Usage notes for likeSED v10, bdlikeSED, and extbdlikeSED macros v11

Macro	Analysis Method
likeSED.py (v12)	Unbinned Likelihood
bdlikeSED.py (v12)	Binned Likelihood
extbdlikeSED.py (v12)	Binned Likelihood, extended source

Changes: (6-*June-2011*)
This version of the usage notes reflects changes made going to v12 of the macros which add the ability to write xml files for the full energy range and individual energy band fits as well as changes to make bdlikeSED.py and extbdlikeSED.py work with ST v9r23p1 (i.e. use gtexpcube2 and take advantage of the BinnedAnalysis object function setEnergyRange). Note that the use of gtexpcube2 means that these two macros will not function with earlier ST versions which did not have this tool. With the use of the BinnedAnalysis.setEnergyRange function the number of files which need to be generated is reduced from 4 per energy band to a total of 5. Additionally, many typos pointed out by users have been corrected.

Contents:

Aims:
The point of these macros is to use the Fermi Science Tools to produce spectral plots for LAT sources. This is done by performing a maximum likelihood analysis (unbinned for likeSED and binned for bdlikeSED and extbdikeSED) in different energy bands. This essentially provides flux measurements in each band which can be plotted with the maximum likelihood model fit to the whole energy range and provide a sense of how well that model describes data. These points should not be used to perform a fit. These macros are meant to combine all of the steps and treat the energy bands in a uniform way as well as accounting for particulars which occur when fitting the LAT data in, relatively, small energy bands.
Note, these macros are not official LAT collaboration endorsements and should not be treated as such, the methods described below are not the only way of producing LAT spectral points so feel free to produce your own macro; however, there are some useful caveats regarding the approach of performing energy band fits contained in the following which are good to keep in mind.

Philosophy:
While running one of these macros does entail performing the full energy range maximum likelihood fit this is not meant to be a one-off process for fitting LAT data. It is expected that one will have, indepednently of the scripts, run at least one, if not several, fits of the data checking the global residuals, possibly trying different spectral models for your source of interest, evaluating if other sources need to be included in the model, etc. (e.g. follow the FSSC analysis threads to get an acceptable fit). At this point, one may chose to produce a separate xml file for the fits in the small energy bands. It is is necessary to produce a separate xml file if your source of interest is not modeled as a PowerLaw or PowerLaw2 over the full energy range; however, one may want to have more sources fixed in the smaller energy band fits than for the full energy range, e.g. if it is found that some sources far from your source of interest have little effect on either the diffuse backgrounds or your source of interest, or you may choose to simply adjust the sources in this xml file to be at the best fit values from your full energy range fit to aid in the convergence of the energy band fits. These scripts are not fail-proof, black boxes. Though much effort has been put into accounting for all potential sources of error and/or bad fits, it may be the case that one will have to manually perform one of the energy band fits and edit the output fits file (more details below). While one can simply follow the example usage below reading on and understanding how the macros fit each energy band and how certain values are arrived at is highly encouraged.

Example Usages:

(For likeSED)

```
from likeSED import *
myobs=UnbinnedObs('myeventfile.fits','mySCfile.fits','myexpmap.fits','myexpcube.fits','myirfs')
mylike=UnbinnedAnalysis(myobs,'full_energy_range.xml','myoptimizer')
inputs=likeInput(mylike,'mysource')
inputs.plotBins()
inputs.fullFit()
sed=likeSED(inputs)
sed.getECent()
sed.fitBands()
sed.Plot()
```

In the above example one first imports the macro and then makes an UnbinnedObs object and an UnbinnedAnalysis object which are for the full energy range fit, see the FSSC documentation on likelihood analysis from python for more details.

