego plots, surface views, etc).

Step 1: Start a ROOT session

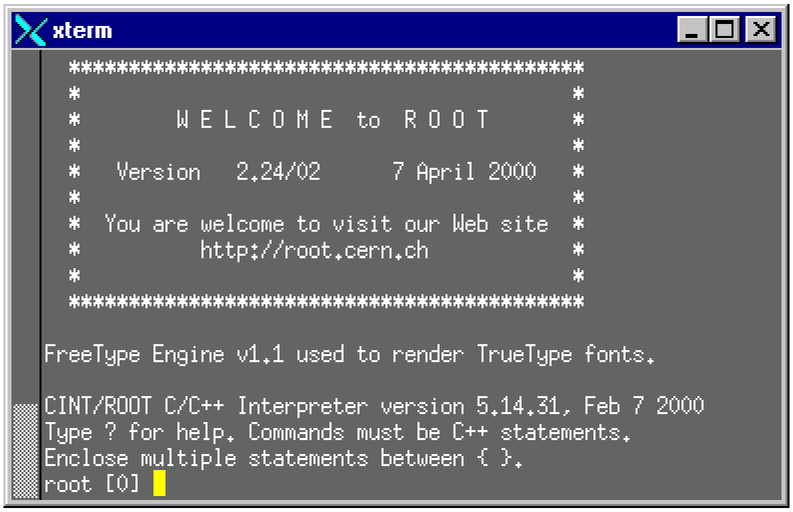
The first thing we need is the sample file. Click here http://www-pat.fnal.gov/root/101/htt_175a.root and copy the file htt_175a.root to your $HOME (or ~) directory.

Now we can get started. From your $HOME directory start root by typing at system prompt

(>) root

A root screen will appear. This is the screen for entering commands.

The ’root prompt’ is where you enter root commands. It starts with the word "root", followed by a number in brackets. The number in brackets is the number of commands that you have entered so far. At this point it is 0, hence it shows “root [0]”.



Before we start, it's a good idea to mention how to end a root session.

You end a ROOT session by typing .q at the root prompt line.

Step 2: Use the ROOT Object Browser

At root [0] prompt line type:

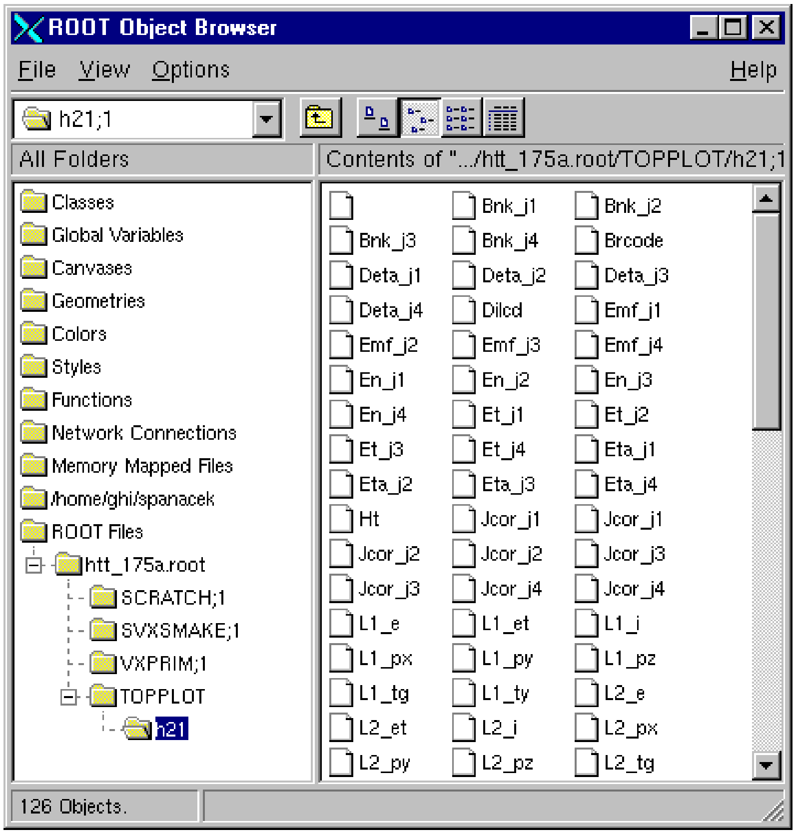
TBrowser b;

This creates a TBrowsr object (b), and it will appear in a separate window a separate window on your screen. The browser lists everything that is part of the current ROOT session.

Open the root file, which has the sample root tree that we will be using in this example. At this point you may wonder what a root tree is. We will show you in a later section how to convert an hbook file into a root file. When such a file is converted the PAW ntuple is converted into a root tree. Root trees are optimized to hold large amount of data. They are explained in detail in the ROOT 102 class, and in the document on “Persistence of ROOT Objects”. For now, we can assume that a tree to root is what an ntuple was to Paw.

To open the root file from the Browser, select the File menu and the Open menu item. Click on the file htt_175a.root and click on the Open button

Now we navigate to the ntuple using the Browser.



Double click on the ROOT Files folder. Beneath it, the name of the root file which we just opened will appear

Double click on the folder next to the file name: htt_175a.root . Another list of folders will appear beneath it.

Double click on the TOPPLOT folder. A folder called h21 appears beneath it. "h21" is actually the ROOT Tree which contains our sample ntuple variables. In a tree these are called leafs.

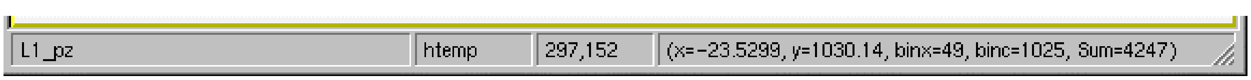
Double click on h21. A list of leafs will appear on the right hand side of the screen. Now your browser should look as shown.

Double click on L1_pz that appears on the right hand side of the screen. ROOT will automatically generate and display a 1-D histogram of that variable.

Step 3: The ROOT Canvas

What you are looking at above is a ROOT canvas. The canvas is the area of the window where you can interact with the ROOT objects. ROOT automatically generated the canvas, named it c1, and displayed the variable L1_pz. There are many things that you can do to modify the canvas. We'll just try a few during this first session:

Display the Event Status: Before we start selecting objects on the canvas, let's look at an example of the event status bar. It helps you see what you are selecting.



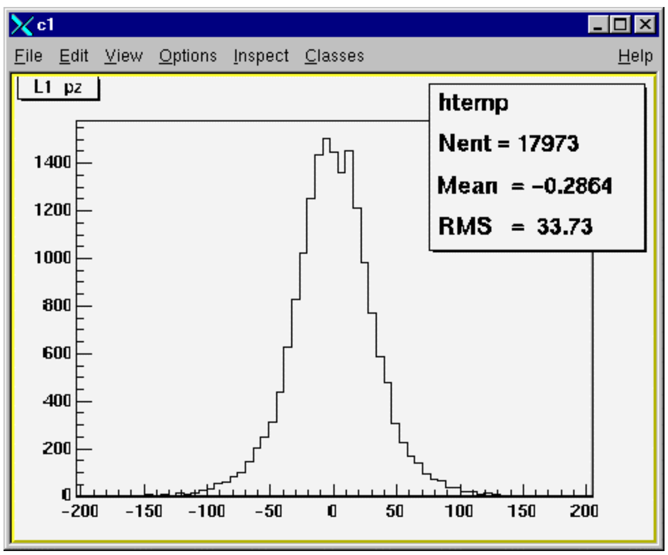
To display this, go to the canvas window and select the Options menu and the Event Status item. The Event Status bar appears at the bottom of the window. You can use it to get the status of the object that you pick e.g. for the histogram it shows the title (L1_pz), the name (htemp) the location on the canvas (297,152), the x and y value of the current cursor position (x= 23.5299, y= 1030.14), the bin number (binx=49), the bin content (binc=1025), and the sum of the bins up to this bin (Sum=4227). Move the cursor around on the histogram and see how these values change.

