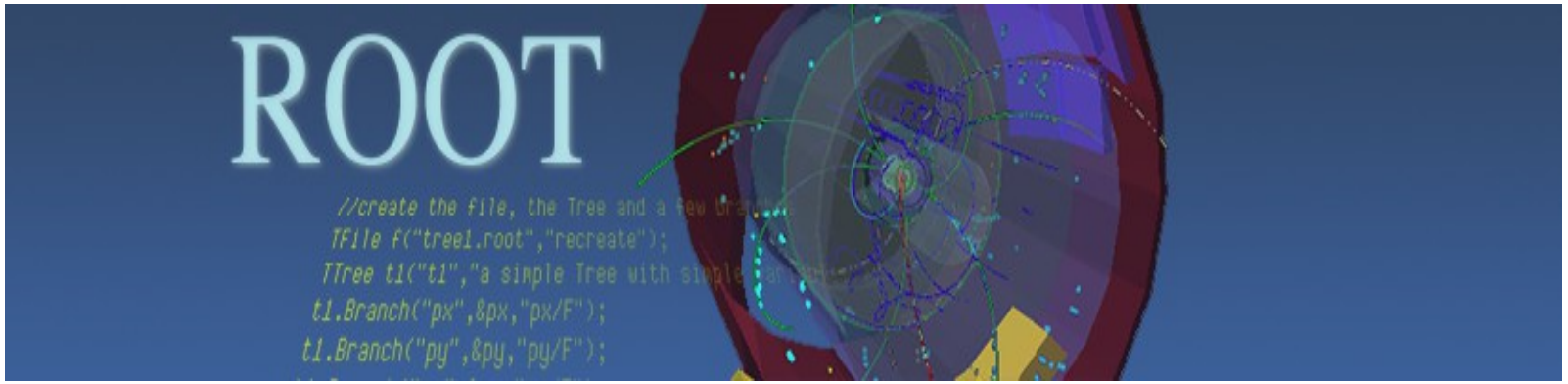
Data Analysis Frameworks

ROOT Data Analysis Frameworks
Computational Physics

Prof. Paul Eugenio
Department of Physics
Florida State University
April 04, 2019

http://root.cern.ch



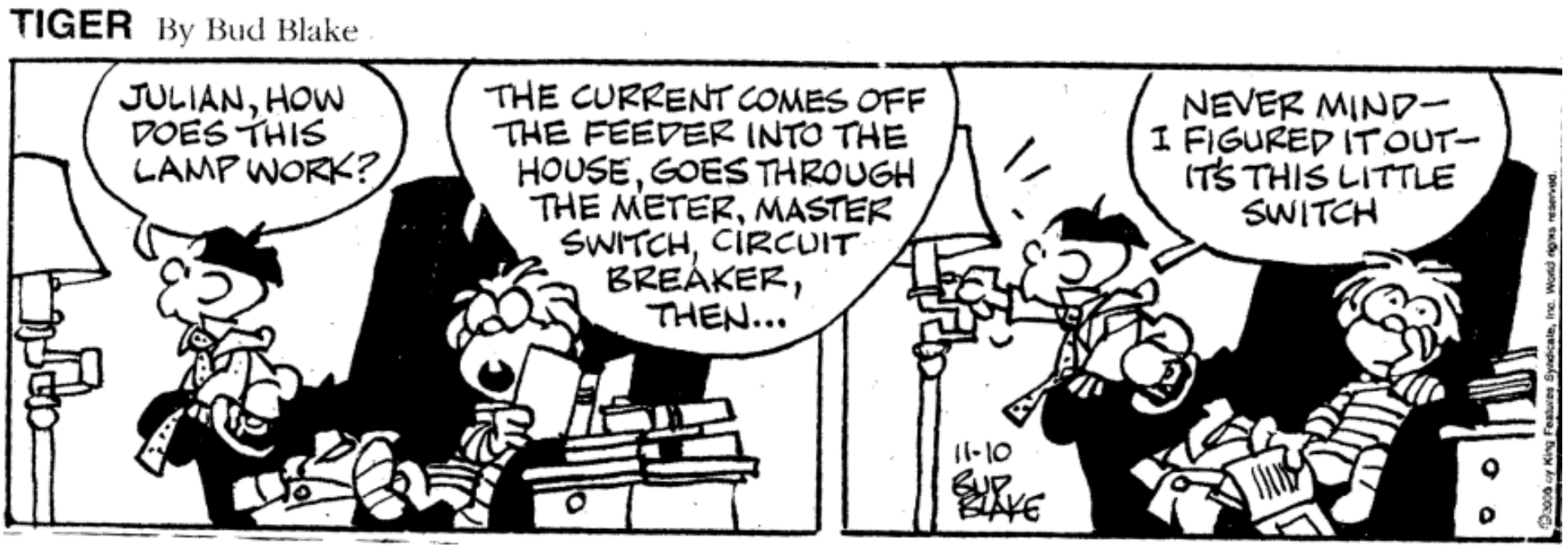
ROOT is an object-oriented framework

aimed at solving the data analysis challenges

of today's high-demand computing in physics

A Framework

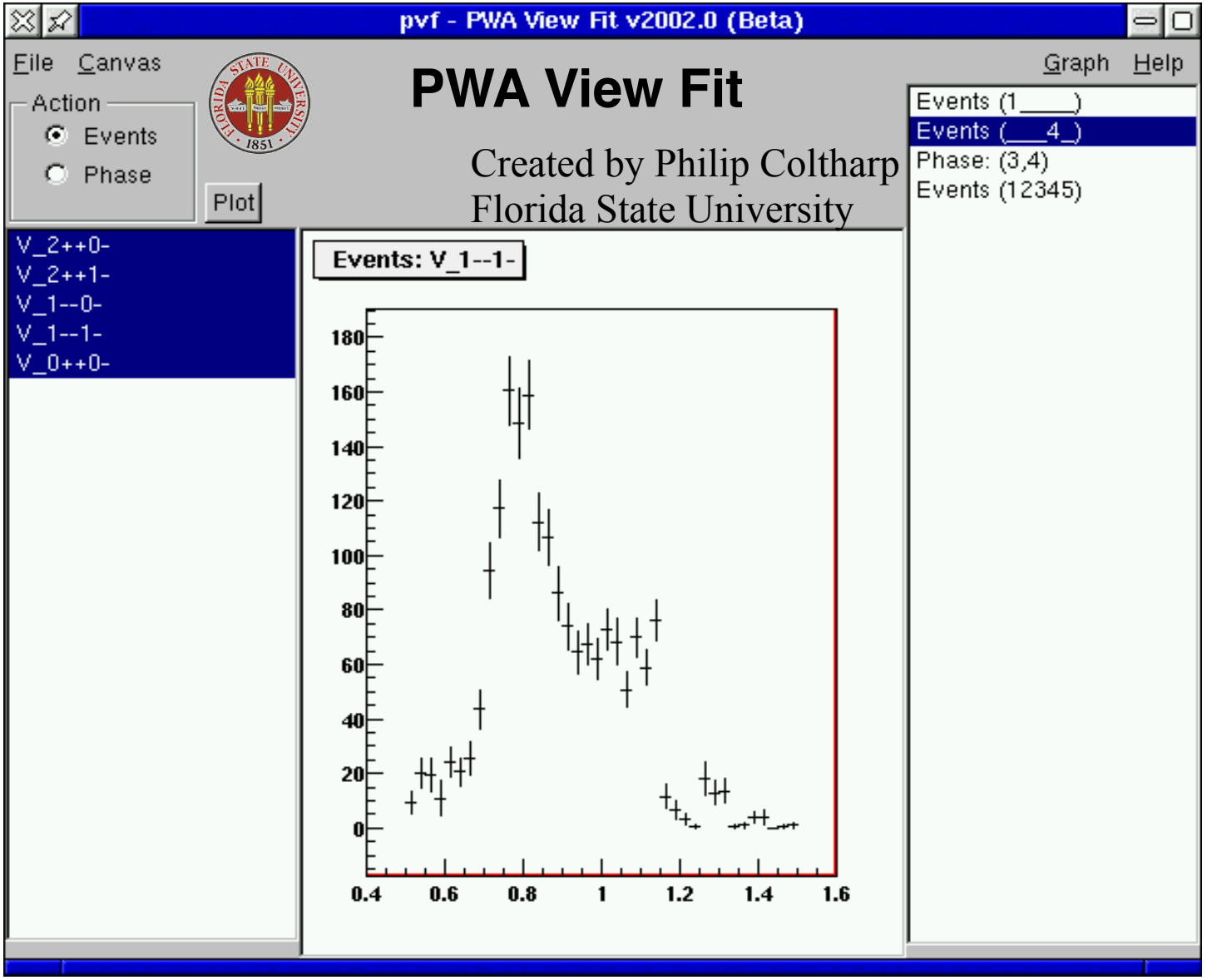
provides utilities and services.