The next step is to make a likeInput object, the first argument is the UnbinnedAnalysis object you just created while the second is a string which corresponds to your source of interest. This will assume that the model used for the UnbinnedAnalysis object, full energy range, should be used for the energy band

fits. If your source of interest is not a PowerLaw or PowerLaw2 in this xml file then you will have to specify a different xml file, more details later.
Next, one calls the plotBins function of the likeInput object, this creates 20 bins, of equal size in log(energy), across the full energy range in the event file given to the UnbinnedObs object and then chooses the highest energy bin to be that which contains the highest energy event found consistent with the position of your source within the 95% containment radius as defined by the instrument response functions (IRFs) specified in the UnbinnedObs object. The default is 20 bins but this number can be specified, going beyond 10 bins per decade is not recommended, even for the brightest sources. The plotBins function then makes all the necessary files for the energy band fits by calling the Fermi Science Tools via the GtApp python module. Then one does the fit for the full energy range with the fullFit function, the default is to assume a tolerance of 1e-3, ABSOLUTE tolerance type, and covar=False but these can be changed. The next step is to create a likeSED object, the only argument for this object is the likeInput object you created earlier.
Then one calls the getECent function of the likeSED object to calculate a weighted average for each of the energy bands. This function uses the full energy range maximum likelihood model as the weights.
Next, one calls the fitBands function, the default is to use a tolerance of 1e-3, ABSOLUTE tolerance type, and to calculate an upper limit in any energy bin for which the source of interest is found to have a test statistic (TS) of <25. These can be changed (e.g. sed.fitBands(tslim=9), for more information see Object Descriptions section). The last step is to call the Plot function, this makes three histograms. The first is a plot of the differential flux spectrum (units of cm^{-2} s^{-2} GeV^{-1}), the second is a nuF_nu or E^2dN/dE spectrum (units of erg cm^{-2} s^{-1}), and the third is a histogram of the TS values in each energy band. All three are plotted vs energy in GeV.

(For bdlikeSED)

```
from bdlikeSED import *
myobs=BinnedObs('mysrcmaps.fits','myexpcube.fits','mybdexpmap.fits','myirfs')
mylike=BinnedAnalysis(myobs,'full_energy_range.xml','myoptimizer')
inputs=bdlikeInput(mylike,'myeventfile.fits','mycountscube.fits','mySCfile.fits','mysource')
inputs.plotBins()
inputs.fullFit()
sed=bdlikeSED(inputs)
sed.getECent()
sed.fitBands()
sed.Plot()
```

The usage for bdlikeSED is very similar to that for likeSED. Two obvious differences are that instead of UnbinnedObs/Analysis objects one creates BinnedObs/Analysis objects and insted of creating likeInput/SED objects one creates bdlikeInput/SED objects. Another important difference is in the necessary inputs for the bdlikeInput object, these extra arguments are necessary to perform binned likeihood fits in the energy bands.
The first argument is the BinnedAnalysis object just created.
The second is the full energy range event file which was, ultimately, the basis of the source maps file given to the BinnedObs object previously created.
The third argument is the counts cube used to make the source maps file given to the BinnedObs object.
The fourth argument is the space craft file used to create the source maps file given to the BinnedObs object.

And the fifth argument is a string corresponding to the name of your source of interest.
Beyond this all of the steps and functionality are the same as for likeSED.

(For extbdlikeSED)

```
from extbdlikeSED import *
myobs=BinnedObs('mysrcmaps.fits','myexpcube.fits','mybdexpmap.fits','myirfs')
mylike=BinnedAnalysis(myobs,'full_energy_range.xml','myoptimizer')
inputs=bdlikeInput(mylike,'myeventfile.fits','mycountscube.fits','mySCfile.fits','mysource')
inputs.plotBins(77000)
inputs.fullFit()
sed=bdlikeSED(inputs)
sed.getECent()
sed.fitBands()
sed.Plot()
```

This macro is designed to produce spectral plots for extended sources. Unbinned likelihood can, in principle, handle extended sources but studies have shown that the fits are not always reliable and binned likelihood is preferred. The usage for extbdlikeSED is the same as for bdlikeSED. The only difference is in the plotBins function, instead of selecting the highest energy based on the maximum energy found consistent with the 95% containment radius the maximum energy must be given to the plotBins function. The macro has not been adjusted to use the bounds of the extended source combined with the IRFs to find the highest energy event consistent with the source and, therefore, this must be decided indpendently via inspection of the event file. Additionally, parts of this macro rely on using a PowerLaw or PowerLaw2 model for the source and do not support other models. Also, there is no support for extended models with different angular extents at different energy ranges in the macro.

