PROCEEDINGS OF SCIENCE

Fermipy: An open-source Python package for analysis of *Fermi*-LAT Data

Matthew Wood*
Kavli Institute for Particle Astrophysics and Cosmology, SLAC National Accelerator Laboratory
E-mail: mdwood@slac.stanford.edu

Regina Caputo
NASA, GSFC
E-mail: regina.caputo@nasa.gov

Eric Charles
Kavli Institute for Particle Astrophysics and Cosmology, SLAC National Accelerator Laboratory
E-mail: echarles@slac.stanford.edu

Mattia Di Mauro
Kavli Institute for Particle Astrophysics and Cosmology, SLAC National Accelerator Laboratory
E-mail: mdimauro@slac.stanford.edu

Jeffrey Magill
Department of Physics and Department of Astronomy, University of Maryland, College Park
E-mail: jmagill@umd.edu

Jeremy Perkins
NASA, GSFC
E-mail: jeremy.s.perkins@nasa.gov

on behalf of the *Fermi*-LAT Collaboration

Fermipy is an open-source python framework that facilitates analysis of data collected by the Fermi Large Area Telescope (LAT). Fermipy is built on the Fermi Science Tools, the publicly available software suite provided by NASA for the LAT mission. Fermipy provides a high-level interface for analyzing LAT data in a simple and reproducible way. The current feature set includes methods for extracting spectral energy distributions and lightcurves, generating test statistic maps, finding new source candidates, and fitting source position and extension. Fermipy leverages functionality from other scientific python packages including NumPy, SciPy, Matplotlib, and Astropy and is organized as a community-developed package following an open-source development model. We review the current functionality of Fermipy and plans for future development.

35th International Cosmic Ray Conference -ICRC217-
10-20 July, 2017
Bexco, Busan, Korea

*Speaker.

1. Introduction

The *Fermi* Large Area Telescope (LAT) is a pair-conversion telescope that is sensitive to gamma rays from below 20 MeV to more than 300 GeV. Since the launch of the *Fermi* observatory in June 2008, the LAT has operated primarily in all-sky survey mode and has now collected more than eight years of data. LAT data classified as photon-like (satisfying the criteria of the photon event classes) are immediately released to the public through the LAT data archive[1] at the Fermi Science Support Center (FSSC). The FSSC also supports a suite of public software tools, the Fermi *ScienceTools* [2], for the reduction and analysis of LAT data. The *ScienceTools* is primarily written in C++ but includes a python interface (`pyLikelihood`) to facilitate scripting analysis in python.

Fermipy is a python software package that provides a high-level interface for LAT data analysis. *Fermipy* is based on the *ScienceTools* and uses `pyLikelihood` to interface with the underlying C++ library. *Fermipy* relies on a number of other open-source python libraries including NumPy [1], Scipy [2], and Astropy [3]. Matplotlib [4] is an optional dependency that can be used to generate analysis visualizations.

Fermipy is organized around an open-source development model and is available to all members of the LAT scientific community. The *Fermipy* source code is hosted on GitHub[3] and is licensed under a 3-clause BSD-style license. Bug reports and proposals for new functionality should be made through the GitHub Issue tracker. Code contributions are accepted via pull requests.

This proceeding provides a short review of some of the main features of *Fermipy*. A more complete and detailed description of the package as well as tutorials are available in the online documentation.[4]

2. Installation

Releases of *Fermipy* are made available through both the pip and conda package manager tools. Instructions for pip- and conda-based installation are provided in the online documentation. [5] Both installation methods require an existing installation of *ScienceTools* provided as precompiled binaries from the FSSC. Pre-built Docker images containing a current release of *Fermipy*, the Fermi *ScienceTools*, and Anaconda python are also provided.

3. Analysis Workflow

Fermipy is designed around a global analysis object (GTAnalysis) that manages the data and model preparation and provides a set of high-level analysis methods. A user executes an analysis by composing a python script that creates a GTAnalysis instance and calls the methods of this object to perform different analysis tasks. A user starts a new analysis by first composing a configuration file

