Fermipy tutorial

PhD course – University of Padova
February 1-2, 2021

Principe Giacomo*[1,2,3]

[1]Dipartimento di Fisica, Università di Trieste, Trieste, Italy
[2]Istituto Nazionale di Fisica Nucleare, Trieste, Italy
[3]INAF - Istituto di Radioastronomia, Bologna, Italy

Introduction

Fermipy is a python package that facilitates analysis of data from the Large Area Telescope (LAT) with the Fermi Science Tools.
The Fermipy package is built on the pyLikelihood interface of the Fermi Science Tools and provides a set of high-level tools for performing common analysis tasks:

- Data and model preparation with the gt-tools (gtselect, gtmktime, etc.).
- Extracting a spectral energy distribution (SED) of a source.
- Generating TS and residual maps for a region of interest.
- Finding new source candidates.
- Localizing a source or fitting its spatial extension.
- • ……..

Fermipy:
https://fermipy.readthedocs.io/en/latest/

Prerequisite (Unbinned Likelihood analysis):
https://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/likelihood_tutorial.html

Fermipy Step by Step: Installation

```
$ curl -OL https://raw.githubusercontent.com/fermiPy/fermipy/master/condainstall.sh
$ export CONDA_PATH=<path to your conda installation>
$ source condainstall.sh
```

Creation of a setup file (setup_fermipy.sh)

```
$ export PYTHON_VERSION=2.7
$ export CONDA_DEPS="scipy matplotlib pyyaml numpy astropy gammapy healpy"
$ # This should point at your conda installation, or at the place you would like to install co
$ export CONDA_PATH="$HOME/minconda"
$ # This is the name that will be given to the conda environment created for fermipy
$ export FERMIPY_CONDA_ENV="fermipy"
$ # This is the command used to install the fermitools.
$ # Set it to an empty string if you do not want to install the fermitools
$ # of if you have already installed them.
$ export ST_INSTALL="conda install -y --name $FERMIPY_CONDA_ENV $FERMI_CONDA_CHANNELS -c $CONDA
$ # This is the command used install fermipy.
$ # If you want to install for source or use a different version of
$ # fermipy you should change this
$ export INSTALL_CMD="conda install -y --name  $FERMIPY_CONDA_ENV -c $CONDA_CHANNELS fermipy"
```

Configuration file

```
data:
  evfile : 'ft1.fits'
  scfile : 'SC.fits'
  ltcube: 'ltcube.fits'

binning:
  roiwidth   : 30.0
  binsz      : 0.1
  binsperdec : 8
  coordsys   : 'GAL'

selection :
  emin : 1000.0
  emax : 2200.0
  tmin : 239557417
  tmax : 541779795
  zmax    : 105
  evclass : 128
  evtype  : 3
  glon: 0.0
  glat: 0.0
  # gtmktime parameters
  filter : 'DATA_QUAL>0 && LAT_CONFIG==1'
  roicut : 'no'

gtlike:
  edisp : True
  irfs : 'P8R3_SOURCE_V2'
  edisp_disable: [isodiff]
  wmap : mask_GP_b2_GC3.fits

model:
  src_roiwidth : 34.0
  galdiff  : IEM.fits
  isodiff  : 'iso_P8R3_SOURCE_V2_v1.txt'
  catalogs: gll_psc_v20.fit
```

The *data* section contains the path to the ft1, ft2 and ltcube

The *binning* part specifies the pixel size, energy bins and ROI width

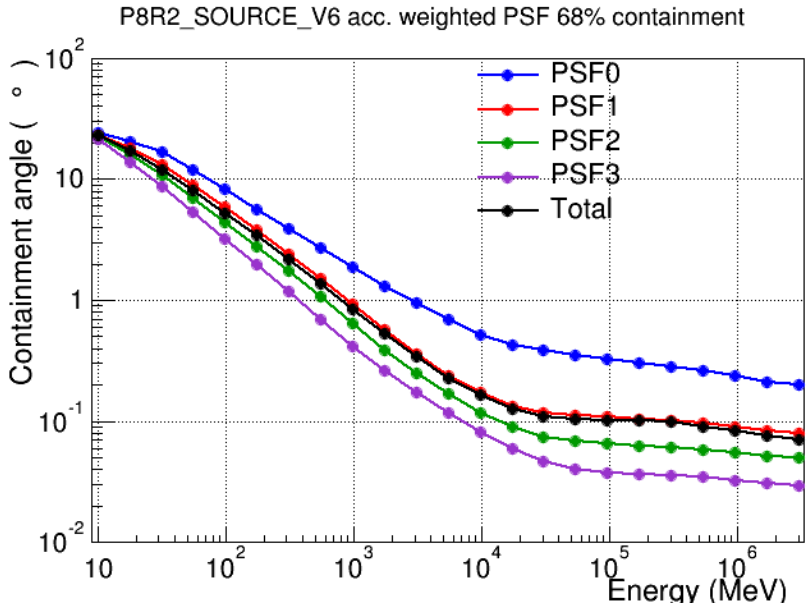
The *selection* part report the energy range, time interval, zmax, center of the ROI and cuts for gtmktime.

gtlike part includes the irfs, energy dispersion and weight map

Model: catalog, IEM and ISO template

Configuration file: multiple components

```
components:
  - { selection : { evtype : 4  } } # PSF0
  - { selection : { evtype : 8  } } # PSF1
  - { selection : { evtype : 16 } } # PSF2
  - { selection : { evtype : 32 } } # PSF3
```



Fermi-LAT performance

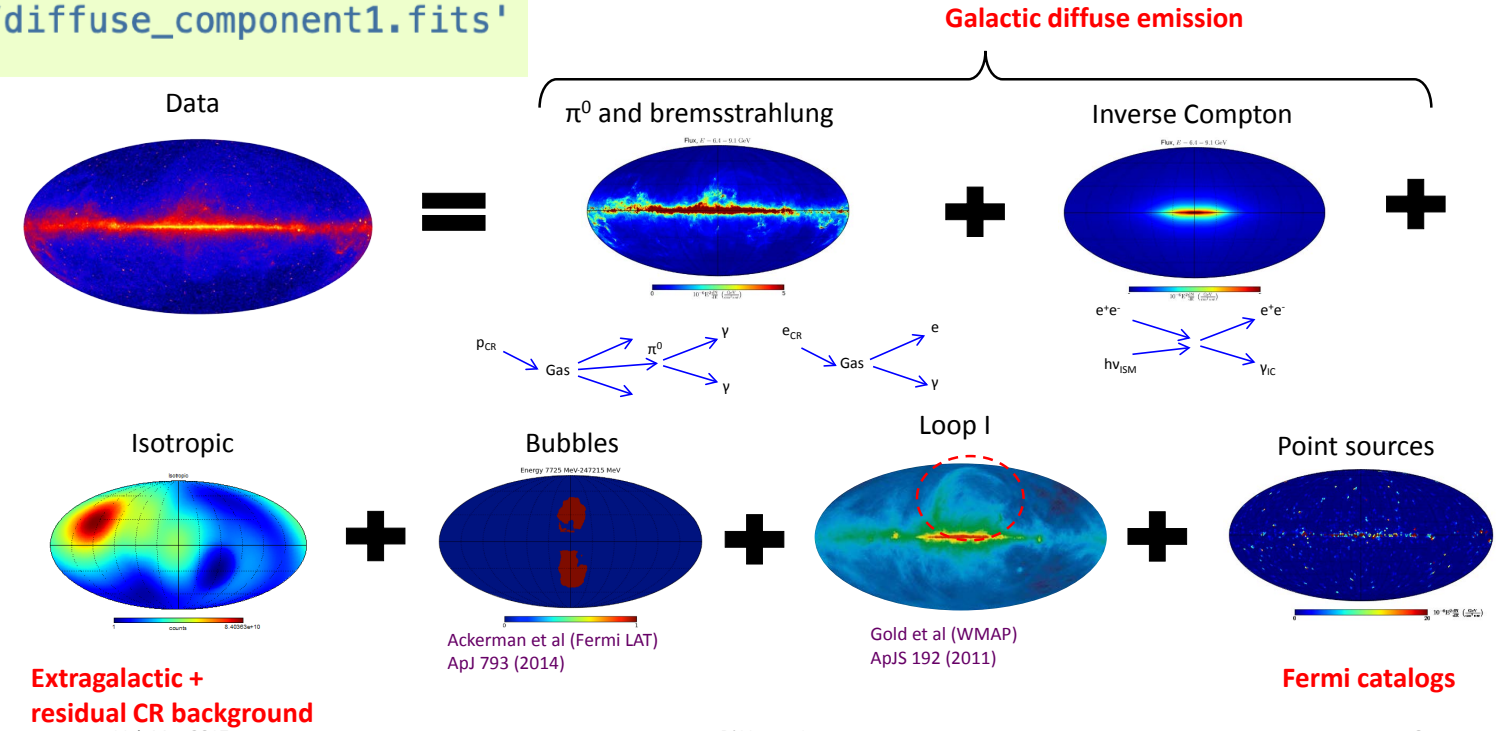
https://www.slac.stanford.edu/exp/glast/groups/canda/lat_Performance.htm

examples

Customize the model

```
model:
  src_roiwidth : 10.0
  galdiff  : '$FERMI_DIFFUSE_DIR/gll_iem_v06.fits'
  isodiff  : '$FERMI_DIFFUSE_DIR/isotropic_source_4years_P8V3.txt'
  catalogs : ['gll_psc_v14.fit']
```

```
model:
  galdiff  :
    - '$FERMI_DIFFUSE_DIR/diffuse_component0.fits'
    - '$FERMI_DIFFUSE_DIR/diffuse_component1.fits'
```