ROOT's Services/Utilities

- Histogramming and Fitting
- Graphics (2D, 3D)
- I/O to file or socket: specialized for histograms, Ntuples (Trees)
- Collection Classes and Run Time Type Identification
- User Interface
  - GUI: Browsers, Panels, Tree Viewer
  - Command Line interface: C++ interpreter CINT
  - Script Processor (C++ compiled ⇔ C++ interpreted)

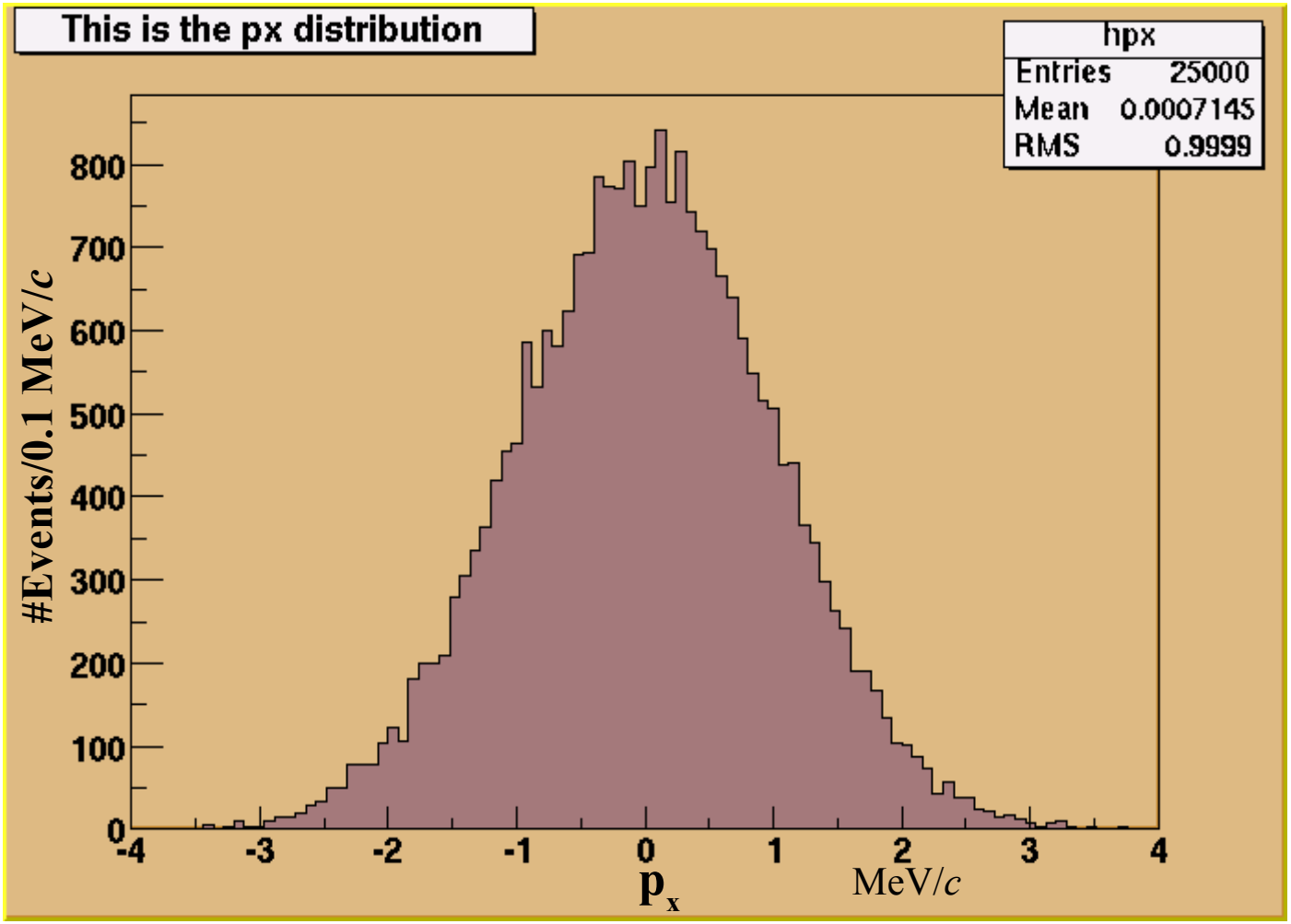
Example GUI Application



Data Analysis & Histograms

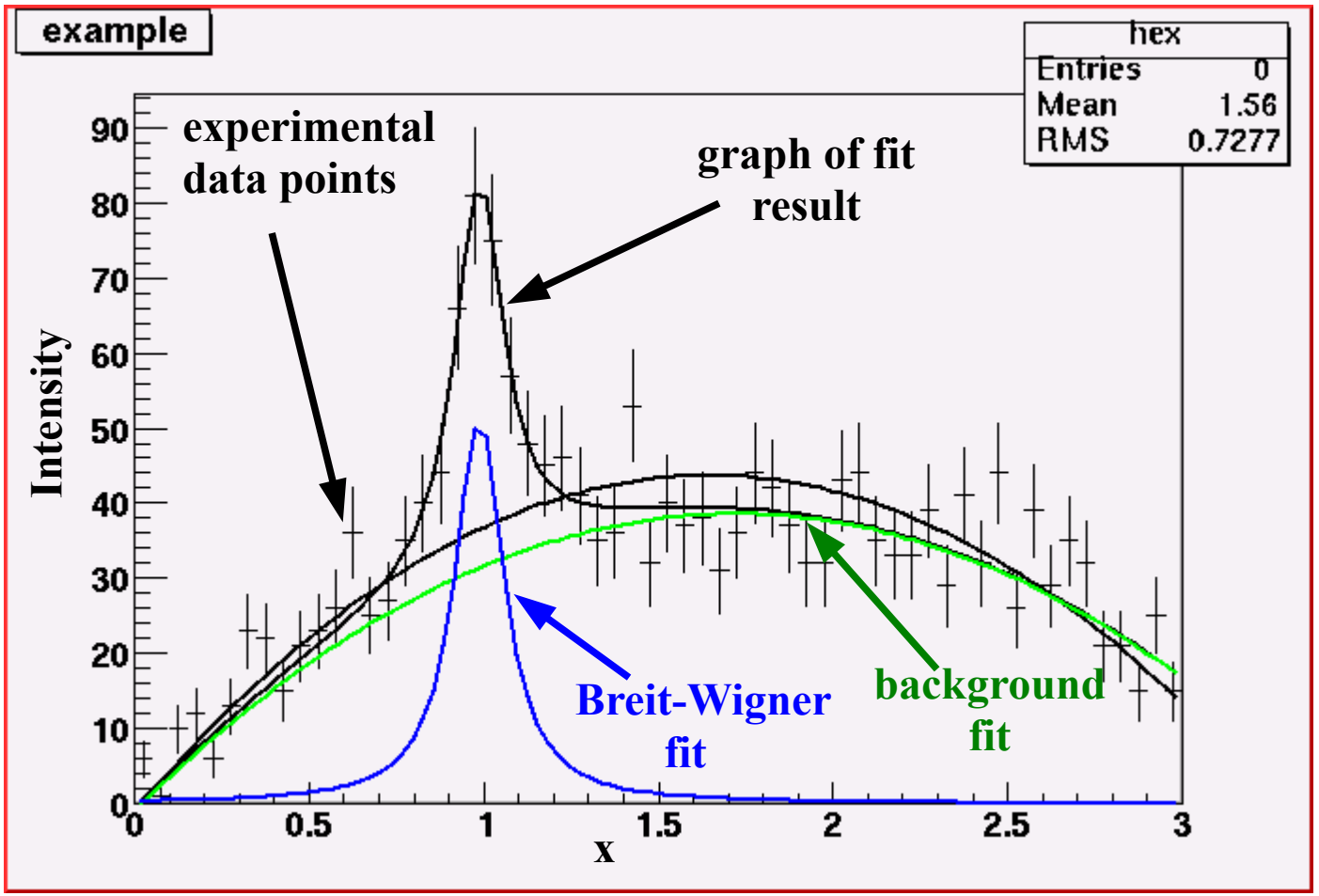
a graph representing a statistical distribution;

- projections from large volumes of data