If you are working interactively and later want to write a macro you can use this information in your macro.

Dragging and Resizing the Canvas or Objects on the Canvas: Like any window you can move, stretch or shrink the whole ROOT canvas or an object in the canvas by pointing, clicking and dragging with the left-hand mouse button. Point to the object you would like to move. When pointed to a sensitive area of the object the cursor will change shape. When the cursor appears as an arrow with a line it indicates the directions in which you can resize the object. A bold cross indicates that you can move the object. You can also modify the end points of the axis. For example, let's modify the end points of the x-axis to display values only from -200 to 200.

Point your mouse at the minus 200-point of the x-axis. The cursor shape should change to a hand.

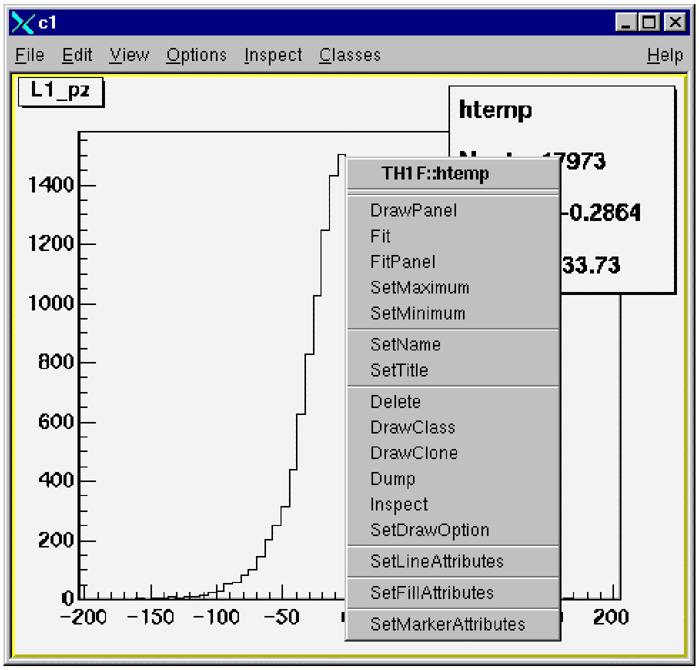
Click the left mouse button and hold. A line appears as soon as you drag it little. Drag the mouse across the x-axis until the 200-point on the x-axis. Release the left mouse button. You should immediately see the new range of the x-axis displayed on your canvas. If you don't see it, select the **Options** menu and click on the **Refresh** option. Your canvas should now look like this:



Right now we have only one canvas, c1, and that is the active canvas. In ROOT you can create and display many canvases on the screen at the same time. The canvas that is active is the one that is color highlighted around its edges. Any action that you tell ROOT to perform interactively is done on the active canvas. Clicking on the canvas with the middle mouse button (or right and left mouse button together) makes the canvas active. For now, we will continue to use only one canvas, but later we will create another canvas and use it to display more than one histogram on the same canvas.

Step 4: The Context Menus

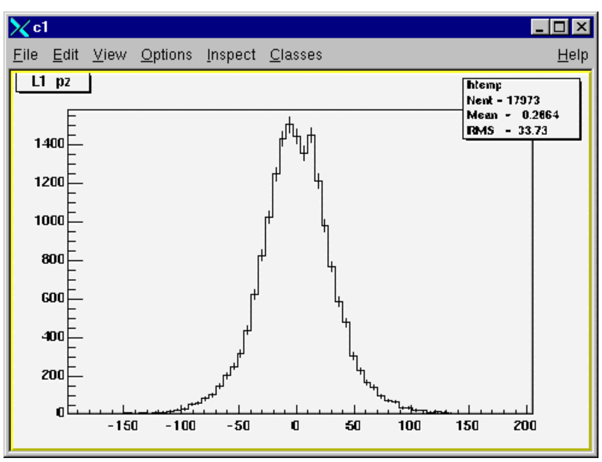
When the right mouse button is clicked on an object in a canvas, the object's context menu will pop up on the screen. This menu is a list of methods that are specific to the object you clicked on. Let's try it with the histogram.

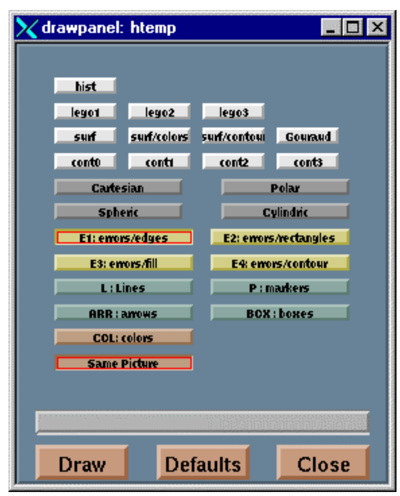


Click your right mouse button over any horizontal part on the outline of the histogram. You will notice the cursor changing to an arrow when the histogram is selected. A context menu appears with the title TH1F::htemp.

There are two special menu items for histograms that appear in this list. One is **DrawPanel**. It brings up another window that gives you options for the way the histogram is drawn and displayed. It includes the standard histogram view as well as various lego views, and additionally, contour plots and surface views. The other option panel for histograms is **FitPanel.** You can choose different ways for performing the fit.

Move the mouse to the **DrawPanel** option and let go. Left mouse click on the **Same Picture** box and the **E1:errrors/edges** box. Then click on the **Draw** button.

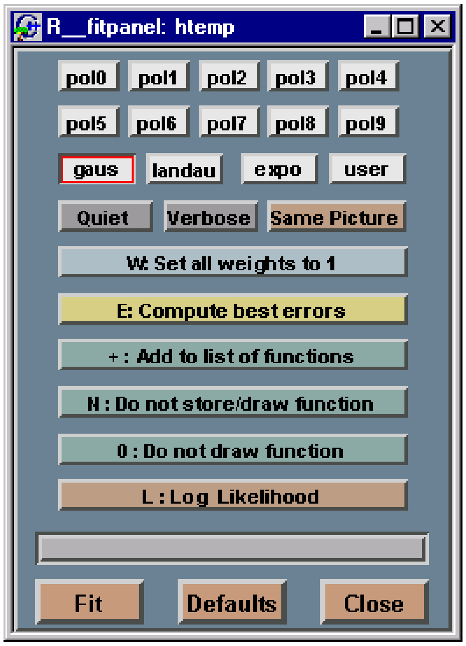




We won't be using the error bar display right now,

so we'll reset the histogram view back to the standard histogram view. To do this, click on the **Defaults** button and then the **Draw** button.

Now we'll use the Fit Panel.

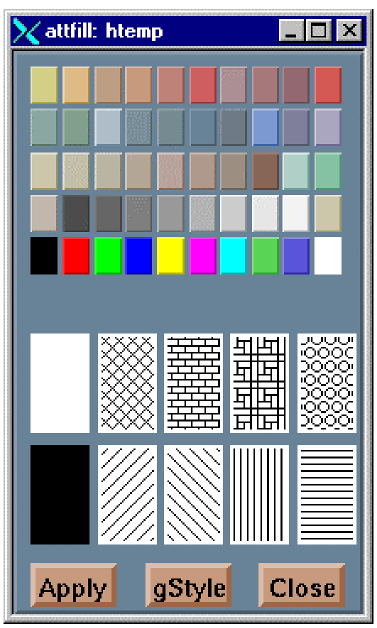


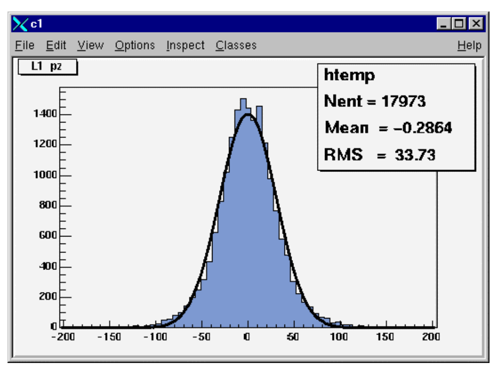
Click your left mouse button on the **FitPanel** menu item. The fit panel appears

Click on **Verbose** to have all the fit information appear on your ROOT command line window when the fit is executed. Click on gaus and then click the Fit button. The fit is displayed on the canvas. (If the fit doesn't immediately display, it should do so by clicking on the "Refresh" option that is under the canvas menu item "Options".)