Object descriptions and full argument lists:

likeInput(LIKE,SrcName,model='',nbins=20,phCorr=1.0)

-Arguments:

LIKE = UnbinnedAnalysis object corresponding to the full energy range fit.

SrcName = String corresponding to the name of your source of interest as it appears in the xml file(s).

model = String corresponding to the xml file to be used in the energy band fits, if this is not specified, i.e. left as the default '', the macro will attempt to use the xml model given to LIKE.

nbins = Integer coresponding to the number of bins across the full energy range one wishes to make, a good rule of thumb is to use 3-5 bins per decade in energy.

phCorr = Optional parameter, which allows for phase-resolved studies where one has selected on phase, if the phase selection is 0.5<=phase<=1.0 then phCorr should be set to 0.5

The likeInput object has access to the files given to the UnbinnedObs and UnbinnedAnalysis objects made previously as well as the IRFs chosen and the region of interest (ROI) and energy cuts applied to the event file given to the UnbinnedObs object. One can also use the likeInput object to access the xml model for the energy band fits and the name of the source of interest.

---

bdlikeInput(LIKE,ft1,cc,ft2,SrcName,model='',nbins=20,phCorr=1.0)

-Arguments: (argument definitions which are identical to those of the likeInput object above are omitted)

ft1 = String corresponding to the file name for the event file used to create the source maps file for the BinnedObs object created previously.

cc = String corresponding to the file name of the counts cube used to make the source maps file for the BinnedObs object.

ft2 = String correspoding to the file name for the space craft file used to create the binned exposure map and exposure cube used in the BinnedObs object.

The bdlikeInput has similar access to files and cuts as described for the likeInput object above.

---

extbdlikeInput(LIKE,ft1,cc,ft2,SrcName,model='',nbins=20,phCorr=1.0)

-Arguments: (argument definitions are identical to those of the bdlikeInput object and are thus omitted)

The functionality is the same as the bdlikeInput object.

---

likeSED(Input)

bdlikeSED(Input)

extbdlikeSED(Input)

-Arguments:

For all three of these objects there is only one argument, the corresponding likeInput/bdlikeInput/extbdlikeInput object as described above. The likeSED/bdlikeSED/extbdlikeSED object has direct access to Input.

Function descriptions and full argument lists for likeInput, bdlikeInput, and extbdlikeInput objects:

If you have made the likeInput/bdlikeInput/extbdlikeInput object and called it inputs, the following functions are called by typing inputs.functionName(...), where the name of the function is used in place of functionName and the necessary arguments are given in the parentheses.

Print()

This function prints the details of the UnbinnedAnalysis (or BinnedAnalysis) object as well as the name of the xml file to be used in the energy band fits and the source of iterest.

---

srcMaxE()

This function cycles through the event file and finds the maximum energy event consistent with the source of interest within the 95% containment radius as defined by the IRFs given to the UnbinnedObs (BinnedObs) object. This function is called by the plotBins function described below. Note, this does not work for extended sources.

---

plotBins(MaxE=0,evclsmin=3,evclsmax=4)

-Arguments:

MaxE = Number corresponding to the maximum energy, in MeV, likely consistent with the source of interest.

evclsmin = minimum value event class selection.

evclsmax = maximum value for event class selection.