- heights of the bars represent observed yields



Fitting Histogram Data

Breit-Wigner Resonance plus a polynomial background

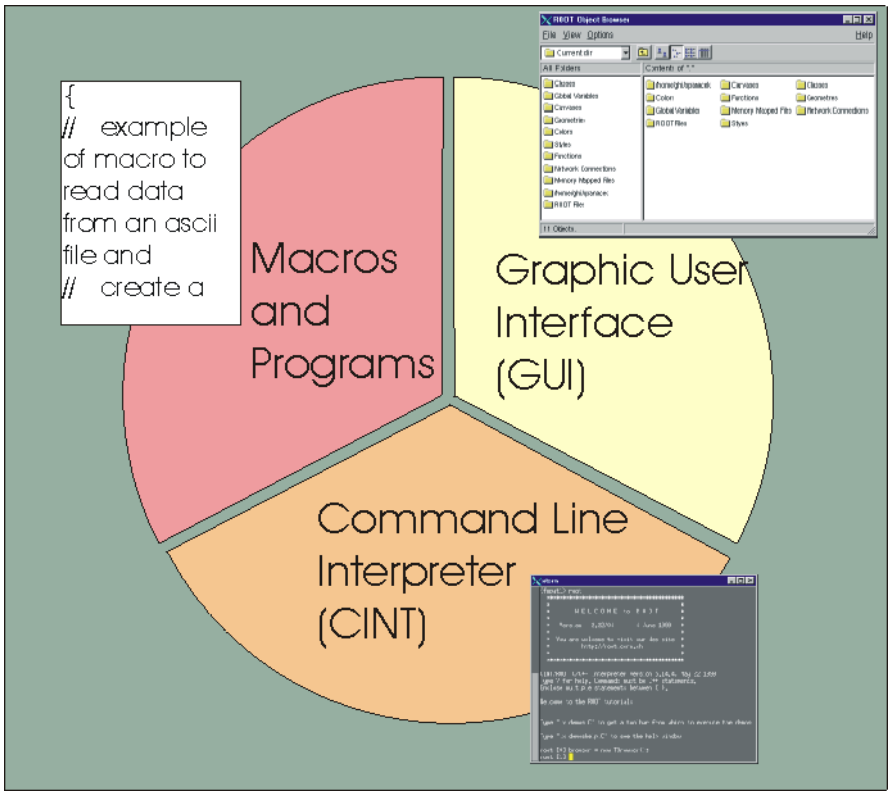


$$BreitWigner\,(x)=\frac{1}{2\pi}\,\frac{I_o\Gamma}{(x-M_0)^2+0.24\,\Gamma_o{}^2}$$

$$Background\,(x)=a_o+b_o\,x+c_o\,x^2$$

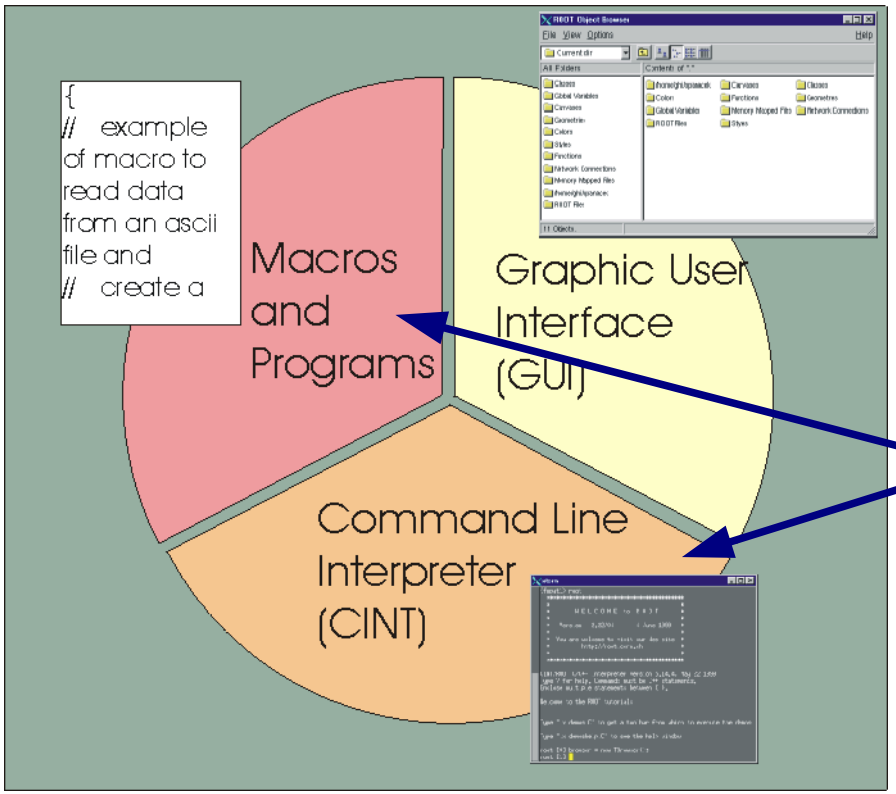
parameters of the fit: a_o, b_o, c_o, I_o, M_o, Γ_o

User Interfaces



- GUI
  windows, buttons, menus
- Root Command line
  CINT (C++ interpreter)
- Macros, applications,
  libraries (C++ compiler and
  interpreter)