Customize the model

```
model:
  sources  :
    - { name: 'SourceA', glon : 120.0, glat : -3.0,
     SpectrumType : 'PowerLaw', Index : 2.0, Scale : 1000, Prefactor : !!float 1e-11,
     SpatialModel: 'PointSource' }
    - { name: 'SourceB', glon : 122.0, glat : -3.0,
     SpectrumType : 'LogParabola', norm : !!float 1E-11, Scale : 1000, beta : 0.0,
     SpatialModel: 'PointSource' }
```

```
model:
  sources  :
    - { name: 'PointSource', glon : 120.0, glat : 0.0,
     SpectrumType : 'PowerLaw', Index : 2.0, Scale : 1000, Prefactor : !!float 1e-11,
     SpatialModel: 'PointSource' }
    - { name: 'DiskSource', glon : 120.0, glat : 0.0,
     SpectrumType : 'PowerLaw', Index : 2.0, Scale : 1000, Prefactor : !!float 1e-11,
     SpatialModel: 'RadialDisk', SpatialWidth: 1.0 }
    - { name: 'GaussSource', glon : 120.0, glat : 0.0,
     SpectrumType : 'PowerLaw', Index : 2.0, Scale : 1000, Prefactor : !!float 1e-11,
     SpatialModel: 'RadialGaussian', SpatialWidth: 1.0 }
    - { name: 'MapSource', glon : 120.0, glat : 0.0,
     SpectrumType : 'PowerLaw', Index : 2.0, Scale : 1000, Prefactor : !!float 1e-11,
     SpatialModel: 'SpatialMap', Spatial_Filename : 'template.fits' }
```

Customize the model during the analysis process

```
from fermipy.gtanalysis import GTAnalysis

gta = GTAnalysis('config.yaml',logging={'verbosity' : 3})

# Remove isodiff from the model
gta.delete_source('isodiff')

# Add SourceA to the model
gta.add_source('SourceA',{ 'glon' : 120.0, 'glat' : -3.0,
               'SpectrumType' : 'PowerLaw', 'Index' : 2.0,
               'Scale' : 1000, 'Prefactor' : 1e-11,
               'SpatialModel' : 'PointSource' })

gta.setup()

# Add SourceB to the model
gta.add_source('SourceB',{ 'glon' : 121.0, 'glat' : -2.0,
                'SpectrumType' : 'PowerLaw', 'Index' : 2.0,
                'Scale' : 1000, 'Prefactor' : 1e-11,
                'SpatialModel' : 'PointSource' })
```

Preliminary steps of the analysis

```
from fermipy.gtanalysis import GTAnalysis

gta = GTAnalysis('config.yaml',logging={'verbosity' : 3})
gta.setup()
```

- gta.setup() runs the gt-tools (gtselect, gtmktime, gtbin, gtcountsmap, gtexposure,..) for selecting the data, creating counts and exposure maps, etc..
- Then gta.print_model():

sourcename	offset	norm	eflux	index	ts	npred free
4FGL J2021.1+3651	0.007	1.784	0.000154	3.12	nan	30462.9
4FGL J2021.9+3609	0.715	0.475	3.62e-06	1.57	nan	63.0
4FGL J2017.9+3625	0.757	0.437	1.93e-05	3.83	nan	4230.8
4FGL J2016.2+3712	1.036	0.348	5.85e-06	2.06	nan	413.6
4FGL J2015.5+3710	1.149	1.947	1.84e-05	2.72	nan	3479.6
4FGL J2017.3+3525	1.612	2.110	1.71e-06	0.41	nan	134.6
4FGL J2013.5+3613	1.641	2.914	3.49e-06	2.85	nan	694.4
4FGL J2030.0+3641	1.799	3.013	1.71e-05	3.32	nan	3293.0

- gta.free_sources(), gta.free_shape(), gta.set_norm() ….
- gta.write_roi() print different control plots: count spectrum, count map,….

Performing a preliminary fit: *gta.optimize()*

gta.optimize(): Iteratively optimize the ROI model. The optimization is performed in three sequential steps:

- Free the normalization of the N largest components that contain a fraction npred_frac and perform a simultaneous fit of the normalization parameters of these components.
- Individually fit the normalizations of all sources that were not included in the first step in order of their npred values. Skip any sources that have NPred < npred_threshold.
- Individually fit the shape and normalization parameters of all sources with TS > shape_ts_threshold where TS is determined from the first two steps of the ROI optimization.

From my experience running multiple time gta.optimize() gives already a model that is practically the same of a complete fit.
https://fermipy.readthedocs.io/en/latest/fitting.html

Performing a preliminary fit: *gta.optimize()*

Parameters:

- **npred_frac** (*float*) – Threshold on the fractional number of counts in the N largest components in the ROI. This parameter determines the set of sources that are fit in the first optimization step.
- **npred_threshold** (*float*) – Threshold on the minimum number of counts of individual sources. This parameter determines the sources that are fit in the second optimization step.
- **shape_ts_threshold** (*float*) – Threshold on source TS used for determining the sources that will be fit in the third optimization step.
- **max_free_sources** (*int*) – Maximum number of sources that will be fit simultaneously in the first optimization step.
- **skip** (*list*) – List of str source names to skip while optimizing.
- **optimizer** (*dict*) – Dictionary that overrides the default optimizer settings.

Key	Type	Description
`loglike1`	`float`	Post-optimization log-likelihood value.
`loglike0`	`float`	Pre-optimization log-likelihood value.
`config`	`dict`	Copy of input configuration to this method.
`dloglike`	`float`	Improvement in log-likehood value.

Performing a preliminary fit: *gta.optimize()*

- The advantage of using this tool is that it is much faster than a fit and running it a few times gives already a model close to the best fit.
- The disadvantage is that it does not return the covariance matrix and the correlation coefficients so you can not make checks for the parameters. Also errors on the SED params are not given.

```
In [7]: optimize= gta.optimize()
2019-12-02 06:10:59 INFO    GTAnalysis.optimize(): Starting
Joint fit  ['galdiff', '4FGL J2028.6+4110e', '4FGL J2021.5+4026', '4FGL J2021.1+3651', 'isodiff']
Fitting shape galdiff TS:  80433.980
Fitting shape 4FGL J2021.5+4026 TS:  15567.866
Fitting shape 4FGL J2021.1+3651 TS:  14692.598
Fitting shape 4FGL J2028.3+3331_added TS:    929.584
Fitting shape 4FGL J2028.6+4110e TS:    888.757
```

```
In [8]: optimize
Out[8]:
{'config': {'max_free_sources': 5,
  'npred_frac': 0.95,
  'npred_threshold': 1.0,
  'optimizer': {'init_lambda': 0.0001,
   'max_iter': 100,
   'min_fit_quality': 2,
   'optimizer': 'MINUIT',
   'retries': 3,
   'tol': 0.001,
   'verbosity': 0},
  'shape_ts_threshold': 25.0,
  'skip': []},
 'dloglike': 6.781330176512711,
 'loglike0': -16796.23910419963,
 'loglike1': -16789.457774023118}
```

Performing a fit: *gta.fit()*

- gta.fit() is a wrapper on the pyLikelihood fit.
- By default gta.fit() will repeat the fit until a fit quality of 3 is obtained. After the fit returns all sources with free parameters will have their properties (flux, TS, NPred, etc.) updated in the ROIModel instance. The return value of the method is a dictionary containing the following diagnostic information about the fit:

Parameters:

- **update** (*bool*) – Update the model dictionary for all sources with free parameters.
- **tol** (*float*) – Set the optimizer tolerance.
- **verbosity** (*int*) – Set the optimizer output level.
- **optimizer** (*str*) – Set the likelihood optimizer (e.g. MINUIT or NEWMINUIT).
- **retries** (*int*) – Set the number of times to rerun the fit when the fit quality is < 3.
- **min_fit_quality** (*int*) – Set the minimum fit quality. If the fit quality is smaller than this value then all model parameters will be restored to their values prior to the fit.
- **reoptimize** (*bool*) – Refit background sources when updating source properties (TS and likelihood profiles).

Output of a *gta.fit()*

Key	Type	Description
`fit_quality`	int	Fit quality parameter for MINUIT and NEWMINUIT optimizers (3 - Full accurate covariance matrix, 2 - Full matrix, but forced positive-definite (i.e. not accurate), 1 - Diagonal approximation only, not accurate, 0 - Error matrix not calculated at all)
`errors`	`ndarray`	Vector of parameter errors (unscaled).
`loglike`	`float`	Post-fit log-likehood value.
`correlation`	`ndarray`	Correlation matrix between free parameters of the fit.
`config`	`dict`	Copy of input configuration to this method.
`values`	`ndarray`	Vector of best-fit parameter values (unscaled).
`dloglike`	`float`	Improvement in log-likehood value.
`fit_status`	int	Optimizer return code (0 = ok).
`covariance`	`ndarray`	Covariance matrix between free parameters of the fit.
`edm`	`float`	Estimated distance to maximum of log-likelihood function.