The maxE parameter must be supplied for extbdlikeSED objects as I have not worked in a method to convolve the PSF with the extended source shape. If the maxE argument is not given, i.e. kept at the default value of 0, the macro will use the pyIrfLoader module to find the highest energy event consistent with your source of interest within the 95% containment radius as defined by the IRFs

specified for the UnbinnedObs (or BinnedObs) object created previously. This function creates the specified number of bins (equally sized in log(energy)) across the full energy range and then creates bins for plotting such that the highest energy bin is that which contains the maxE value. The function selects the necessary energy band event files and creates the other files necessary for performing unbinned or binned likelihood fits in each energy band. The files are named such that they contain the name of the source of interest (with white spaces removed), the number of bins across the full energy range, the energy band number, the type of file if necessary (i.e. sm for source maps, em for exposure map), and the IRFs chosen if necessary. If files with the same names exist they will not be overwritten and a message will be printed to the screen
saying so.
For bdlikeSED.py and extbdlikeSED.py there is an additional optional argument myForce(=False by default) which is used to force the exposure calculation to be done using gtexpcube; however, this should only be used if you have run a ST version with gtexpcube2 (forthcoming) and thus have a par file for that ST. GtApp will see the parfile and assume that the tool exists and not actually check.
Note that for bdlikeSED.py and extbdlikeSED.py the plotBins (and customBins, see below) function does not have evclsmin and evclsmax arguments as no additional event files need to be created with v12.

---

customBins(MIN,MAX,evclsmin=3,evclsmax=4)

-Arguments:

MIN = A list which contains the low edges, in MeV, of user defined bins.

MAX = A list which contains the high edges, in MeV, of user defined bins.

This function allows the user to define their own bins and use those instead of the bins as defined by plotBins. This function will also make all necessary files automatically. Note, the macro does not distinguish files made via custom bins from those made via the plotBins function scheme. As such, it is necessary to note how the files were made and, if you plan to use a different binning scheme, delete the files made previously so that new files can be made.

---

makeFiles(evclassmin,evclsmax)

This function makes the files necessary to perform unbinned or binned likelihood fits in each energy band, it is called by the plotBins and customBins functions and shoud not be called directly. The two arguments are passed in from the plotBins and customBins functions. Note that for bdlikeSED.py and extbdlikeSED.py the makeFiles function does not have any arguments.

---

getfullFit(Emin=0,Emax=0,expCorrect=False,writeXML=False)

-Arguments:

Emin = Number corresponding to the minimum energy, in MeV, for which one wants the full energy range fit to be plotted. Allows for extrapolation to compare with lower energy measurements. If not specified, will be set to low edge of first energy bin.

Emax = Number corresponding to the maximum energy, in MeV, for which one wants the full energy range fit to be plotted. Allows for extrapolation to compare with higher energy measurements. If not specified, will be set to high edge of last energy bin.

e**xpCorrect** = Boolean telling the macro whether or not the xml file provided for the full energy

range fit needs to have the normalizations corrected to account for the phase cut corresponding to the phCorr value specified previously. In particular, if this is true the macro will divide all normalizations by the phCorr value to give the correct values accounting for the fact that the exposure calculations do not take phase into account.

writeXML = Boolean telling the macro whether or not the full energy range model should be written out as an xml file or not, default is False.

This function allows one to access the fit information from the UnbinnedAnalysis (BinnedAnalysis) object if the fit has already been run. It records the fit values across the energy range to be plotted as a TGraph. If the fit was run with covar=True it will also record information for a bowtie plot from Emin to Emax as well. This function will print the model information for you source of interest to the screen.

---

fullFit(ftol=1e-3,CoVar=False,toltype='ABS',Emin=0,Emax=0,writeXML=False)

-Arguments: (argument definitions identical to those defined above will be omitted)

ftol = Tolerance for fit across full energy range.

CoVar = Flag to be given to the UnbinnedAnalysis (BinnedAnalysis) object when performing the fit to calculate, or not, the covariance information. Acceptable arguments are True or False, no quotes.

toltype = String, 'ABS' for ABSOLUTE tolerance type and 'REL' for RELATIVE tolerance type. For the Fermi Science Tools v9r15p2 and later the default is ABSOLUTE so that has been adopted here as well.