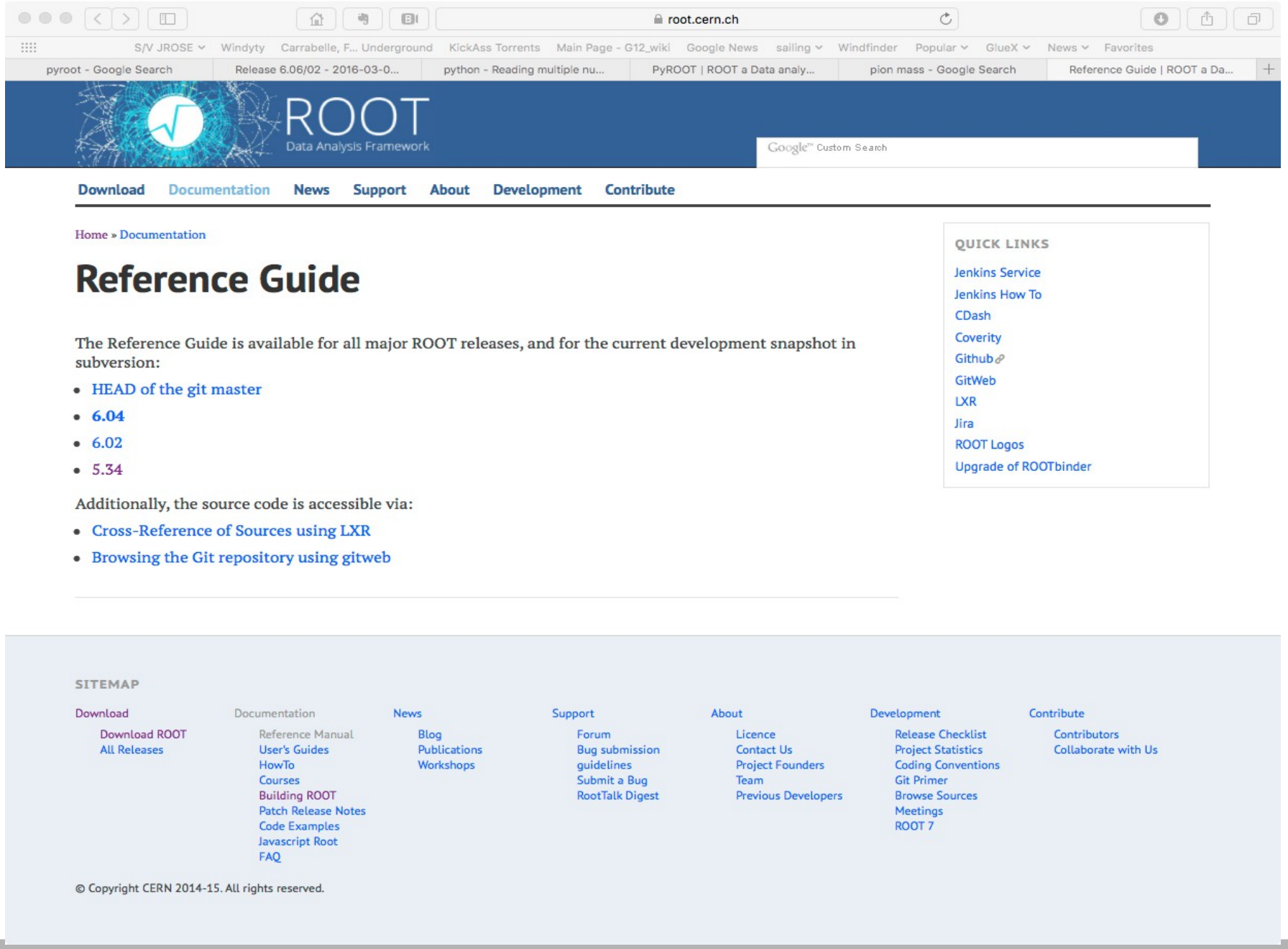
User Interfaces



- GUI
  windows, buttons, menus
- Root Command line
  CINT (C++ interpreter)
- Macros, applications, libraries (C++ compiler and interpreter)
- **PyROOT** ***Python*** module allows for command line usage via iPython, applications and macros through module import ROOT and high performance run-time bindings to original C++

ROOT Online Reference Guide

https://root.cern.ch/



https://root.cern.ch/pyroot

ROOT
Data Analysis Framework

Google™ Custom Search

Menu

Home » Interpreter

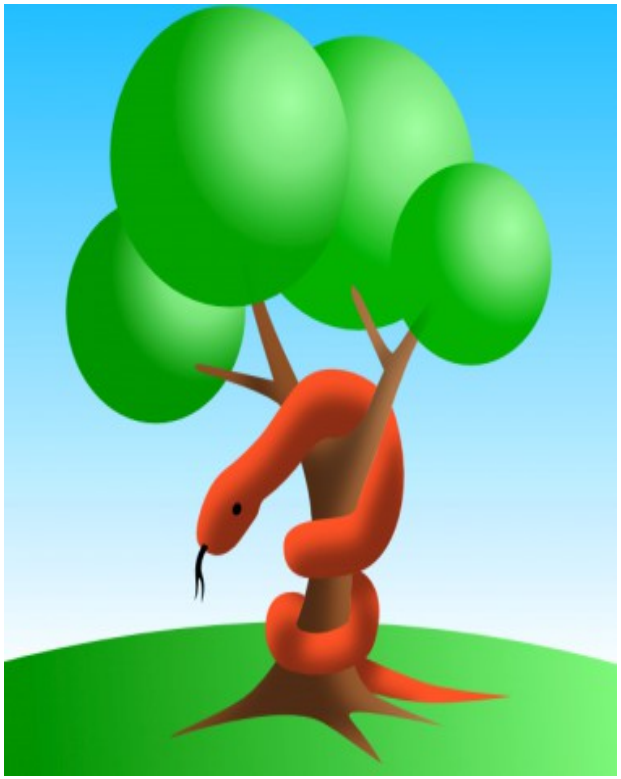
NAVIGATE THROUGH THIS BOOK

- Interact with CINT at ROOT's prompt
» Interacting with Shared Libraries: rootcint
- PyPyROOT
- PyROOT

PyROOT

How to use ROOT with Python (PyROOT)

PyROOT is a Python extension module that allows the user to interact with any ROOT class from the Python interpreter. This is done generically using the ROOT dictionary, therefore there is no need to generate any Python wrapper code to include new ROOT classes. At the same time PyROOT offers the possibility to execute and evaluate any Python command or start a Python shell from the ROOT/CINT prompt. Further details are available in the PyROOT manual.



More pythonistic PyROOT: rootpy

The generality with which PyROOT creates bindings based on dictionaries makes that it is close to the original C++, which is great for e.g. sharing documentation. However, where it is possible to do so automatically, e.g. for handling of container classes, "pythonizations" are available. If that is not enough for your needs, then check out the rootpy project for a more pythonistic PyROOT.