Now we'll set the fill color.

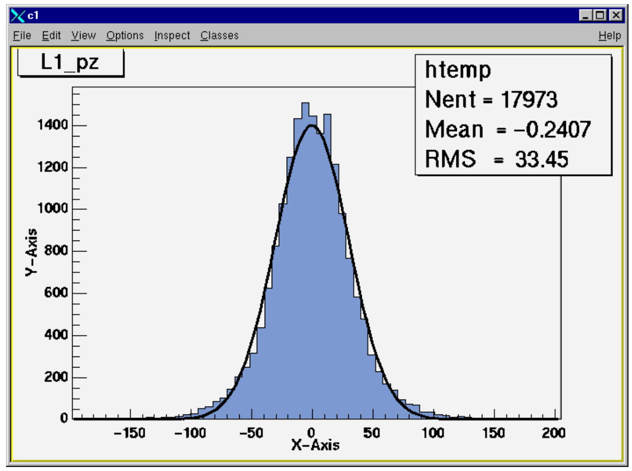
Go back and click on any horizontal part on the outline of the histogram to bring up the histogram context menu again. Click on **SetFillAttributes**. This time, the fill panel appears. Choose a color and click on the **Apply** button. Your histogram will be filled with that color and look as shown.





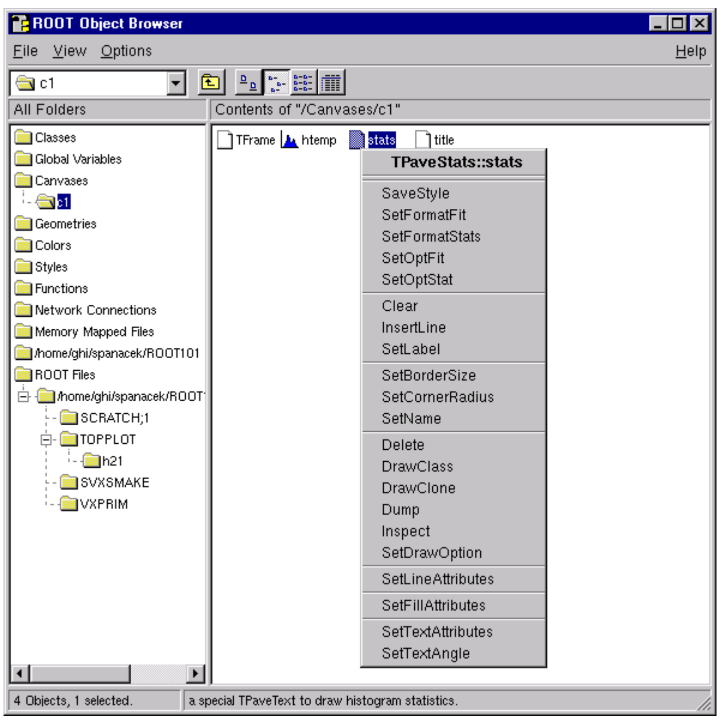
Step 5: Setting the Axis Title:

You can also use the context menu of the axis to give it a title. To do this right click on the x-axis. This brings up the context menu. Select **SetTitle** and type "**X-Axi**s" in the text box and click **OK**. You will see the title appears, but it is all the way on the right side of the axis. To center it, bring up the context menu of the x-axis again by right clicking on the x-axis. Select **CenterTitle**. The text box will display **kTrue**, and that is what you want. Just select **OK**, and the title is now centered. You can add the title to the y-axis by repeating this procedure on the y-axis.

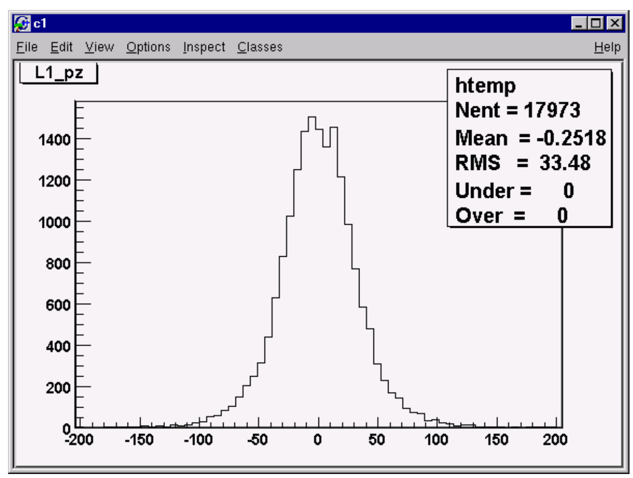


Your histogram should look like the one here.

Another way to get to the context menus of objects on the canvas is to use the ROOT Browser. On our canvas is a statistics box that shows the number of entries (Nent), mean and RMS. We would also like to see the overflow and underflow in this box. We could do that by right clicking on it and setting the option. To show you another way to do this we will use the browser.



Go to your browser window. If you don't have one, you can start one from your canvas by selecting the **Start Browser** item in the **Inspect** menu.



Double click on the **Canvases** folder. A folder labeled c1 will appear beneath it.

Double click on **c1** and the list of objects on c1 appear on the right hand side of the browser. Right mouse click on **stats** to bring up the context menu for it. Left mouse click on the option **SetOptStat**. A dialog box will appear.

Enter **111111** (this is a PAW convention for displaying all statistics.) You may have to refresh the canvas by selecting the **Options** menu and the **Refresh** menu item. Now the new statistics should be visible. You can resize the statistics box to make it more readable on the canvas.

Step 6: The Class and Members Reference

Right now, with the above example on hand, is a good opportunity to introduce the ideas of classes and objects that ROOT uses. TPaveStats::stats is the header of the context menu which appeared when we right clicked on the statistics box. What this means is that this object, which we clicked on, is an object of the TPaveStats class. In order to find out about it, we are going to look up the TPaveStats class in the "Classes and Members Reference" index on the ROOT web site. Each class used in ROOT is listed in the index with a description of the class and its methods. The location for the Class Index is:

http://root.cern.ch/root/html/ClassIndex.html

Go to TPaveStats in the index and click on it. The window for TPaveStats appears. Click on class description and it will take you to a description of the class with all of its methods. Here is where you can find out what it means to set the options to (111111), just as we did in the step-by-step session.

Making use of the index is a very good way to become familiar with ROOT classes and their uses.

Step 7: The Canvas Editor

The ROOT canvas has several menus these are: File, Edit, View, Options, Inspect and Classes. We've already mentioned **Refresh** and **Event Status** that appear under the **Option** menu. Next we are going to make use of the **Editor,** an option under the **Edit** menu, which brings up a new window. This window allows you to interactively create new graphical objects on the canvas.

1. Bring up the Editor from the Edit menu selecting the **Editor** item. Click on the **Arrow** button. Go to your canvas c1 and place the mouse at the point you want to begin the arrow. Click left button and keeping it pressed, drag to the end point of the arrow. Release the mouse button to set the arrow in the canvas.

