NovaFermi

Release 0.0

Peter Craig

Nov 25, 2025

CONTENTS:

CHAPTER

ONE

ANALYZE_FERMI MODULE

Written by Peter Craig (craigpe1@msu.edu) Last updated 11/15/25

Runs Fermi data analysis for novae Includes functions to: 1. Run a binned likelihood analysis to check TS value for a nova 2. Generate a light curve 3. Search for maximum TS value

`analyze_fermi.`**`FermiTools_UpperLim`**(*params*, *fheader*)

For comparison, here is the FermiTools Upper Limit mechanism

`analyze_fermi.`**`TS_Map`**(*params*, *input_file*, *clobber*)

Function to generate TS Maps. Will build two files, one with the full source model list, called name_TSmap_resid.fits, and one with the nova model removed, called name_TSmap_background_resid.fits.

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **input_file** (*string : name of xml input file*)

Return type

None

`analyze_fermi.`**`bin_data`**(*params*, *clobber*, *fheader*)

Function to generate counts maps / cubes

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **clobber** (*boolean : If true, overwrite existing files*)
- **fheader** (*string : Unique ID added to file names to avoid name conflicts*)

Return type

None

`analyze_fermi.`**`binned_likelihood`**(*params*, *tstart*, *tend*, *clobber=False*, *fheader=''*)

Function to run the full binned likelihood analysis pipeline Will run this once, and produces a TS value, and a flux :param params: :type params: dict : parameter dict from read_parameters :param tstart: :type tstart: float : time (MET) for data start :param tstart: :type tstart: float : time (MET) for data end :param clobber: :type clobber: boolean : If true, overwrite existing files :param fheader: :type fheader: string : Unique ID added to file names to avoid name conflicts

Return type

None

`analyze_fermi.`**`cal_to_met`**(*date_time*)

Function to compute Fermi MET

Parameters

`date_time` (*`datetime object : Should contain time you'd like to`*) – convert to MET

Returns

MET

Return type

float : Fermi MET in seconds

`analyze_fermi.`**`cleanup`**(*params*, *fheader*)

Short function to remove files produced during a given likelihood run. Intent is to reduce file volume when generating light curves.

WARNING! This function will delete files; do not run unless you are sure that you want to remove these files. Intended to cleanup all of the sizeable files produced during the run.

Parameters

- **`params`** (*`dict : parameter dict from read_parameters`*)
- **`fheader`** (*`string : file id`*)

`analyze_fermi.`**`compute_upper_lim`**(*params*, *fheader*)

Function to compute the upper limit on the flux for a source This is an implementation of the profile-likelihood method, makes an assumption about what is reasonable for nova fluxes. Probably a bit inefficient, but the algorithm is straightforward Uses a profile likelihood method to find upper limit at given CL Algorithm description: Begins by freezing spectral parameters to some standard nova selections (see setup_pl function that sets the actual model). Runs one optimizer to fit model with all nova params frozen except for the normalization. This provides the max likelihood (L0). The goal is then to find the (larger) flux where the likelihood L satisfies 1.35 = -2 * (log(L)-log(L0)), currently done using a simple bisection root finder that will find the root between the normalization parameter at L0 and the max allowed norm (which is unreasonably bright). Assumes that there is one root in the likelihood criterion. Different confidence levels correspond to different likelihood differences (not currently supported, we assume a 95% CL).

Parameters

- **`params`** (*`dict : parameter dict from read_parameters`*)
- **`model`** (*`string : name of input xml file`*)

Returns

ULF

Return type

float : upper limit flux

`analyze_fermi.`**`data_selection`**(*params*, *tstart*, *tend*, *clobber*, *fheader*)

Function to run the data selection Runs the gtselect and mktime FermiTools tasks to run the data selection Will produce the following FITS files: source_filtered.fits (gtselect) source_filtered_gti.fits (mktime)

Parameters

- **`params`** (*`dict : parameter dict from read_parameters`*)
- **`tstart`** (*`float : time (MET) for data end`*)
- **`tstart`**
- **`clobber`** (*`boolean : If true, overwrite existing files`*)

- **fheader** (*string : Unique ID added to file names to avoid name conflicts*)

Return type
None

`analyze_fermi.`**`fit_model`**(*params*, *fheader*, *get_like*, *inmod='No'*, *opt='NewMINUIT'*)

Function to run the model fitting steps This version is the recommended fitting process First runs a quick analysis with DRMNFB to get close to the approx sltn. Follows this with a NewMINUIT optimization run to finalize the model.