High performance PyROOT: cppyy

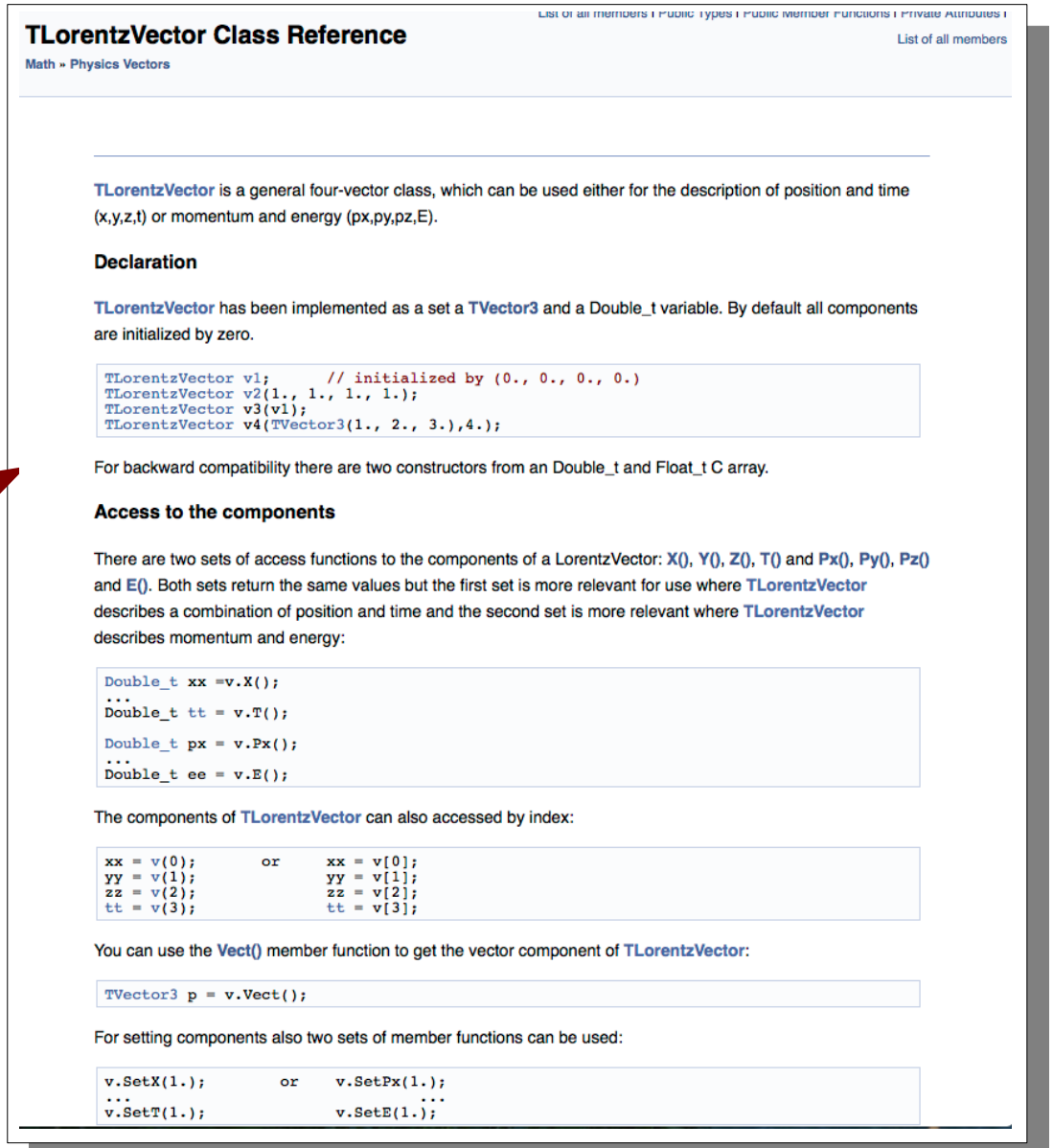
A new approach is based on cppyy, a run-time bindings generator like PyROOT for the PyPy project. It has both a CINT and a pure-Reflex back-end and is, depending on use, 50x-100x faster than PyROOT. A pre-installed version based on the CINT backend is available on /afs as pypyroot.

ROOT Online C++ Guide

https://root.cern.ch/doc/master/classTLorentzVector.html

Index of PHYSICS classes

TFeldmanCousins .....
TGenPhaseSpace .....
TLorentzRotation .....
TLorentzVector .....
TRotation .....
TVector2 .....
TVector3 .....



TLorentzVector Class Reference

List of all members

Math » Physics Vectors

TLorentzVector is a general four-vector class, which can be used either for the description of position and time (x,y,z,t) or momentum and energy (px,py,pz,E).

Declaration

TLorentzVector has been implemented as a set a TVector3 and a Double_t variable. By default all components are initialized by zero.

```
TLorentzVector v1;      // initialized by (0., 0., 0., 0.)
TLorentzVector v2(1., 1., 1., 1.);
TLorentzVector v3(v1);
TLorentzVector v4(TVector3(1., 2., 3.),4.);
```

For backward compatibility there are two constructors from an Double_t and Float_t C array.

Access to the components

There are two sets of access functions to the components of a LorentzVector: X(), Y(), Z(), T() and Px(), Py(), Pz() and E(). Both sets return the same values but the first set is more relevant for use where TLorentzVector describes a combination of position and time and the second set is more relevant where TLorentzVector describes momentum and energy:

```
Double_t xx =v.X();
...
Double_t tt = v.T();

Double_t px = v.Px();
...
Double_t ee = v.E();
```

The components of TLorentzVector can also accessed by index:

```
xx = v(0);       or     xx = v[0];
yy = v(1);              yy = v[1];
zz = v(2);              zz = v[2];
tt = v(3);              tt = v[3];
```

You can use the Vect() member function to get the vector component of TLorentzVector:

```
TVector3 p = v.Vect();
```

For setting components also two sets of member functions can be used:

```
v.SetX(1.);       or    v.SetPx(1.);
...                     ...
v.SetT(1.);             v.SetE(1.);
```

ROOT Online C++ Guide

https://root.cern.ch/doc/master/classTLorentzVector.html

PyROOT provides access to the ROOT C++ objects

```
from ROOT import TLorentzVector

piPlus = TLorentzVector(-0.12, -0.19, 1.68, 1.70)
piMinus = TLorentzVector()
piMinus.SetPxPyPzE(0.56, 0.24, 1.9, 2.1)

twoPions = piPlus + piMinus
print("The 2 pion invariant mass is ", twoPions.Mag(), "GeV/c**2." )
```

```
The 2 pion invariant mass is 1.19 GeV/c**2.
```

Using ROOT , TNtuples, & TFiles

Data Trees & Ntuples

Get src at

http://hadron.physics.fsu.edu/~eugenio/comphy/examples/ntp.py

Create a data table of the function
$f(x) = e^{-x} \sin^2(3x)$
for x between 0 and 5 in steps of 0.01