2. Click on the **Text** button. Click the left mouse button at the point

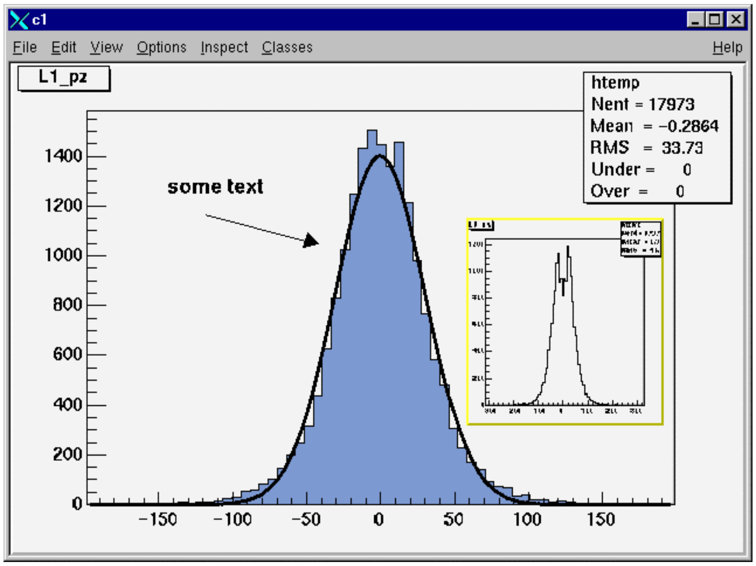
in the canvas where you want to put the text. Type in the text and then press carriage return. If you want to move the text to another place on the canvas, you can point to the text while keeping the left mouse button pressed and then drag the text to a new position on the canvas.

Deleting an object

1. If you want to delete an object, right mouse click on the object and then choose Delete from the context menu. When the dialog box appears, leave it empty and click OK.

2. Click on the **Pad** button. This allows you to make a pad (a sort of mini-canvas) within the canvas. Just as above, click the left mouse button at the point where you want to place the pad on the canvas. Keeping the button pressed, drag your mouse across to draw the size of the pad you want. Release the left mouse button to set the pad in place.

3. Now you can draw another histogram in the new pad you've created. First you have to make sure that the pad is active. It should be highlighted with a colored line around it. If it isn't you can make the pad active by clicking on it with the middle mouse button. Then go to the ROOT Browser screen that has the list of variables.



4. Double click on the variable that you want to display in the pad (I've double clicked on L1_py). It should appear in the pad within the canvas.

5. If you wanted to print the canvas, you click on **Print** under the canvas menu option **File.** Root saves the file as a postscript file that you can print later. The file is saved in the current directory as c1.ps.

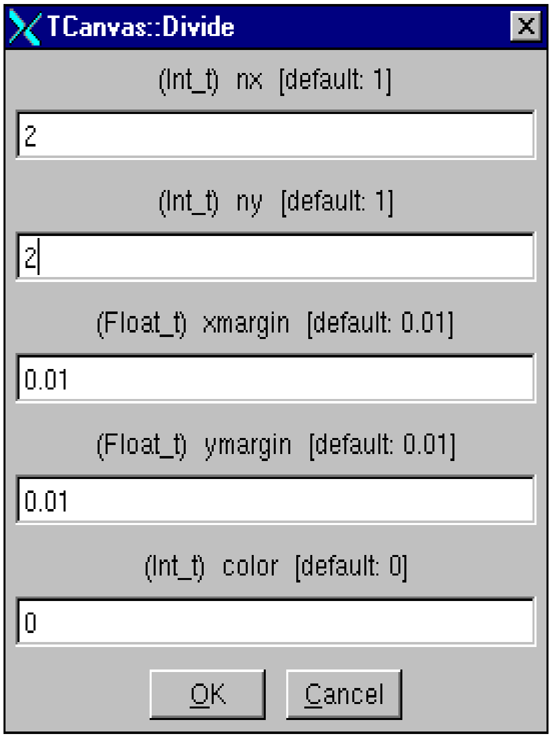
ROOT has a full explanation of **How to Use the graphics Editor** located at:

http://root.cern.ch/root/HowtoEdit.html

Step 8: A Canvas Displaying Multiple Histogram

Click on the **New Canvas** option on the **File** menu of canvas c1. A new canvas is automatically created that is named **c1_n2**. This new canvas should now be color highlighted around its edges to indicate that it is the active canvas. If it isn't, click on the canvas with the middle mouse button (or right and left together) to make it active.

Point the mouse within the empty canvas and right mouse click to bring up the context menu for the canvas. Click on the property **Divide**. A box should pop up which looks like this. To make a canvas with four pads, you enter the options as shown and click on OK.



Now you have an empty canvas with 4 pads in the display. We'll use the 4 pads to set up four different displays of the variable L1_px.

To do this:

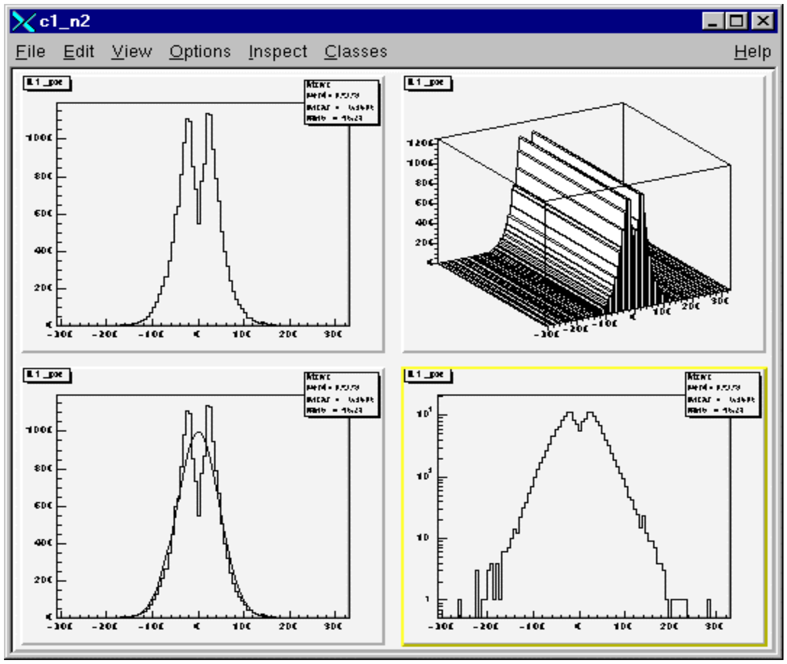
Click on the upper left pad with the middle mouse button to make it the active pad. This will be our pad that displays the standard histogram view. Then go to the root browser showing the list of variables and double click on the variable L1_px.

Click on the upper right pad with middle mouse button to make it active. Then, just as above, go to the root browser and double click on the variable L1_px again. Go to the outline of the histogram and right mouse click to bring up the context menu for this pad's histogram. From the Properties list, choose **DrawPanel**. In the DrawPanel window, click on **lego1** and then **Draw**. When you have a lego plot like this you can use the left mouse and rotate it. Press down on the left mouse button and drag it. You can see the transparent cube moving with the mouse. Let go, and the plot is drawn with the new perspective.

As above, click on the lower left canvas pad with middle mouse button to make it active and then fill it with the variable L1_px. Right mouse click on the histogram outline and choose the **FitPanel** option from the Properties list. In the FitPanel window, click **gaus**, **set all weights to 1**, and then **Fit**.

Make lower right canvas pad active and fill with variable L1_px. This time, point your mouse in the outer canvas part of the pad to bring up the canvas context menu for that display. Click on **SetLogy.** The pad will display a logarithmic view of the variable.