This function will actually perform the fit for the full energy range with the specified tolerance and tolerance type. If you want a bowtie plot then specify CoVar=True. Emin and Emax serve the same function as in the getfullFit function. This function will print the model information for you source of interest to the screen. For more information on the tolerance type see the FSSC documentation.

Function descriptions and full argument lists for likeSED, bdlikeSED, and extbdlikeSED objects:
If you have made the likeSED/bdlikeSED/extbdlikeSED object and called it sed, the following functions are called by typing sed.functionName(...), where the name of the function is used in place of functionName and the necessary arguments are given in the parentheses.

---

getECent()

This function creates a list of weighted energy centers, in GeV, to use when fitting the energy bands and when plotting the results (more details later). The weighting is done as follows...assume that the full energy range fit model is some function of energy F(E), then the center energy is calculated as sum(F(E_i)*E_i)/sum(F(E_i)) where i runs from 0 to 99 and E_0 is the low edge of the energy bin and E_99 is the high edge of the energy bin.

---

customECent(Cent)

-Arguments:

Cent = A list containing user defined center energies, in MeV, for each bin.

This function allows the user to define center energies for the bins in a manner different from the spectrally weighted approach of the getECent function described above.

---

fitBands(ftol=1e-3,tslim=25,toltype='ABS',opt='NewMinuit',lastbinUL=False,rescaleAll=False,writeXML=False)

-Arguments:

ftol = Tolerance for the fit in each energy band.

tslim = TS value below which a 95% confidence level upper limit is calculated.

toltype = Tolerance type to use for each fit 'ABS' is ABSOLUTE and 'REL' is RELATIVE.

opt = Optimizer to use for the energy band fits.

lastbinUL = Flag to force the last bin to be an upper limit. This forces an upper limits calculation in the last bin regardless of the TS value of the source and the value of tslim. However, if this flag is False and the source of interest is found with a TS < tslim in the last bin an upper limit will still be calculated.

rescaleAll = Boolean, the macro will change the scale value of the normalization parmaeter to have a better starting point for the fit (i.e. if one starts with a prefactor of 1e-9 for all energy bands the higher energy bands may not converge well) for the source of interest, set this to True if you want it done for all sources with free normalization parameters for consistency.

This is the function that actually performs the likelihood fits in each energy band. If no value is specified for an argument then the default value, listed above, will be used. Specifics of the fitting are described later. Note that the default value of 25 for tslim is rather strict and you may want to try lowering this value.

Plot(plot=True)

-Arguments:

plot = Flag to display, or not, the output plots.

This function takes all of the information from the full energy range fit and the energy band fits and produces the three plots described earlier. It also saves the information in a fits file with naming convention 'SrcName_Nbins_likeSEDout.fits' (where SrcName is replaced by the name of your source of interest, N is replaced by the integer number of bins across the full energy range specified in the likeInput object, and likeSEDout is replaced with bdlikeSEDout and extbdlikeSED out for the correspoding macros). The TCanvases produced are also saved in a root file with naming convention 'SrcName_Nbins_histos.root' (with replacements as described above). If plot=False, useful when running in the background, set output to /dev/null; the TCanvases will flicker only momentarily whereas if left at the default vale of True the TCanvases stay drawn and editing of axes ranges and moving of the TLegend can be done. This function will automatically save .eps versions of the three TCanvases

---

residuals(plot=True)

The sole argument for this function serves the same purpose as that in the Plot function described above. This function will plot the residuals defined as ([energyBand_i fit results]-[full energy range model at E_center_i])/[full energy range model at E_center_i], where the E_center_i values are from the getECent or customECent functions. This will save the TCanvas as a root file with naming convention 'SrcName_Nbins_residuals.root' (with replacements as described above for the Plot function) and will produce a .eps version of the TCanvas.