Examples of *gta.fit()*

Fit to HESS1825-137 3-10 GeV. Tried to do the fit with all the sources as in the 4FGL:

```
In [23]: gta.fit()
2019-12-02 06:35:19 INFO    GTAnalysis.fit(): Starting fit.
2019-12-02 06:38:08 ERROR   GTAnalysis.fit(): MINUIT failed with status
code 102 fit quality 2
```

Then I changed all the curved SEDs into power-law:

```
In [27]: gta.fit()
2019-12-02 06:41:19 INFO    GTAnalysis.fit(): Starting fit.
2019-12-02 06:42:46 INFO    GTAnalysis.fit(): Fit returned successfully.
Quality:   3 Status:   0
2019-12-02 06:42:46 INFO    GTAnalysis.fit(): LogLike:   -16810.484
DeltaLogLike:    10252.595
```

```
In [29]: gta.print_model()
2019-12-02 06:48:36 INFO    GTAnalysis.print_model():
```

sourcename	offset	norm	eflux	index	ts	npred	free
4FGL J2021.1+3651_a	0.275	20.770	4.57e-05	3.04	14821.17	4573.6	*
4FGL J2017.9+3625_a	0.510	6.654	4.45e-06	3.81	480.40	493.1	*
4FGL J2015.5+3710_a	0.933	1.453	6.12e-06	2.63	775.26	582.4	*
4FGL J2013.5+3613_a	1.375	2.943	8.08e-07	4.42	34.36	96.6	*
4FGL J2017.3+3525_a	1.442	0.002	7.57e-07	0.00	31.47	51.6	*
4FGL J2022.3+3840_a	1.957	4.829	5.8e-07	5.00	19.34	74.9	*
4FGL J2030.0+3641_a	2.053	8.515	5.15e-06	3.88	785.18	571.6	*
4FGL J2018.5+3852_a	2.106	0.073	1.85e-06	1.57	110.96	153.9	*
4FGL J2009.9+3544	2.237	2.119	7.91e-07	0.50	31.92	56.6	*
4FGL J2026.6+3449	2.390	1.809	6.71e-07	0.67	27.27	48.1	*
4FGL J2035.0+3632_a	3.064	1.070	3.19e-06	2.85	388.44	308.4	*
4FGL J2025.3+3341_a	3.293	0.262	1.18e-06	2.59	64.45	109.3	*
4FGL J2021.5+4026_a	3.680	41.904	5.53e-05	3.36	15039.92	5849.2	*
4FGL J2028.3+3331_a	3.687	5.853	5.31e-06	3.61	928.00	559.5	*
4FGL J2021.0+4031e_	3.748	1.110	1.28e-05	2.03	243.51	1129.3	*
4FGL J2004.3+3339_a	4.435	0.326	3.42e-06	2.08	336.02	289.0	*
4FGL J1959.0+3844_a	4.540	0.401	4e-07	3.55	24.23	41.4	*
4FGL J1958.5+3512	4.580	0.494	7.58e-07	3.66	39.91	78.0	*
4FGL J2028.6+4110e_	4.717	18.629	0.000126	2.34	940.29	6615.7	*
4FGL J2023.4+4127	4.725	0.466	1.2e-06	2.27	31.19	104.1	*
4FGL J2023.2+3153	4.945	2.533	8.61e-07	2.03	49.57	70.2	*
4FGL J2032.2+4127_a	5.265	2.628	3.24e-05	1.99	98.48	333.9	*
isodiff	---	0.762	0.00457	2.24	9.99	977.8	*
galdiff	---	0.917	0.0357	-0.02	78830.71	50149.8	*

Customizing the model for the fit

By default all models parameters are initially fixed. The free_source() and free_sources() methods can be use to free or fix parameters of the model.

```
# Free Normalization of all Sources within 3 deg of ROI center
gta.free_sources(distance=3.0,pars='norm')

# Free all parameters of isotropic and galactic diffuse components
gta.free_source('galdiff')
gta.free_source('isodiff')
```

The minmax_ts and minmax_npred arguments to *free_sources()* can be used to free or fixed sources on the basis of their current TS or Npred values:

```
# Free sources with TS > 10
gta.free_sources(minmax_ts=[10,None],pars='norm')

# Fix sources with TS < 10
gta.free_sources(minmax_ts=[None,10],free=False,pars='norm')

# Fix sources with 10 < Npred < 100
gta.free_sources(minmax_npred=[10,100],free=False,pars='norm')
```

Performing a fit: printing or saving the output model

We can check the results of the optimization step by calling *print_roi():*

```
gta.print_roi()
```

After the fitting is complete we can write the current state of the model with *write_roi():*

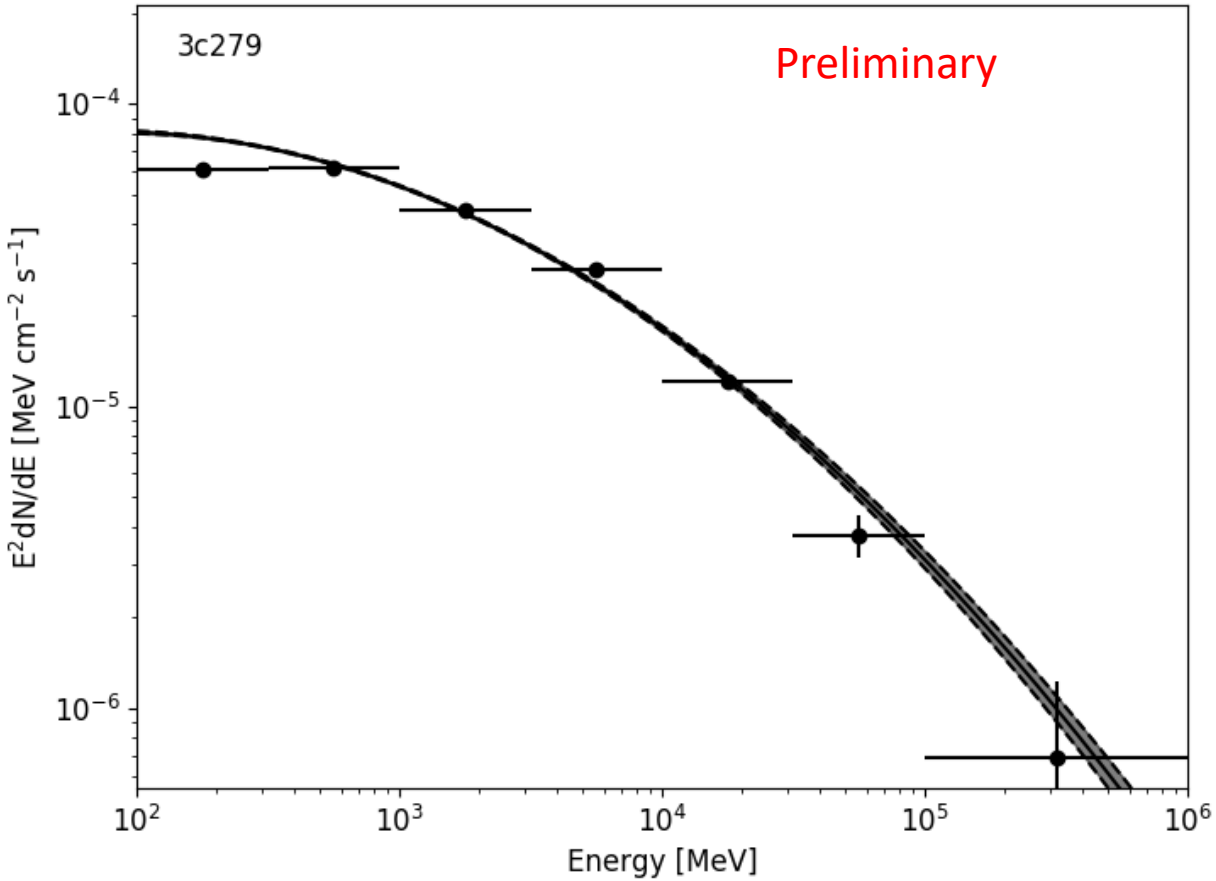
```
gta.write_roi('fit_model')
```

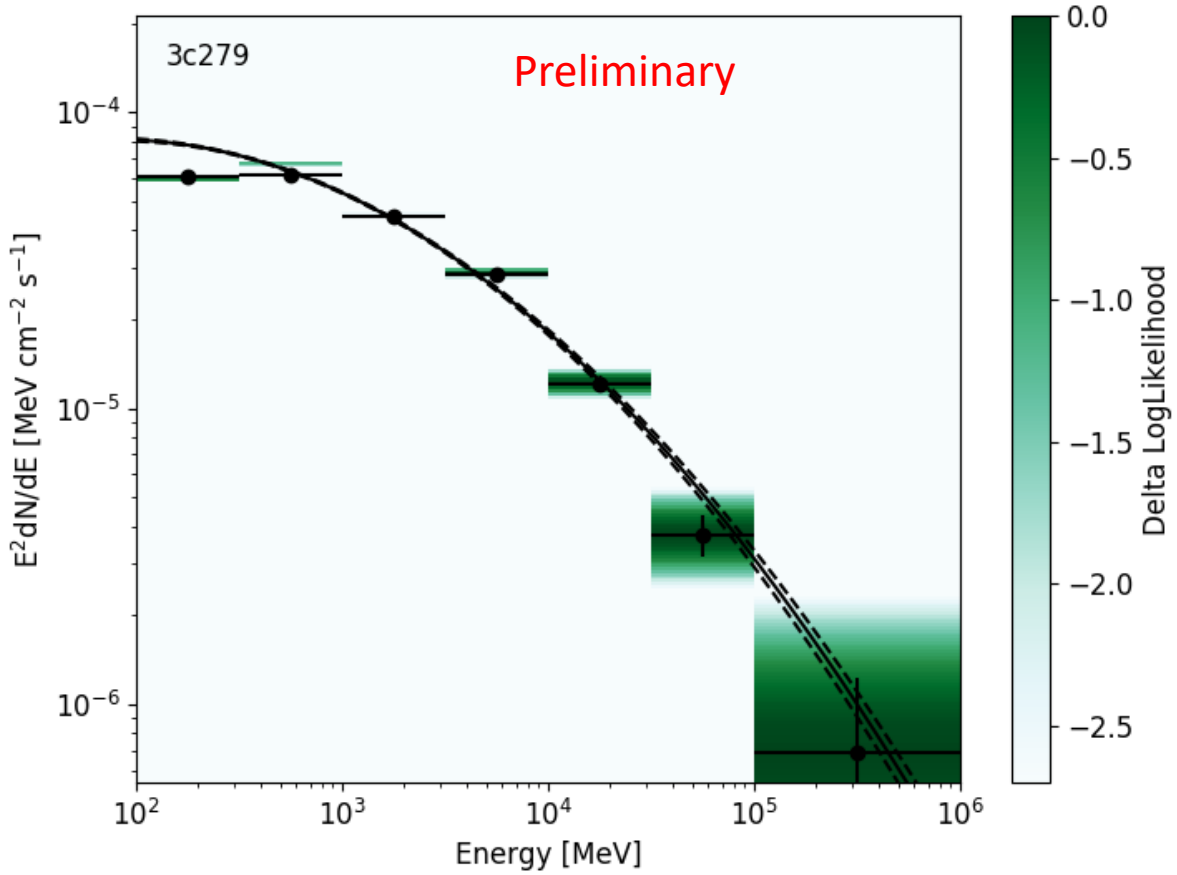
Performing a fit: *gta.sed()*

The *sed()* method computes a spectral energy distribution (SED) by performing independent fits for the flux normalization of a source in bins of energy. The normalization in each bin is fit using a power-law spectral parameterization with a fixed index.

```
# Run analysis with default energy binning
sed = gta.sed('sourceA')

# Override the energy binning and the assumed power-law index
# within the bin
sed = gta.sed('sourceA', loge_bins=[2.0,2.5,3.0,3.5,4.0,4.5,5.0], bin_index=2.3)

# Profile background normalization parameters with prior scale of 5.0
sed = gta.sed('sourceA', free_background=True, cov_scale=5.0)
```

gta.sed(): examples





Lightcurve analysis

lightcurve() fits the charateristics of a source in a sequence of time bins.

