Python Analysis / LATAnalysisScripts / Lightcurves / Fitting Issues

Jeremy S. Perkins, FSSC
Fermi Summer School 2012

You probably should follow along with the data I'm using today and go back to your ROI after the tutorial.

Introduction

- Python is a programming language similar to Tcl, Perl, Ruby, Scheme or Java
    - Very clear, readable syntax
    - Embeddable within apps as a scripting interface
    - http://wiki.python.org/moin/BeginnersGuide
    - http://python4astronomers.github.com/
- All of the Science Tools are included as python modules with some additional capabilities like
    - Upper limits
    - Direct access to results (and data)

Quick Usage:

```
[user@localhost ~]$ python
Python 2.7.2 (default, Apr 10 2012, 10:33:45)
[GCC 4.2.1 (Apple Inc. build 5666)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import pyLikelihood
>>> from UnbinnedAnalysis import *
>>> help(UnbinnedAnalysis)
```

Check out the python tutorial for examples:
http://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/python_tutorial.html

We're going to use the *LATAnalysisScripts* today instead which is are modules built on top of the python interface.

Philosophy and Why Should You Care

- LATAnalysisScripts is a set of three python libraries to make the analysis of LAT data straightforward and powerful.
- Uses standard python tools to make installation and use easy and intuitive.
- Vetted by the FSSC so should be ‘official’
- Checks for many common errors
- Easy to document, easy to use, easy to repeat
- Documentation!
- Exposes special features.

LATAnalysisScripts Structure

- Three 'independent' libraries (quickAnalysis, quickLike and quickPlot).
- A single config file
- A common naming convention for files.
- Detailed, clear error reporting
- Detailed, clear logging of results and commands.

Download and Install

- Download and install pyds9

```
[user@localhost ~]$ wget http://hea-www.harvard.edu/saord/download/ds9/
python/pyds9-1.3.tar.gz
[user@localhost ~]$ tar zxvf pyds9-1.3.tar.gz
[user@localhost ~]$ cd pyds9-1.3
[user@localhost ~]$ python setup.py install
```

- Copy make2FGLxml into the python tree

```
[user@localhost ~]$ cp make2FGLxml.py $FERMI_DIR/lib/python2.7/site-
packages/
```

- Unpack and intall LATAnalysisScripts

```
[user@localhost ~]$ tar zxvf LATAnalysisScripts-1.9.tar.gz
[user@localhost ~]$ cd LATAnalysisScripts-1.9
[user@localhost ~]$ python setup.py install
```

Our Simple Analysis

- We're going to derive an upper limit on Swift J164449.3+573451 during a short time period.
  - Search Center (RA,Dec) = (251.2054,+57.5808)
  - Radius = 30 degrees
  - Start Time (MET) = 322963202 seconds (2011-03-28T00:00:00)
  - Stop Time (MET) = 323568002 seconds (2011-04-04T00:00:00)
  - Minimum Energy = 100 MeV
  - Maximum Energy = 300000 MeV