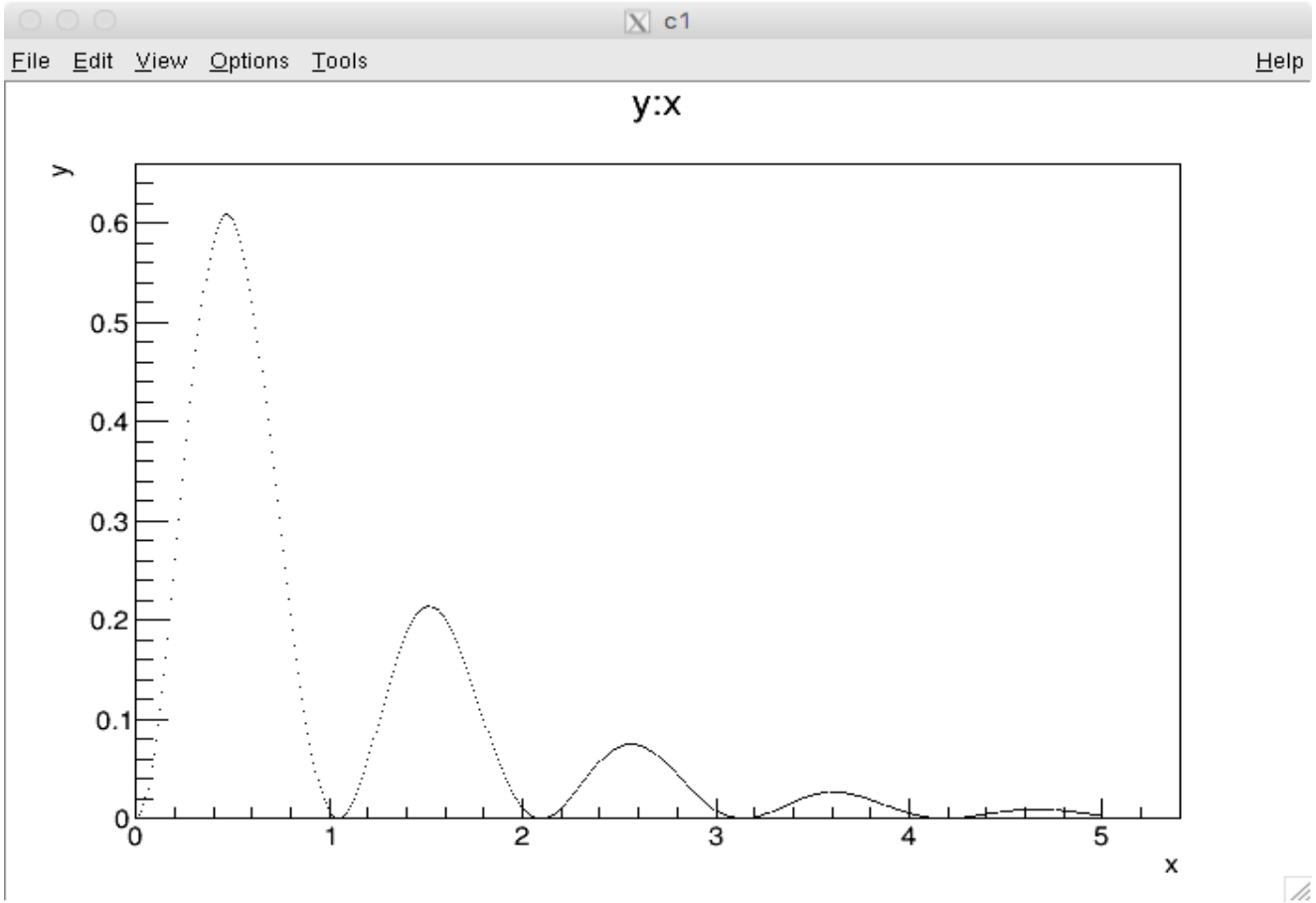
Data Trees & Ntuples

```
""" ....
"""
from __future__ import division, print_function
import numpy as np
import ROOT

def func( aX ):
   """ ... """
   return np.exp( -aX ) * np.sin( 3.0 * aX )**2

xMin = 0.0
xMax =  5.0
deltaX = 0.01

# Create an instance of a TNtuple object
ntuple = ROOT.TNtuple("ntp1", "My 1st Ntuple", "x:y" )

for x in np.arange( xMin, xMax, deltaX ):
   ntuple.Fill( x, func(x) )

# Create a plot of func(x) vs x
ntuple.Draw("y:x")

raw_input("Enter [return] to exit")
```

ntuple is a data container object with name of "ntp1", title of "My 1st Ntuple", and two data elements: x, y

Example: npt.py

```
hpc-login-39 79% ntp.py
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
Enter [return] to exit
```



Saving Root Objects in a ROOT File

Get src at
http://hadron.physics.fsu.edu/~eugenio/comphy/examples/ntpSaved.py

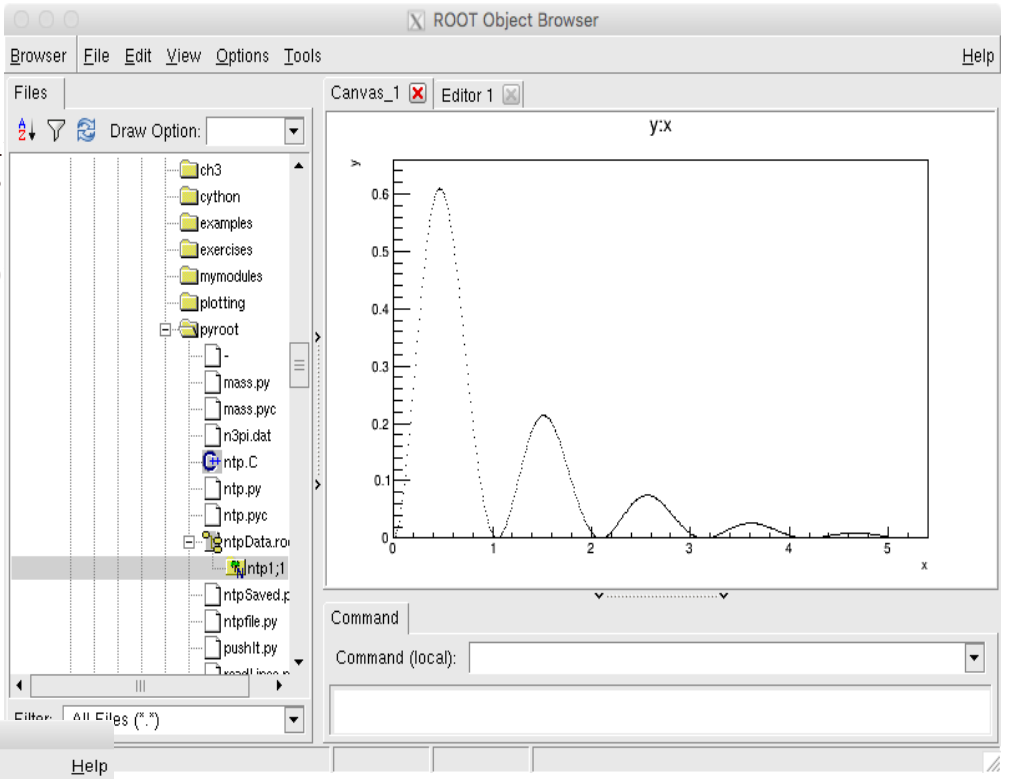
```
#! /usr/bin/env python
"""
 ntpSaved.py: PyROOT example of saving ROOT objects for later use

   . . .
"""
from __future__ import division, print_function
import numpy as np
import ROOT

def func( aX ):
    """ ... """
    return np.exp( -aX ) * np.sin( 3.0 * aX )**2

# open a ROOT TFile
rootFile = ROOT.TFile("ntpData.root","RECREATE")
# Create an instance of a Tntuple object
ntuple = ROOT.TNtuple("ntp1", "My First Ntuple", "x:y" )

xMin = 0.0
xMax =  5.0
deltaX = 0.01


for x in np.arange( xMin, xMax, deltaX ):
    ntuple.Fill( x, func(x) )

# Save the ROOT TFile containing ROOT objects
rootFile.Write()
```