By default the lightcurve method will run an end-to-end analysis in each time bin using the same processing steps as the baseline analysis.

There are several options which cacn be used to reduce the lightcurve computation time.

- The multithread option splits the analysis of time bins across multiple cores
- The use_scaled_srcmap option generates an approximate source map for each time bin by scaling the source map of the baseline analysis by the relative exposure.

lightcurve() makes the analysis using different iterations.

- Leave free all the SED parameters of the sources in the model (normalization and shape).
- In the fit is not successful fix the shape.
- Then fixes the SED of the sources with increasingly higher TS.
- Then fix the sources beyond one degree from the center of the ROI.

Lightcurve analysis

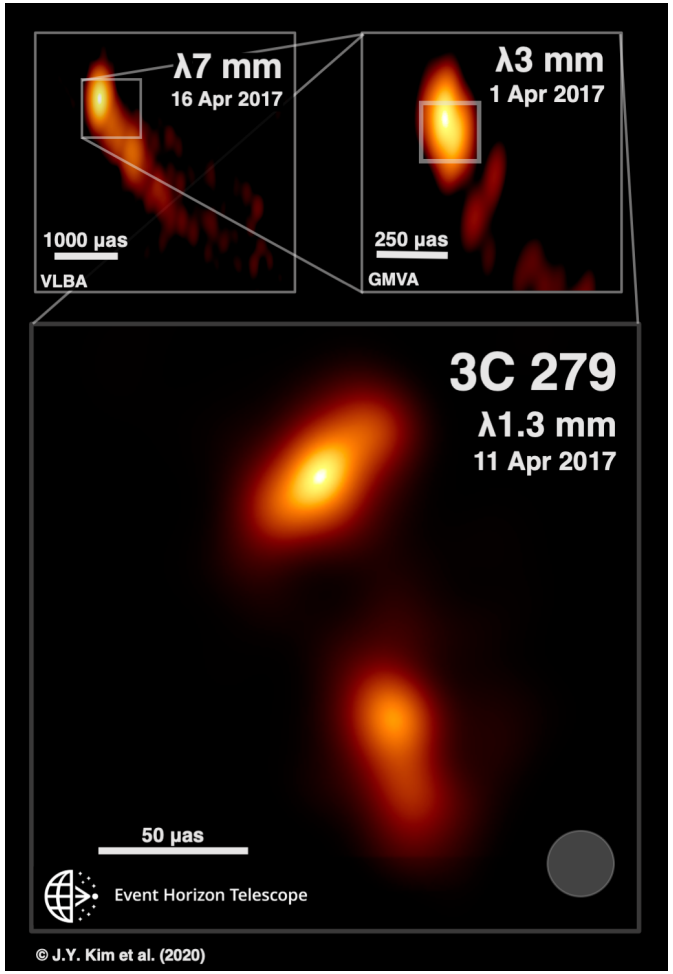
Parameters:

- **name** (*str*) – source name
- **binsz** (*float*) – Set the lightcurve bin size in seconds. (default : 86400.0)
- **free_background** (*bool*) – Leave background parameters free when performing the fit. If True then any parameters that are currently free in the model will be fit simultaneously with the source of interest. (default : False)
- **free_params** (*list*) – Set the parameters of the source of interest that will be re-fit in each time bin. If this list is empty then all parameters will be freed. (default : None)
- **free_radius** (*float*) – Free normalizations of background sources within this angular distance in degrees from the source of interest. If None then no sources will be freed. (default : None)
- **free_sources** (*list*) – List of sources to be freed. These sources will be added to the list of sources satisfying the free_radius selection. (default : None)
- **make_plots** (*bool*) – Generate diagnostic plots. (default : False)
- **max_free_sources** (*int*) – Maximum number of sources that will be fit simultaneously with the source of interest. (default : 5)
- **multithread** (*bool*) – Split the calculation across number of processes set by nthread option. (default : False)
- **nbins** (*int*) – Set the number of lightcurve bins. The total time range will be evenly split into this number of time bins. (default : None)
- **nthread** (*int*) – Number of processes to create when multithread is True. If None then one process will be created for each available core. (default : None)
- **outdir** (*str*) – Store all data in this directory (e.g. "30days"). If None then use current directory. (default : None)
- **save_bin_data** (*bool*) – Save analysis directories for individual time bins. If False then only the analysis results table will be saved. (default : True)
- **shape_ts_threshold** (*float*) – Set the TS threshold at which shape parameters of sources will be freed. If a source is detected with TS less than this value then its shape parameters will be fixed to values derived from the analysis of the full time range. (default : 16.0)
- **systematic** (*float*) – Systematic correction factor for TS:subscript: `var` . See Sect. 3.6 in 2FGL for details. (default : 0.02)

```
# split lightcurve across 2 cores
lc = gta.lightcurve('sourceA', nbins=2, multithread=True, nthread=2)
```

Key	Type	Description
`name`	`str`	Name of Source,
`tmin`	`ndarray`	Lower edge of time bin in MET.
`tmax`	`ndarray`	Upper edge of time bin in MET.
`fit_success`	`ndarray`	Did the likelihood fit converge? True if yes.
`config`	`dict`	Copy of the input configuration to this method.
`ts_var`	`float`	TS of variability. Should be distributed as χ^2 with $n-1$ degrees of freedom, where n is the number of time bins.

Lightcurve analysis: examples – 3C279

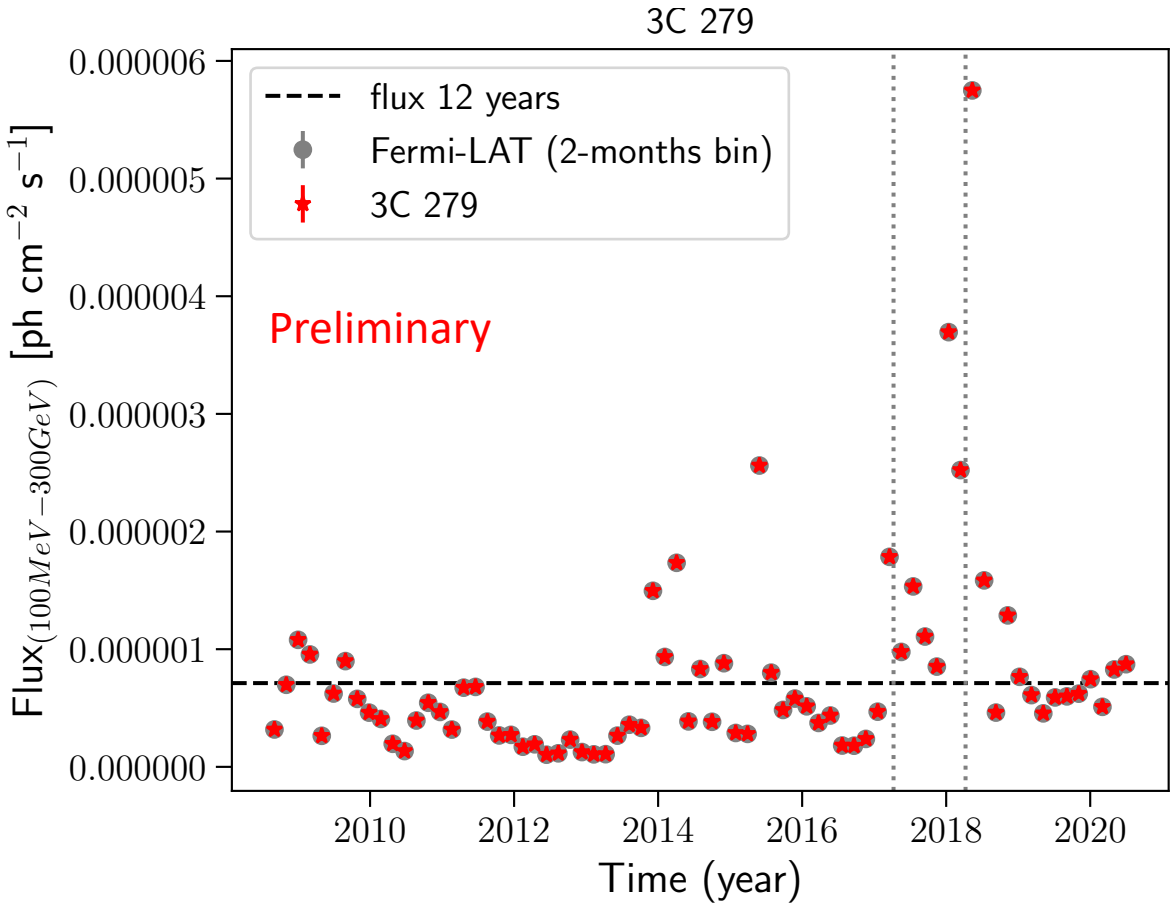


Flux 12 years (2008-2020):
F = (7.12 ± 0.03) x10^{-7} ph cm^{-2} s^{-1}
TS=232890