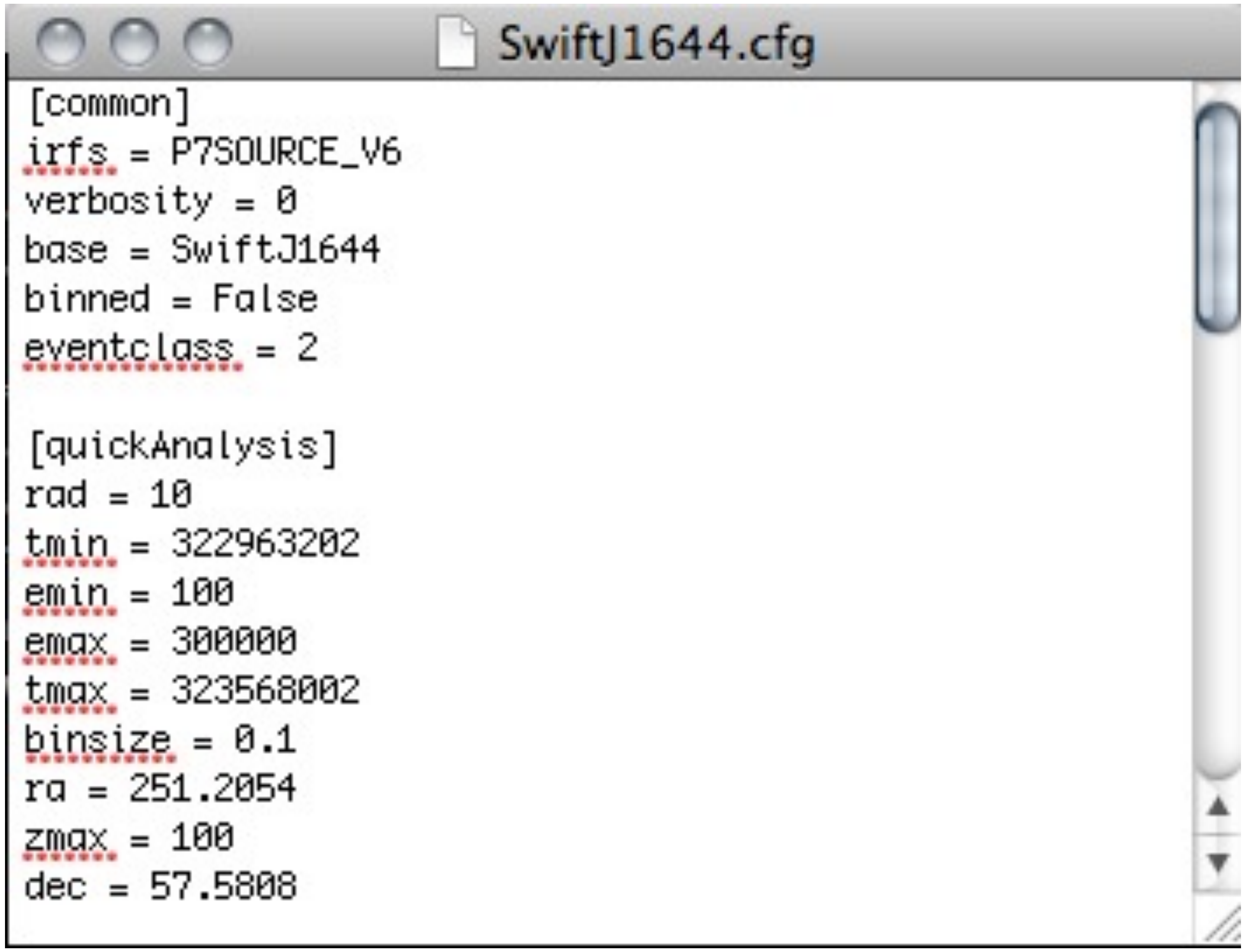
http://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/data/LATAnalysisScripts/L120404155224B0489E7F72_PH00.fits
http://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/data/LATAnalysisScripts/L120404155224B0489E7F72_SC00.fits

Initialize the analysis

```
[user@localhost]$ quickAnalysis -i
Creating example configuration file called example.cfg
2012-02-01 11:56:00,717 - quickAnalysis - INFO - Created quickAnalysis
object: irfs=P7SOURCE_V6, verbosity=0, binned=False, base=example,
eventclass=2, rad=10, tmin=INDEF, ra=0, emin=100, emax=300000, tmax=INDEF,
dec=0, zmax=100, binsize=0.1,
2012-02-01 11:56:00,718 - quickAnalysis - INFO - wrote common config to
example.cfg.
2012-02-01 11:56:00,718 - quickAnalysis - INFO - wrote quickAnalysis
config to example.cfg.
[user@localhost]$ cp example.cfg SwiftJ1644.cfg
```

Now open 'SwiftJ1644.cfg' in a text editor.

