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 encouraged to try to prepare your laptop before 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 materials on the website is designed so that you learn it by yourself. They are all downloadable.

Analysis Workflow.

Typical Workflow.

- Read ROOT files.(At least that's what framework gets us)
- Plot stuff.
- Multivariate analysis. Cuts, classifiers etc.
- Fit. MINUIT ,ROOFIT or your favorite fitting package.

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.

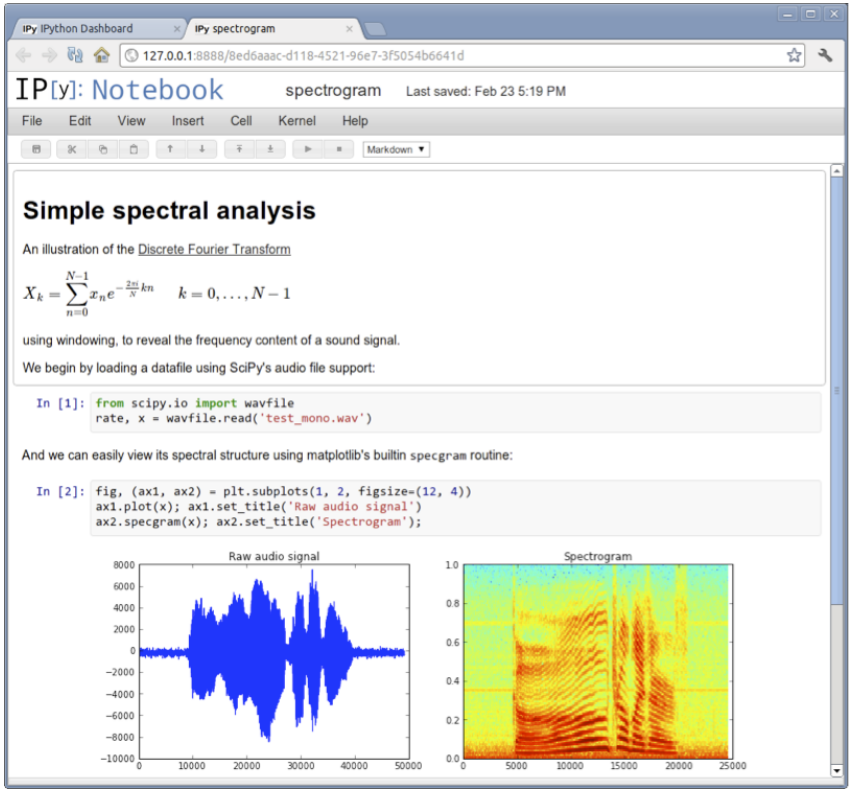
What's better about Python etc.?

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. 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.
- Autocomplete. Docstring. IPython magic.
- inumpy
  - https://github.com/piti118/inumpy



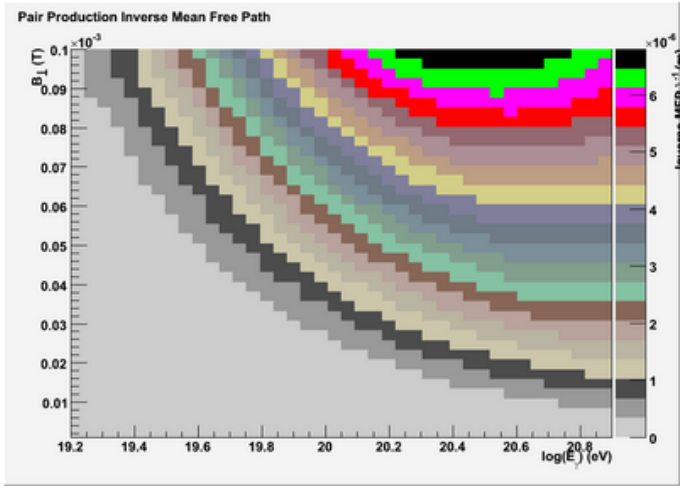

```
mc = root2rec('SP1005_Run1.root','cand')
```

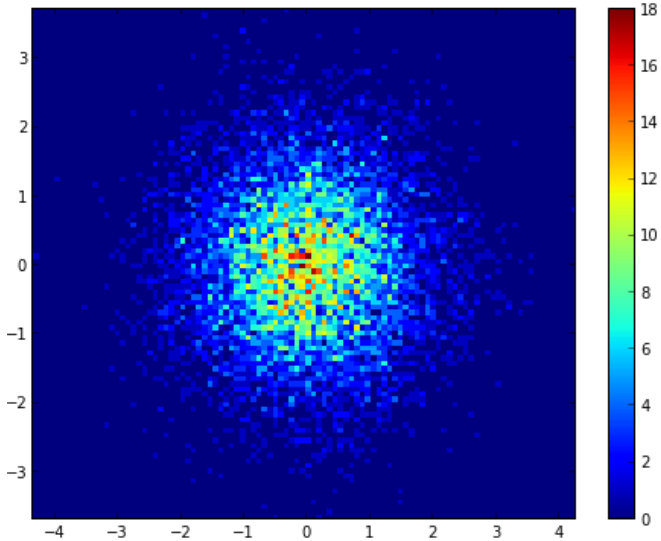
```
mc.
```

```
mc.Aplanarity
mc.BAbsCosGammaRoeThrust
mc.BAbsCosThetaBStar
mc.BAbsCosThetaCMPV
mc.BAbsCosThetaPrime
mc.BAbsCosThetaTStar
mc.BBU_mb465_mupi20_mug27_weight
mc.BCosGammaRoeThrust
mc.BCosThetaBStar
mc.BCosThetaCMPV
```