Particulars of the energy band fits:

The fits are done one energy band at a time. The first step is to create the UnbinnedObs/Analysis (BinnedObs/Analysis) objects for a given energy band and set it to the tolerance type specified in the fitBands function. Once the fit results have been recorded for a given energy band the objects created

for it are deleted to avoid memory issues for unbinned likelihood, for binned likelihood only the BinnedAnalysis object is deleted as we can use the same BinnedObs object for each energy band. The fitting has slightly different steps if the source of interest is modeled as a PowerLaw or PowerLaw2 owing simply to the differences in the parameters of each model but the basic approach is the same for both cases.
The macro cycles through the sources in the model and for those which have free normalization parameters and have PowerLaw2 spectral models the UpperLimit and LowerLimit parameters are set to the high and low edges of the energy bin, respectively. This is to avoid large errors when trying to fit for UpperLimit and LowerLimit values far beyond the bounds of the energy band.
The macro then does a preliminary fit using a tolerance of 1 (and then trying 0.1 and 10 if an error occurs) using the optimizer specified in the fitBands function to get a good starting point and to look for point sources which have a negative or zero TS value in that energy band. Those negative or zero TS sources which have free normalizations and are not the source of interest are then removed from the fit. This is done as it has been found that such sources can lead to a bad covariance matrix and thus incorrect error bars, usually this will manifest as unbelievably small error bars. If this preliminary fit fails a message is printed to the screen and, in the event of unreasonable error bars, you may want to manually fit that energy band.
The macro then attempts to fit the function, starting from the preliminary fit with negative or zero TS sources removed, with the optimizer and fit tolerance given to the fitBands function. If an error occurs the macro will try to lower and then raise the tolerance by a factor of 10. If the fit will not converge an error message is printed and values of zero are recorded for that energy band. If this occurs you should attempt to fit that energy band manually and then you can edit the fits file output from the Plot function and create new plots using the redraw macro described elsewhere.
When the fit finishes successfully the necessary information is recorded and if the source of interest is found to have a TS < tslim an upper limit is calculated using the UpperLimits module included with the Fermi Science Tools. This calculates a 95% confidence level upper limit on the integrated flux of your source (from the bin low edge to the bin high edge). If the first upper limit calculation is unsuccessful the fit will be tried again with a tolerance 1/10 that specified in the fitBands function to attempt to find a better starting point for the calculation. If this fit finishes successfully the TS value of the source of interest is checked again and, provided it is still < tslim, the upper limits calculation is tried again. If the upper limits calculation still fails the best fit value with zero error is used and a warning is printed. Note that this will be plotted as an upper limit by the Plot function so you will want to manually refit this bin and modify the output fits file accordingly.

<u>When source of interest has PowerLaw spectral model:</u>
When using a PowerLaw model for the source of interest the differential flux value for that energy band can be taken as the value of the Prefactor if the Scale parameter is set to the center of the energy band. If you are not familiar with the parameters of the PowerLaw spectral model please review the FSSC documentation. The macro does just that. It also changes the scale of the Prefactor to be closer to what one might expect given the full energy range model. This sounds somewhat circular but all it does is ensure a good starting place for the fit. In particular, the code calculates the integral flux for your source of interest in the full energy range model between the bin low and high edges and divides that by the width of the bin. The code then takes the log10 of this value and casts it as an integer (call it J) and sets the scale of the Prefactor to be 10**J. So if your xml file has the prefactor starting at 1e-7 ph/cm^2/s/MeV but in a given energy band your full energy range function has an integral flux divided by bin width of 5.2e-9 the code will start your source of interest at 1e-9 ph/cm^2/s/MeV. This simply

helps the fit converge more consistently; however, depending on the tolerance level used for the fit and if you have a very steeply falling spectrum, this rescaling can lead to upper limits in the higher energy bins which are unrealistically too low. Therefore, to help avoid this problem the exponent for the new scale value is set to max(J,-14).
When using a PowerLaw model, the macro saves the prefactor value and plots that for the differential flux spectrum. Then it simply multiplies by the center of each energy band squared to produce the E^2dN/dE plot, with conversion factors to get units of erg cm^-2 s^-1.