SwiftJ1644.cfg



```
[common]
irfs = P7SOURCE_V6
verbosity = 0
base = SwiftJ1644
binned = False
eventclass = 2

[quickAnalysis]
rad = 10
tmin = 322963202
emin = 100
emax = 300000
tmax = 323568002
binsize = 0.1
ra = 251.2054
zmax = 100
dec = 57.5808
```

Common Section

quickAnalysis Section

Some housekeeping.

```
[user@localhost]$ ls -1 *PH* > SwiftJ1644.list
[user@localhost]$ ln -s L120404155224B0489E7F72_SC00.fits SwiftJ1644_SC.fits
```

Now run...

```
[user@localhost]$ quickAnalysis -a -n SwiftJ1644
...output suppressed.
```

This does everything up to the likelihood calculation and stores all of the output in a log file called 'SwiftJ1644_quickAnalysis.log'.

Create a Model File

```
[user@localhost]$ quickAnalysis -x -n SwiftJ1644
Creating XML model file from 2FGL
2012-02-01 13:48:49,305 - quickAnalysis - INFO - Reading from config file
(SwiftJ1644.cfg)
2012-02-01 13:48:49,306 - quickAnalysis - INFO - Reading common variables...
2012-02-01 13:48:49,306 - quickAnalysis - INFO - Reading quickAnalysis
variables...
2012-02-01 13:48:49,307 - quickAnalysis - INFO - Created quickAnalysis object:
irfs=P7SOURCE_V6, verbosity=0, base=SwiftJ1644, binned=False, eventclass=2,
rad=20, tmin=322963202, emin=100, emax=300000, tmax=323568002, binsize=0.1,
ra=251.2054, zmax=100, dec=57.5808,
2012-02-01 13:48:49,307 - quickAnalysis - CRITICAL - SwiftJ1644_model.xml doesn't
exist.
2012-02-01 13:48:49,307 - quickAnalysis - INFO - SwiftJ1644_model.xml doesn't
exist, will create a new one.
This is make2FGLxml version 04r1.
For use with the gll_psc_v02.fit and gll_psc_v05.fit and later LAT catalog files.
Creating file and adding sources for 2FGL
Added 33 point sources and 0 extended sources
2012-02-01 13:48:50,298 - quickAnalysis - INFO - NOTE: if there are extended
sources in your ROI, make sure the correspoinding diffuse template is in the
working directory.
```

This model doesn’t include our source of interest. Add the following to the top of the SwiftJ1644_model.xml file

```
<source name="SwiftJ1644" type="PointSource">
 <spectrum type="PowerLaw2">
  <parameter free="true" max="10000.0" min="0.0001" name="Integral" scale="1e-07"
value="1.0"/>
  <parameter free="true" max="5.0" min="0.0" name="Index" scale="-1.0" value="2.0"/>
  <parameter free="false" max="500000.0" min="20.0" name="LowerLimit" scale="1.0"
value="100.0"/>
  <parameter free="false" max="500000.0" min="20.0" name="UpperLimit" scale="1.0"
value="300000.0"/>
 </spectrum>
 <spatialModel type="SkyDirFunction">
  <parameter free="false" max="360.0" min="-360.0" name="RA" scale="1.0" value="251.2054"/>
  <parameter free="false" max="90.0" min="-90.0" name="DEC" scale="1.0" value="57.5808"/>
 </spatialModel>
</source>
```

Edit your ‘SwiftJ1644.cfg’ file to include the quickLike configuration section (you can also run ‘quickLike -i’ to generate and example).

```
[quickLike]
sourcename = SwiftJ1644
model = SwiftJ1644_model.xml
drmtol = 0.1
mintol = 0.0001
```

Start up python and setup the likelihood session.

```
[user@localhost]$ python
Python 2.6.5 (r265:79063, Nov 9 2010, 12:49:33)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>from quickLike import *
>>>qL = quickLike('SwiftJ1644', True)
...output supressed
>>>qL.makeObs()
...output supressed
>>>qL.initMIN(modelFile="SwiftJ1644_model.xml")
...output supressed
```

