Improving Analysis Workflow with IPython

Piti Ongmongkolkul & Chih-hsiang Cheng

December 15, 2012

Advertisement

Time and Location:

- Setup Session. 1 hour TBD TBA.
- Tutorial. 3 hour TBD TBA.

Prepare your laptop in advance

Visit this url or instruction. You are recommended to try to prepare your laptop in advance of the setup session. If there is any problem, just come to Setup Session. We will try our best to help.

Can't attend the tutorial?

The material on the website is design so that you learn it by yourself. They are all downloadable.

Analysis Workflow.

Typical Workflow.

- 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.

There were a lot of missing pieces here and there when we began. We looked at what we need and implemented those missing pieces. We think we got most of them.

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 one of 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.

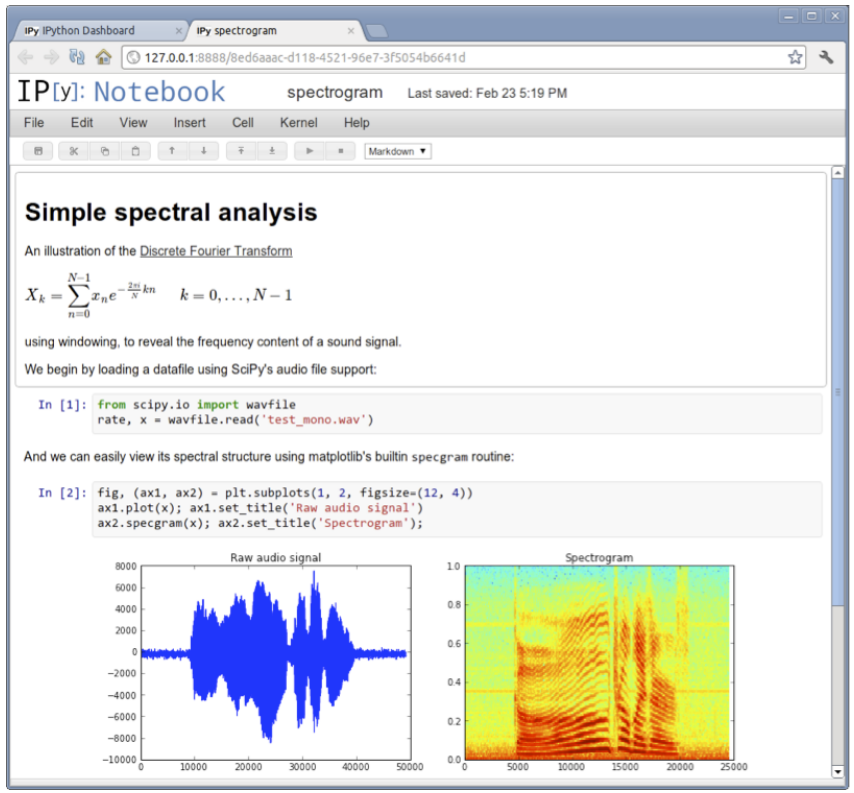
What's better about Python etc.?

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++. Static typing. 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. root_numpy is as fast as C++. There is also rootpy which use root_numpy as backend.

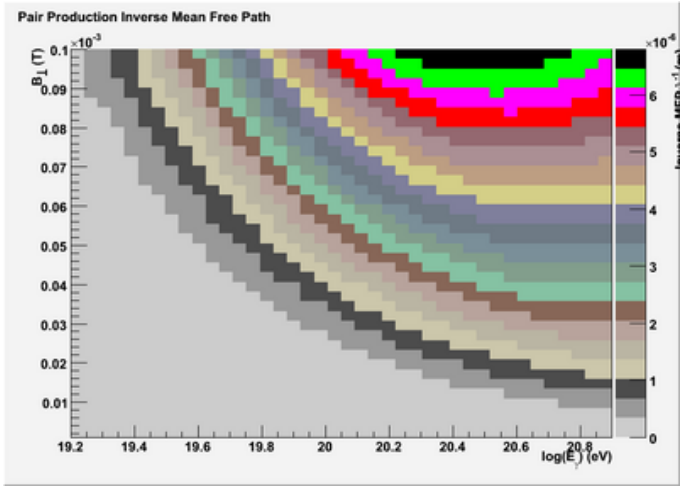
Interactive Environment

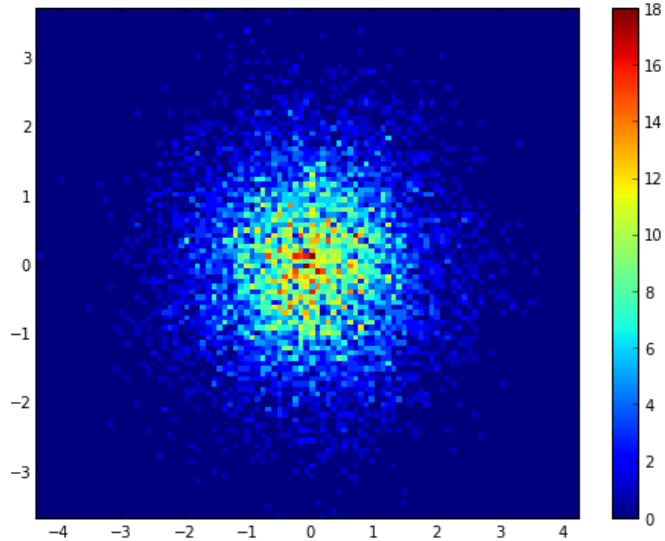
- ROOT interactive environment is not so good for doing analysis. This applied to both new TBrowser and command prompt environment.
- IPython Notebook environment.
- http://ipython.org/
- Mathematica. Maple. Matlab. Sage.
- Type command. See output. Edit command. See output.
- Immediate inline feedback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



Plots looks nice by default

- Needs tons of work to make it looks OK. They changed it recently though.
- Gray background by default. Why? Really?
- 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





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 automagically recognizes argument names as parameters. No need to repeat yourself.

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

- Minuit and Likelihood/Chi2 construction
  - https://github.com/piti118/RTMinuit
  - https://github.com/piti118/dist_fit

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