When source of interest has PowerLaw2 spectral model:

When using a PowerLaw2 model for the soruce of interest the differential flux value for that energy band is taken to be the integral flux from the bin high edge to low edge divided by the bin width. The macro does just this, setting the UpperLimit and LowerLimit parameters of the source to the bin high and low edges, respectively, as described previously and recording the Integral parameter of the fit. The Plot function divides this value by the bin width. If you are unfamiliar with the parameters of the PowerLaw2 spectral model please consult the FSSC documentation. Similar to how the scale of the Prefactor is adjusted for a PowerLaw model, the macro adjusts the scale of the Integral parameter to be closer to what one might expect given the full energy range fit. To do this, the code calculates the integral flux of the full fit function from the bin low edge to the bin high edge and then calculates the log10 of this flux and casts it as an integer (call it J) and then sets the scale of the Integral parameter to be 10**J. Similar to the case when using a PowerLaw spectral model, the index of the new scale value is set to max(J,-14). The macro also records the Index parameter when using a PowerLaw2 model to facilitate plotting. As noted above, the Integral values are divided by the bin width to get the differential flux values in the Plot function. However, to get the nuF_nu points the Integral parameter is integrated up using the best fit Index parameter.

Differences when dealing with extended sources:

When fitting an extended source it should be noted that the Prefactor(Integral) parameter in the PowerLaw(PowerLaw2) model has units of cm^{-2} s^{-1} MeV^{-1} sr^{-1} (cm^{-2} s^{-1} sr^{-1}), this means that an extra step is necessary. After running the full energy range fit the sr value must be calculated. For a PowerLaw model, the power law integral equation is solved for the prefactor. This value is then divided by the Prefactor parameter to give the sr value. For a PowerLaw2 model the integral flux is calculated between the UpperLimit and LowerLimit parameters of the model, this value is then divided by the Integral parameter to get the sr value. This value is then multiplied by the Prefactor or Integral values in the energy band fits to get the proper units for the plots.

---

redraw.py

I decided that it might be useful to have a macro with some functions to easily redraw the output plots and access the info in the default output root and fits file from the likeSED and bdlikeSED macros. As such, I've made redraw.py. This macro allows you to simply redraw the plots using the output root file or make TGraphs from the fits file which can be used to draw on other plots like multiwavelength SEDs, the macro allows you to change the units of the flux and energy columns if you desire by simply specifying the necessary multiplication factors. There are usage notes at the end of the .py file, but I've reproduced them below for ease.

This is meant to be a supplement to the likeSED family of macros and aid in replotting the output

spectra and in using the spectra on larger band SEDs.

There are 3 functions of interest:
redrawSED(filename)

-Arguments:

filename = String corresponding to the name of the output root histograms file from one of the likeSED macros.

This function merely accesses the root file, which is automatically made when calling the Plot function of a extbd/bd/likeSED object, to redraw the default canvases.

---
getflxGraph(filename,flxmod=1.,energymod=1.)

-Arguments:

filename = String corresponding to the name of the output fits file from one of the likeSED macros.

flxmod = Float, necessary conversion factor to go from default units (erg cm^{-2} s^{-1}) to desired units.

Energymod = Float, same as above but for energy units, default is GeV.

This function accesses the fits file, made automatically when calling the Plot function, and makes a TGraphAsymmErrors which can be used to plot on a full band SED. The optional arguments allow one to use different units for the flux and energy axes.

---
getmodGraph(filename,flxmod=1.,energymod=1.,covar=False)

-Arguments: (argument definitions which are identical to those of the likeInput object above are omitted)

covar = Boolean, if true means full energy range fit was run with covar=true so bowtie information exists and you want it.

This function accesses the same fits file getflxGraph does but makes a TGraph for the full energy range model (and bowtie plot) for use on a full band SED.