Unload our source of interest and do the fit without it first.

```
>>>qL.unLoadSource('SwiftJ1644')
2012-02-02 12:50:51,589 - quickLike - INFO - Removed SwiftJ1644 from the model
and saved it.
>>>qL.fitMIN()
...some output suppressed
2012-02-02 12:55:08,062 - quickLike - INFO - NEWMINUIT Fit Finished. Total TS:
-11469.3411544
2012-02-02 12:55:08,063 - quickLike - INFO - NEWMINUIT Fit Status: 156
2012-02-02 12:55:08,063 - quickLike - INFO - NEWMINUIT fit Distance:
1.35106292821e-05
2012-02-02 12:55:08,063 - quickLike - ERROR - NEWMINUIT DID NOT CONVERGE!!!
2012-02-02 12:55:08,063 - quickLike - ERROR - The fit failed the following tests:
HasMadePosDefCovar HasPosDefCovar HasAccurateCovar
...some output suppressed
```

Note that our fit didn’t converge...

Let’s see if we can figure out why...

Check for parameters at their limits

```
>>>qL.paramsAtLimit()
2012-02-02 15:51:32,050 - quickLike - ERROR - The Prefactor (0.000100002150128)
of _2FGLJ1531.0+5725 is close (2.1501276315e-05) to its lower limit (0.0001)
2012-02-02 15:51:32,062 - quickLike - ERROR - The Index (4.99999994135) of
_2FGLJ1604.6+5710 is close (1.17301164337e-08) to its upper limit (5.0)
2012-02-02 15:51:32,063 - quickLike - ERROR - The Index (4.92274288596) of
_2FGLJ1656.5+6012 is close (0.0154514228086) to its upper limit (5.0)
2012-02-02 15:51:32,065 - quickLike - ERROR - The Index (4.98785968966) of
_2FGLJ1559.0+5627 is close (0.00242806206775) to its upper limit (5.0)
```

Reload our source and check for weak sources.

```
>>>qL.reLoadSource()
2012-02-02 15:52:42,865 - quickLike - INFO - Reloaded saved source.
>>>qL.removeWeak()
...output supressed
```

Remove all the free sources with TS < 1.0

```
>>>qL.removeWeak(tslimit=1.0, RemoveFree=True)
...output supressed
```

Unload our source again and re-fit.

```
>>>qL.reLoadSource()
>>>qL.removeWeak(tslimit=0.0, RemoveFree=True, RemoveFixed=True)
>>>qL.unLoadSource('SwiftJ1644')
2012-02-02 17:26:51,589 - quickLike - INFO - Removed SwiftJ1644 from the model and
saved it.
>>>qL.fitMIN()
2012-02-02 17:27:01,039 - quickLike - INFO - NEWMINUIT Fit Finished. Total TS:
-11470.0702022
2012-02-02 17:27:01,039 - quickLike - INFO - NEWMINUIT Fit Status: 0
2012-02-02 17:27:01,039 - quickLike - INFO - NEWMINUIT fit Distance: 3.52763544413e-06
```

It converged, but let's double check for problems.

```
>>>qL.paramsAtLimit()
2012-02-02 17:29:45,456 - quickLike - ERROR - The Index (4.99999570933) of
_2FGLJ1559.0+5627 is close (8.58133958737e-07) to its upper limit (5.0)
```

Looks like we're having trouble fitting this parameter. Let's try fixing it to the catalog value.

Copy the output of our fit to a new file.

```
>>>exit()
[user@localhost]$ cp SwiftJ1644_likeMinuit.xml SwiftJ1644_minimal.xml
```

Open this new file in a text editor and set the index of 2FGLJ1559.0+5627 to it's catalog value (2.09846) and set it to fixed (free="0").

Also, make sure that SwiftJ1644 is there somewhere.