Flux 1 month (25/3 – 25/4 2017):
F = (20.4 ± 0.6) x10^{-7} ph cm^{-2} s^{-1}
TS=9254

Flux 1 month (8/4 –8/5, 2018):
F = (63.2 ± 0.8) x10^{-7} ph cm^{-2} s^{-1}
TS=51815

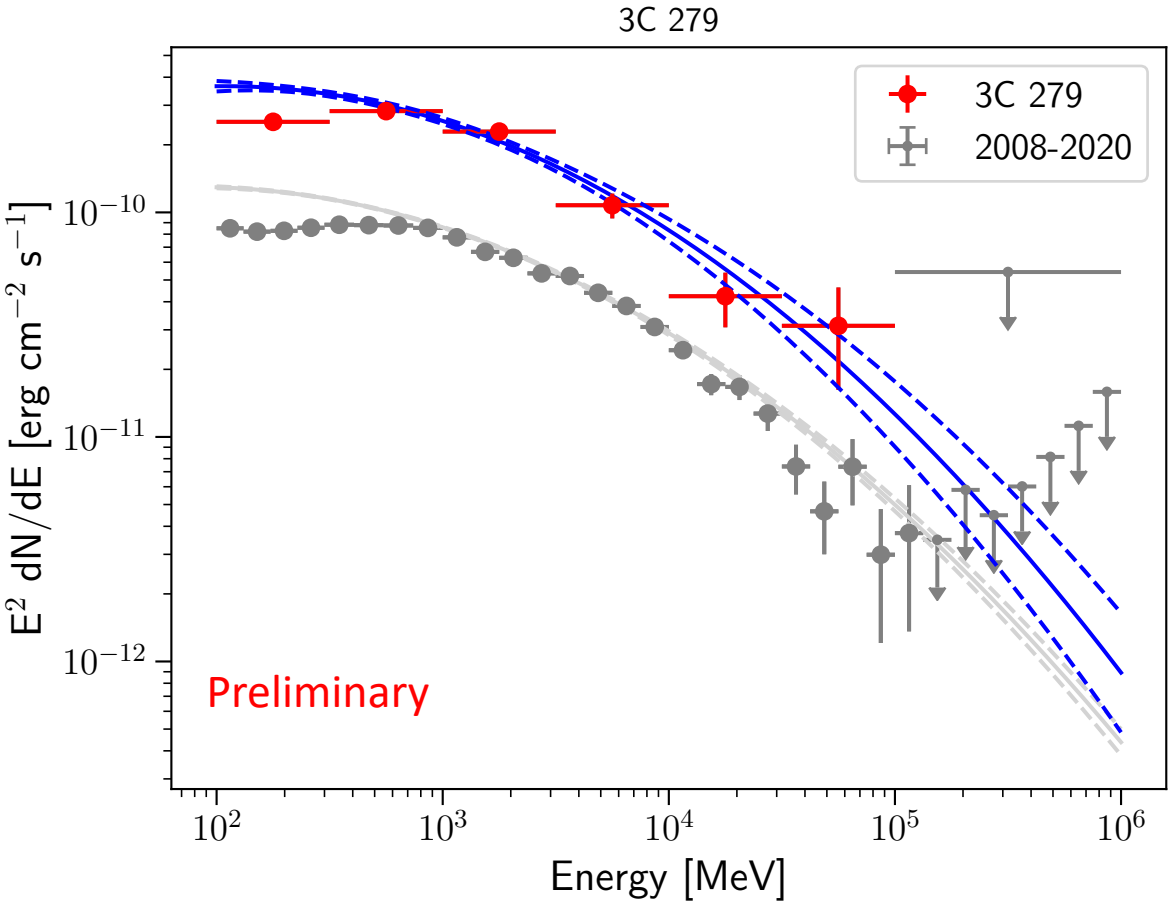
2 months bin

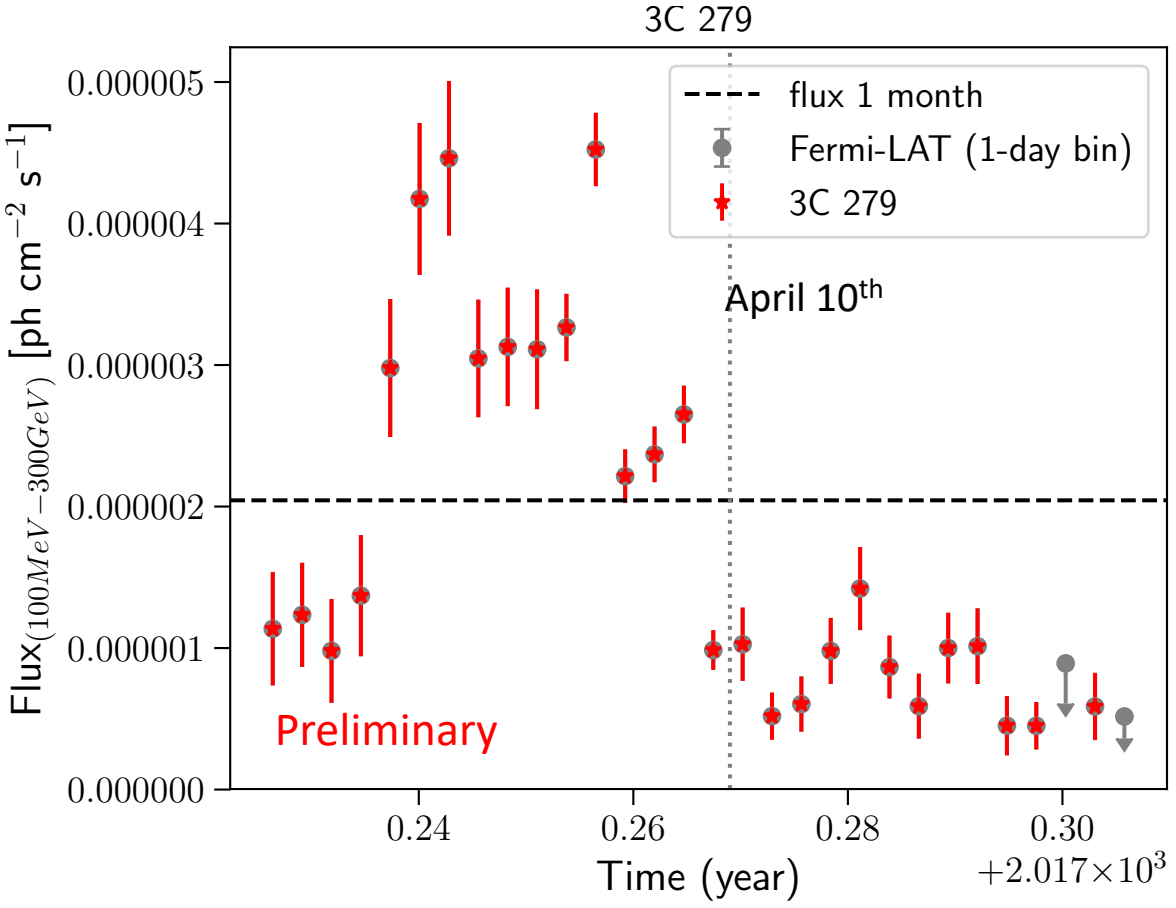


Lightcurve analysis: examples – 3C279

LogParabola 12 years: *Alpha*: 2.32±0.04, Beta: 0.063 ±0.0257 E_0: 1 GeV, N_0: 5.53 ±0.03 x10^{-11} erg cm^{-2} s^{-1}

LogParabola* 25th March – 25th April, *2017: *Alpha*: 2.32±0.03, Beta: 0.07±0.02, E_0: 1 GeV, N_0: 1.60 ±0.05 x10^{-10} erg cm^{-2} s^{-1}