[1] https://fermi.gsfc.nasa.gov/cgi-bin/ssc/LAT/LATDataQuery.cgi
[2] https://fermi.gsfc.nasa.gov/ssc/data/analysis/software/
[3] https://github.com/Fermipy/fermipy
[4] http://fermipy.readthedocs.org/
[5] http://fermipy.readthedocs.org/en/latest/install.html

```yaml
data:
  evfile : ft1.lst
  scfile : ft2.fits
binning:
  roiwidth   : 10.0
  binsz      : 0.1
  binsperdec : 8
selection :
  emin : 100
  emax : 316227.76
  zmax    : 90
  evclass : 128
  evtype  : 3
  tmin    : 239557414
  tmax    : 428903014
  target : 'mkn421'
gtlike:
  edisp : True
  irfs : 'P8R2_SOURCE_V6'
  edisp_disable : ['isodiff','galdiff']
model:
  src_roiwidth : 15.0
  galdiff  : '$FERMI_DIFFUSE_DIR/gll_iem_v06.fits'
  isodiff  : 'iso_P8R2_SOURCE_V6_v06.txt'
  catalogs : ['3FGL']
```

Figure 1: A YAML configuration for a *Fermipy* analysis of the gamma-ray blazar Mkn 421.

that defines analysis parameters including the data selection, region-of-interest (ROI) geometry, and model specification. *Fermipy* uses the YAML format for its configuration files and groups related parameters into blocks. A sample configuration file is shown in Figure 1. Configurations can easily be constructed programatically with a YAML serialization and parsing library such as PyYAML.

Figure 2 shows a sample analysis script that could be using in conjunction with the configuration file shown in Figure 1. The configuration file sets the parameters used to initialize the analysis object and its path is passed to the analysis object constructor. The `setup` method is called to run the data and model preparation by executing the appropriate gt-tools: *gtselect*, *gtmktime*, *gtbin*, *gtltcube*, *gtexpcube2*, and *gtsrcmaps*.

Once the analysis object is initialized, a typical analysis sequence involves optimizing the parameters of the background model, generating a TS or residual map of the region to assess the quality of the model, and extracting characteristics for a source of interest (flux, TS, spectral fit parameters, SED, etc.). In the example script shown in Figure 2, the `optimize` method is used to obtain a reasonable baseline model for the region by fitting the parameters of all components in the ROI. The `localize` and `sed` methods are then used to determine the best-fit position of Mkn 421 and extract its spectral energy distribution (SED). Finally the `write_roi` method is used to write the results of the analysis to an output file.

Many of the *Fermipy* methods return a results dictionary that can be used for subsequent post-processing or visualization. Results can also be serialized to a FITS file by passing `write_fits=True` or a numpy file by passing `write_npy=True`. The analysis state can be saved with the `write_roi` method which generates a FITS file containing a table with one row per source in the ROI. This file

```python
# Initialize the analysis object
from fermipy.gtanalysis import GTAnalysis
gta = GTAnalysis('config.yaml')
# Setup the analysis
gta.setup()
# Optimize spectral parameters
gta.optimize()
# Localize a source of interest
loc = gta.localize('Mkn421', make_plots=True)
# Extract an SED for a source of interest
sed = gta.sed('Mkn421', make_plots=True)
# Generate a TS map of the region
tsmap = gta.tsmap()
# Save the current analysis state to a file
gta.write_roi('fit0')
```

Figure 2: An analysis script demonstrating the sequence of methods that would be used to perform a basic analysis of an individual source (Mkn 421).

can be used in conjunction with the `load_roi` method to restore the analysis to a previous state.

4. Analysis Methods

Fermipy provides a number of high-level analysis methods that automate common analysis tasks. The following sections provide a brief description of some of these methods.

4.1 sed

The `sed` method computes the spectral energy distribution (SED) of a source in the ROI model. The first argument is the name of a source component. Optional arguments can be provided to control energy binning and how nearby background sources will be handled in the fit. The SED is computed by replacing the spectral model of the source with a power law and performing an independent fit of its normalization in each analysis energy bin. By default the method will fit the source normalization using a power-law with $\Gamma = 2$. The power-law index used to fit the source flux in each bin can be controlled with the `bin_index` and `use_local_index` parameters.