Let's do this again with our new model.

```
[user@localhost]$ python
Python 2.6.5 (r265:79063, Nov 9 2010, 12:49:33)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>from quickLike import *
>>>qL = quickLike('SwiftJ1644', True)
...output supressed
>>>qL.makeObs()
...output supressed
>>>qL.initMIN(modelFile="SwiftJ1644_minimal.xml")
...output supressed
>>>qL.unLoadSource('SwiftJ1644')
2012-02-02 17:38:14,752 - quickLike - INFO - Removed SwiftJ1644 from the model
and saved it.
>>>qL.fitMIN()
...output supressed
2012-02-02 17:39:57,667 - quickLike - INFO - NEWMINUIT Fit Finished. Total TS:
-11470.7003776
2012-02-02 17:39:57,668 - quickLike - INFO - NEWMINUIT Fit Status: 0
2012-02-02 17:39:57,668 - quickLike - INFO - NEWMINUIT fit Distance:
3.4341373915e-05
>>>qL.paramsAtLimit()
```

It converges and the parameters are ok.

Calculate the upper limit.

```
>>>qL.reLoadSource()
2012-02-09 17:41:08,647 - quickLike - INFO - Reloaded saved source.
>>>qL.calcUpper('SwiftJ1644', Emax=10000)
...output supressed
2012-02-02 17:45:50,554 - quickLike - INFO - SwiftJ1644 UL:
3.48e-07 ph/cm^2/s for emin=100.0, emax=10000.0, delta(logLike)
=1.35
```

Useful Functions

(more details via the help function)

- calcUpper: calculates an upper limit
- customERange: changes the range of the fit
- paramsAtLimit: prints out any parameters that are at their limits
- printSource: prints the details of a specific source
- removeWeak: removes weak sources
- unLoadSource: removes a source but saves it
- reLoadSource: restores the source that was unloaded

Also lots in python...

- Check out our documentation.
- Use the help function.
- Ask the help desk.
- Use the LATAnalysisScripts as an example
    - You could even contribute! The scripts are on github and you can add features or fix bugs.