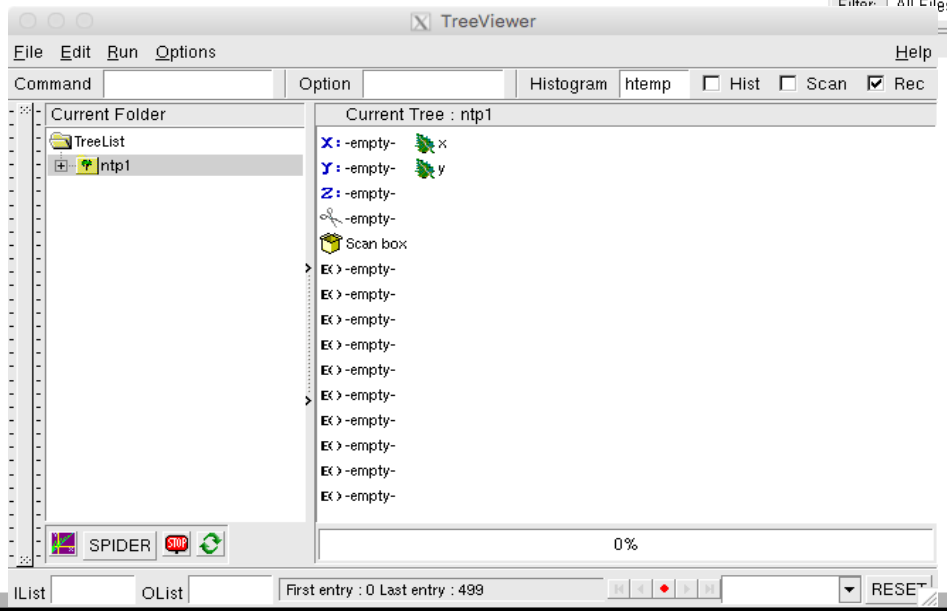
Running PyRoot Interactively via TBrowser

```
hpc-login-39 92% python
Python 2.7.5 (default, Nov 6 2016, 07:46:25)
[GCC 4.8.5 20140120 (Red Hat 4.8.5-11)] on lin
Type "help", "copyright", "credits" or "licens
>>> import ROOT
>>> ROOT.TBrowser()
<ROOT.TBrowser object ("Browser") at 0x282e650
>>> (TFile *) 0x7f9790112240
```

From TBrowser open root file nptData.root

Then right click over "ntp1" and select "StartViewer"



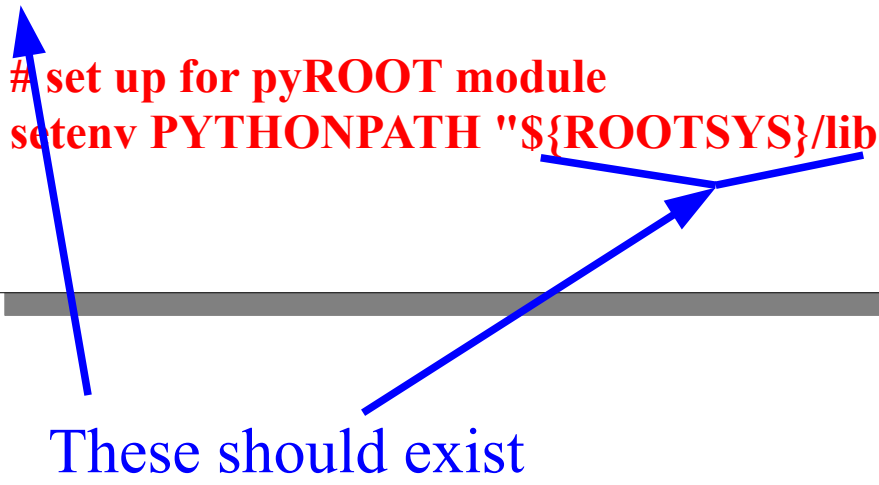


Getting Started

- Environment
  - Edit your "~/.cshrc" file

Add these lines near the end of file:

```
# set up for ROOT
setenv  ROOTSYS   ~eugenio/root
setenv   LD_LIBRARY_PATH     "${LD_LIBRARY_PATH}:$ROOTSYS/lib"
set  path=( $path $ROOTSYS/bin )

# set up for pyROOT module
setenv PYTHONPATH "${ROOTSYS}/lib:${HOME}/python/mymodules:./mymodules"
```



Getting Started

No error messages after delay, then all is working

Open new terminal session and Check python

Open an interactive python session and try

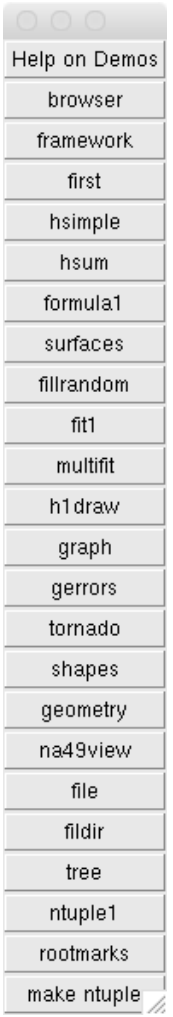
```
hpc-login-25 515% python
>>> import ROOT
>>>
```

Get my examples online and give PyROOT a test run

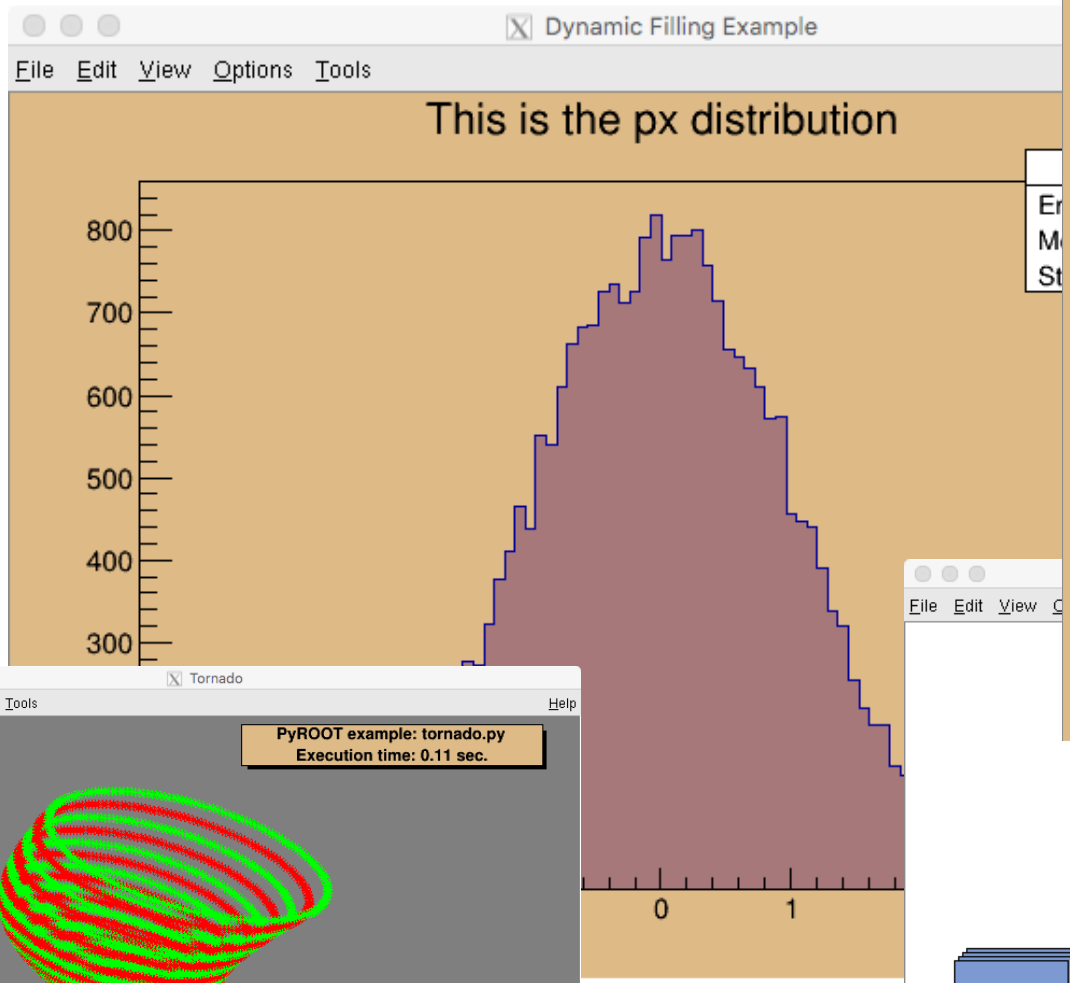
```
75% wget http://hadron.physics.fsu.edu/~eugenio/comphy/examples/ntp.py
76% wget http://hadron.physics.fsu.edu/~eugenio/comphy/examples/ntpSaved.py
77% chmod +x ntp.py ntpSaved.py
78% ntp.py
```