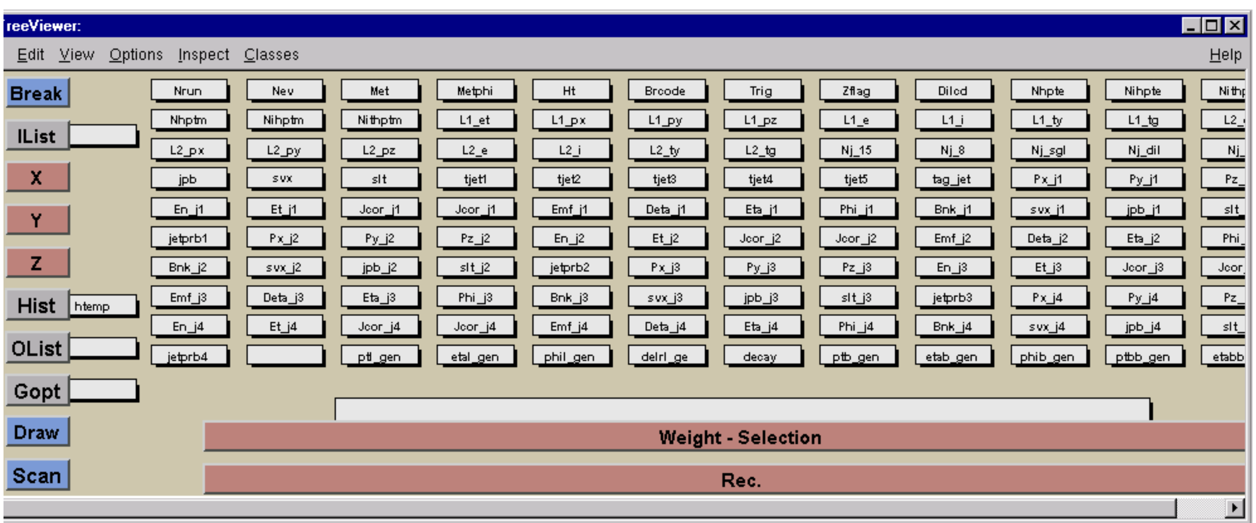
Again, you can print the canvas display by clicking on Print from the File item menu on the canvas. It saves the canvas as a postscript file that you can print later.



We've finished setting up 4 pads on the canvas with 4 views of the L1_px variable and it should look like this.

Step 9: The TreeViewer

Root introduces a class for organizing and storing data that it calls a Tree. Without getting too involved in the nitty-gritty of this concept, the Tree architecture extends the concept of the Ntuple to all complex objects or data structures found in Raw Data tapes and DST's. The idea is that the same data model, same language, same style of queries can be used to all data sets in one experiment.



The viewer for TTree brings up a separate window that allows you to manipulate the contents of a tree, with all of its members and methods. We can bring up a Tree Viewer screen for our ntuple file " h21" by typing in the command:

h21->StartViewer();

The TreeViewer appears. The white boxes represent all of the variables in the tree.

The set of buttons on the left of the canvas allow you to perform the following operations:

Button	What does it do?
Draw	Display the active variable(s) that are placed in the X, Y, Z buttons
Scan	Same with TTree::Scan output style (not for beginners)
Break	Interrupt the current event loop
OList	Creates a new TEventList using the current selection
IList	Actives a TEventList as an input selection
Hist	Redefines the default histogram (default = htemp)
Cpad	Specifies the output pad (default = c1)
Gopt	Used to specify graphic options
X	To select the X variable
Y	To select the Y variable
Z	To select the Z variable
Weight-Selection	To select the weight/Selection expression
REC	To record the command in the history file

In addition, the vertical slider on the far left hand side of the canvas can be used to select the minimum and maximum events range.

Now let's use it to set up a 2 dimensional lego view of L1_px and L1_py with a selection cut of L1_py >50.

To do this:

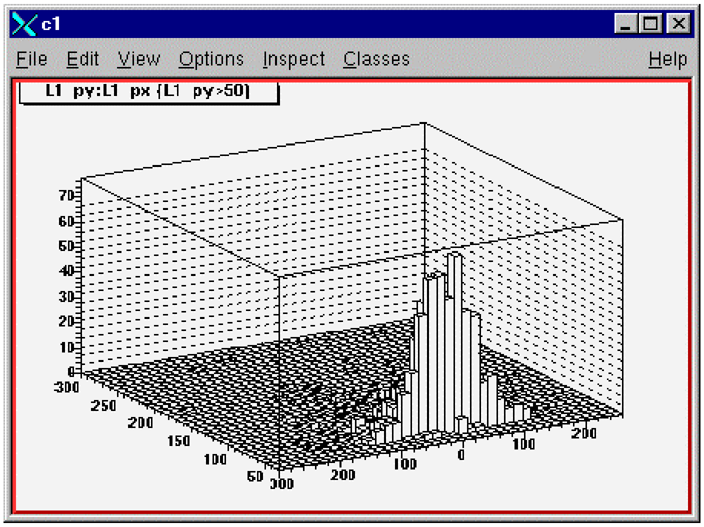
Left mouse click on L1_px and drag and drop it to the X button. Left mouse click on L1_py and drag and drop it to the Y button

Right mouse click on the white box next to the Gopt button. Select the item "Set Label". An option box appears on your screen. Type in: lego then click "OK"

Right mouse click on the white box next to the Weight -Selection button. Select the item "Set Label". An option box appears on your screen. Type in: L1_py>50 and then click

"OK"

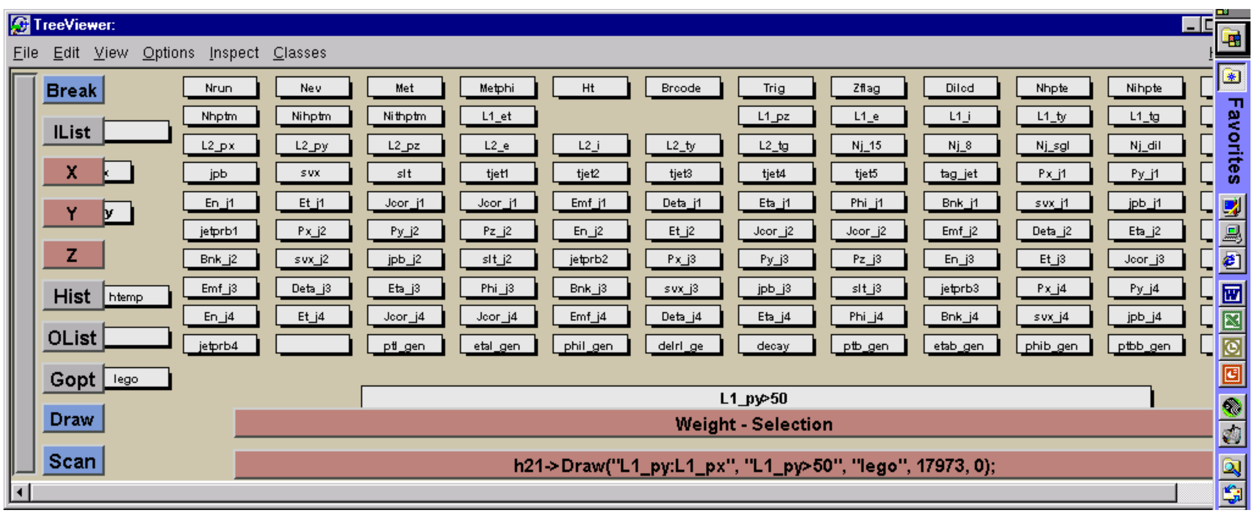
Click on the Draw button



When the "Draw" button is clicked, The TTreeViewer object assembles the information you've set up and draws the event with the corresponding arguments. While the event loop is executing, a red box inside the slider bar indicates the progress in the loop.

While the "Draw" button is executed, the event loop can be interrupted by clicking on the "Break" button. The current status of the histogram is shown.

All variables can be dragged and dropped to their destination (which can be X, Y, Z or Weight/Selection).