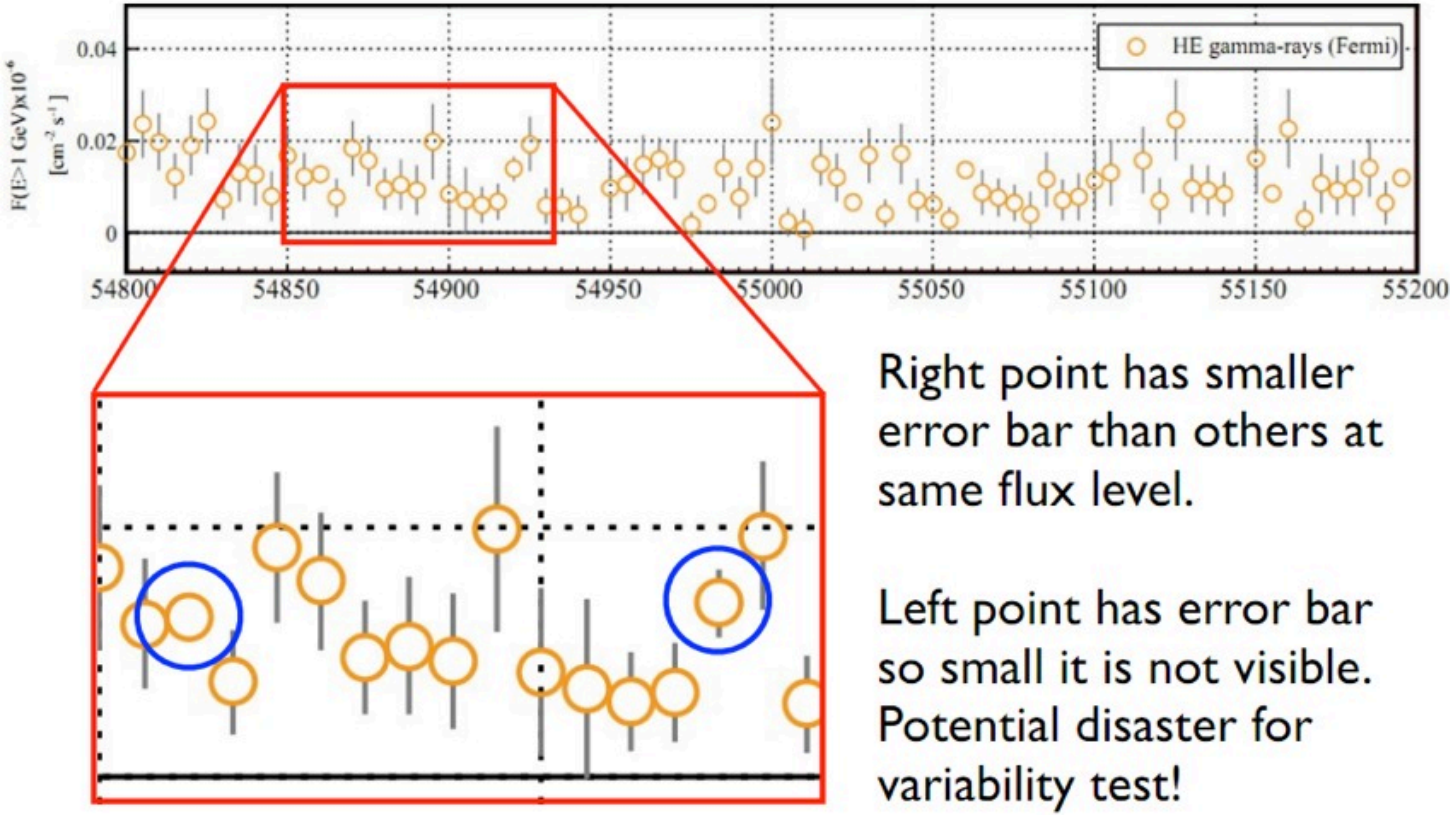
Notes on Light Curves

- Types of Light Curves:
  - Regular Binning with likelihood
  - Adaptive Binning with likelihood
  - Aperture photometry

Lightcurve How To

- Perform a standard analysis over the full energy range ('DC' Analysis).
- Determine your binning (~$TS_{DC}/25$).
- Prepare a model file:
  - Freeze all spectral shapes of background sources
  - Freeze all parameters of weak sources (~$TS_{DC}/N_{bin} < 4$ or 9)
  - Freeze the spectral shape of your SoI?
- Decide when to show upper limits vs. flux points
- Use gtselect (or the gt_app 'filter' in python) to bin the data and run the likelihood method.
- Check for problems (see next slide, pertains to DC analysis as well).
- Compute upper limits if needed.