The return value of the method is a dictionary containing the characteristics of the source in each energy bin: flux, flux errors, flux upper limits, TS, and predicted counts. When run with `make_plots=True`, the method generates diagnostic plots showing the comparison between the measured flux in each bin and the best-fit global parameterization of the source (see left panel of Figure 3). In each energy bin, a profile likelihood versus source flux is also extracted that can be used to compute flux confidence intervals under different assumptions or re-fit the global spectrum using a different spectral parameterization. The output FITS file is generated using the SED FITS file format documented at the `gamma-astro-data-formats` repository.[6]

4.2 extension

The `extension` method can be used to test a source for spatial extension. It accepts as its first argument the name of a source and optional arguments that control the spatial model and

[6] https://gamma-astro-data-formats.readthedocs.io/en/latest/spectra/flux_points/index.html

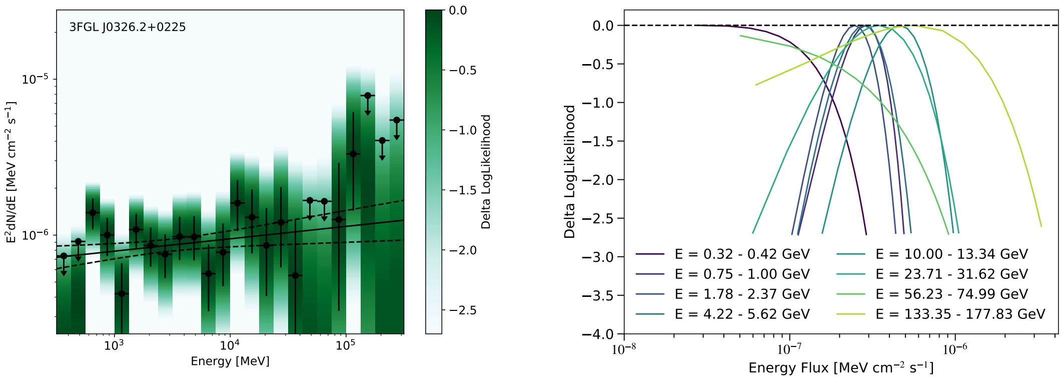

Figure 3: *Left:* Diagnostic plot generated by the `sed` method showing the comparison of the spectral points with the best-fit global parameterization. Black points show the best-fit flux or 95% flux upper limit measured in each analysis energy bin. The color scale indicates the likelihood profile versus source flux extracted in each energy bin. The black solid and dashed lines show the uncertainty band (butterfly) of the global parameterization. *Right:* Likelihood profiles versus energy flux in eight energy bins extracted from the FITS file for the SED shown in the left panel.

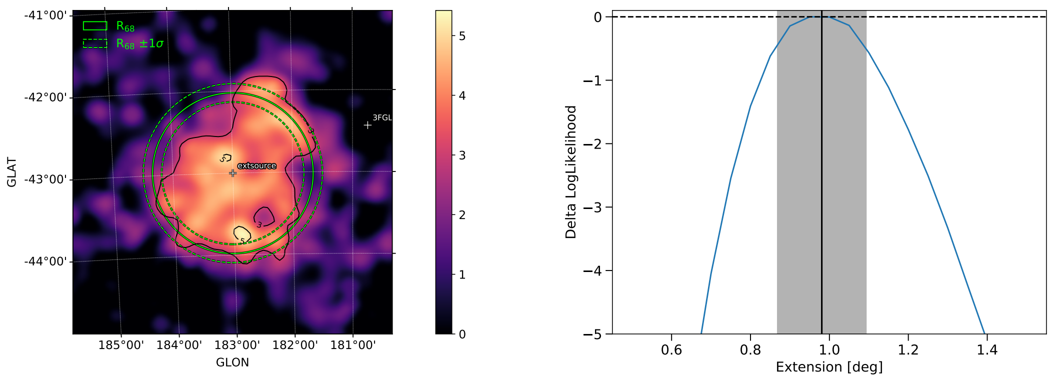

Figure 4: *Left:* Diagnostic plot generated by the `extension` method showing the residual TS map of the source overplotted with the best-fit extension (green solid and dashed lines). *Right:* Likelihood profile versus angular size generated by the `extension` method. The vertical black line and grey shaded region indicate the best-fit value and 1σ errors on the extension.

whether nearby background sources are profiled in the fit. The spatial extension test is performed by substituting the existing spatial model with an extended one (e.g. a 2D Gaussian or 2D Disk) and profiling the likelihood as a function of the spatial size parameter. The resulting likelihood profile is used to evaluate likelihood ratio with respect to a point-source model (TS_{ext}), the best fit extension, and errors and upper limits on the extension. Figure 4 shows some of the diagnostic plots generated by this method.

4.3 tsmap

The `tsmap` method can be used to rapidly generate a test statistic map of an ROI using the

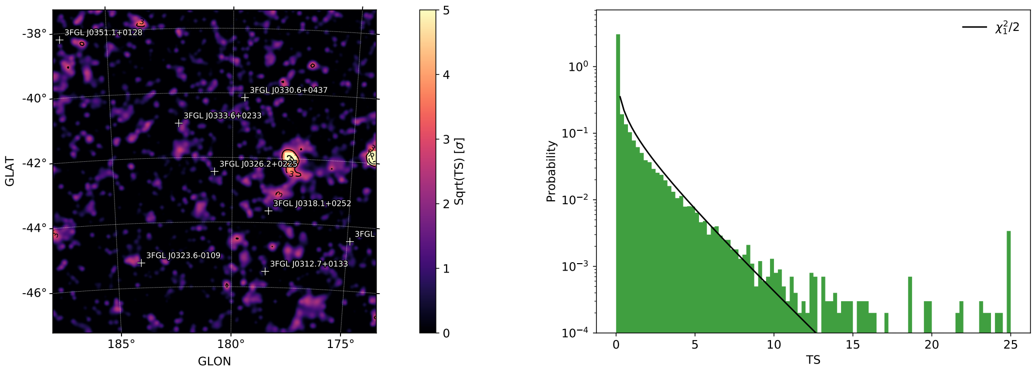

Figure 5: Diagnostic plots generated by the `tsmap` method. *Left:* Map of the test-source test statistic versus position in the ROI. TS isocontours are shown corresponding to . *Right:* Histogram of TS values from the TS map. The black curve shows the expected distribution for a likelihood ratio test with one degree of freedom..

same algorithm as the *gttsmap* tool. A test source is placed at the center of each spatial pixel and a maximum-likelihood fit is performed to determine its best-fit amplitude and TS. The spectral and spatial characteristics of the test source are controlled with the `model` parameter (by default a point source with power-law index of 2 will be used).

To reduce computation time, `tsmap` fixes the background model and restricts the calculation to pixels in the vicinity of the test source position. These simplifications allow TS maps to be generated much more rapidly than with the *gttsmap* application. A TS map of a typical ROI (e.g. a data cube with $30 \times 100 \times 100$ pixels) can be calculated on a single CPU core in a few minutes. When run with `make_plots=True` the method generates a set of diagnostic plots including those shown in Figure 5. The method also generates a FITS file containing maps of TS and test-source normalization.

4.4 find_sources

The `find_sources` method is an iterative source-finding algorithm that identifies new source candidates using the likelihood ratio method. In each iteration a TS map of the ROI is generated using the source model specified with the `model` argument. Candidate sources are identified from the N highest peaks in the TS map (where N is set with the `sources_per_iter` parameter) that have a TS greater than a certain threshold (`sqrt_ts_threshold`) and are at least a certain distance (`min_separation`) from another peak with higher TS. A new source is added at each peak satisfying these criteria with a position derived by fitting a parabola to the TS surface in a local patch around the source. After each iteration a new TS map is generated using an updated model including the sources added in the last iteration. The analysis is stopped when no additional source candidates are found or the number of iterations exceeds `max_iter`.

Figure 6 illustrates the application of this method to a blended pair of sources. The first iteration finds a source in the center of the map. When this source is added to the model, a second fainter candidate becomes visible that is identified as a point source in the second iteration.