You will also see the assembled root command that was executed by clicking and dragging. The command is displayed in the text box below the Weight-Selection box.

This is root's way of capturing the mouse events and offering a way to replay them. The command is saved in the command history file and can be retrieved to use in a macro or on the command line directly.

You can create a new variable by clicking with the right mouse button on the TTree viewer canvas and selecting "CreateNewVar". When the option box appears, type an expression to create your variable. For example: L1_py**2+L1_px**2.
The variable is created and will be displayed in a new box on the tree viewer. If you click

on draw (or double-click on the box) the new variable will be drawn with all of your current setup for events range, cuts, graphic options, and weights-selections.

At any time the canvas can be saved in a Root file by with the menu "File" and menu item "Save as Canvas.root". If the current canvas is called c1, it will generate a file c1.root. You can restore the canvas in a later session by opening the file and drawing the canvas.

Root > TFile f("c1.root");
Root > c1.Draw();

Step 10: Ending the ROOT Session

To end the root session, at the root prompt line type:

.q

We're just about to end this first step-by-step session, so let's review what we have done. We've touched upon many of the options that are available to you when using ROOT's GUI with a canvas. We've used ROOT's Editor menu to create new graphical objects on a canvas. We've begun to learn about ROOT's Object Browser and have used context menus to change the graphic display and to make very basic fits and displays on the canvas. Lastly, we've used the Tree Viewer to plot a two dimensional histograms. Coming up is Part 2 - which deals with ROOT commands.

ROOT COMMANDS

In the previous sections we've made use of the mouse and clicked our way through ROOT. You must be curious by now, how these things are done on the root command line. In this section we show you how to do the things we did with the mouse, from the command line with root commands.

Root commands come in three flavors, C++ commands, CINT commands, and Shell commands. Here are examples of all three:

1) C++ syntax:
   root[0] TBrowser * b = new TBrowser();

2) CINT commands start with a **"."**

   root [1].?

   // this command will list all the CINT commands

   root [2].X [filename]

// load [filename] and execute function [filename]

root [3].L [filename]

// load [filename]

3) SHELL commands start with a **".!"**
   root[4] .! ls

   This command will list the contents of the current shell directory

C++ commands are combined to build macros, the scripts that you will later be writing in order to create, modify and display your final output. The commands are case sensitive, so make sure you've entered them properly.

The Tab-Tab Feature

When you enter commands, ROOT provides a very helpful tab-tab feature. You type in a variable followed by an arrow, for example:

gStyle-> and then enter a tab

This prints out on your display a listing of all the methods and members of gStyle.

This lists all of the things that you can do with gStyle at this point.

Now you can continue by typing in:

gStyle-> SetOpt<tab>

It will display a list of all the methods and members that start with SetOpt.

Now type:

gStyle-> SetOptS<tab>

It completes the line to say: gStyle-> SetOptStat. Type

gStyle-> SetOptStat(<tab>

A list of the parameters that are expected for this method will appear. If the method has multiple signatures all of them are displayed.

The Command HistoryFile

A history of the commands that you entered during ROOT sessions is stored in text file called $ROOTSYS/.root_hist. Commands can be copied out of this text file and placed into the text file that you create as the script (the series of running commands) of your macro.

Open a Root File and Display the Browser

Before we start I would like to explain a couple of things about the syntax used on the root command line that always puzzles people. First, you have the option to leave off the semicolon (;) at the end of a root command when working on the command line. The difference between having it and leaving it off is that when you leave it off the return value of the command will be printed on the next line. For example:

> **root [0] 23+5** // no semicolon prints the return value
> **(int)28**
> **root [1] 23+5;** // semicolon no return value is printed
> **root [2]**

The second leniency of CINT is that is allows you to use "." and "->" interchangeably. If you have learned C++, this will bother you and look strange. A pointer p still works when used as p.Print() rather than p->Print(). This is a well-meaning feature of CINT that takes getting used to, especially for the C++ expert.

ROOT Command	What does this command do?
.q	End a ROOT session
TFile *my = new TFile("htt_175a.root");	Open a file
TBrowser *b = new TBrowser();	Open the ROOT Object Browser
my->ls();	List contents of a file called "my"

my->cd("TOPPLOT");
 	
gDirectory->ls();
	Change directory to make TOPPLOT the current directory (the directory which contains the ntuples)
	
	List contents of current directory, now TOPPLOT

h21->Print();	To find names of variables in the ntuple file

Drawing Histograms

ROOT Command	What does this command do?
h21->Draw("L1_pz");	To display the variable "L1_pz" as a 1-D histogram
c1->Print();	To print the display of canvas "c1". ROOT saves it as a ps file to print later.

h21->Draw("L1_pz>>myh1");	To display the variable "L1_pz" and save to a named histogram called "myh1"

Adding Text

Here are some examples of commands for adding text to your display, setting the fill colors of objects, and the color and style of markers that are used for drawing a histogram with error bars. To see a menu of colors available to use in ROOT with the numbers assigned to them, you go to the "**Colors**" option under canvas item menu "**View**". For marker styles, you go to "**Markers**", also under canvas item menu "**View**".

ROOT Command	What does this command do?

t1 = new TpaveLabel(-235,1217,-44,1387,
"A label","br");
t1->SetFillColor(6);
myh1->Draw();
t1->Draw();
	Create a pave label
	
	Set a fill color for it
	Display canvas "myh1"
	Display the label on the canvas

myh1->SetFillColor(2); myh1->Draw();	Set the fill color of "myh1" to red and display

h21->SetMarkerColor(4);
h21->SetMarkerStyle(8);
h21->Draw("L1_py","","e1");
	Set marker color of "h21" to blue
	Set marker style of "h21" to a filled circle
	Draw "L1_py" with error bars
	(Other displays are "e2", "e3", and "e4")

Axis Title

ROOT Command	What does this command do?
myh1->GetXaxis()->SetTitle("X-Axis");	Give the x-axis the name "X-Axis"
myh1->GetXaxis()->CenterTitle(1);	Center the name of the x-axis.
myh1->GetYaxis()->SetTitle("Y-Axis");	Give the y-axis the name "Y-Axis"
myh1->GetYaxis()->CenterTitle(1);	Center the name of the y-axis.

Adding Cuts

ROOT Command	What does this command do?
h21->Draw("L1_px","(L1_px>0) ");	Display "L1_px" where "L1_px" is greater than zero
h21->Draw("L1_py","(L1_px<0) ");	Display "L1_py" where "L1_px" is less than zero
h21->Draw("L1_px","(L1_py<.5) && (L1_pz>10)");	Chained cuts

Changing Statistics

ROOT Command	What does this command do?
gStyle->SetOptStat(111111);	Set statistics to display under and overflow

Fits and Functions

ROOT Command	What does this command do?
myh1->Fit("gaus");	Fit the saved histogram " myh1" with a gaussian fit.
myh1->Fit("pol9");	Fit the histogram " myh1" with a polynomial fit of degree N, in this example pol9.

f1= new TF1("f1","abs(sin(x)/x)",0,10);
 	
f1->Draw();
	Create a user-designed function and draw it

TF1 *e1 = new TF1("e1","expo",0,200); myh1->Fit("e1","R");	Declare a exponential function to fit over a sub range (Note: the "R" in the Fit command stands for "range". If you leave out the "R" as an argument in the Fit command, the range of fit is the current range of the histogram(in this case, the full range) To reset the range to what you want, you would have to use the SetRange

e1->SetRange(0,200);
 	
 myh1->Fit("e1","R");
	command:
	
	 If "R" previously left out, to reset the range of fit "e1" from 0 to 200
	
	 Then enter the fit command again, using "R"' option)

TF1 *g1 = new TF1("g1","gaus",-200.,200);
 	