Plots looks nice by default

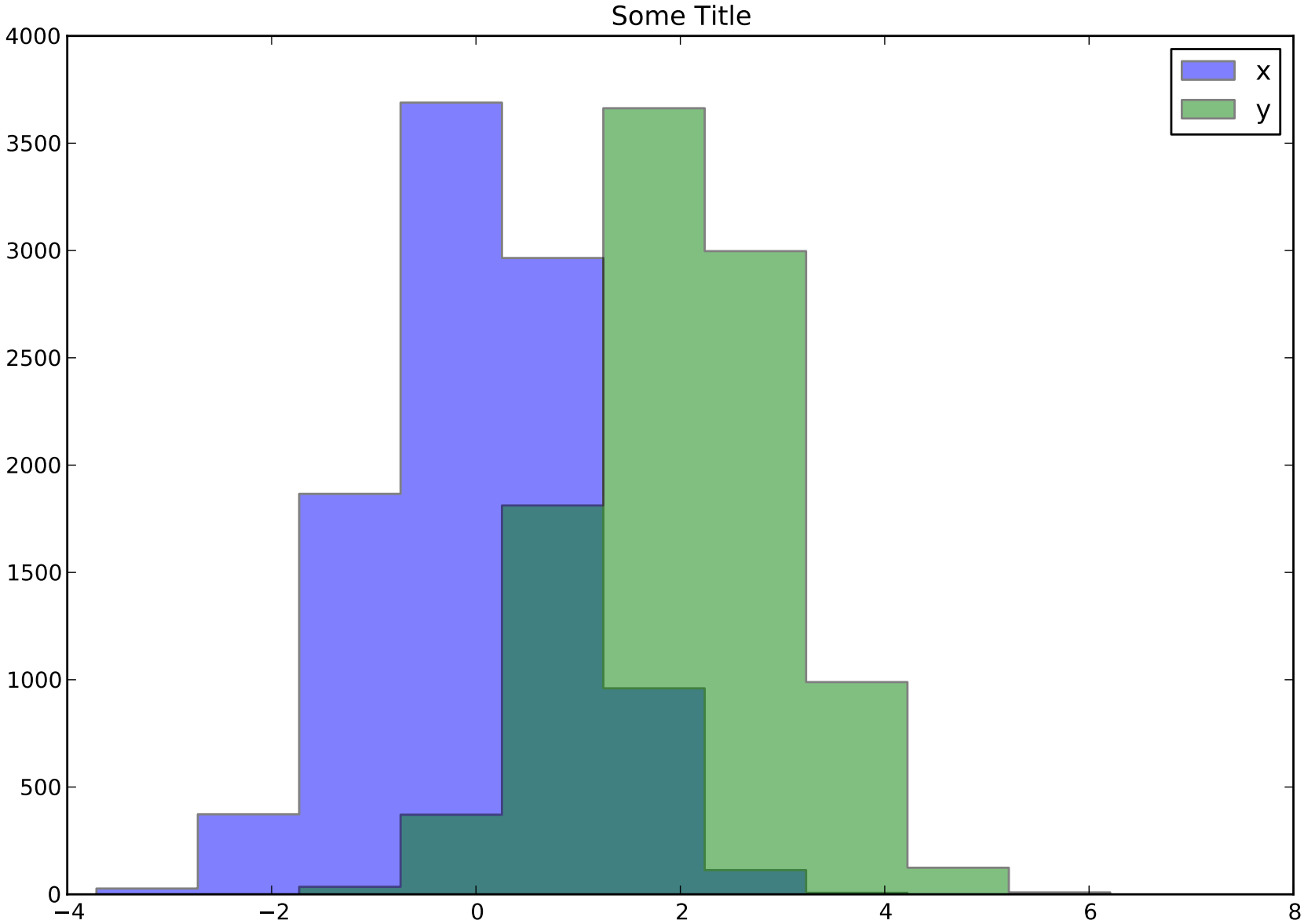
- Needs tons of work to make ROOT plot looks OK. They changed it recently though.
- Gray background by default. Why? Really?
- Default color for COLZ.
  - Legend says they are the 16 color supported by color screen back then.
- No transparent color!!
- Matplotlib. Python plotting library. http://matplotlib.org/
- Huge Gallery http://matplotlib.org/gallery.html
- Extensive documentation.





Plotting syntax

Let's try to make a simple plot



Plotting Syntax

ROOT. Black magic.

```
tree->Draw("x");
TH1F *xhist = (TH1F*)gPad->GetPrimitive("htemp");
htemp->SetLineColor(kRed);
tree->Draw("y>>h2","same");
TH1F *yhist = (TH1F*)gPad->GetPrimitive("h2");
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. Named argument.

```
hist([x,y], histtype='stepfilled', label=['x','y'],
            color=['blue','green'], alpha=0.5)
title("Some Title")
legend() #yep that simple.
```

Bonus

```
In [5]: x = randn(10000)
        y = randn(10000)+2
        figure(figsize=(10,7))
        hist(x, histtype='stepfilled', label='x',alpha=0.5);
        hist(y, histtype='stepfilled', label='y',alpha=0.5);
        hist(x, bins=10, range=None, normed=False, weights=None, cumulative=False,
        bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=None,
        log=False, color=None, label=None, hold=None, **kwargs)
        Call signature::
        (array([  18,  128,  526, 1465, 2568, 2688, 1721,  686,  163,   37]), array([-3.63379696,
        -1.49456578, -0.78148872,
```

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/χ^2 construction. With introspection and much more.
  - https://github.com/piti118/RTMinuit
  - https://github.com/piti118/dist_fit

```
def pdf(x, mu, sigma, alpha):
    return complicated_function(x,mu,sigma,alpha)
lh = BinLH(pdf,data)#knows about mu, sigma, alpha
m = Minuit(lh)
m.migrad()
```