Driving Science Tools with python

David Sanchez*

19th March 2008

Python is a powerfull tool to drive the Science Tools (ST), get results and draw plot. We give some informations to drive ST and get results. Not all functions are given here but this is enough to begin.

Using Ftools

In this section, we give some instructions to drive the Science Tools with python, in particular to make an unbinned analysis. In general, you have to provide the same information as in command line. After, you can run the tool. For an explanation of the tools, see the workbook (*http://glast-ground.slac.stanford.edu/workbook/)*.

Selection of events *gtselect* In order to configure gtselect, the following lines are mandatory :

```
filter['infile'] = FT1_file
filter['outfile'] = Filtered_eventfile
filter['ra'] = ra
filter['dec'] = dec
filter['rad'] = radius
filter['emin'] = Emin
filter['emax'] = Emax
filter['tmin'] = tstart_run
filter['tmax'] = tend_run
filter['eventClass'] = -1
```

FT1_file, Filtered_eventfile are string and the others are number. You can also add optional configurations. The option's names are the same as in the Pfile (see the workbook). To run gtselect, type the command :

```
filter.run()
```

Diffuse responses *gtdiffrsp* The configuration of gtdiffresps use the same schema:

*David Sanchez IN2P3/CNRS, dsanchez@llr.in2p3.fr

```
diffResps['evfile'] = Filtered_eventfile
diffResps['scfile' ]= FT2_file
diffResps['srcmdl'] = xml_src_model_file
diffResps['irfs'] = irfs
```

All variables are string. To run the tool :

```
diffResps.run()
```

Exposure cube : *gtltcube* Configuration of gtltcube :

```
expCube['evfile'] = Filtered_eventfile
expCube['scfile'] = FT2_file
expCube['outfile'] = file_expCube
expCube['dcostheta'] = 0.025
expCube['binsize'] = 1
```

All variables are string. To run the tool :

```
expCube.run()
```

Exposure map : *gtexpmap* Configuration of gtexpmap :

```
expMap['evfile'] = Filtered_eventfile
expMap['scfile'] = FT2_file
expMap['expcube'] = Cube_file
expMap['outfile'] = expMap_file
expMap['irfs'] = irfs
expMap['srcrad'] = 10+radius
```

All variables are string except radius. To run the tool

```
expMap.run()
```

Fitting data *gtlike* The following function allows to use gtlike in a smart way :

```
def runLike(evfile, scfile, expMap, expCube, srcModel, outfile):

    obs = UnbinnedObs(evfile, scfile, expMap=expMap, expCube=expCube, irfs=irfs)
    foo = UnbinnedAnalysis(obs, srcModel=srcModel, optimizer='MINUIT')
    foo.ftol=1e-7
    try:
      foo.fit(covar = True)
    except:
      foo.fit(covar = True)
    foo.logLike.writeXml(outfile)
    return foo
```

It creates a python object from the class UnbinnedObs and a other from the class UnbinnedAnalysis. The command 'foo.fit' calls the optimizer. *runLike* returns the UnbinnedAnalysis object. To call this function, the command is :

```
fit = runLike(Filtered_eventfile, FT2_file, expMap_file, Cube_file,xml_src_model_file
```

Getting results

Plotting results, computing TS value If *fit* is a python object given by the function *runlike*, to plot the the model and residual, you have to enter :

```
fit.plot()
```

The TS value is given by :

```
TS=fit.Ts(srcname)
```

Getting results for power law and broken power law First we define a power law by :

$$\phi = N0 \left(\frac{E}{E_0} \right)^{-\gamma}$$

and a broken power law by :

if $E < E_{break}$:

$$\phi = N0 \left(\frac{E}{E_{break}} \right)^{-\gamma_1}$$

if $E > E_{break}$:

$$\phi = N0 \left(\frac{E}{E_{break}} \right)^{-\gamma_2}$$

To get the value and the error of the Prefactor (Hereafter srcname is the name of the source which we are interested in):

```
scale = fit[srcname].funcs['Spectrum'].getParam('Prefactor').getScale()
N0 = fit[srcname].funcs['Spectrum'].getParam('Prefactor').value()
error_N0_mev=fit[srcname].funcs['Spectrum'].getParam('Prefactor').error()
```

For a power law, the other parameters are E0 and Gamma :

```
E0=fit[srcname].funcs['Spectrum'].getParam('Scale').value()

#~~~~~~~~~~~~~~~#
Gamma=-fit[srcname].funcs['Spectrum'].getParam('Index').value()
error_Gamma=fit[srcname].funcs['Spectrum'].getParam('Index').error()
```

For a broken power law, we have to get the value and error of Ebreak, Gamma1, Gamma2

```
Ebreak=fit[srcname].funcs['Spectrum'].getParam('BreakValue').value()
error_Ebreak=fit[srcname].funcs['Spectrum'].getParam('BreakValue').error()

#~~~~~~~~~~~~~~~#
Gamma1=-fit[srcname].funcs['Spectrum'].getParam('Index1').value()
```

```
err_Gamma1=fit[srcname].funcs['Spectrum'].getParam('Index1').error()

#~~~~~~~~~~~~~~~#
Gamma2=-fit[srcname].funcs['Spectrum'].getParam('Index2').value()
err_Gamma2=fit[srcname].funcs['Spectrum'].getParam('Index2').error()
```

Usefull functions The covariance matrix is computed during the fit and you can access it by :

```
covar = fit.covariance
```

To get the name of the parameter number i:

```
fit[i].getName()
```

To get the name of the source j :

```
fit[j].srcName
```