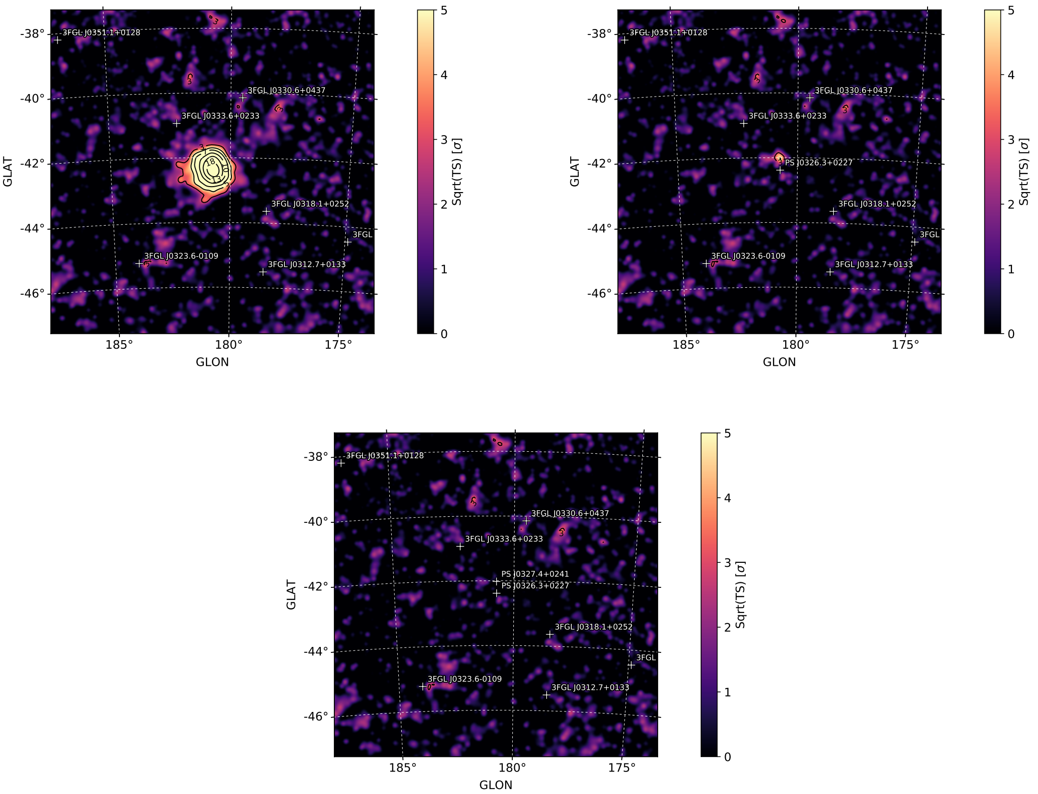

Figure 6: Illustration of the `find_sources` method. The top left panel shows the TS map of the ROI prior to running the method. The right and bottom panels show the TS maps obtained after the first and second iterations..

5. Conclusions

The *Fermipy* package provides an end-to-end tool for high-level analysis of LAT data that facilitates research on a broad range of topics (AGN, Galactic sources, DM searches). Although the package is now quite mature, development of *Fermipy* remains active. Future releases will focus on the following areas:

- Support for analysis of data pixelized with HEALPix.
- Functionality for evaluating systematic uncertainties from the effective area and PSF.
- Improved integration with other open-source gamma-ray analysis software such as *Gammapy* [5], *3ML* [6], and *Gammalib/ctools* [7].
- Improved support for multi-threaded analysis.
- Better standardization of FITS output formats through the `gamma-astro-data-formats` effort.

We aim to make *Fermipy* a community-supported effort and on this basis we welcome contributions from all members of the LAT scientific community.

Acknowledgments

The *Fermi*-LAT Collaboration acknowledges support for LAT development, operation and data analysis from NASA and DOE (United States), CEA/Irfu and IN2P3/CNRS (France), ASI and INFN (Italy), MEXT, KEK, and JAXA (Japan), and the K.A. Wallenberg Foundation, the Swedish Research Council and the National Space Board (Sweden). Science analysis support in the operations phase from INAF (Italy) and CNES (France) is also gratefully acknowledged.

References

[1] T. Oliphant, *Guide to NumPy*. Trelgol Publishing, 2006.

[2] E. Jones, T. Oliphant, P. Peterson et al., *SciPy: Open source scientific tools for Python*, 2001–.

[3] Astropy Collaboration, T. P. Robitaille, E. J. Tollerud, P. Greenfield, M. Droettboom, E. Bray et al., *Astropy: A community Python package for astronomy*, A&A **558** (Oct., 2013) A33, [1307.6212].

[4] J. D. Hunter, *Matplotlib: A 2d graphics environment*, *Computing in Science and Engg.* **9** (May, 2007) 90–95.

[5] A. Donath, C. Deil, M. P. Arribas, J. King, E. Owen, R. Terrier et al., *Gammapy: An open-source Python package for gamma-ray astronomy*, in *34th International Cosmic Ray Conference (ICRC2015)*, vol. 34 of *International Cosmic Ray Conference*, p. 789, July, 2015, 1509.07408.

[6] G. Vianello, R. J. Lauer, P. Younk, L. Tibaldo, J. M. Burgess, H. Ayala et al., *The Multi-Mission Maximum Likelihood framework (3ML)*, *ArXiv e-prints* (July, 2015) , [1507.08343].

[7] J. Knödlseder, M. Mayer, C. Deil, J.-B. Cayrou, E. Owen, N. Kelley-Hoskins et al., *GammaLib and ctools. A software framework for the analysis of astronomical gamma-ray data*, A&A **593** (Aug., 2016) A1, [1606.00393].