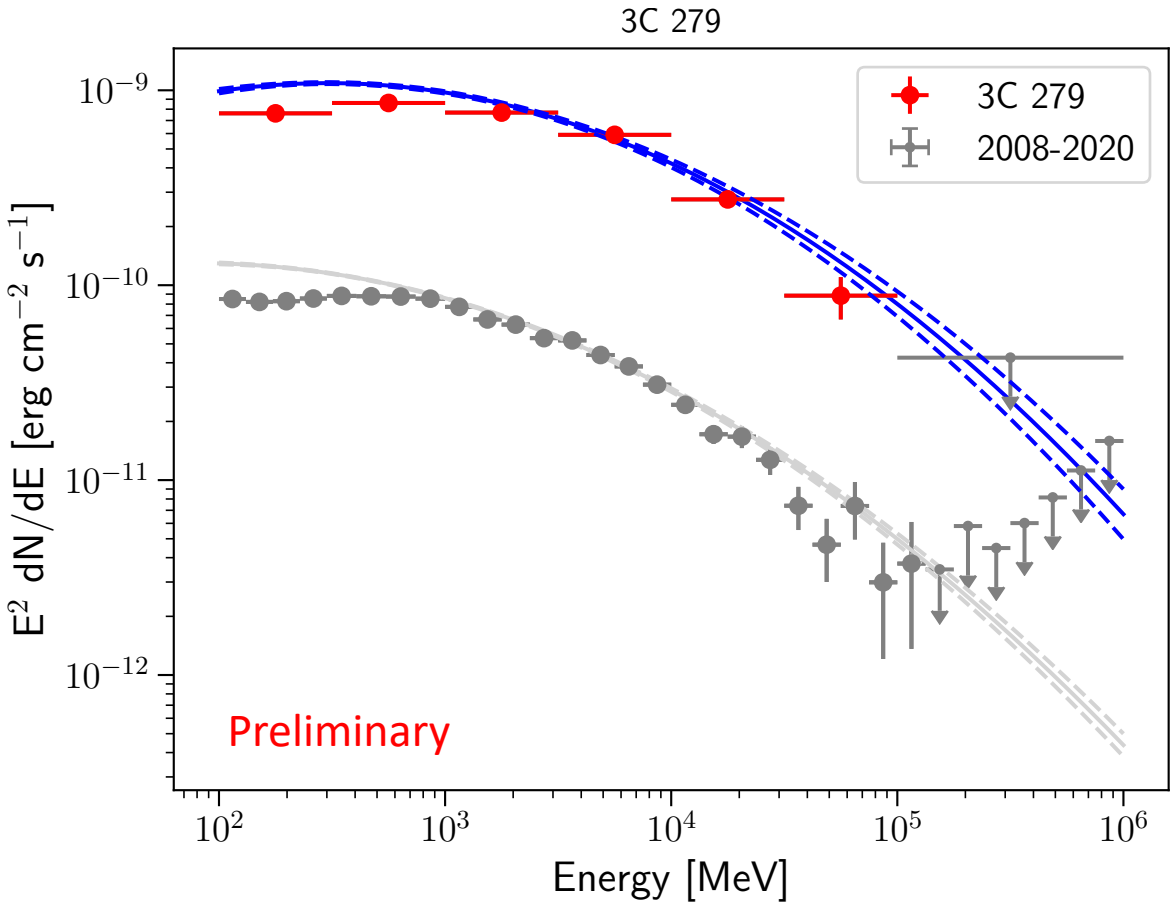


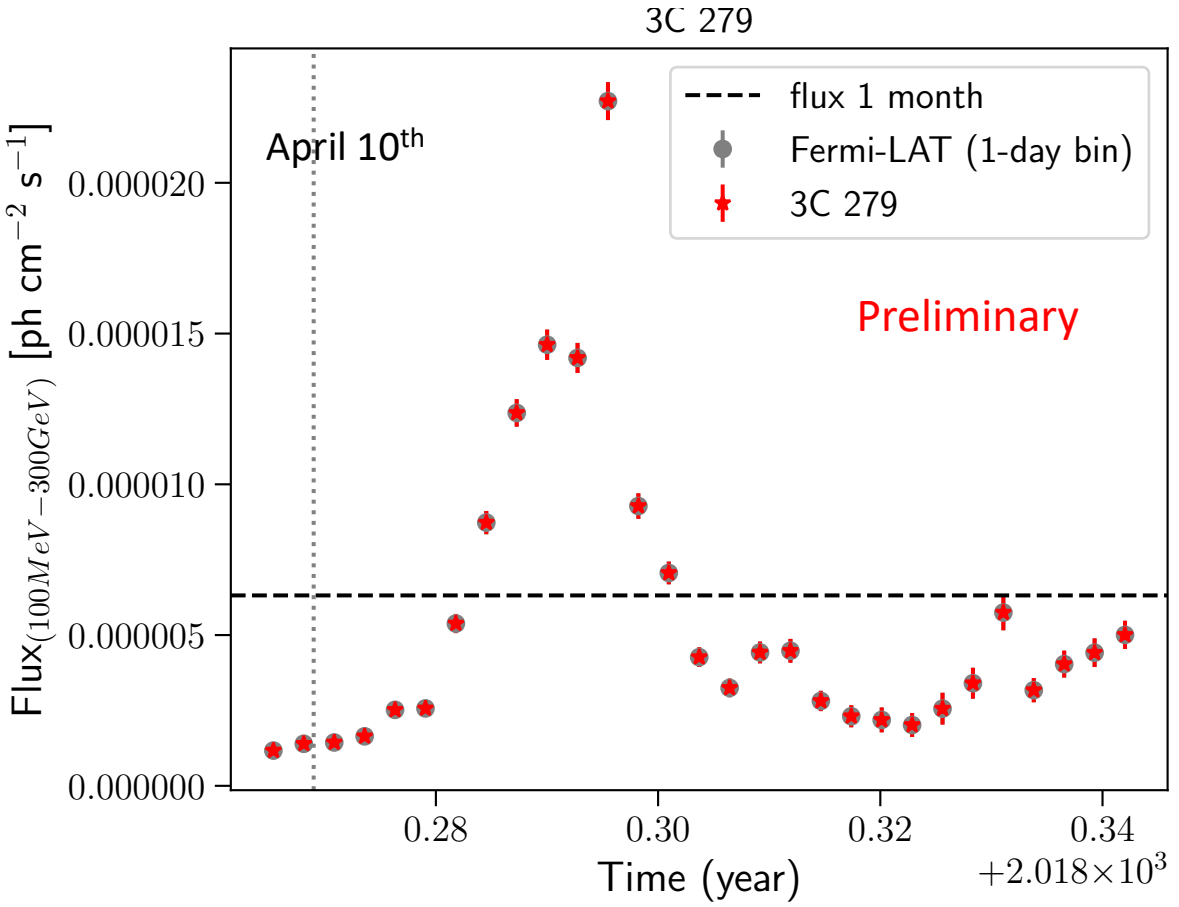


Lightcurve analysis: examples – 3C279

LogParabola 12 years: *Alpha*: 2.32±0.04, Beta: 0.063 ±0.0257 E_0: 1 GeV, N_0: 5.53 ±0.03 x10^{-11} erg cm^{-2} s^{-1}

LogParabola 8th April – 8th May, 2018: *Alpha*: 2.19±0.01, Beta: 0.076±0.007, E_0: 1 GeV, N_0: 6.09 ±0.09 x10^{-10} erg cm^{-2} s^{-1}





Extension analysis

The *extension()* method executes a source extension analysis for a given source by computing a likelihood ratio test with respect to the no-extension (point-source) hypothesis and a best-fit model for extension. The best-fit extension is found by performing a likelihood profile scan over the source width (68% containment) and fitting for the extension that maximizes the model likelihood.

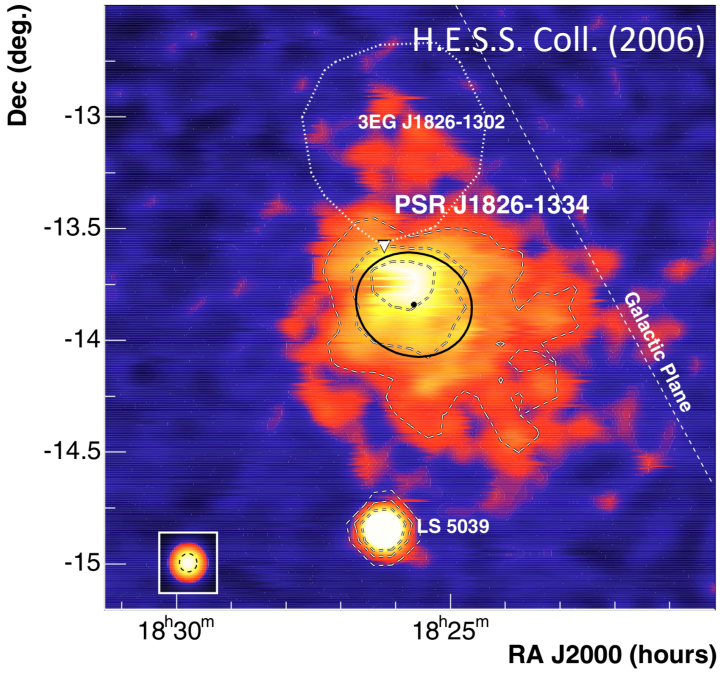
```
# Free a nearby source that maybe be partially degenerate with the
# source of interest.  The normalization of SourceB will be refit
# when testing the extension of sourceA
gta.free_norm('sourceB')
gta.extension('sourceA', free_background=True)

# Fix all background parameters when testing the extension
# of sourceA
gta.extension('sourceA', free_background=False)

# Free normalizations of sources within 2 degrees of sourceA
gta.extension('sourceA', free_radius=2.0)
```

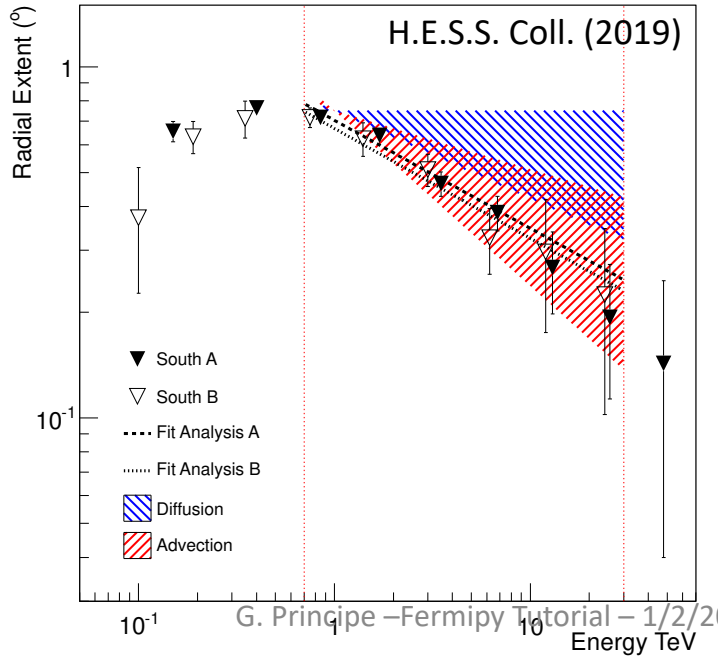
Extension analysis: example – The Pulsar Wind Nebula HESS J1825-137

