Performing Analysis with IPython

Piti Ongmongkolkul & Chih-hsiang Cheng

December 14, 2012

Outline

- What do we in an analysis.
- ROOT and what's missing.
- Python
- IPython Notebook
- Reading RootFile

What do we do in an analysis?

- We read a ROOT file.(At least that's what framework gets us)
- We plot stuff.
- We perform multivariate technique. (Cuts, classifiers etc.)
- We use MINUIT or ROOFIT or your favorite fitting package. To extract observables.

ROOT

- De facto high-energy physics analysis environment. Has been around forever.
- IO (writing reading file). This is done right. I'd say it's the best you can find commercial or free.
- You can Plot stuff.
- Has TMVA. SPR supports ROOT out of the box(ish).
- Written in C++. Fast...(somewhat). You can write C++ and link against it.
- Has interactive environment. The notorious CINT. This will be change Cling soon. But, it will still be a C++ interpreter. TBrowser doesn't help much.

The Language

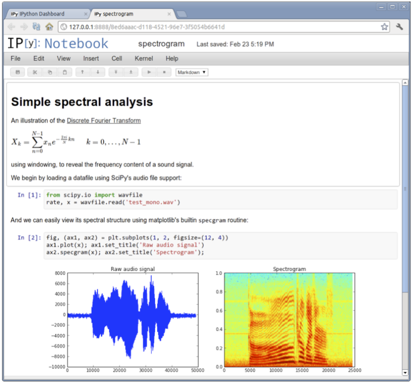
- A lot of problem with ROOT is not really ROOT problem.
- C++ is a very verbose static type language. Good for other things but not a dynamic work like data analysis.
- C++. Repeat yourself like crazy.

  -
    ```
    TFile f("myfile.root");
    TTree* tree = dynamic_cast<TTree*>f.Get("tree");
    float x;
    tree->SetBranchAddress("x",&x);
    tree->GetEntry(10);
    cout << x << endl;
    ```
- Python. root_numpy. https://github.com/rootpy/root_numpy

  -
    ```
    data = root2rec("myfile.root")#treename is optional
    print data.x[10]
    ```
  - There is PyROOT. But it is very slow for doing basic stuff like reading file. We wrote a library to do this as fast as C++. There is also rootpy which use root_numpy as backend.

Interactive Environment

- ROOT interactive environment is not so good for doing analysis. This applied to both new TBrowser and command prompt environment.
- IPython Notebook environment.
- http://ipython.org/
- Mathematica. Maple. Matlab. Sage.
- Type command. See output. Edit command. See output.
- Immediate inline feedback is the key. No separate windows.
- Save it along with output. Come back and view/re-execute later.



Simple spectral analysis

An illustration of the Discrete Fourier Transform

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

using windowing, to reveal the frequency content of a sound signal.

We begin by loading a datafile using SciPy's audio file support:

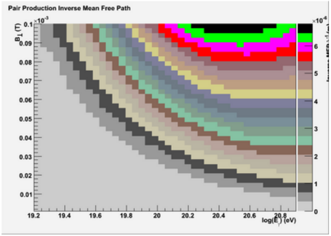
```
In [1]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

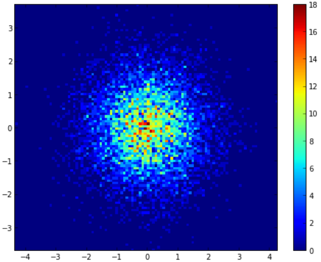
And we can easily view its spectral structure using matplotlib's builtin `specgram` routine:

```
In [2]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

Ugly Plot

- Needs tons of work to make it looks OK. They changed it recently though.
- Gray background by default. How many of you have Gray background in your slides?
- Default color for COLZ.
  - Legend says they are the 16 color supported by color screen back then.
- No transparent color!!
- Matplotlib. Python plotting library. http://matplotlib.org/
- huge Gallery http://matplotlib.org/gallery.html





Horrible Plotting Syntax

- ROOT. Black magic.

```
tree->Draw("x");
TH1F *xhist = (TH1F*)gPad->GetPrimitive("htemp");
htemp->SetLineColor(kRed);
tree->Draw("y>>anotherhist","same");
TH1F *yhist = (TH1F*)gPad->GetPrimitive("anotherhist");
yhist->SetLineColor(kBlue);
htemp->SetTitle("Magic!!!");
Legend* leg = new TLegend(0.1,0.7,0.48,0.9);
leg->SetHeader("The Legend Title");
leg->AddEntry(xhist,"x");
leg->AddEntry(yhist,"y");
leg->Draw();
```

- Matplotlib. Much more intuitive.

```
hist(tree.x, label="x", color="red", hist_type="step")
hist(tree.y, label="y", color="blue", hist_type="step")
title("That is the way it should be")
legend(loc="upper right") #yep that simple.
```

Multivariate Analysis and Fitting

- Python has tons of packages to do multivariate analysis.
  - Most popular one is scikit-learn http://scikit-learn.org/
  - A Bunch of neural network library too.
- Fitting takes advantage of python introspection. It automatically recognize function argument name. No need to repeat yourself.

```python
def f(x,y,z):
    return (x-2)**2+(y-3)**2+(z-4)**2
m = Minuit(f)
m.migrad()
print m.values #{"x":2.,"y":3.,"z":4.}
```

  - https://github.com/piti118/RTMinuit
  - https://github.com/piti118/dist_fit

```python
lh = BinLH(pdf,data)#automatically read pdf arguments
m = Minuit(lh)
m.migrad()
```

Tutorial

Let’s see how we can use all these to create a better workflow.