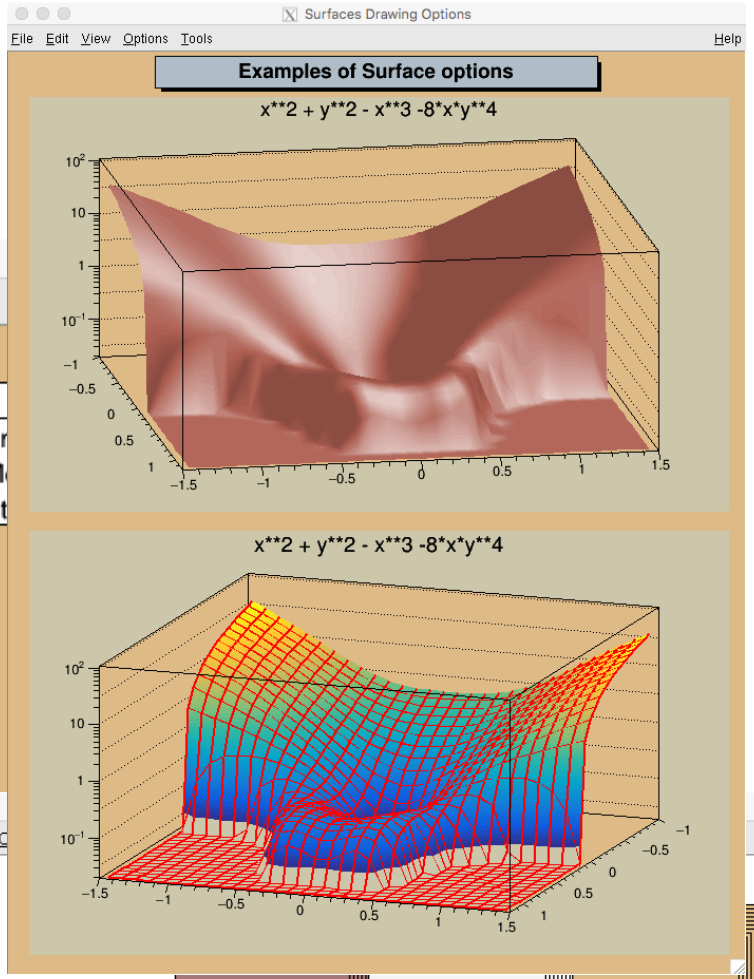
Exit from python and copy over the pyroot examples

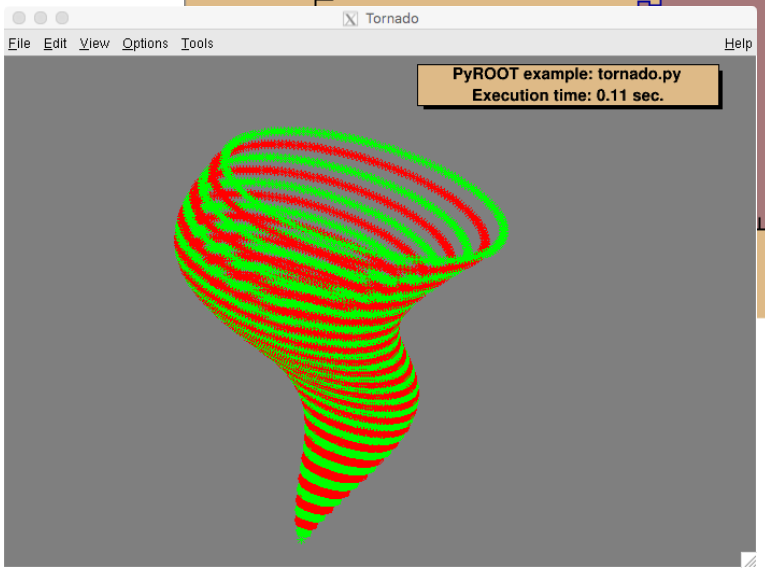
```
physics 71% mkdir pyroot-examples/
physics 72% cd pyroot-examples/
physics 73% cp $ROOTSYS/tutorials/pyroot/* .
physics 74% python demo.py
```

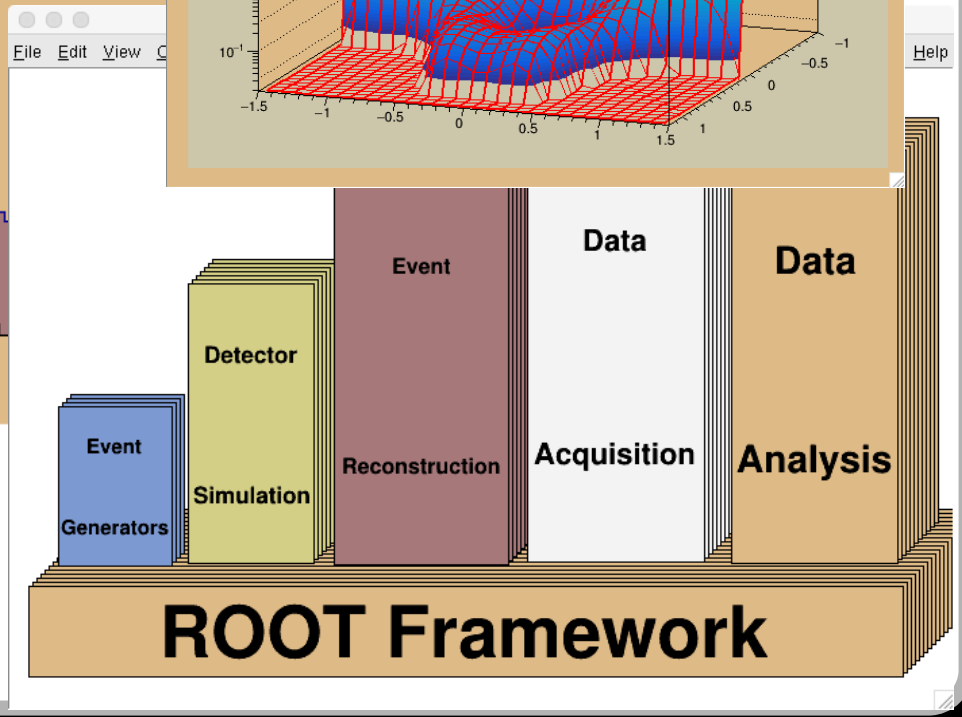


Let's get working









Let's get rolling