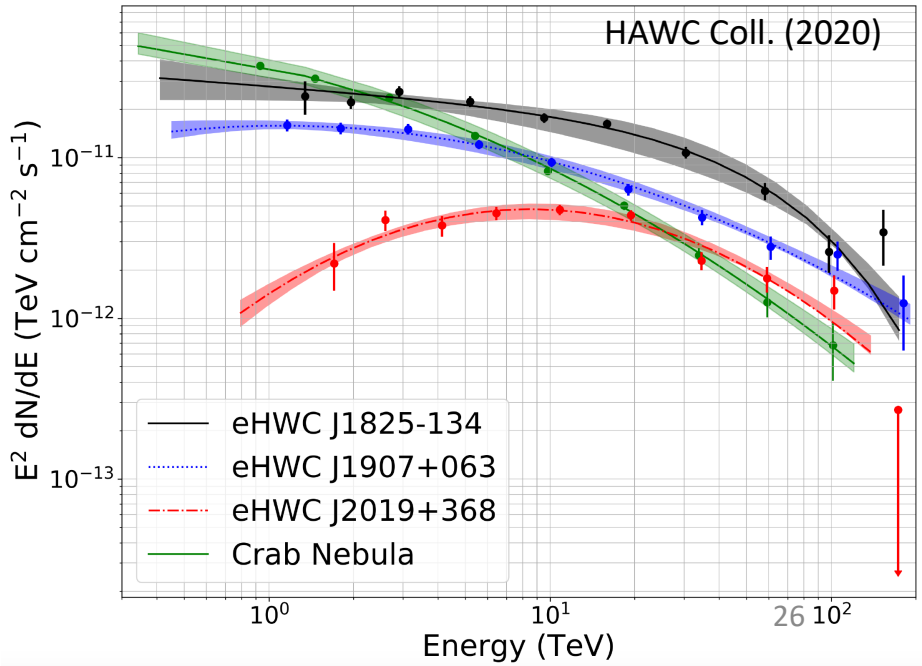
- First detected in the H.E.S.S. Galactic Plane Survey (2005)
- XMM-Newton/Suzaku: reveal diffuse x-ray emission size~0.1°
- Recent H.E.S.S. results (2019):
  - *with a size >100 pc is the largest PWN currently known*
  - TeV energy dependent morphology
- HESS J1825-137 is one of the three sources detected above 100 TeV by HAWC (2020), making it a promising Pevatron candidate.

Powered by the pulsar PSR J1826-1334 (PSR B1823-13):

- *Characteristic age = 21 kyr*
- *Period = 101 ms*
- *Distance = 4kpc*







Extension analysis: example – The Pulsar Wind Nebula HESS J1825-137

We performed the *first energy dependent extension* and spectral analysis of HESS J1825-137 in the GeV domain using **11.6 years** of ***Fermi-LAT*** data.

Principe et al. 2020, A&A, 640, 13

Analysis procedure:

1) General analysis (on the entire energy range 1 GeV - 1 TeV):

- Optimization, localization, and spectral analysis

2) Energy-resolved morphological study (2DGaussian, Radial profile)

Extension analysis in 5 energy bins (4 bins in 1-100 GeV, 1 bin in 100 GeV – 1TeV)

The initial model taken from the FGES paper (Ackermann et al. 2018):

Spatial Model: **2DGaussian**

- Sigma = 0.79°
- RA = 276.296°
- DEC = -13.992°

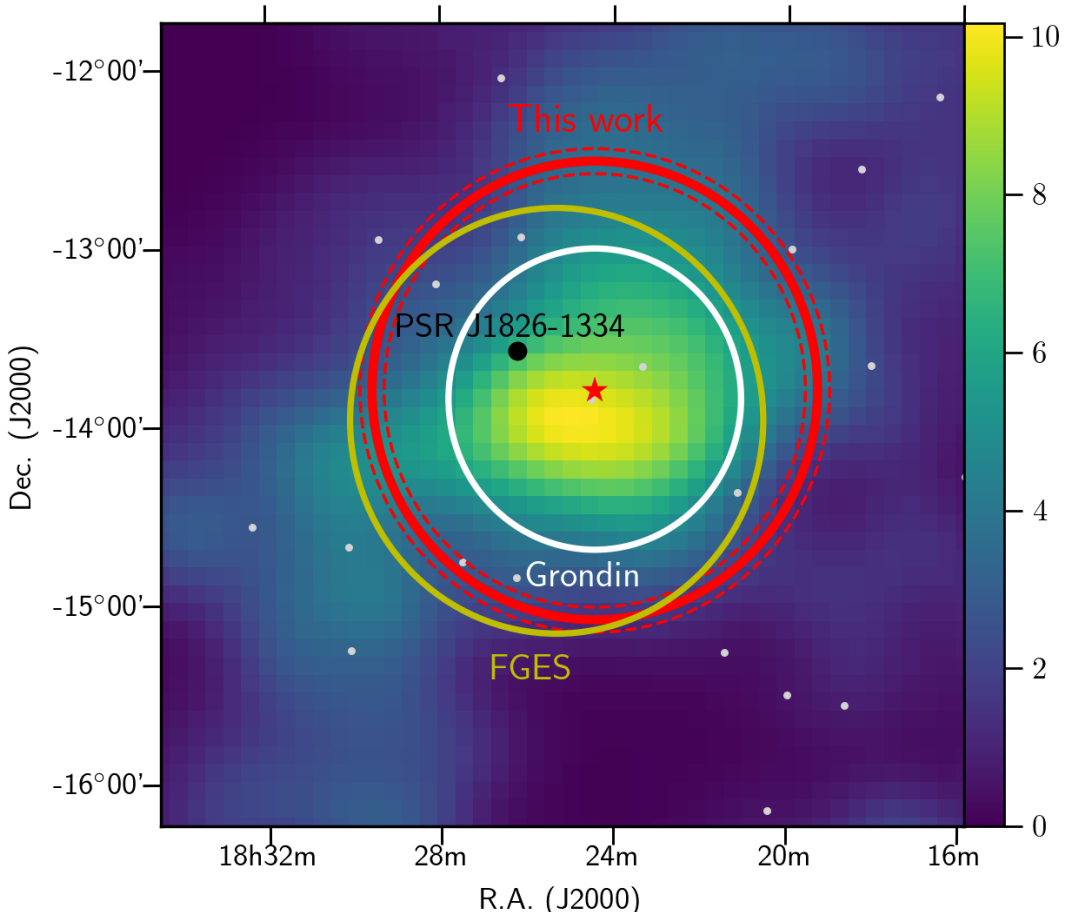
Spectrum Type: **LogParabola**

Models: 4FGL, standard LAT diffuse model, and optimized model for the Galactic plane *(*Ackermann et al. *2017)*

Data Selection	Values
IRFs	P8R3v2
Time Interval	11.6 years
Energy Range	1 GeV – 1 TeV
Energy Bins	8 per decade (for spectra)
Zenith angle	105°
ROI (pixel) size	15° (0.1°)

Extension analysis: example – The Pulsar Wind Nebula HESS J1825-137

RA, DEC: (276.11, -13.80)
Extension: 1.30 ° ± 0.06° (~150 pc)
(TS_{ext}=1040, ~30σ)



Fermi-LAT previous studies. **Grondin+(2011):** 20 months 1 – 100 GeV.
FGES (2017): 6 years, 10 GeV – 1 TeV