 TF1 *g2 = new TF1("g2","gaus",-200., 0);
 	
 myh1->Fit("g1","R");
 	
 myh1->Fit("g2","R+");
	Display 2 subranges of fits on one display using built in function gaussian function

f2 = new TF1("f2","[0]/(x*x+[1]*[1])" ,-100,100);
 	
 f2->SetParameters(1000000,30);
 	
 myh1->Fit("f2","R");
	Create a user designed function to use in a fit
	
	 Set initial values of parameter
	
	 Fit the function to histogram "myh1"

Multi-histogram display

ROOT Command	What does this command do?
myc1= new TCanvas("myc1","My canvas");	Manually create a new canvas

myc1->Divide(2,2);
 	
 myc1_1->cd();
 	
 h21->Draw("L1_px>>his1");
 	
 myc1_2->cd();
 	
 his1->DrawCopy("lego1");
 	
 myc1_3->cd();
 	
 his1->Fit("gaus");
	Automatically generate 4 pads in the new canvas "myc1" and then set the display for each pad
	
	 Set pad1 "myc1_1" as active pad and display standard histogram view of "L1_px", saving the display as a named histogram "his1"
	
	 Set pad2 active, and draw a copy of "his1" as lego view
	
	 Set pad3 active , and draw a copy of "his1" with a gaussian fit

myc1_4->cd();
myc1_4->SetLogy(1);
his1->Draw();
	Set pad4 active and set to display a logarithmic histogram. Display "L1_px" in it.

Global Variables

ROOT Command	What does this command do?
gEnv->Print();	Display the settings of the current ROOT environment
gObjectTable->Print();	Display list of objects known in current ROOT session
gStyle->SetOptStat(111111);	Set statistics to full display

Two Dimensional Display

Here, let's just start afresh. We'll quit ROOT and start a new session where we will open our example root file and again change the directory to TOPPLOT in order to work with 2 variables. ROOT keeps a history of the commands you have entered in a file that is called root_history.txt and you can recall previous commands by using the arrow keys on your keyboard

ROOT Command	What does this command do?

TFile *my = new TFile("htt_175a.root");
my->cd("TOPPLOT");
h21->Draw("L1_px");
h21->Draw("L1_py", ,"same");
	To also display "L1_py" on the current display that shows L1_px

h21->Draw("L1_px:L1_py");	Display L1_px and L1_py as a scatter plot
h21->Draw("L1_px:L1_py", ,"lego");	Draw as a lego plot

h21->Draw("L1_px:L1_py", "L1_pz<50");	Draw with a cut
h21->Draw("L1_px:L1_py", "L1_pz<50","cont");	Draw with cut as a contour plot

Creating a Root File

Although we haven't covered this in the tutorial, here is an example of commands to create a Root file**.** Files can contain any kind of ROOT objects, such as histograms or canvases. When you create a histogram or root tree, ROOT will automatically attach it to the ROOT file. Other objects, such as a new canvas that you create and want to put it into a saved ROOT file, have to be explicitly attached to the file.

The choices for the second parameter on the TFile constructor are: RECREATE, NEW, CREATE, READ, and UPDATE. The NEW and CREATE have the same effect. They will create a new file and open it for writing, but if the file already exists it is not opened. RECREATE will also create a new file and if the file already exists it will be overwritten. UPDATE will open an existing file for writing. READ will open an existing file as read only.

In the box below are commands to create a new ROOT file, and then create and save a new histogram and a new canvas in the ROOT file that you have created.

ROOT Interactive Command	What does this command do?
gROOT->Reset();	Clear ROOT
TCanvas *mc1 = new TCanvas("mc1","My canvas");	Create a new canvas
TFile *hfile = new TFile("newfile.root","RECREATE");	Create a new **.root** file that we name as "newfile.root"
TH1F *hr = new TH1F("hr"," Random",20,0,10);	Create a new histogram "hr" and fill the histogram with random numbers

gRandom->SetSeed();
{
for (Int_t i=0; i<500; i++) {
Float_t random = gRandom->Rndm(1)*10;
	To start a multi line command from the command line, you type an opening curly bracket.
	The prompt changes and you can type the next line of the multi-line command.
	The prompt stays in the multi-line mode

`hr->Fill(random);`
`}`
`}`
	until you enter a matching closing curly bracket. At that time, the prompt returns to the single line mode.

`hr->Draw();`	Draw the histogram "hr"
`mc1->Modified();` `mc1->Update();`	Update the canvas "mc1"
`hfile->Append(mc1);`	Explicitly attach the canvas to the file (If the object is a histogram or a ntuple/tree, ROOT automatically attaches it to the file)
`hfile->Write();`	Save the file

`TDirectory *d = hfile->mkdir("MyTop");`	Create a subdirectory of your ROOT file with a pointer named "d"

`TCanvas *mc2= new TCanvas("mc2","My canvas");`
`f1= new TF1("f1","abs(sin(x)/x)",0,10);`
`f1->Draw();`
	Make a canvas and fill with a display

`d->Append(mc2);`	Add the canvas to the subdirectory
`gDirectory->Append(mc2);`	Another way to add canvas to subdirectory
`hfile->Write();`	Save file
`hfile->Close();`	Close file

Helpful Hints

Converting a PAW ntuple file into a ROOT file

At this point, you might want to try out ROOT using a ntuple file of your own. There is a program called **h2root** which converts files created with PAW into ".root" files that can be used with ROOT. To do the conversion, at the system prompt, you just have to type:

h2root filename.hbook filename.root

If your file contained a HBOOK ntuple, it is automatically converted into a ROOT tree. In ROOT terms, it becomes an object of the TTree class. If your file had a HBOOK histogram, it is automatically converted into a histogram object of ROOT's TH1F class.

Once your file is converted you can start using it with ROOT.

Reading an ASCII file and making a ROOT ntuple

ROOT's official homepage from CERN contains a Tutorial file, and one of the tutorials is an example of a macro that reads data from an ASCII file and creates a root file with a histogram and a ntuple. This tutorial is located at:

http://root.cern.ch/root/html/examples/basic.C.html

You can modify this macro and use it for your own ASCII files.

Saving the ROOT Canvas as a Macro

In the above examples of ROOT commands there is an example for putting a pave label into a histogram display. The easiest way to do this is to use the graphic interface, just as we did in Part 1. You can set up a Pave Label using the **Editor** that is located under the canvas menu item **Edit** . Once you have the display set up in a way you like, you can use the option **Save as canvas.C** , located under the canvas menu item **File.** This creates a skeletal macro with the default name of the canvas you have on display followed by ".C" (e.g. c1.C). The skeletal macro would have the ROOT command coding for the pave label with its position on the canvas. The macro will require some editing, but can be a useful basis for a macro that you would create for your own use. Additionally, a history of the commands that you entered during ROOT sessions is stored in text file called **root_history.txt**. Commands can be copied out of this text file and placed into the text file that you create as the script (the series of running commands) of your macro.

Further Documentation About ROOT

We hope this tutorial has given you a small introduction to ROOT and that you will now be interested in going out into the wide world of ROOT to learn and make use of its more advanced functionality when using it as a physics analysis tool. There is a wealth of material about ROOT available via the web, particularly through the homepage of ROOT where the creators of ROOT, Rene Brun ,Fons Rademakers and their team, have provided an enormous amount of documentation **.**

The ROOT Homepage is the **official ROOT web site**. The location is: http://root.cern.ch. Some very useful locations within this site to the beginner user are given below. All of them can be accessed from the listing that appears on the Homepage.