Gotchas



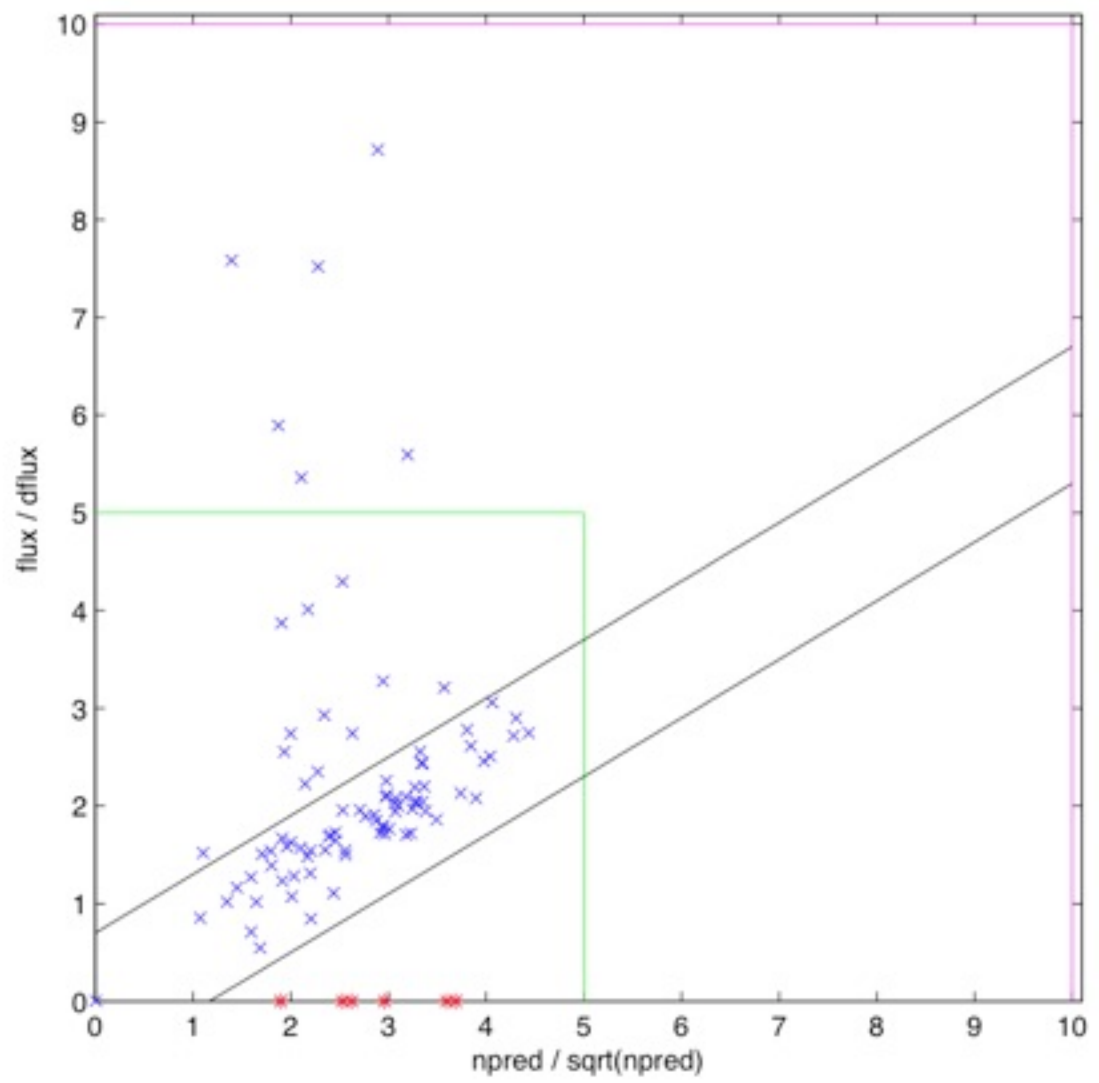
Right point has smaller error bar than others at same flux level.

Left point has error bar so small it is not visible. Potential disaster for variability test!

Common problem is that the fit doesn't converge and the error matrix isn't calculated. Then the error bars are wrong and any modeling based on those data will be wrong.

How to check

$\mathrm{Flux_{err}}$/Flux vs. sqrt(Npred)/Npred



Also, look for convergence.

- When using the MINUIT optimizer look for output that says "ERR MATRIX NOT POS-DEF".
- Get the return code from MINUIT/NEW-MINUIT (look at the LATAnalysisScripts for how to do this).

How to Fix (pertains to DC analysis as well)

- Problem is usually a parameter at the limit
  - Usually spectral shapes - just freeze them.
  - Weak background sources - freeze them or remove them.
- Try increasing the overall tolerance
  - In python do '>>> like.tol = 1e-8'
  - In gtlike set 'ftol = 1e-8' on command line
- Simplify, Simplify, Simplify.
  - More parameters means more complexity. Freeze parameters from the outside in.

Break...

- What do you want to do? Some suggestions:
  - Work up a quick loop to generate a light curve of the 3C 279 data.
  - Use python to refine the model of your favorite source.
  - Look in the user contributed tools for more tasty treats.
  - Use pyfits to load in the IRF data (or any data in FITS format).