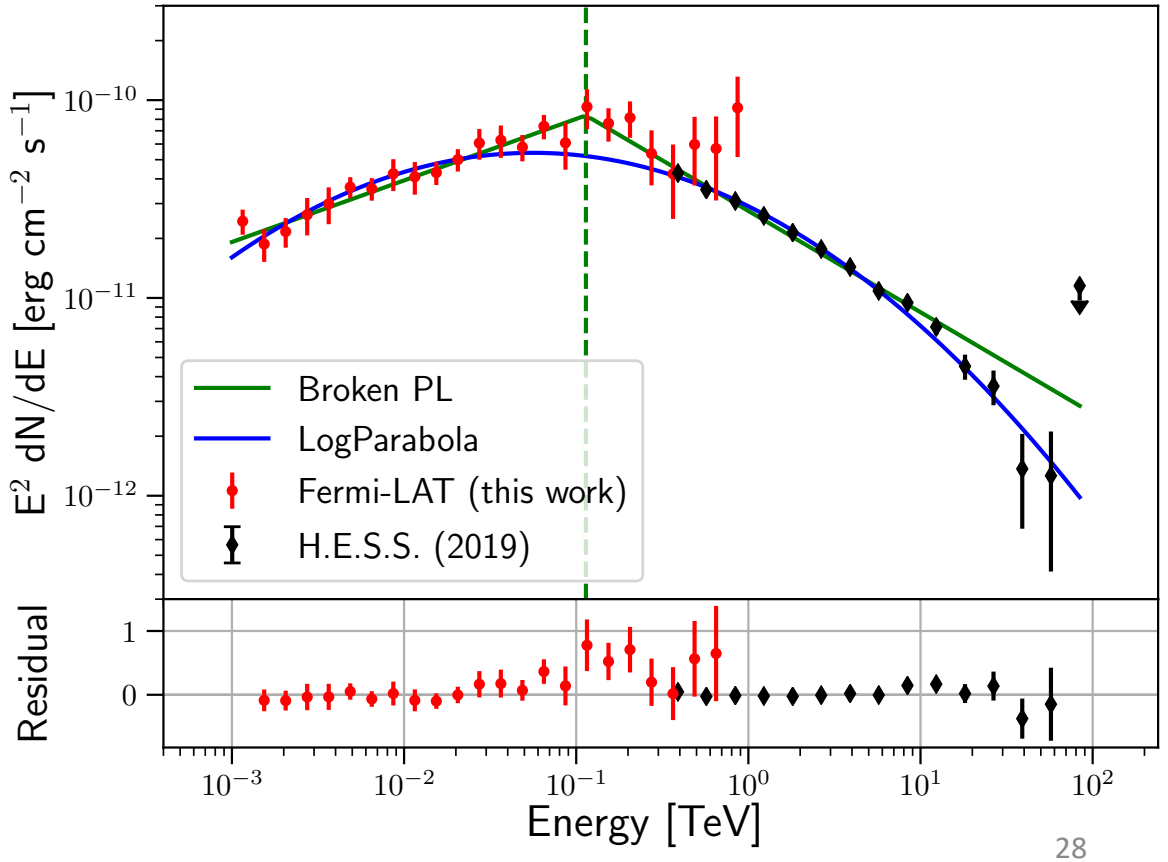
Principe et al. 2020, A&A, 640, 13

LogParabola

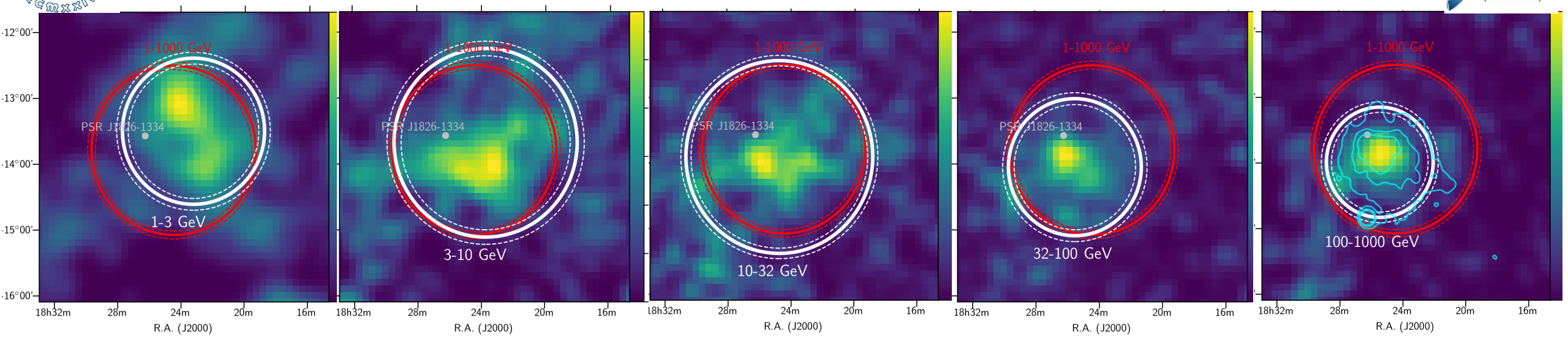
Alpha: 2.15±0.05, Beta: 0.075±0.02, E_0: 154 GeV, N_0: 6.0 x10^{-11} erg cm^{-2} s^{-1}

Broken PL

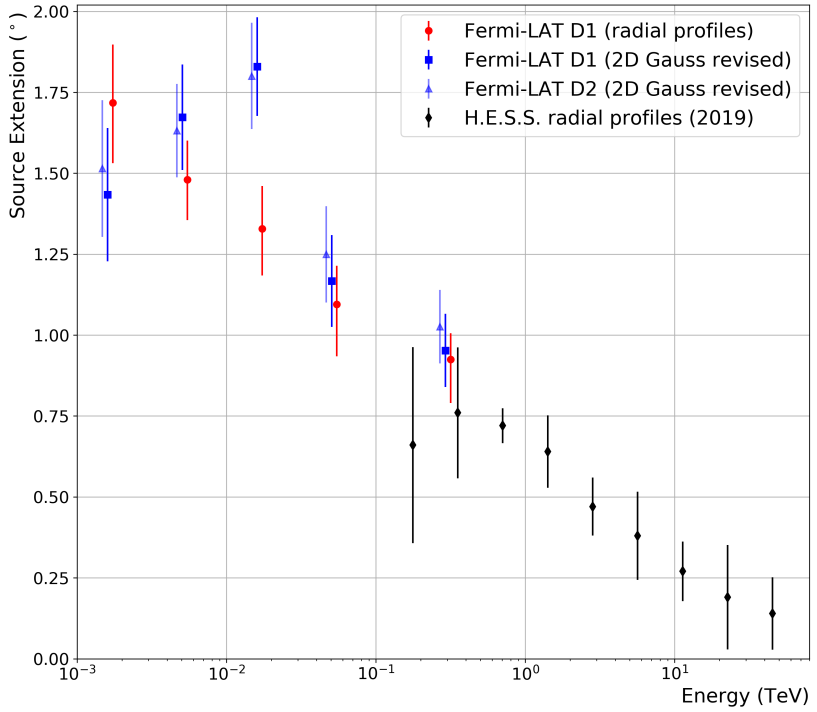
Γ_1: 1.69±0.03, Γ_2: 2.51±0.01, **E_0: 114 GeV,** N_0: 8.37 x10^{-11} erg cm^{-2} s^{-1}



Extension analysis: example – The Pulsar Wind Nebula HESS J1825-137



Principe et al. 2020, A&A, 640, 13



Radial profile method: radial distance at which the emission drops to 1/e relative to the maximum *starting from the PSR position* (only in one hemisphere due to the asymmetry of the PWN).

Source finding

find_sources() is an iterative source-finding algorithm that uses peak detection on a TS map to find new source candidates.

```
model = {'Index' : 2.0, 'SpatialModel' : 'PointSource'}
srcs = gta.find_sources(model=model, sqrt_ts_threshold=5.0,
                        min_separation=0.5)
```

Checks on the fit

Fit quality: A "good" fit corresponds to a value of "fit quality= 3"; if you get alower value it is likely that there is a problem with the error matrix.

According to the Minuit documentation possible values for "fit quality" are:

0 - Error matrix not calculated at all

1 - Diagonal approximation only, not accurate

2 - Full matrix, but forced positive-definite (i.e. not accurate)

3 - Full accurate covariance matrix (After MIGRAD, this is the indication of normal convergence.)

• TS and residual map: running gta.tsmap and gta.residmap you can make the TS and residual maps.

• gta.write_roi(): This makes a series of plots such as: count map, count spectrum

Checklist for your analysis

This is a list of things you should check in your analysis:

- The TS and/or residual map should not have structured or large scale residuals.
- The fit quality should be at least 2.
- Check covariance matrix and correlation coefficients.
- The count spectrum should has residuals of a few % level.

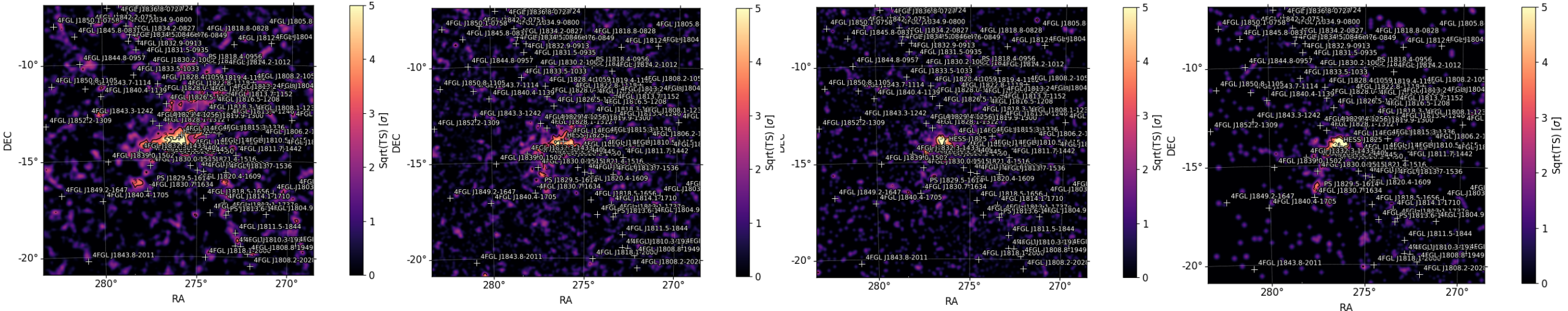
A few suggestions:

- Include sources from the catalog at least a few degrees beyond the edge of the ROI.
- If the fit is not successful (fit quality 0 or 1 and maybe also with 2), run a few times gta.optimize()
- Run the search for new sources gta.find_sources().
  - An iterative source-finding algorithm that uses likelihood ratio (TS) maps of the region of interest to find new sources. After each iteration a new TS map is generated incorporating sources found in the previous iteration. The method stops when the number of iterations exceeds max_iter or no sources exceeding sqrt_ts_threshold are found.

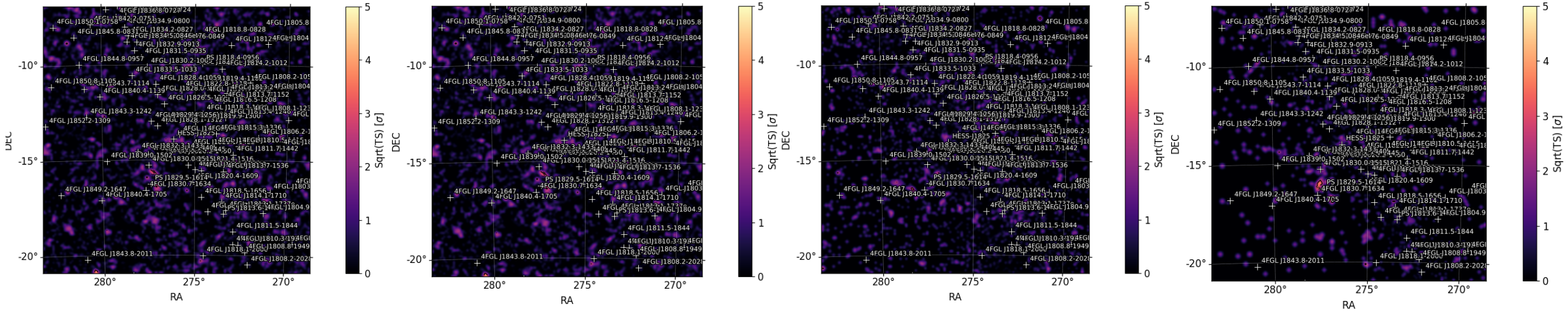
TS map – HESS J1825

Excess maps: Data – model excluding the studied source

3-10 GeV | 10-30 GeV | 30-100 GeV | 100-1000 GeV



Residual maps: Data – model excluding the studied source



If the fit is not successful

If the fit is not successful there are a few things you may try.

- Run a few time gta.optimize() and then gta.fit()
- See if there are SED parameters of some source that take crazy values (gta.get_params).
- If you are running an analysis in a small energy range (1-10 GeV, 1-100 GeV, above 10 GeV,…) substitute the SED of the sources from PLEC or LP to PL.
- Run the fit to each source leaving free to vary only the sources within a few degree.
- ……