Search the ROOT Web Site. **For on-line help** or a search on a particular topic it's very useful to use this facility. The location is: http://root.cern.ch/root/Search.phtml

The ROOT Tutorials are a description of tools that you will be using in ROOT for a PAW style analysis. It gives examples of creating histograms/ntuples, fits, functions,etc. with **sample macros.** . The location is: http://root.cern.ch/root/Tutorials.html

The ROOT How To's are an introduction to the infrastructure of ROOT and give examples of **how to write code within ROOT classes.** . The location is: http://root.cern.ch/root/Howto.html

Classes and Members Reference Guide is an index to ROOT classes with **full class description, including options** to commands using the class. The location is http://root.cern.ch/root/html/ClassIndex.html

The Publication List is a list of ROOT related publications. At the bottom of the list is non official ROOT documentation. It is located at: http://root.cern.ch/root/Publications.html

There is a ROOT talk mailing list. Subscribe via:

http://root.cern.ch/cgi-bin/registration.pl

To submit a bug report: http://pcroot.cern.ch/root-bugs.

You can search and browse the Bug Database before submitting your own bug report.

Setting up Root at Femilab

1. Log on to cdfsga, fnalu, fnpat1, or d02ka

2. On cdfsga type:

```
setup root v2_23_11 –q KCC_3_3
```

You may need to set up your environment. At the system prompt type:

```
source ~cdfsoft/.cshrc
```

On fcdfsgi2, root should be already setup

On fnalu type:

```
setup root v2_23_11
```

On d02ka type:

```
setup root v2_23_11 -q KCC_3_3:exception
```

On fnpat1 type:

```
setup root v2_23_11 -q KCC_3_3
```

3. Startup root by typing at the system prompt:

```
root
```

4. Quit root by typing .q at the root prompt

```
root[0] .q
```

Appendix A: ROOT Command Reference

Axis: give the axis a title	myh1->GetXaxis()->SetTitle("X-Axis");
Axis: center the title	myh1->GetXaxis()->CenterTitle(1);
Browser: create a ROOT Browser	TBrowser *b = new TBrowser();
Canvas: manually create a canvas	myc1=new TCanvas("myc1","My canvas");
Canvas: automatically generate 4 pads	myc1->Divide(2,2);
Canvas: set pad1 "myc1_1" as active pad	myc1_1->cd();
Canvas: print the display of a canvas	c1->Print();
Canvas: refresh canvas "mc1" on screen	mc1->Update();
Cuts: examples	h21->Draw("L1_py","(L1_px<0) "); h21->Draw("L1_px","(L1_py<.5) && (L1_pz>10)");
Directory: list contents of current directory	gDirectory->ls();
Directory: change directory	my->cd("TOPPLOT");
Directory: create a subdirectory	TDirectory *d = hfile->mkdir("MyTop");
Directory: add canvas "mc2" to subdir Directory: add object to current directory	d->Append(mc2); gDirectory->Append(mc2);
End a ROOT session	.q
File: Open a file	TFile *my = new TFile("htt_175a.root");
Files: create new .root file	TFile *hfile = new TFile("newfile.root","RECREATE");
Files: save file	hfile->Write();
Files: close file	hfile->Close();
File: list contents of a file	my->ls();
Files: attach object to a file	hfile->Append(mc1);

Fits: fit histogram "his1" with a gaussian fit	his1->Fit("gaus");
Fits: fit histogram with a polynomial fit	his1->Fit("pol9");
Functions: create user designed function	f1= new TF1("f1","abs(sin(x)/x)",0,10);
Functions: exponential function to fit over a subrange	TF1 *e1 = new TF1("e1","expo",0,200); myh1->Fit("e1","R");
Function : set initial values of a parameter	f2->SetParameters(1000000,30);
Global variables: current environment	gEnv->Print();
Global var: list of objects in current session	gObjectTable->Print();
Global variable: clearing ROOT	gROOT->Reset();
Histogram: draw histogram "his1"	his1->Draw();
Histogram: draw a copy as lego view	his1->DrawCopy("lego1");
Histogram : Draw with error bars	h21->Draw("L1_py","","e1");
Histogram: set a fill color	t1->SetFillColor(6);
Histogram: set marker color	h21->SetMarkerColor(2);
Histogram: set marker style	h21->SetMarkerStyle(8);
Histogram: set to log. scale	myc1_4->SetLogy(1);
Histogram: save display as a named hist.	h21->Draw("L1_px>>his1");

Histogram: create & fill with random nos.	TH1F *hr = new TH1F("hr"," Random",20,0,10);
	gRandom->SetSeed();
	{
	for (Int_t i=0; i<500; i++) {
	Float_t random = gRandom->Rndm(1)*10;
	hr->Fill(random);
	}
	}

Pave label: Create a pave label	t1 = new TPaveLabel(-235,1217,-44,1387,

	"A label","br");
Range: set range of a fit	e1->SetRange(0,200);
Range: fit with range option	myh1->Fit("e1","R");

Range: set 2 subranges of fits on one display	TF1 *g1 = new TF1("g1","gaus",-200.,200);
	TF1 *g2 = new TF1("g2","gaus",-200., 0);
	myh1->Fit("g1","R");
	myh1->Fit("g2","R+");

Scatter plot: display "L1_px" and "L1_py"	h21->Draw("L1_px:L1_py");
Statistics: set to display under and overflow	gStyle->SetOptStat(111111);
Tree: create a TTree Viewer	h21->StartViewer();
Tree: display as a 1-D histogram	h21->Draw("L1_pz");
Tree: display & save as named histogram	h21->Draw("L1_pz>>myh1");
Tree: find names of variables in file	h21->Print();
Tree: " py" on same display as " px"	h21->Draw("py", ,"same");
Tree: 2 -D histogram as a scatter plot	h21->Draw("L1_px:L1_py");
Tree: 2-D scatter plot histogram with a cut	h21->Draw("L1_px:L1_py", "L1_pz<50");
Tree: 2-D contour histogram with cut	h21->Draw("L1_px:L1_py", "L1_pz<50","cont");
Tree: 2-D as a lego plot	h21->Draw("L1_px:L1_py", ,"lego");

Appendix B: The Canvas Menus

The ROOT canvas has a drop down menu which includes: File, Edit, View, Options, Inspect and Classes. There are also Context Menus for options

File : Most of the options for this canvas menu item are fairly self evident, but we'd particularly like to call attention to some of the **Save as** options

Option	What does it do?
Save as canvas.C	Generates a macro automatically which contain the ROOT command statements that correspond to the picture that you created interactively in canvas "c1"
Save as canvas.gif	Saves the canvas display as a .gif file
Save as canvas.ps	Saves the canvas display as a postscript file
Save as canvas.eps	Saves as an encapsulated postscript file

Edit : has options to Clear Pad and Clear Canvas. Also has option Editor

Option	What does it do?
Editor	Brings up the built-in graphics editor window which you can use to draw and edit basic primitives (arcs, arrows, text, paves, etc) starting from an empty canvas or as an addition to your current display A good description of how to use the editor is located at: http://root.cern.ch/root/HowtoEdit.html

View : Includes among the options:

Option	What does it do?
Colors	Brings up a separate window display of the colors available to use in ROOT with the numbers assigned to them
Markers	Brings up a separate window display of the marker types available to use in ROOT with the numbers assigned to them

Options : Includes among the options: **Show Statistics**, **Show Histogram Title**, and **Show Fit Parameters** which set the respective items to either be displayed or not displayed on the canvas. Other useful options, which we made use of in Part 1, include:

Option	What does it do?
Event Status	Displays the Event Status bar at the bottom of the canvas display window. Gives information about coordinate points on the canvas
Refresh	Refreshes the canvas display

Inspect : has 2 options

Option	What does it do?
ROOT	Brings up the Inspect window which gives information about all TROOT members being used in the current ROOT session
Start Browser	Brings up a new ROOT Browser window. Can be used by clicking to display and modify attributes of objects that are part of the current ROOT session.

Classes : has an option for Class Tree. It is more applicable for non-novice users of ROOT.