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **fheader** (*string : Unique ID added to file names to avoid name conflicts*)
- **get_like** (*boolean : If true, return logL*)

Return type
None

`analyze_fermi.`**`gen_model`**(*params*, *clobber*, *fheader*)

Function to create an input model file

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **clobber** (*boolean : If true, overwrite existing files*)

Return type
None

`analyze_fermi.`**`gen_srcmap`**(*params*, *clobber*, *fheader*)

Function to generate source maps for binned likelihood analysis

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **clobber** (*boolean : If true, overwrite existing files*)
- **fheader** (*string : Unique ID added to file names to avoid name conflicts*)

Return type
None

`analyze_fermi.`**`gen_ul_xml`**(*input_file*, *output_file*, *name*, *smodel*)

Simple function to setup the UL xml files
Parameters

name : string : name of our source flux : string : flux to install for our model

Return type
None

`analyze_fermi.`**`generate_residuals`**(*params*, *clobber*, *fheader*)

Function to create residuals between the counts map and the model map. Simply generates a model map with the FermiTools, then takes the difference between that and a similar counts map

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **clobber** (*boolean : If true, overwrite existing files*)
- **fheader** (*string : Unique ID added to avoid filename conflicts*)

Return type
None

`analyze_fermi.`**`light_curve_multiproc`**(*params*, *clobber*, *log='mp_log'*)

Function to build a light curve Uses window width in parameter file, and step size This is the multiprocessing version of this function

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **clobber** (*boolean : If true, overwrite existing files*)
- **log** (*string : base file name to load data*)

Return type
None

`analyze_fermi.`**`light_curve_singleproc`**(*params*, *clobber*, *log='results.csv'*)

Function to build a light curve Uses window width in parameter file, and step size This is the simple version of this function, useful for debugging any issues encountered during the analysis. Runs on a single core.

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **clobber** (*boolean : If true, overwrite existing files*)
- **log** (*string : file to save lc data*)

Return type
None

`analyze_fermi.`**`likelihood_wrapper`**(*run_pars*)

Simple wrapper function to run the likelihood analysis using only a single argument. Makes it easier for the multi-processing functions to run many likelihood analyses. Will automatically compute upper limits if the test statistic from the main fit is less than 4. Set up_lim_lc to no in the parameter file to disable this behavior.

Parameters
run_pars (*list : list of all likelihood parameters.*) – Should contain, in this order: params : dict : parameter dict from read_parameters tstart : float : time (MET) for data start tstart : float : time (MET) for data end clobber : boolean : If true, overwrite existing files fheader : string : Unique ID added to avoid filename conflicts log : filename to save data to cleanup : boolean : If true, delete large intermediate files

Return type
Flux , Flux_Error , TS

`analyze_fermi.`**`lt_exp_maps`**(*params*, *clobber*, *fheader*)

Generates livetime cubes and exposure maps for binned likelihood Fermi analysis

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **clobber** (*boolean : If true, overwrite existing files*)
- **fheader** (*string : Unique ID added to file names to avoid name conflicts*)

Return type
None

`analyze_fermi.`**`met_to_tpeak`**(*met*, *params*)

Function to compute the time in days since peak given a Fermi MET

Parameters

- **met** (*float : Fermi MET*)
- **params** (*dict : parameter dict from read_parameters*)

Returns
t_peak

Return type
float : time since peak in days

`analyze_fermi.`**`print_params`**(*params*)

Simple function to print out our parameters

Parameters
params (*dict : parameter dictionary from read_parameters*)

Return type
None

`analyze_fermi.`**`read_parameters`**(*pfile*)

Function to read analysis parameter file All parameter options should get set in this file See template parameter file for available parameters

Parameters
pfile (*string : name of parameter file*)

Returns
params

Return type
dict : contains all analysis options and parameters

`analyze_fermi.`**`setup_events_file`**(*clobber=False*)

Simple function to setup files listing all data, and identifies the spacecraft file. Just leave all data / spacecraft files in current directory and this will prep data as need be.

Parameters
clobber (*boolean : If true, will overwite any existing event list*)

Returns

- **infile** (*string : name of event filename*)
- **scfile** (*string : name of spacecraft file*)

`analyze_fermi.`**`setup_pl`**(*params*, *flux*, *index*, *free=False*)

Simple function to setup a power law model for a source used for upper limit computations

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **model** (*string : name of input xml file*)

Returns
model

Return type
string : string ready to be written into xml file

`analyze_fermi.`**`setup_tsmap_xml`**(*params*, *input_file*)

Function to build a xml input file suitable for computing background TSMaps. Basically just takes an xml file and strips the model for our source. Will also freeze out model parameters (otherwise, runtime quickly becomes intractable. Takes in the xml file for the model that we want to use.

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **input_file** (*string : name of xml input file*)

Return type
None

`analyze_fermi.`**`tpeak_to_met`**(*time*, *params*)

Function to compute Fermi MET, given a time relative to nova peak.

Parameters

- **time** (*float : Time since peak (negative for before peak) in days*)
- **params** (*dict : parameter dict from read_parameters*)

Returns
MET

Return type
float : Fermi MET in seconds

CHAPTER

TWO

PLOT_LC MODULE

Short routine to collect and plot the results of a light curve generated with analyze_fermi.py

`plot_lc.`**`plot_light_curve`**(*params*, *display=False*)

Function to plot light curve results from analyze_fermi Assumes that you are running in the directory with the data TODO: Update this routine to accept data directory arguments Takes in a parameter dictionary, and will produce a TS plot (Showing TS as a function of time) and will produce a light curve complete with uncertainties and upper limits as appropriate. All times will be plotted relative to the peak listed in the given parameter file.

Parameters

- **params** (*dict : parameter dict from read_parameters*)
- **display** (*boolean : If true, run plt.show() to display figures*)

Return type

None

CHAPTER THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX