In [1]:
```
#import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.colors as colors
import matplotlib as mpl
import numpy as np
import glob
import astropy.units as u
import tables
import gc
import time

from astropy.coordinates import SkyCoord, AltAz
from astropy.coordinates.erfa_astrom import ErfaAstromInterpolator, erfa_astrom
from astropy.time import Time
from lstchain.reco.utils import location, add_delta_t_key, get_effective_time
from os import path
from scipy.stats import binned_statistic
from pyirf.statistics import li_ma_significance
from lstchain.spectra.crab import crab_magic
from ctapipe.containers import EventType
from pyirf.spectral import CRAB_MAGIC_JHEAP2015
from ctapipe.coordinates import CameraFrame, TelescopeFrame
from gammapy.stats import WStatCountsStatistic
from ctapipe_io_lst import LSTEventSource
from ctapipe.io import read_table

%matplotlib inline

#%load_ext filprofiler
```

In [2]:
```
#
# This is a notebook to calculate the flux sensitivity using real Crab observations.
#
# NOTE: the inputs of this notebook are DL2 files of Crab observations,
# both for source-independent and source-dependent analyses.
#
# The source-independent files must contain the event-wise nominal position
# of Crab (src_x, src_y), in a table called "source_position". This can be
# achieved by processing standard DL2 files with $LSTCHAIN/scripts/lstchain_dl2_add_sourcepos.py
# (this is done like this because it is too costly to compute the position every time we need it)
#
#
# The samples below are those used for the ApJ LST-1 performance (or "Crab") paper:
# NOTE!! The full sample (34.2 h of effective time) takes about 3 hours to process in the IT cluster!
# The time is mostly to read the data and fill the "super-histograms" from which the event statistics
# are derived. One can fill the histograms once, and the run only the (much faster) second part of the
# notebook with different settings for the sensitivity calculation.
#
# source-independent dataset
tag1 = "source-independent"
#dataset1 = glob.glob("/fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2/process_wi
dataset1 = glob.glob("/fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_w

# Source-dependent dataset
# Created with RFs trained using gammas out to 1 deg, AND additional src-indep params:
tag2 = "source-dependent+, ring-wobble-0-1-trained"
dataset2 = glob.glob("/fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combi

dataset1.sort()
dataset2.sort()
```

In [3]:
```
#
# Settings for sensitivity calculations:
# (NOTE: one can change these afterwards, to obtain results with different conditions just
# running the second part of the notebook, with no need to re-read all data!!!)
#
alpha = 0.2 # Li & Ma's "alpha", ratio of the ON to the OFF exposure
# This what we *assume* for the calculation, because it is the standard value.
# Note however that for a single LST, and to avoid large systematics, we use only one off
# region to estimate the background (in source-dependent analysis; in source-independent
# analysis it is one region up to 250 GeV and 3 regions for >250 GeV). We then "extrapolate"
# that background to what we would have in case we really had alpha=0.2...


# Assumed (effective) observation time for the calculation (note: it has nothing to do with
# the actual observation time of the sample used for the calculation!):
obs_time = 50 * u.h
# obs_time = 1/60. * u.h # 1 minute
# obs_time = 0.083333333 * u.h  # 5 minutes
# obs_time = 0.5 * u.h
# obs_time = 0.7 * u.h
# obs_time = 5 * u.h
```

```
# The 5-sigma condition is so standard that we do not make it configurable! It is hardcoded.
# Additional sensitivity conditions:
min_n_gammas = 10 # Minimum number of excess events
min_backg_percent = 5 # Excess must be at least this percentage of the background
```

In [4]:
```
# Binnins of gammaness, Alpha, thetas and energy for all calculations:

# Very fine binnings to allow cut optimization:
gammaness_bins = np.linspace(0, 1, 2001)
alphabins = np.linspace(0, 60, 3001)
theta2bins = np.linspace(0, 0.6, 6001)

logenergy_bins = np.linspace(-2.0, 2.2, 22) # 5 per decade

evt_id_bins = [-0.5, 0.5, 1.5]
# for event_id%2 (to separate odd & even event numbers).
# Even event-id's end up in the first bin; odd event-id's in the second one.
# We separate them so that we can optimize the cuts in one and apply them to
# the other (and vice versa)

# For source-independent analysis we will use one off region up to 250 GeV.
# Three off regions for higher energies:
first_ebin_with_3offs = 7 # i.e. from ~250 GeV onwards

# Intensity cuts:
min_intensity = 50
max_intensity = 1e10
```

In [5]:
```
#
# CUTS TO ENSURE THE QUALITY OF THE REAL DATA EXCESS FROM WHICH THE SENSITIVITY WILL BE COMPUTED!
#
# We set here the conditions to ensure that all used histogram bins (defined by Ebin, gammaness cut and
# angular cut) have, in the data sample, a reliable excess which we can use for estimating sensitivity:
#
min_signi = 3   # below this value (significance of the test source, Crab, for the *actual* observation
                # time of the sample and obtained with 1 off region) we ignore the corresponding cut
                # combination

min_exc = 0.002 # in fraction of off. Below this we ignore the corresponding cut combination.
min_off_events = 10 # minimum number of off events in the actual observation used. Below this we
                    # ignore the corresponding cut combination.

# Note that min_exc is set to a pretty low value! In the observations published in the LST1 performance
# paper the background at low E is stable to 0.5% (i.e. 0.005)
```

In [6]:
```
#
# For the calculation of the uncertainty of sensitivity:
backg_syst = 0.01 # relative background systematic uncertainty

backg_normalization = False # generally normalization is not needed, background is uniform enough.
norm_range = np.array([[0.1, 0.16], [20., 59.8]]) # deg2 for theta2,  deg for Alpha
```

In [7]:
```
sa = LSTEventSource.create_subarray(tel_id=1)
focal = sa.tel[1].optics.effective_focal_length
```

In [8]:
```
# The source position src_x, src_y (in m), stored in "source_position", is calculated by
# lstchain_dl2_add_sourcepos.py using the effective focal length (29.30565 m), which means
# that it is "consistent" with the reco values reco_src_x, reco_src_y (which are affected
# by the telescope's optical aberration)
#
print(focal)
```

```
29.30565 m
```

In [9]:
```
on_events = [None, None]
off_events = [None, None]

# In on_events and off_events (which are basically 5-axis histograms):
# axis 0 has two bins. Indicates analysis type. index=0: source-independent;  index=1: source-dependent
#
# axis 1 has two bins. index=0: even event-id events;   index=1: odd-event-id events
# axis 2 is the energy axis, see logenergy_bins above
# axis 3 is the gammaness axis, see gammaness_bins above
# axis 4 is the angular axis (theta or Alpha), see theta2bins and alphabins above
```

In [10]:
```
#%%filprofile

tablename = "/dl2/event/telescope/parameters/LST_LSTCam"
livetimes = []

livetime = 0


for ifile, file in enumerate(dataset1):
```

```
print(ifile+1, '/', len(dataset1), ': ', file, time.asctime(time.gmtime()))

tb = read_table(file, tablename)

# See above the note on the source_position table!
tb_extra = read_table(file, "source_position/table")

lt, _ = get_effective_time(tb)
livetime += lt


dx = np.rad2deg((tb['reco_src_x']-tb_extra['src_x'])/focal.to_value(u.m))
dy = np.rad2deg((tb['reco_src_y']-tb_extra['src_y'])/focal.to_value(u.m))
tb['theta2_on']  = (dx**2 + dy**2).astype('float32')

dx = np.rad2deg((tb['reco_src_x']+tb_extra['src_x'])/focal.to_value(u.m))
dy = np.rad2deg((tb['reco_src_y']+tb_extra['src_y'])/focal.to_value(u.m))
tb['theta2_off'] = (dx**2 + dy**2).astype('float32')

dx = np.rad2deg((tb['reco_src_x']+tb_extra['src_y'])/focal.to_value(u.m))
dy = np.rad2deg((tb['reco_src_y']-tb_extra['src_x'])/focal.to_value(u.m))
tb['theta2_off_90'] = (dx**2 + dy**2).astype('float32')

dx = np.rad2deg((tb['reco_src_x']-tb_extra['src_y'])/focal.to_value(u.m))
dy = np.rad2deg((tb['reco_src_y']+tb_extra['src_x'])/focal.to_value(u.m))
tb['theta2_off_270'] = (dx**2 + dy**2).astype('float32')

tb['odd_or_even'] = (tb['event_id']%2).astype('float32')


# Filter to select cosmics (=shower events)
noped = (tb['event_type'] != EventType.SKY_PEDESTAL.value)
nocal = (tb['event_type'] != EventType.FLATFIELD.value)
cosmics = noped & nocal

mask = ((tb['intensity']>min_intensity) &
        (tb['intensity']<max_intensity) &
        cosmics)

on, _  = np.histogramdd(np.array([tb['odd_or_even'][mask],
                                  tb['log_reco_energy'][mask].astype('float32'),
                                  tb['gammaness'][mask].astype('float32'),
                                  tb['theta2_on'][mask]]).T,
                        bins=[evt_id_bins, logenergy_bins, gammaness_bins, theta2bins])

on = on.astype('float32')


if on_events[0] is None:
    on_events[0] = on
else:
    on_events[0] += on


off, _ = np.histogramdd(np.array([tb['odd_or_even'][mask],
                                  tb['log_reco_energy'][mask].astype('float32'),
                                  tb['gammaness'][mask].astype('float32'),
                                  tb['theta2_off'][mask]]).T,
                        bins=[evt_id_bins, logenergy_bins, gammaness_bins, theta2bins])
off = off.astype('float32')

# For bins >= first_ebin_with_3offs we fill the off with the average of 3 off regions
off[:,first_ebin_with_3offs:,:,:] *= 1/3.
high_e_mask = mask & (tb['log_reco_energy'] >= logenergy_bins[first_ebin_with_3offs])

off += np.histogramdd(np.array([tb['odd_or_even'][high_e_mask],
                                tb['log_reco_energy'][high_e_mask].astype('float32'),
                                tb['gammaness'][high_e_mask].astype('float32'),
                                tb['theta2_off_90'][high_e_mask]]).T,
                      bins=[evt_id_bins, logenergy_bins, gammaness_bins,
                            theta2bins])[0].astype('float32') / 3.

off += np.histogramdd(np.array([tb['odd_or_even'][high_e_mask],
                                tb['log_reco_energy'][high_e_mask].astype('float32'),
                                tb['gammaness'][high_e_mask].astype('float32'),
                                tb['theta2_off_270'][high_e_mask]]).T,
                      bins=[evt_id_bins, logenergy_bins, gammaness_bins,
                            theta2bins])[0].astype('float32') / 3.

if off_events[0] is None:
    off_events[0] = off
else:
    off_events[0] += off

on = None
```

```
    off = None
    tb = None
    tb_extra = None
    gc.collect() # memory clean-up


print("Live time:", livetime.to(u.h))
livetimes.append(livetime)
```

```
1 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl2
_LST-1.Run02914.h5 Thu Sep  7 21:02:36 2023
2 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl2
_LST-1.Run02929.h5 Thu Sep  7 21:03:32 2023
3 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl2
_LST-1.Run02930.h5 Thu Sep  7 21:04:38 2023
4 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl2
_LST-1.Run02931.h5 Thu Sep  7 21:05:34 2023
5 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl2
_LST-1.Run02932.h5 Thu Sep  7 21:06:30 2023
6 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl2
_LST-1.Run02933.h5 Thu Sep  7 21:07:26 2023
7 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl2
_LST-1.Run02934.h5 Thu Sep  7 21:08:38 2023
8 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl2
_LST-1.Run02949.h5 Thu Sep  7 21:09:40 2023
9 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl2
_LST-1.Run02950.h5 Thu Sep  7 21:10:48 2023
10 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02967.h5 Thu Sep  7 21:11:58 2023
11 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02968.h5 Thu Sep  7 21:12:55 2023
12 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02969.h5 Thu Sep  7 21:14:00 2023
13 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02970.h5 Thu Sep  7 21:15:02 2023
14 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02971.h5 Thu Sep  7 21:16:00 2023
15 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02972.h5 Thu Sep  7 21:17:00 2023
16 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02973.h5 Thu Sep  7 21:18:33 2023
17 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02974.h5 Thu Sep  7 21:19:45 2023
18 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02975.h5 Thu Sep  7 21:21:00 2023
19 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02976.h5 Thu Sep  7 21:22:12 2023
20 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02977.h5 Thu Sep  7 21:23:14 2023
21 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02988.h5 Thu Sep  7 21:24:21 2023
22 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02989.h5 Thu Sep  7 21:25:35 2023
23 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02990.h5 Thu Sep  7 21:26:34 2023
24 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02991.h5 Thu Sep  7 21:27:38 2023
25 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run02992.h5 Thu Sep  7 21:28:34 2023
26 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03004.h5 Thu Sep  7 21:29:40 2023
27 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03005.h5 Thu Sep  7 21:30:43 2023
28 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03006.h5 Thu Sep  7 21:31:58 2023
29 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03007.h5 Thu Sep  7 21:32:57 2023
30 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03008.h5 Thu Sep  7 21:34:22 2023
31 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03093.h5 Thu Sep  7 21:35:19 2023
32 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03094.h5 Thu Sep  7 21:36:21 2023
33 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03095.h5 Thu Sep  7 21:37:21 2023
34 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03096.h5 Thu Sep  7 21:38:41 2023
35 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03231.h5 Thu Sep  7 21:39:54 2023
36 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03232.h5 Thu Sep  7 21:40:58 2023
37 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03243.h5 Thu Sep  7 21:41:54 2023
38 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03270.h5 Thu Sep  7 21:43:16 2023
39 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03271.h5 Thu Sep  7 21:44:17 2023
40 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03272.h5 Thu Sep  7 21:45:18 2023
41 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03273.h5 Thu Sep  7 21:46:18 2023
42 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03274.h5 Thu Sep  7 21:47:24 2023
43 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03275.h5 Thu Sep  7 21:48:21 2023
```

```
44 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03276.h5 Thu Sep  7 21:49:27 2023
45 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03277.h5 Thu Sep  7 21:50:29 2023
46 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03278.h5 Thu Sep  7 21:51:32 2023
47 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03279.h5 Thu Sep  7 21:52:32 2023
48 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03318.h5 Thu Sep  7 21:53:21 2023
49 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03319.h5 Thu Sep  7 21:54:13 2023
50 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03320.h5 Thu Sep  7 21:55:14 2023
51 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03321.h5 Thu Sep  7 21:56:25 2023
52 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03328.h5 Thu Sep  7 21:57:36 2023
53 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03329.h5 Thu Sep  7 21:58:58 2023
54 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03330.h5 Thu Sep  7 21:59:53 2023
55 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03338.h5 Thu Sep  7 22:01:16 2023
56 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03339.h5 Thu Sep  7 22:02:08 2023
57 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03340.h5 Thu Sep  7 22:03:05 2023
58 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03355.h5 Thu Sep  7 22:04:03 2023
59 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03356.h5 Thu Sep  7 22:05:31 2023
60 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03373.h5 Thu Sep  7 22:06:25 2023
61 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03598.h5 Thu Sep  7 22:07:29 2023
62 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03599.h5 Thu Sep  7 22:08:24 2023
63 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03600.h5 Thu Sep  7 22:09:25 2023
64 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03601.h5 Thu Sep  7 22:10:25 2023
65 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03615.h5 Thu Sep  7 22:11:35 2023
66 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03632.h5 Thu Sep  7 22:12:53 2023
67 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03633.h5 Thu Sep  7 22:13:55 2023
68 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03634.h5 Thu Sep  7 22:14:31 2023
69 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03635.h5 Thu Sep  7 22:15:17 2023
70 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03672.h5 Thu Sep  7 22:16:01 2023
71 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03673.h5 Thu Sep  7 22:16:56 2023
72 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03674.h5 Thu Sep  7 22:17:45 2023
73 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03675.h5 Thu Sep  7 22:18:32 2023
74 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03676.h5 Thu Sep  7 22:19:38 2023
75 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03677.h5 Thu Sep  7 22:20:21 2023
76 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03706.h5 Thu Sep  7 22:21:13 2023
77 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03707.h5 Thu Sep  7 22:22:10 2023
78 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run03708.h5 Thu Sep  7 22:23:05 2023
79 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run04067.h5 Thu Sep  7 22:23:54 2023
80 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run04068.h5 Thu Sep  7 22:24:43 2023
81 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run04086.h5 Thu Sep  7 22:25:43 2023
82 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run04087.h5 Thu Sep  7 22:26:57 2023
83 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run06045.h5 Thu Sep  7 22:28:03 2023
84 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run06073.h5 Thu Sep  7 22:28:59 2023
85 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run06304.h5 Thu Sep  7 22:29:46 2023
86 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run06872.h5 Thu Sep  7 22:30:54 2023
```

```
87 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run06873.h5 Thu Sep  7 22:31:55 2023
88 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run06874.h5 Thu Sep  7 22:32:48 2023
89 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run06875.h5 Thu Sep  7 22:33:47 2023
90 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run06892.h5 Thu Sep  7 22:34:46 2023
91 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run06893.h5 Thu Sep  7 22:35:51 2023
92 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run06894.h5 Thu Sep  7 22:36:53 2023
93 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run06895.h5 Thu Sep  7 22:37:54 2023
94 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run07097.h5 Thu Sep  7 22:39:07 2023
95 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run07098.h5 Thu Sep  7 22:40:11 2023
96 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run07099.h5 Thu Sep  7 22:41:18 2023
97 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run07133.h5 Thu Sep  7 22:42:38 2023
98 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run07136.h5 Thu Sep  7 22:43:15 2023
99 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/dl
2_LST-1.Run07161.h5 Thu Sep  7 22:44:00 2023
100 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07195.h5 Thu Sep  7 22:44:59 2023
101 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07196.h5 Thu Sep  7 22:45:51 2023
102 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07197.h5 Thu Sep  7 22:46:17 2023
103 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07199.h5 Thu Sep  7 22:47:13 2023
104 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07200.h5 Thu Sep  7 22:48:05 2023
105 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07227.h5 Thu Sep  7 22:48:56 2023
106 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07228.h5 Thu Sep  7 22:49:48 2023
107 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07231.h5 Thu Sep  7 22:50:40 2023
108 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07232.h5 Thu Sep  7 22:51:21 2023
109 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07233.h5 Thu Sep  7 22:52:14 2023
110 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07253.h5 Thu Sep  7 22:53:06 2023
111 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07254.h5 Thu Sep  7 22:54:28 2023
112 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07255.h5 Thu Sep  7 22:55:28 2023
113 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07256.h5 Thu Sep  7 22:56:32 2023
114 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07274.h5 Thu Sep  7 22:57:07 2023
115 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07275.h5 Thu Sep  7 22:58:05 2023
116 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07276.h5 Thu Sep  7 22:59:00 2023
117 / 117 :  /fefs/aswg/workspace/abelardo.moralejo/Crab_performance_paper/data_v0.9.9/DL2_sourcepos_with_v010/d
l2_LST-1.Run07277.h5 Thu Sep  7 22:59:36 2023
Live time: 34.19721046934775 h
```

In [70]:

```
#%%filprofile
# Columns to be kept in the case of source-dependent analysis (again, so save memory):

columns_srcdep = ["('on', 'expected_src_x')",
                  "('on', 'expected_src_y')",
                  "('on', 'alpha')",
                  "('off_180', 'alpha')",
                  "('on', 'gammaness')",
                  "('off_180', 'gammaness')",
                  "('on', 'reco_energy')",
                  "('off_180', 'reco_energy')"]
columns = ['obs_id',
           'event_id',
           'intensity',
           'event_type',
           'x', 'y', 'psi']


tablename = "/dl2/event/telescope/parameters/LST_LSTCam"
tablename_srcdep = "/dl2/event/telescope/parameters_src_dependent/LST_LSTCam"

livetime = 0
```

```
on_events[1] = None
off_events[1] = None

for ifile, file in enumerate(dataset2):

    print(ifile+1, '/', len(dataset2), ':  ', file, time.asctime(time.gmtime()))

    tb = read_table(file, tablename)
    tb['odd_or_even'] = (tb['event_id']%2).astype('float32')


    lt, _ = get_effective_time(tb)
    livetime += lt

    tb_srcdep = read_table(file, tablename_srcdep)

    tb_srcdep.rename_columns(["('on', 'expected_src_x')", "('on', 'expected_src_y')",
                              "('on', 'alpha')", "('off_180', 'alpha')",
                              "('on', 'gammaness')", "('off_180', 'gammaness')",
                              "('on', 'reco_energy')", "('off_180', 'reco_energy')"],
                             ['src_x', 'src_y', 'alpha_on', 'alpha_off',
                              'gammaness_on', 'gammaness_off', 'reco_energy_on', 'reco_energy_off'])

    noped = (tb['event_type'] != EventType.SKY_PEDESTAL.value)
    nocal = (tb['event_type'] != EventType.FLATFIELD.value)
    cosmics = noped & nocal

    mask = ((tb['intensity']>min_intensity) &
            (tb['intensity']<max_intensity) &
            cosmics)

    on, _  = np.histogramdd(np.array([tb['odd_or_even'][mask],
                                      np.log10(tb_srcdep['reco_energy_on'][mask]),
                                      tb_srcdep['gammaness_on'][mask],
                                      tb_srcdep['alpha_on'][mask]]).T,
                            bins=[evt_id_bins, logenergy_bins, gammaness_bins, alphabins])
    on = on.astype('float32')

    off, _  = np.histogramdd(np.array([tb['odd_or_even'][mask],
                                      np.log10(tb_srcdep['reco_energy_off'][mask]),
                                      tb_srcdep['gammaness_off'][mask],
                                      tb_srcdep['alpha_off'][mask]]).T,
                            bins=[evt_id_bins, logenergy_bins, gammaness_bins, alphabins])
    off = off.astype('float32')

    if on_events[1] is None:
        on_events[1] = on
        off_events[1] = off
    else:
        on_events[1] += on
        off_events[1] += off

    on = None
    off = None
    tb = None
    tb_srcdep = None
    gc.collect() # memory cleanup


print("Live time:", livetime.to(u.h))
livetimes.append(livetime)
```

```
1 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_axi
s_1deg/DL2/data/dl2_LST-1.Run02914.h5 Fri Sep  8 05:57:41 2023
2 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_axi
s_1deg/DL2/data/dl2_LST-1.Run02929.h5 Fri Sep  8 05:58:08 2023
3 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_axi
s_1deg/DL2/data/dl2_LST-1.Run02930.h5 Fri Sep  8 05:58:18 2023
4 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_axi
s_1deg/DL2/data/dl2_LST-1.Run02931.h5 Fri Sep  8 05:58:46 2023
5 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_axi
s_1deg/DL2/data/dl2_LST-1.Run02932.h5 Fri Sep  8 05:59:13 2023
6 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_axi
s_1deg/DL2/data/dl2_LST-1.Run02933.h5 Fri Sep  8 05:59:46 2023
7 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_axi
s_1deg/DL2/data/dl2_LST-1.Run02934.h5 Fri Sep  8 06:00:21 2023
8 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_axi
s_1deg/DL2/data/dl2_LST-1.Run02949.h5 Fri Sep  8 06:00:49 2023
9 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_axi
s_1deg/DL2/data/dl2_LST-1.Run02950.h5 Fri Sep  8 06:01:16 2023
10 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02967.h5 Fri Sep  8 06:01:48 2023
11 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02968.h5 Fri Sep  8 06:02:16 2023
12 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02969.h5 Fri Sep  8 06:02:45 2023
13 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02970.h5 Fri Sep  8 06:03:12 2023
14 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02971.h5 Fri Sep  8 06:03:40 2023
15 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02972.h5 Fri Sep  8 06:04:14 2023
16 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02973.h5 Fri Sep  8 06:04:49 2023
17 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02974.h5 Fri Sep  8 06:05:19 2023
18 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02975.h5 Fri Sep  8 06:05:48 2023
19 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02976.h5 Fri Sep  8 06:06:26 2023
20 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02977.h5 Fri Sep  8 06:06:55 2023
21 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02988.h5 Fri Sep  8 06:07:25 2023
22 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02989.h5 Fri Sep  8 06:08:01 2023
23 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02990.h5 Fri Sep  8 06:08:12 2023
24 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02991.h5 Fri Sep  8 06:08:40 2023
25 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run02992.h5 Fri Sep  8 06:09:03 2023
26 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03004.h5 Fri Sep  8 06:09:27 2023
27 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03005.h5 Fri Sep  8 06:09:55 2023
28 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03006.h5 Fri Sep  8 06:10:24 2023
29 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03007.h5 Fri Sep  8 06:10:51 2023
30 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03008.h5 Fri Sep  8 06:11:17 2023
31 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03093.h5 Fri Sep  8 06:12:07 2023
32 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03094.h5 Fri Sep  8 06:12:35 2023
33 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03095.h5 Fri Sep  8 06:13:10 2023
34 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03096.h5 Fri Sep  8 06:13:56 2023
35 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03231.h5 Fri Sep  8 06:14:25 2023
36 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03232.h5 Fri Sep  8 06:14:52 2023
37 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03243.h5 Fri Sep  8 06:15:19 2023
38 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03270.h5 Fri Sep  8 06:15:44 2023
39 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03271.h5 Fri Sep  8 06:16:21 2023
40 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03272.h5 Fri Sep  8 06:16:53 2023
41 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03273.h5 Fri Sep  8 06:17:19 2023
42 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03274.h5 Fri Sep  8 06:17:54 2023
43 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03275.h5 Fri Sep  8 06:18:26 2023
```

```
44 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03276.h5 Fri Sep  8 06:18:56 2023
45 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03277.h5 Fri Sep  8 06:19:25 2023
46 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03278.h5 Fri Sep  8 06:19:58 2023
47 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03279.h5 Fri Sep  8 06:20:26 2023
48 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03318.h5 Fri Sep  8 06:20:51 2023
49 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03319.h5 Fri Sep  8 06:21:25 2023
50 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03320.h5 Fri Sep  8 06:22:07 2023
51 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03321.h5 Fri Sep  8 06:22:35 2023
52 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03328.h5 Fri Sep  8 06:23:07 2023
53 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03329.h5 Fri Sep  8 06:23:37 2023
54 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03330.h5 Fri Sep  8 06:23:53 2023
58 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03355.h5 Fri Sep  8 06:26:09 2023
59 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03356.h5 Fri Sep  8 06:26:41 2023
60 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03373.h5 Fri Sep  8 06:27:18 2023
61 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03598.h5 Fri Sep  8 06:27:55 2023
62 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03599.h5 Fri Sep  8 06:28:25 2023
63 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03600.h5 Fri Sep  8 06:28:55 2023
64 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03601.h5 Fri Sep  8 06:29:29 2023
65 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03615.h5 Fri Sep  8 06:29:57 2023
66 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03632.h5 Fri Sep  8 06:30:33 2023
67 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03633.h5 Fri Sep  8 06:31:00 2023
68 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03634.h5 Fri Sep  8 06:31:19 2023
69 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03635.h5 Fri Sep  8 06:31:53 2023
70 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03672.h5 Fri Sep  8 06:32:19 2023
71 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03673.h5 Fri Sep  8 06:32:53 2023
72 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03674.h5 Fri Sep  8 06:33:22 2023
73 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03675.h5 Fri Sep  8 06:33:51 2023
74 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03676.h5 Fri Sep  8 06:34:18 2023
75 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03677.h5 Fri Sep  8 06:34:46 2023
76 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03706.h5 Fri Sep  8 06:35:16 2023
77 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03707.h5 Fri Sep  8 06:35:43 2023
78 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run03708.h5 Fri Sep  8 06:36:12 2023
79 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run04067.h5 Fri Sep  8 06:36:45 2023
80 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run04068.h5 Fri Sep  8 06:37:11 2023
81 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run04086.h5 Fri Sep  8 06:37:43 2023
82 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run04087.h5 Fri Sep  8 06:37:59 2023
83 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run06045.h5 Fri Sep  8 06:38:31 2023
84 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run06073.h5 Fri Sep  8 06:38:59 2023
85 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run06304.h5 Fri Sep  8 06:39:26 2023
86 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run06872.h5 Fri Sep  8 06:40:07 2023
87 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run06873.h5 Fri Sep  8 06:40:44 2023
88 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run06874.h5 Fri Sep  8 06:41:16 2023
89 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run06875.h5 Fri Sep  8 06:41:49 2023
```

```
90 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run06892.h5 Fri Sep  8 06:42:43 2023
91 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run06893.h5 Fri Sep  8 06:43:33 2023
92 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run06894.h5 Fri Sep  8 06:44:10 2023
93 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run06895.h5 Fri Sep  8 06:44:49 2023
94 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run07097.h5 Fri Sep  8 06:45:24 2023
95 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run07098.h5 Fri Sep  8 06:45:57 2023
96 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run07099.h5 Fri Sep  8 06:46:36 2023
97 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run07133.h5 Fri Sep  8 06:47:08 2023
98 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run07136.h5 Fri Sep  8 06:47:24 2023
99 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_ax
is_1deg/DL2/data/dl2_LST-1.Run07161.h5 Fri Sep  8 06:47:52 2023
100 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07195.h5 Fri Sep  8 06:48:20 2023
101 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07196.h5 Fri Sep  8 06:48:49 2023
102 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07197.h5 Fri Sep  8 06:48:58 2023
103 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07199.h5 Fri Sep  8 06:49:26 2023
104 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07200.h5 Fri Sep  8 06:49:57 2023
105 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07227.h5 Fri Sep  8 06:50:18 2023
106 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07228.h5 Fri Sep  8 06:50:52 2023
107 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07231.h5 Fri Sep  8 06:51:18 2023
108 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07232.h5 Fri Sep  8 06:51:34 2023
109 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07233.h5 Fri Sep  8 06:52:05 2023
110 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07253.h5 Fri Sep  8 06:52:26 2023
111 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07254.h5 Fri Sep  8 06:53:03 2023
112 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07255.h5 Fri Sep  8 06:53:42 2023
113 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07256.h5 Fri Sep  8 06:54:14 2023
114 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07274.h5 Fri Sep  8 06:54:32 2023
115 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07275.h5 Fri Sep  8 06:55:04 2023
116 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07276.h5 Fri Sep  8 06:55:47 2023
117 / 117 :   /fefs/aswg/workspace/seiya.nozaki/Crab_performance_paper/20221027_v0.9.9_crab_tuned/combined_off_a
xis_1deg/DL2/data/dl2_LST-1.Run07277.h5 Fri Sep  8 06:56:05 2023
Live time: 34.19721046934775 h
```

In [72]:
```
# Index 0 refers to the source-independent analysis, index 1 to the source-dependent analysis.
# If the Crab data samples are the same (just different analysis method), the live times should
# be the same:

print(livetimes[0].to(u.h))
print(livetimes[1].to(u.h))
```

```
34.19721046934775 h
34.19721046934775 h
```

In [79]:
```
# Show the excess:
# plt.errorbar(0.5*(bins[1:]+bins[:-1]), onevts-offevts, yerr=(onevts+offevts)**0.5, fmt='o', markersize=3)
# plt.xlabel('Alpha (deg)')
# plt.ylabel('Excess events')
# plt.grid()
# plt.show()
```

In [82]:
```
# Prepare arrays to contain the cumulative sums, from each gammaness value to 1, and from 0 to each
# value of theta (or Alpha):

cum_on_events = [np.copy(on_events[0]), np.copy(on_events[1])]
cum_off_events = [np.copy(off_events[0]), np.copy(off_events[1])]
excess_events = [on_events[0]-off_events[0], on_events[1]-off_events[1]]
```

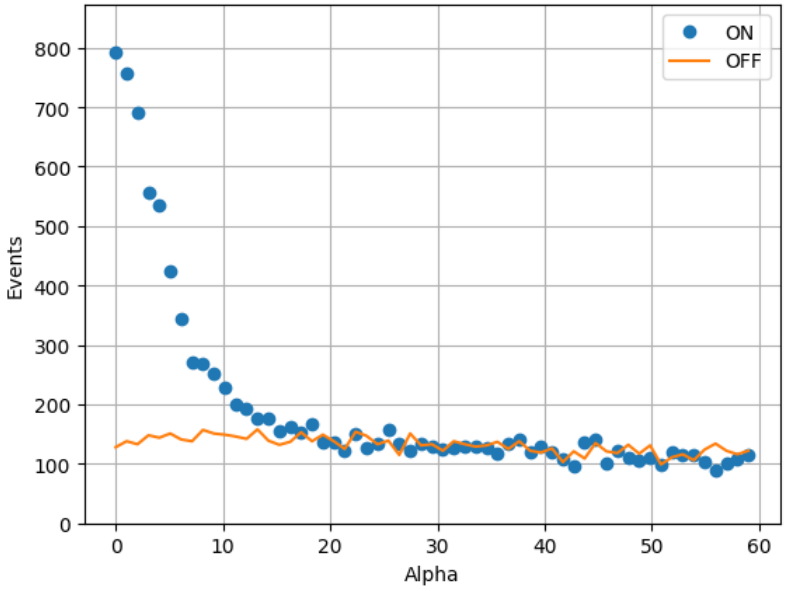
In [83]:
```
cum_on_events[0].shape
# 0 indicates source-dependent analysis.
# The four axis are odd/even event_id's; energy; gammaness_cut; theta2_cut;
```

Out[83]: (2, 21, 2000, 6000)

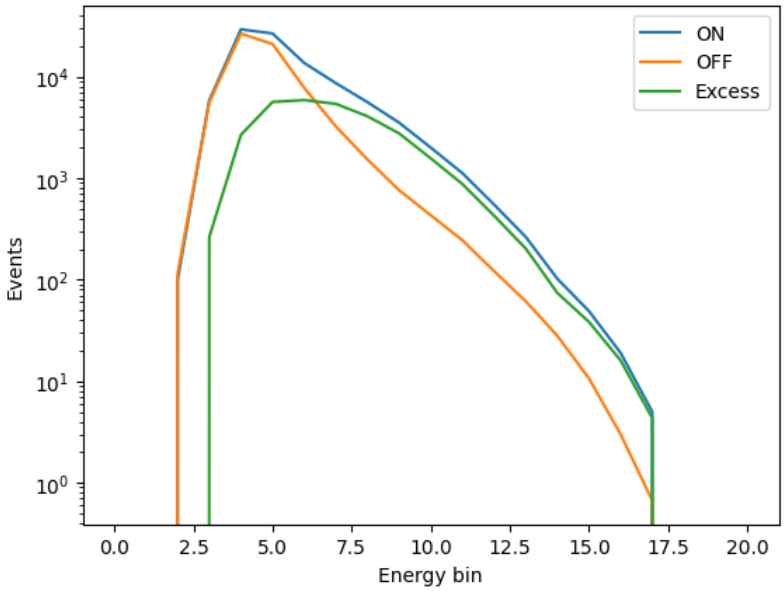
In [84]:
```
#%%filprofile

# Obtain the cumulative histograms: integrate in gammaness and in theta (or alpha).

for evtid in range(cum_on_events[0].shape[0]):
    for energyid in range(cum_on_events[0].shape[1]):
        # gammaness_bins and theta / alpha bins arrays are of bin edges...
        # Note that actual number of bins in histograms is that number -1

        for i in reversed(range(len(gammaness_bins)-2)):
            cum_on_events[0][evtid, energyid, i,:] += cum_on_events[0][evtid, energyid, i+1,:]
            cum_off_events[0][evtid, energyid, i,:] += cum_off_events[0][evtid, energyid, i+1,:]
            cum_on_events[1][evtid, energyid, i,:] += cum_on_events[1][evtid, energyid, i+1,:]
            cum_off_events[1][evtid, energyid, i,:] += cum_off_events[1][evtid, energyid, i+1,:]


        for j in range(len(theta2bins)-1):
            if j == 0:
                continue
            cum_on_events[0][evtid, energyid, :, j] += cum_on_events[0][evtid, energyid, :, j-1]
            cum_off_events[0][evtid, energyid, :, j] += cum_off_events[0][evtid, energyid, :, j-1]

        for j in range(len(alphabins)-1):
            if j == 0:
                continue
            cum_on_events[1][evtid, energyid, :, j] += cum_on_events[1][evtid, energyid, :, j-1]
            cum_off_events[1][evtid, energyid, :, j] += cum_off_events[1][evtid, energyid, :, j-1]
```

In [85]:
```
cum_excess_events = [[],[]]
cum_excess_events[0] = cum_on_events[0] - cum_off_events[0]
cum_excess_events[1] = cum_on_events[1] - cum_off_events[1]
```

In [299…
```
#
# Just a plot to illustrate the content of the cumulative histograms, and to check the overall good
# agreement of the on- and off- alpha distributions. Note that it will not be perfect, some gammas
# may be present in the OFF at middle-Alpha values, especially for soft gammaness cuts!
# source-dependent analysis (1), even event_id's (0)
#
Ebin = 6 # Energy bin
gammanessbin = 1900 # gammaness cut (in bins)
alpha_rebin = 50  # rebinning for better view
ON = np.diff(cum_excess_events[1][0][Ebin][gammanessbin][::alpha_rebin] +
             cum_off_events[1][0][Ebin][gammanessbin][::alpha_rebin])

xx = np.linspace(alphabins.min(), alphabins.max(), len(ON)+1)

plt.plot(xx[:-1], ON, 'o', label='ON')
plt.plot(xx[:-1], np.diff(cum_off_events[1][0][Ebin][gammanessbin][::alpha_rebin]), label='OFF')
plt.ylim(0, 1.1*np.max(ON))
plt.xlabel('Alpha')
plt.ylabel('Events')
plt.grid()
plt.legend()
plt.show()
```



In [86]:
```
# Just a check plot: even event_id's, source-independent analysis, and some specific gammaness & theta2 cuts
plt.plot(cum_on_events[0][0,:,1400,200], label='ON')
plt.plot(cum_off_events[0][0,:,1400,200], label='OFF')
plt.plot(cum_excess_events[0][0,:,1400,200], label='Excess')
plt.yscale('log')
plt.xlabel('Energy bin')
plt.ylabel('Events')
plt.legend()
plt.show()
```



In [87]:
```
# Having excess and OFF we do not need ON:
cum_on_events[0] = None
cum_on_events[1] = None
cum_on_events = None
gc.collect() # memory clean-up
```

Out[87]: 14234

In [169…
```
# Check for size of histograms in GB:
for i, type_str in enumerate(['Source-independent:', 'Source-dependent:']):
    print(type_str,
          excess_events[i].size*excess_events[i].itemsize/1e9,
          cum_excess_events[i].size*cum_excess_events[i].itemsize/1e9,
          cum_off_events[i].size*cum_off_events[i].itemsize/1e9,
          off_events[i].size*off_events[i].itemsize/1e9,
          on_events[i].size*on_events[i].itemsize/1e9)
```

```
Source-independent: 2.016 2.016 2.016 2.016 2.016
Source-dependent: 1.008 1.008 1.008 1.008 1.008
```

In [89]:
```
# Look for other large objects in memory (TEST)
# import sys
# sorted([(x, sys.getsizeof(globals().get(x))) for x in dir()], key=lambda x: x[1], reverse=True)
```

Out[89]:
```
[('cosmics', 4199542),
 ('high_e_mask', 4199542),
 ('mask', 4199542),
 ('_i50', 23012),
 ('nevts_off', 12112),
 ('nevts_on', 12112),
 ('_i42', 6468),
 ('_i10', 4976),
 ('_i22', 4517),
 ('_i80', 4517),
 ('_i70', 3952),
 ('_i12', 3951),
 ('_i47', 3562),
 ('_i33', 3296),
 ('_i2', 2757),
 ('_i56', 2349),
 ('data', 2312),
 ('_i4', 1688),
 ('_i32', 1670),
 ('_i26', 1421),
 ('_i84', 1421),
 ('_i34', 1331),
 ('_i58', 1242),
 ('_31', 1240),
 ('_i52', 1204),
 ('CameraFrame', 1200),
 ('_i3', 1191),
 ('_i51', 1094),
 ('dataset1', 1080),
 ('dataset2', 1080),
 ('ErfaAstromInterpolator', 1072),
 ('EventType', 1072),
 ('LSTEventSource', 1072),
 ('SkyCoord', 1072),
 ('TelescopeFrame', 1072),
 ('WStatCountsStatistic', 1072),
 ('erfa_astrom', 1072),
 ('_i39', 1060),
 ('_i43', 1045),
 ('_i5', 1025),
 ('_i53', 998),
 ('_i1', 961),
 ('AltAz', 904),
 ('Time', 904),
 ('_i44', 876),
 ('_i37', 851),
 ('_i38', 816),
 ('angular_cut', 816),
 ('delta_sensitivity', 816),
 ('delta_sensitivity_5s', 816),
 ('gammaness_cut', 816),
 ('In', 792),
 ('_ih', 792),
 ('_i55', 779),
 ('_i40', 695),
 ('_i16', 680),
 ('_i74', 680),
 ('_i45', 649),
 ('Out', 640),
 ('_oh', 640),
 ('_i20', 637),
 ('_i78', 637),
 ('_i57', 631),
 ('_i14', 601),
 ('_i72', 601),
 ('_i62', 578),
 ('_i35', 559),
 ('_i9', 556),
 ('_i46', 550),
 ('_i48', 533),
 ('xx', 512),
 ('xxoff', 512),
 ('_i65', 506),
 ('_i49', 494),
 ('_i61', 491),
 ('integral_sensitivity', 464),
 ('num_excess_events', 464),
 ('num_off_events', 464),
 ('reco_energy', 464),
 ('sensitivity', 464),
 ('sensitivity_5s', 464),
 ('_i36', 430),
 ('_i28', 414),
 ('_i86', 414),
 ('_iii', 414),
```

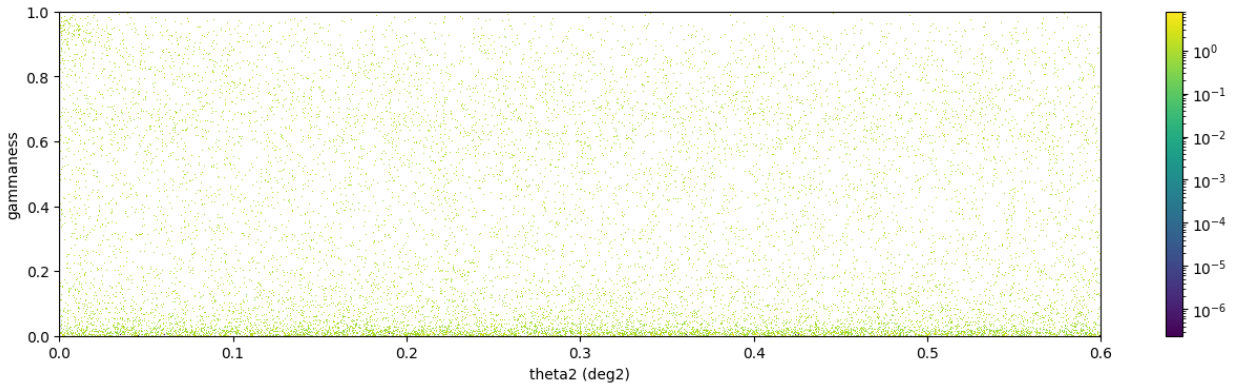
```
('_i68', 388),
('_i24', 381),
('_i82', 381),
('_i', 378),
('_i30', 378),
('_i88', 378),
('_i8', 377),
('_i69', 366),
('_i29', 359),
('_i87', 359),
('_ii', 359),
('_i6', 354),
('_i18', 346),
('_i76', 346),
('_i41', 318),
('yy', 312),
('yyoff', 312),
('_68', 280),
('rate_per_s_m2', 280),
('rebin_factor', 280),
('_i21', 259),
('_i79', 259),
('_i27', 196),
('_i85', 196),
('_i25', 190),
('_i83', 190),
('file', 186),
('_i66', 170),
('norm_bins', 160),
('norm_range', 160),
('_i31', 155),
('_i89', 155),
('_i19', 149),
('_i77', 149),
('add_delta_t_key', 144),
('binned_statistic', 144),
('calc_flux_3conditions', 144),
('calc_flux_for_N_excess', 144),
('calc_flux_for_N_sigma', 144),
('calc_flux_for_x_percent_backg', 144),
('crab_magic', 144),
('dfde', 144),
('find_bin_indices', 144),
('get_effective_time', 144),
('li_ma_significance', 144),
('open', 144),
('plot_MAGIC_sensitivity_fraction', 144),
('plot_rebinned', 144),
('read_table', 144),
('_i7', 142),
('dx', 136),
('dy', 136),
('etev', 136),
('_i13', 128),
('_i71', 128),
('columns', 120),
('columns_srcdep', 120),
('flux', 120),
('flux_5s', 120),
('flux_5s_minus', 120),
('flux_5s_plus', 120),
('flux_minus', 120),
('flux_plus', 120),
('focal', 120),
('livetime', 120),
('location', 120),
('lt', 120),
('obs_time', 120),
('target_obs_time', 120),
('__doc__', 113),
('_54', 112),
('_63', 112),
('_64', 112),
('alphabins', 112),
('gammaness_bins', 112),
('indices0', 112),
('indices1', 112),
('logenergy_bins', 112),
('nocal', 112),
('noped', 112),
('theta2bins', 112),
('_i11', 106),
('tablename_srcdep', 105),
('_i54', 95),
('_i60', 91),
('tablename', 91),
```

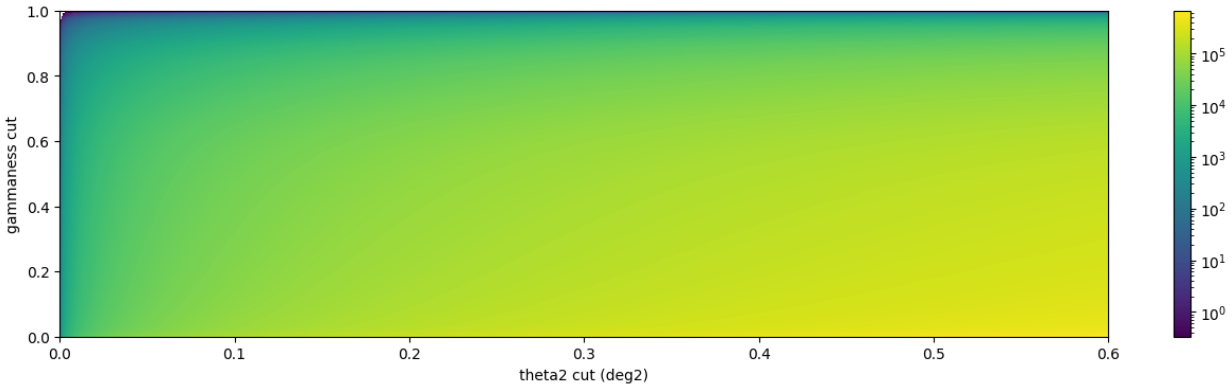
```
('tag2', 91),
('_i23', 89),
('_i81', 89),
('_', 88),
('evt_id_bins', 88),
('livetimes', 88),
('_i59', 87),
('_25', 72),
('_59', 72),
('_60', 72),
('_83', 72),
('__builtin__', 72),
('__builtins__', 72),
('colors', 72),
('cum_excess_events', 72),
('cum_off_events', 72),
('cut_indices', 72),
('detectable_flux', 72),
('excess_events', 72),
('flux_for_10_excess', 72),
('flux_for_5_sigma', 72),
('flux_for_5percent_backg', 72),
('gc', 72),
('glob', 72),
('indices', 72),
('integrate', 72),
('lima_signi', 72),
('min_angle_cut', 72),
('mpl', 72),
('np', 72),
('off_events', 72),
('on_events', 72),
('path', 72),
('plt', 72),
('signi', 72),
('sys', 72),
('tables', 72),
('time', 72),
('u', 72),
('Output_filename', 71),
('_i64', 69),
('tag1', 67),
('_i63', 66),
('_dh', 64),
('_i15', 64),
('_i17', 64),
('_i73', 64),
('_i75', 64),
('get_ipython', 64),
('__name__', 57),
('_i67', 57),
('table', 56),
('__', 49),
('___', 49),
('CRAB_MAGIC_JHEAP2015', 48),
('exit', 48),
('fig', 48),
('quit', 48),
('sa', 48),
('_67', 32),
('index', 32),
('max_excess', 32),
('min_excess', 32),
('nexcess', 32),
('noff', 32),
('recoE', 32),
('start_bin', 32),
('total_excess', 32),
('total_off', 32),
('_29', 28),
('_87', 28),
('analysis_type', 28),
('energyid', 28),
('even_or_odd', 28),
('evtid', 28),
('excess_in_norm_region', 28),
('first_ebin_with_3offs', 28),
('i', 28),
('iebin', 28),
('ifile', 28),
('j', 28),
('k', 28),
('min_backg_percent', 28),
('min_intensity', 28),
('min_n_gammas', 28),
('min_off_events', 28),
```

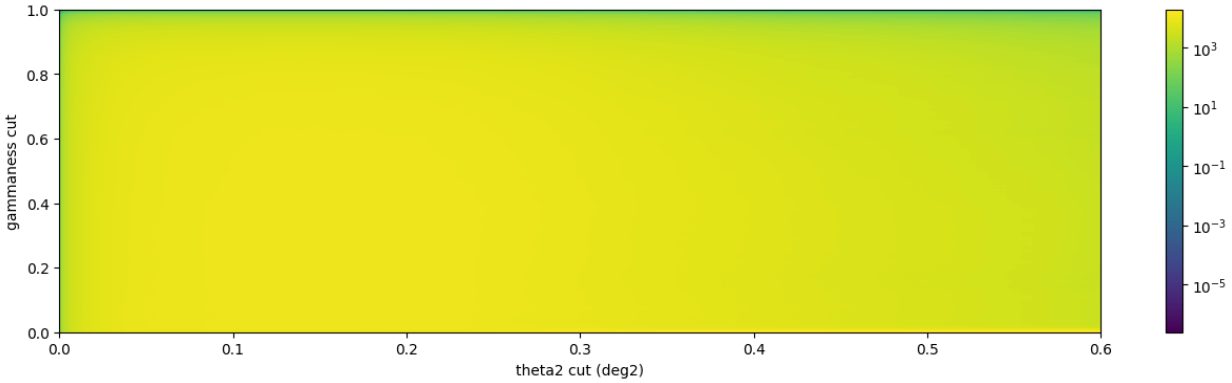
```
('min_signi', 28),
('off_in_norm_region', 28),
('off_norm_factor', 28),
('_11', 24),
('_13', 24),
('_23', 24),
('_71', 24),
('_81', 24),
('alpha', 24),
('backg_normalization', 24),
('backg_syst', 24),
('max_intensity', 24),
('min_exc', 24),
('other_half', 24),
('__loader__', 16),
('__package__', 16),
('__spec__', 16),
('cum_on_events', 16),
('dummy', 16),
('dummy2', 16),
('off', 16),
('on', 16),
('table1', 16),
('table2', 16),
('table2_srcdep', 16),
('tb', 16),
('tb_extra', 16),
('tb_srcdep', 16)]
```

In [90]:

```
# Just a test plot
evtid = 1 # Plot odd events
energyid = 7 # energy bin 7

# Excess events, per bin:
plt.figure(figsize=(16,4))
plt.pcolormesh(theta2bins, gammaness_bins, excess_events[0][evtid][energyid], norm=colors.LogNorm())
plt.colorbar()
plt.xlabel('theta2 (deg2)')
plt.ylabel('gammaness')
plt.show()

# Cumulative off events:
plt.figure(figsize=(16,4))
plt.pcolormesh(theta2bins, gammaness_bins, cum_off_events[0][evtid][energyid], norm=colors.LogNorm())
plt.colorbar()
plt.xlabel('theta2 cut (deg2)')
plt.ylabel('gammaness cut')
plt.show()


# Cumulative excess events:
plt.figure(figsize=(16,4))
plt.pcolormesh(theta2bins, gammaness_bins, cum_excess_events[0][evtid][energyid], norm=colors.LogNorm())
plt.colorbar()
plt.xlabel('theta2 cut (deg2)')
plt.ylabel('gammaness cut')
plt.show()
```







In [128…
```
##############################################################################################
#                                                                                            #
# SECOND PART OF THE NOTEBOOK!!! RUN FROM HERE OF YOU JUST WANT TO RE-RUN THE CUT OPTIMIZATION  #
# USING THE SAME DATA SAMPLES!                                                               #
#                                                                                            #
##############################################################################################
```

In [129…
```
##############################################################################################
#                                                                                            #
# UNCOMMENT BELOW THE SETTINGS YOU WANT TO MODIFY (RELATIVE TO WHAT WAS SET AT THE BEGINNING)   #
#                                                                                            #
##############################################################################################

# alpha = 0.2 # Li & Ma's "alpha", ratio of the ON to the OFF exposure
# This what we *assume* for the calculation, because it is the standard value.
# Note however that for a single LST, and to avoid large systematics, we use only one off
# region to estimate the background (in source-dependent analysis; in source-independent
# analysis it is one region up to 250 GeV and 3 regions for >250 GeV). We then "extrapolate"
# that background to what we would have in case we really had alpha=0.2...


# Assumed (effective) observation time for the calculation (note: it has nothing to do with
# the actual observation time of the sample used for the calculation!):
# obs_time = 50 * u.h
# obs_time = 1/60. * u.h # 1 minute
# obs_time = 0.083333333 * u.h  # 5 minutes
# obs_time = 0.5 * u.h
# obs_time = 0.7 * u.h
# obs_time = 5 * u.h


# The 5-sigma condition is so standard that we do not make it configurable! It is hardcoded.
# Additional sensitivity conditions:
# min_n_gammas = 10 # Minimum number of excess events
# min_backg_percent = 5 # Excess must be at least this percentage of the background
#
```

In [171…
```
##############################################################################################
#                                                                                            #
# UNCOMMENT BELOW THE SETTINGS YOU WANT TO MODIFY (RELATIVE TO WHAT WAS SET AT THE BEGINNING)   #
#                                                                                            #
##############################################################################################

#
# CUTS TO ENSURE THE QUALITY OF THE REAL DATA EXCESS FROM WHICH THE SENSITIVITY WILL BE COMPUTED!
#
# We set here the conditions to ensure that all used histogram bins (defined by Ebin, gammaness cut and
# angular cut) have, in the data sample, a reliable excess which we can use for estimating sensitivity:
#
# min_signi = 3   # below this value (significance of the test source, Crab, for the *actual* observation
```

```
                # time of the sample and obtained with 1 off region) we ignore the corresponding cut
                # combination

# min_exc = 0.002 # in fraction of off. Below this we ignore the corresponding cut combination.
# min_off_events = 10 # minimum number of off events in the actual observation used. Below this we
                    # ignore the corresponding cut combination.

# Note that min_exc is set to a pretty low value! In the observations published in the LST1 performance
# paper the background at low E is stable to 0.5% (i.e. 0.005)
```

In [172…

```
#
# For the calculation of the uncertainty of sensitivity:
backg_syst = 0.01 # relative background systematic uncertainty

backg_normalization = False # generally normalization is not needed, background is uniform enough.
norm_range = np.array([[0.1, 0.16], [20., 59.8]]) # deg2 for theta2,  deg for Alpha
```

In [173…

```
#
# Function to compute the fraction of the Crab flux that results in a given significance, for the
# observation time and Li&Ma's alpha set above.
# This is based on the observed excess and background (cumul_excess and cumul_off). This is done for all
# bins of cumul_excess and cumul_off, that is, for all possible gammaness and theta2/Alpha cuts
#
# In order to get more reliable results:
# We can require the *actually observed* excess to be a minimum fraction of the background (min_exc)
# We can require the *actually observed* excess to have a significance of at least min_signi (computed
# assuming just 1 off region)
# By "actually observed" we mean that it is not the excess (or significance) extrapolated for a
# 50h-observation or whatever, but the actually obtained result in the input data sample.
#
def calc_flux_for_N_sigma(N_sigma, cumul_excess, cumul_off,
                          min_signi, min_exc, min_off_events, alpha,
                          target_obs_time, actual_obs_time):

    time_factor = target_obs_time.to_value(u.h) / actual_obs_time.to_value(u.h)

    start_flux = 1
    flux_factor = start_flux * np.ones_like(cumul_excess)

    good_bin_mask = ((cumul_excess > min_exc*cumul_off) &
                     (cumul_off > min_off_events))

    flux_factor = np.where(good_bin_mask, flux_factor, np.nan)

    # First calculate significance (with 1 off) of the excesses in the provided sample, with no scaling.
    # We will only use the cut combinations where we have at least min_signi sigmas to begin with...
    # NOTE!!! float64 precision is essential for the arguments of li_ma_significance!

    lima_signi = li_ma_significance((flux_factor*cumul_excess + cumul_off).astype('float64'),
                                    cumul_off.astype('float64'),
                                    alpha=1)

    # Set nan in bins where we do not reach min_signi:
    flux_factor = np.where(lima_signi > min_signi, flux_factor, np.nan)


    # Now calculate the significance for the target observation time_
    lima_signi = li_ma_significance((time_factor*(flux_factor*cumul_excess +
                                                  cumul_off)).astype('float64'),
                                    (time_factor*cumul_off/alpha).astype('float64'),
                                    alpha=alpha)


    # iterate to obtain the flux which gives exactly N_sigma:
    for iter in range(4):
        # print(iter)
        tolerance_mask = np.abs(lima_signi-N_sigma)>0.001 # recalculate only what is needed
        flux_factor[tolerance_mask] *= (N_sigma / lima_signi[tolerance_mask])
        # NOTE!!! float64 precision is essential here!!!!
        lima_signi[tolerance_mask] = li_ma_significance((time_factor*(flux_factor[tolerance_mask]*
                                                                      cumul_excess[tolerance_mask]+
                                                                      cumul_off[tolerance_mask])).astype('float
                                                        (time_factor*cumul_off[tolerance_mask]/alpha).astype('
                                                        alpha=alpha)
    return flux_factor, lima_signi
```

In [174…

```
#%%filprofile

lima_signi = [[], []]
flux_for_5_sigma = [[], []]

# Compute the flux (in Crab fraction), for all cuts, which results in 5 sigma for the sensitivity conditions
# defined above:

for k in range(2):
```

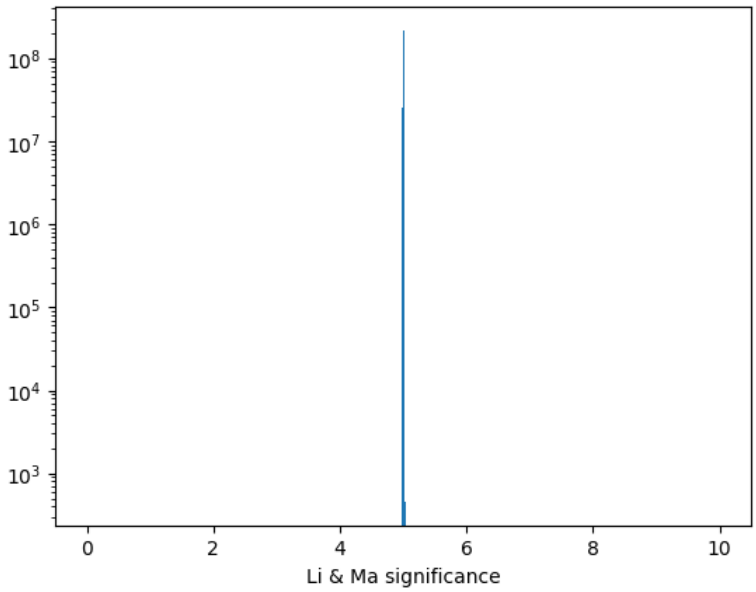
```
    flux_for_5_sigma[k], lima_signi[k] = calc_flux_for_N_sigma(5, cum_excess_events[k], cum_off_events[k],
                                                               min_signi, min_exc, min_off_events, alpha,
                                                               obs_time, livetimes[k]*0.5)
# NOTE: livetime is divided by 2 because calculation is done separately for the odd- and even-event_id samples!
```

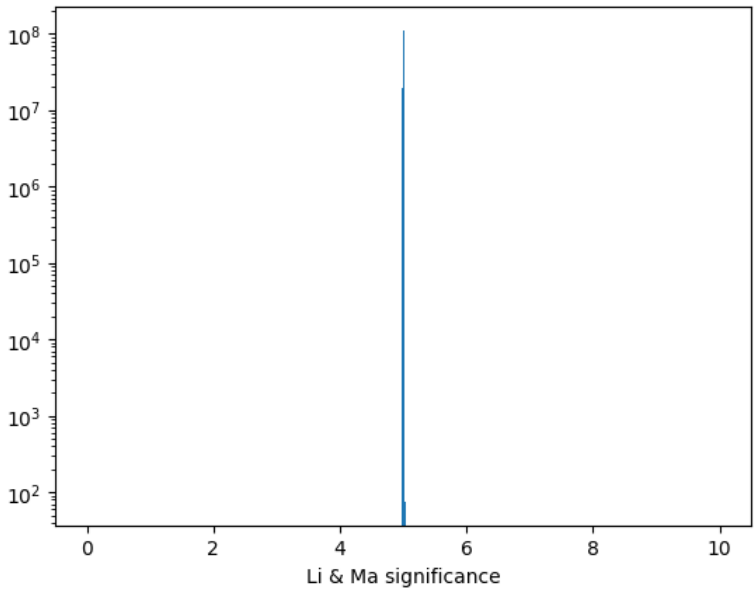
In [175…

```
# Now make sure we only consider bins with valid flux in *both* samples (odd- and even-events).
# This is because we will use the optimal cuts obtained with odd- events to apply them to even- events,
# and vice-versa. If the flux is nan for one of the two, we set it to nan for both.

for k in range(2):
    mask = np.isnan(flux_for_5_sigma[k][0]) | np.isnan(flux_for_5_sigma[k][1])
    flux_for_5_sigma[k] = np.where(mask, np.nan, flux_for_5_sigma[k])
    lima_signi[k] = np.where(mask, np.nan, lima_signi[k])
```

In [176…

```
# Check that significances are indeed 5 (or very close - they come from an interative procedure)
# Each entry in the histograms is a cut combination.

plt.hist(lima_signi[0].flatten(), bins=500, range=(0, 10), log=True)
plt.xlabel('Li & Ma significance')
plt.show()

plt.hist(lima_signi[1].flatten(), bins=500, range=(0, 10), log=True)
plt.xlabel('Li & Ma significance')
plt.show()
```





In [177…

```
#
# Function to calculate the flux needed to obtain a given excess, for all cut combinations:
#
def calc_flux_for_N_excess(N_excess, cumul_excess, target_obs_time, actual_obs_time):
    time_factor = target_obs_time.to_value(u.h) / actual_obs_time.to_value(u.h)
    return (N_excess/(cumul_excess*time_factor))

#
```

```
# Do the computation (JUST FOR TEST!!):
#
# flux_for_10_excess = [[], []]
# for k in range(2):
#     flux_for_10_excess[k] = calc_flux_for_N_excess(min_n_gammas, cum_excess_events[k], obs_time, livetimes[k]
#     # livetime divided by 2 because calculation is done separately for the odd- and even-event_id samples!

# flux_for_10_excess = [np.where(np.isnan(flux_for_5_sigma[0]), np.nan, flux_for_10_excess[0]),
#                       np.where(np.isnan(flux_for_5_sigma[1]), np.nan, flux_for_10_excess[1])]
```

In [178…

```
#
# Function to calculate the flux needed to obtain a given percent of the background,
# for all cut combinations:
#
def calc_flux_for_x_percent_backg(percent, cumul_excess, cumul_off):
    # In fraction of the flux of the test source (Crab):
    return percent/100*cumul_off/cumul_excess

#
# Do the computation (JUST FOR TEST!!):
#
# flux_for_5percent_backg = [[], []]
# for k in range(2):
#     flux_for_5percent_backg[k] = calc_flux_for_x_percent_backg(min_backg_percent, cum_excess_events[k],
#                                                                cum_off_events[k])

# flux_for_5percent_backg = [np.where(np.isnan(flux_for_5_sigma[0]), np.nan, flux_for_5percent_backg[0]),
#                            np.where(np.isnan(flux_for_5_sigma[1]), np.nan, flux_for_5percent_backg[1])]
```

In [179…

```
#
# Function to calculate the flux (for all cut combinations) which fulfills
# all 3 conditions of sensitivity (standard: at least 5-sigma significance,
# at least 10 excess events, and the excess being at least 5% of the background):
#
def calc_flux_3conditions(cumul_excess, cumul_off,
                          min_signi, min_exc, min_off_events, alpha,
                          target_obs_time, actual_obs_time,
                          min_n_gammas, min_backg_percent):

    f1, _ = calc_flux_for_N_sigma(5, cumul_excess, cumul_off,
                                  min_signi, min_exc, min_off_events, alpha,
                                  target_obs_time, actual_obs_time)
    f2 = 0
    f3 = 0

    if min_n_gammas > 0:
        f2 = calc_flux_for_N_excess(min_n_gammas, cumul_excess, target_obs_time, actual_obs_time)
    if min_backg_percent > 0:
        f3 = calc_flux_for_x_percent_backg(min_backg_percent, cumul_excess, cumul_off)

    return np.maximum(np.maximum(f1, f2), f3)
```

In [180…

```
#
# Definitions of min_signi, min_exc and min_off_events (to consider a cut combination valid) are above
#
# Do the computation of the detectable flux (for all cut combinations) according to the sensitivity conditions:
#
detectable_flux = [[],[]]

#Minimum flux fulfilling all three conditions:
for k in range(2):
    detectable_flux[k] = calc_flux_3conditions(cum_excess_events[k], cum_off_events[k],
                                               min_signi, min_exc, min_off_events, alpha,
                                               obs_time, livetimes[k]*0.5,
                                               min_n_gammas, min_backg_percent)
```

```
/tmp/ipykernel_10179/3511943101.py:6: RuntimeWarning: divide by zero encountered in true_divide
  return (N_excess/(cumul_excess*time_factor))
/tmp/ipykernel_10179/3629251318.py:7: RuntimeWarning: divide by zero encountered in true_divide
  return percent/100*cumul_off/cumul_excess
/tmp/ipykernel_10179/3629251318.py:7: RuntimeWarning: invalid value encountered in true_divide
  return percent/100*cumul_off/cumul_excess
```

In [181…

```
# Just in case, make sure we only consider bins with valid flux in *both* samples (odd- and even-events):
for k in range(2):
    mask = np.isnan(detectable_flux[k][0]) | np.isnan(detectable_flux[k][1])
    detectable_flux[k] = np.where(mask, np.nan, detectable_flux[k])
```

In [182…

```
sensitivity = [[[], []], [[], []]] # [analysis_type, odd_or_even, energy]
reco_energy = [[[], []], [[], []]]
signi = [[[], []], [[], []]]

cut_indices = [[[], []], [[], []]] # [analysis_type, odd_or_even, energy]

# Tweak: sometimes the minimization at low Ereco ends up with very tight alpha cut
```

```
# just because of a fluke... We try to avoid it here, by setting minimum "reasonable"
# cut values for the low-E bins:
#
min_angle_cut = [np.zeros(len(logenergy_bins)-1), np.zeros(len(logenergy_bins)-1)]
min_angle_cut[0][:5] = 0.02
min_angle_cut[1][:5] = 5

# Now, in each subsample (odd/even event_id's) we will find which cuts provide the best sensitivity
# (= minimum detectable flux), and will apply thoise cuts to the *other* subsample.

for analysis_type in range(2): # source-independent and source-dependent analyses
    for even_or_odd in range(2):
        # We optimize the cuts with the sample indicated by even_or_odd,
        # and apply them on the complementary sample, "other_half":
        if even_or_odd == 0:
            other_half = 1
        else:
            other_half = 0

        for iebin in range(len(logenergy_bins)-1):
            reco_energy[analysis_type][other_half].append(10**(0.5*(logenergy_bins[iebin]+logenergy_bins[iebin+

            # Now find the cuts which provide the minimum detectable flux using the events with odd event_id.

            # Except if we have only nan values... :
            if np.sum(~np.isnan(detectable_flux[analysis_type][even_or_odd][iebin])) == 0:
                sensitivity[analysis_type][other_half].append(np.nan)
                signi[analysis_type][other_half].append(np.nan)
                cut_indices[analysis_type][other_half].append([0, 0])
                continue

            if analysis_type == 0:
                start_bin = np.where(theta2bins>min_angle_cut[0][iebin])[0][0] - 1
            else:
                start_bin = np.where(alphabins>min_angle_cut[1][iebin])[0][0] - 1

            index = np.nanargmin(detectable_flux[analysis_type][even_or_odd][iebin,:,start_bin:])

            indices = list(np.unravel_index(index,
                                            detectable_flux[analysis_type][even_or_odd][iebin, :, start_bin:].s
            indices[1] += start_bin

            # Now get & store the minimum detectable flux with these cuts but using the other half of the event
            # Keep also the best-cut indices for later use

            sensitivity[analysis_type][other_half].append(detectable_flux[analysis_type][other_half][iebin, ind
            signi[analysis_type][other_half].append(lima_signi[analysis_type][other_half][iebin, indices[0], in
            cut_indices[analysis_type][other_half].append(indices)

        sensitivity[analysis_type][other_half] = np.array(sensitivity[analysis_type][other_half])
        reco_energy[analysis_type][other_half] = np.array(reco_energy[analysis_type][other_half])
        signi[analysis_type][other_half] = np.array(signi[analysis_type][other_half])
        cut_indices[analysis_type][other_half] = np.array(cut_indices[analysis_type][other_half])
```

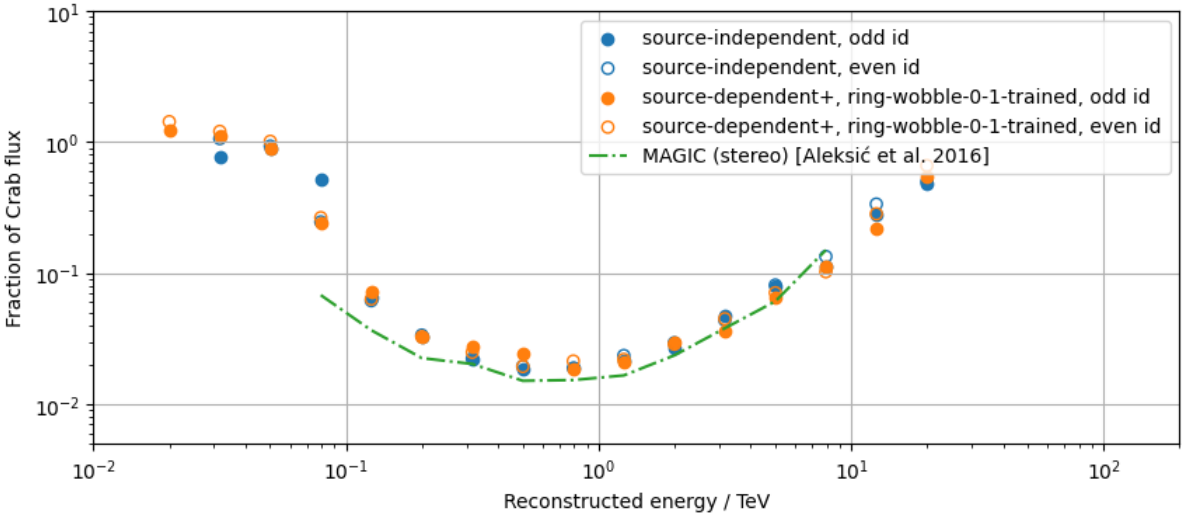
In [183…

```
#
# MAGIC sensitivity in Crab fraction:
#
def plot_MAGIC_sensitivity_fraction():
    s = np.loadtxt('magic_sensitivity.txt', skiprows = 1)
    energy = (s[:,0] * u.GeV).to(u.TeV)
    percentage = s[:,5]
    plt.plot(energy, percentage/100.,  '-.', label='MAGIC (stereo) [Aleksić et al. 2016]', color='tab:green')
    return
```

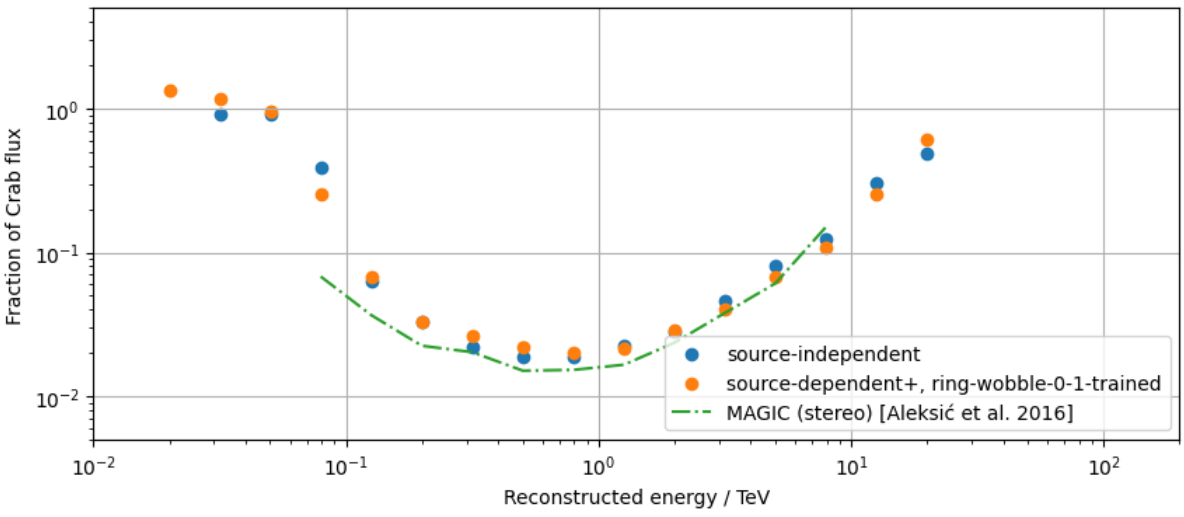
In [184…

```
plt.figure(figsize=(10,4))
plt.scatter(reco_energy[0][1], sensitivity[0][1], marker='o', facecolors='tab:blue', edgecolors='tab:blue',
            label=tag1+", odd id")
plt.scatter(reco_energy[0][0], sensitivity[0][0], marker='o', facecolors='none', edgecolors='tab:blue',
            label=tag1+", even id")

plt.scatter(reco_energy[1][1], sensitivity[1][1], marker='o', facecolors='tab:orange', edgecolors='tab:orange',
            label=tag2+", odd id")
plt.scatter(reco_energy[1][0], sensitivity[1][0], marker='o', facecolors='none', edgecolors='tab:orange',
            label=tag2+", even id")

plt.yscale('log')
plt.xscale('log')

plt.ylim(0.005, 10)
plt.xlim(0.01, 200)
plt.ylabel("Fraction of Crab flux")
plt.xlabel("Reconstructed energy / TeV")

plot_MAGIC_sensitivity_fraction()
```

```
plt.legend()
plt.grid()
plt.show()
```

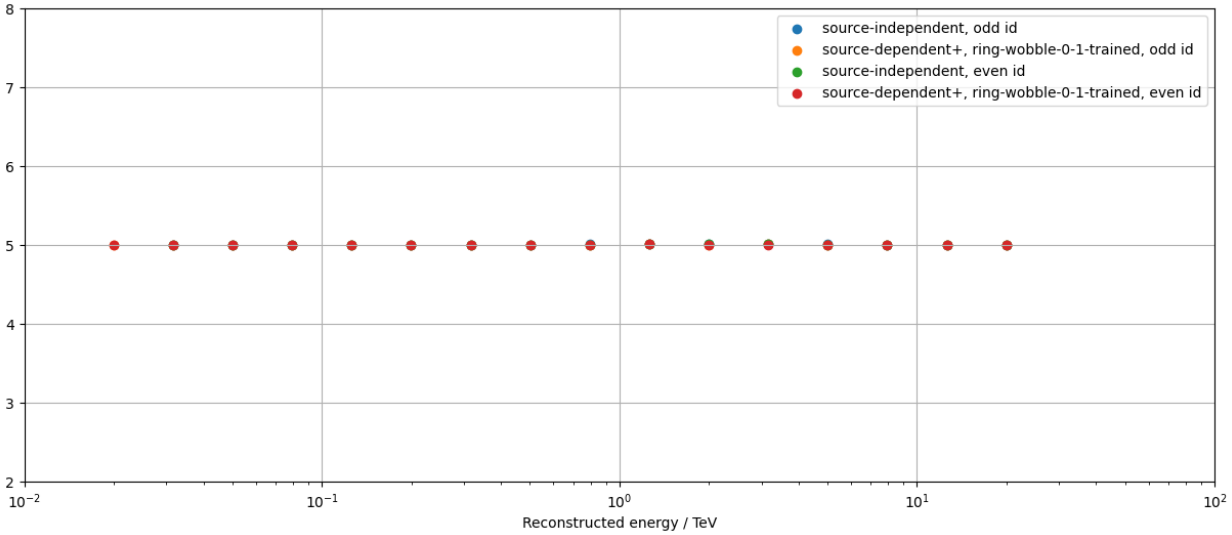


In [185…
```
plt.figure(figsize=(10,4))
plt.scatter(reco_energy[0][1], 0.5*(sensitivity[0][1]+sensitivity[0][0]),
            marker='o', facecolors='tab:blue', edgecolors='tab:blue',
            label=tag1)

plt.scatter(reco_energy[1][1], 0.5*(sensitivity[1][1]+sensitivity[1][0]), marker='o', facecolors='tab:orange',
            label=tag2)

plt.yscale('log')
plt.xscale('log')

plt.ylim(0.005, 5)
plt.xlim(0.01, 200)
plt.ylabel("Fraction of Crab flux")
plt.xlabel("Reconstructed energy / TeV")

plot_MAGIC_sensitivity_fraction()

plt.legend(loc='lower right')
plt.grid()
plt.show()
```



In [186…
```
# Check the significance for the optimal cuts, for all the energy bins:
plt.figure(figsize=(15,6))
plt.scatter(reco_energy[0][1], signi[0][1], label=tag1+", odd id")
plt.scatter(reco_energy[1][1], signi[1][1], label=tag2+", odd id")
plt.scatter(reco_energy[0][1], signi[0][0], label=tag1+", even id")
plt.scatter(reco_energy[1][1], signi[1][0], label=tag2+", even id")

plt.xscale('log')

plt.ylim(2, 8)
plt.xlim(0.01, 100)
plt.legend()
plt.xlabel('Reconstructed energy / TeV')
```

```
plt.grid()
plt.show()
```

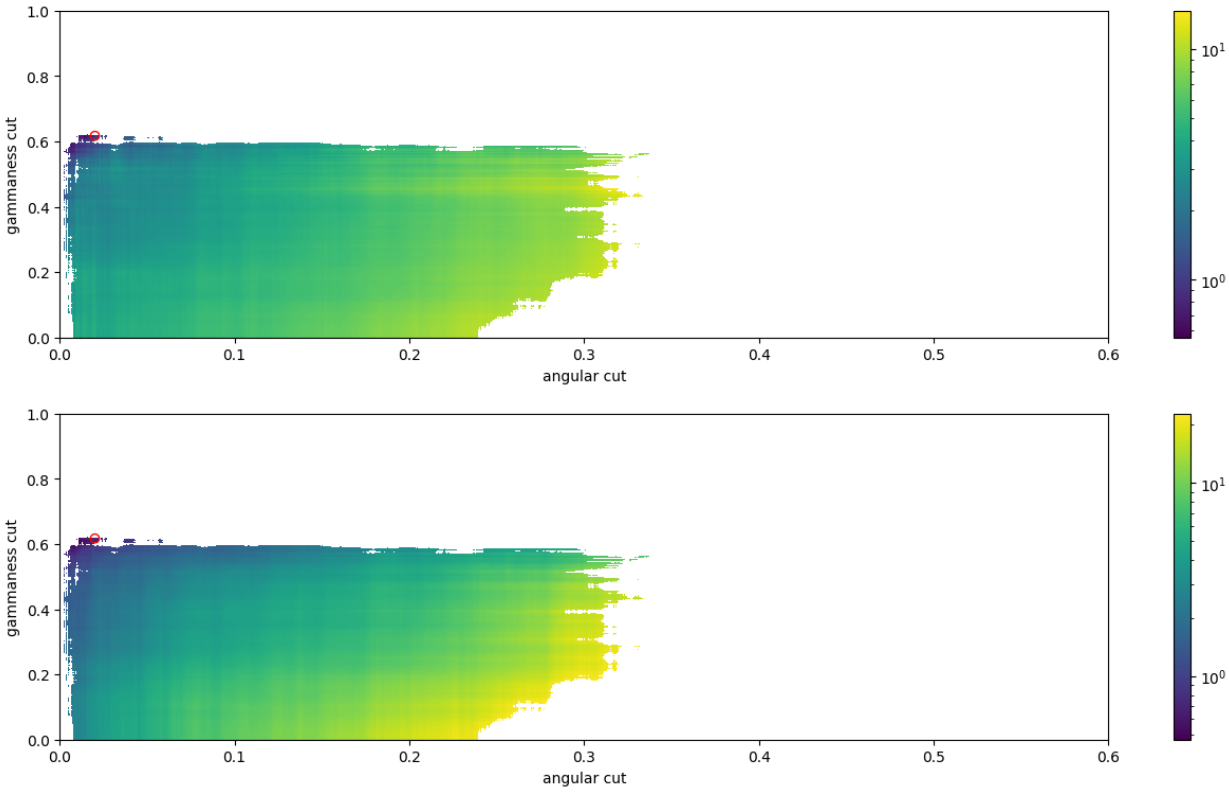


In [187…

```
# Just a test plot to see how the minimum flux is found on even-numbered events, and applied to odd-numbered ev
energyid = 2 # for energy bin
analysis_type = 0 # 0 source-independent, 1 source-dependent

print(10**(0.5*(logenergy_bins[energyid]+logenergy_bins[energyid+1])), "TeV")

if analysis_type == 0:
    start_bin = np.where(theta2bins>min_angle_cut[0][energyid])[0][0] - 1
else:
    start_bin = np.where(alphabins>min_angle_cut[1][energyid])[0][0] - 1

index = np.nanargmin(detectable_flux[analysis_type][0][energyid, :, start_bin:])
indices = list(np.unravel_index(index, detectable_flux[analysis_type][0][energyid, :, start_bin:].shape))
indices[1] += start_bin
# convert to indices in gammaness axis & angle axis

print("Minimum:", np.nanmin(detectable_flux[analysis_type][0][energyid, :, start_bin:]))
print(detectable_flux[analysis_type][0][energyid, indices[0], indices[1]])
print(detectable_flux[analysis_type][0][energyid,
                                        indices[0]-3:indices[0]+4,
                                        indices[1]-3:indices[1]+4])

gammaness_cut = gammaness_bins[indices[0]]
angular_cut = theta2bins[indices[1]+1]  # +1 because we want the bin's upper edge
if analysis_type == 1:
    angular_cut = alphabins[indices[1]+1]

print()
print("With the sample of even-numbered events:")
print('Minimum flux, gammaness & angular cut bins:', indices)
print('Cut values and minimum detectable flux (in fraction of Crab):',
      gammaness_cut, angular_cut, detectable_flux[analysis_type][0][energyid, indices[0], indices[1]])
print('Excess, Off events (for input sample), Li & Ma significance (in target t_obs for detectable flux):')
print('    ', cum_excess_events[analysis_type][0][energyid, indices[0], indices[1]],
      cum_off_events[analysis_type][0][energyid, indices[0], indices[1]],
      f'{lima_signi[analysis_type][0][energyid, indices[0], indices[1]]:.3f}')

plt.figure(figsize=(16,4))
if analysis_type == 0:
    plt.pcolormesh(theta2bins, gammaness_bins, detectable_flux[analysis_type][0][energyid], norm=colors.LogNorm
else:
    plt.pcolormesh(alphabins, gammaness_bins, detectable_flux[analysis_type][0][energyid], norm=colors.LogNorm(

plt.colorbar()
plt.scatter([angular_cut], [gammaness_cut], marker='o', facecolors='none', edgecolors='red')

plt.ylabel('gammaness cut')
plt.xlabel('angular cut')

# plt.xlim(0, 0.5)
# plt.ylim(0.65, 0.75)
plt.show()

plt.figure(figsize=(16,4))
if analysis_type == 0:
    plt.pcolormesh(theta2bins, gammaness_bins, detectable_flux[analysis_type][1][energyid], norm=colors.LogNorm
else:
    plt.pcolormesh(alphabins, gammaness_bins, detectable_flux[analysis_type][1][energyid], norm=colors.LogNorm(
plt.colorbar()
```

```
plt.scatter([angular_cut], [gammaness_cut], marker='o', facecolors='none', edgecolors='red')

plt.ylabel('gammaness cut')
plt.xlabel('angular cut')
# plt.xlim(0, 0.5)
# plt.ylim(0.65, 0.75)
plt.show()


print('With the sample of odd-numbered events:')
print('Applied cuts (from the other sample) and minimum detectable flux (in fraction of Crab):',
      gammaness_cut, angular_cut, detectable_flux[analysis_type][1][energyid, indices[0], indices[1]])
print('Excess, Off events (for input sample), Li & Ma significance (in target t_obs for detectable flux):')
print('    ', cum_excess_events[analysis_type][1][energyid, indices[0], indices[1]],
      cum_off_events[analysis_type][1][energyid, indices[0], indices[1]],
      f'{lima_signi[analysis_type][1][energyid, indices[0], indices[1]]:.3f}')
```

```
0.03162277660168379 TeV
Minimum: 0.88832486
0.88832486
[[0.8724 0.8535 0.8965 0.9251 0.9438 0.9465 0.9663]
 [0.8795 0.8619 0.9087 0.9333 0.9526 0.9577 0.9783]
 [0.8752 0.8574 0.9023 0.9221 0.9386 0.9461 0.9667]
 [0.8436 0.8268 0.8695 0.8883 0.9017 0.9089 0.9309]
 [0.8509 0.8375 0.8798 0.8991 0.913  0.9203 0.9458]
 [0.8766 0.8621 0.9076 0.9228 0.9376 0.9479    nan]
 [0.9164 0.9023    nan    nan    nan    nan    nan]]

With the sample of even-numbered events:
Minimum flux, gammaness & angular cut bins: [1234, 200]
Cut values and minimum detectable flux (in fraction of Crab): 0.617 0.0201 0.88832486
Excess, Off events (for input sample), Li & Ma significance (in target t_obs for detectable flux):
     394.0 7000.0 5.000
```




```
With the sample of odd-numbered events:
Applied cuts (from the other sample) and minimum detectable flux (in fraction of Crab): 0.617 0.0201 0.7657725
Excess, Off events (for input sample), Li & Ma significance (in target t_obs for detectable flux):
     466.0 7137.0 5.000
```

In [188…

```
#
# Function to find the cut indices (i.e. bin indices of the histos) which correspond to certain cuts
#
def find_bin_indices(gcut, angcut, analysis_type):
    # Find bin edge which is closest to the cut value:
    if analysis_type == 0:
        angcut_index = np.nanargmin(np.abs(theta2bins-angcut)) - 1
    else:
        angcut_index = np.nanargmin(np.abs(alphabins-angcut)) - 1

    gcut_index = np.nanargmin(np.abs(gammaness_bins-gcut))

    return gcut_index, angcut_index
```

In [189…

```
#
# Plot a rebinned histogram (for better visualization)
#
def plot_rebinned(x, y, yerr, rebin, label):
    xx = np.array([0.5*(x[i]+x[i+rebin]) for i in range(0, len(x)-1, rebin)])
    yy = np.array([np.sum(y[i:i+rebin]) for i in range(0, len(y)-1, rebin)])
    yyerr = np.array([(np.sum(yerr[i:i+rebin]**2))**0.5 for i in range(0, len(yerr)-1, rebin)])
    plt.errorbar(xx, yy, yerr=yyerr, fmt='o', markersize=3, label=label)
    return xx, yy
```

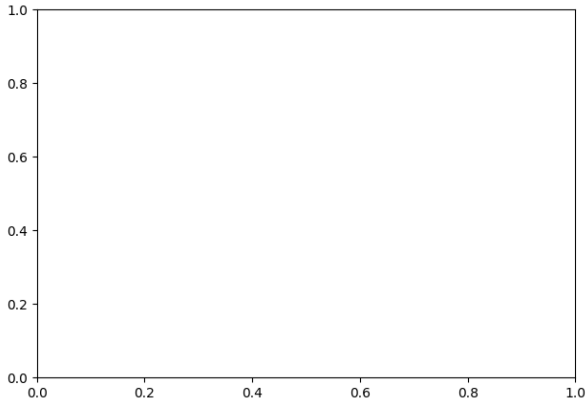
In [190…

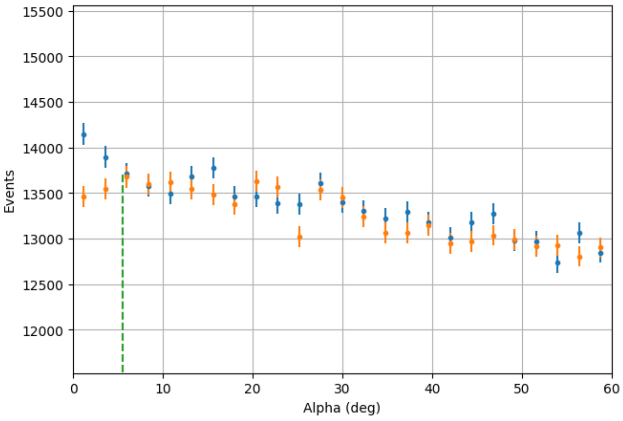
```
#
# Now we recalculate the sensitivities, and also check individual theta2 / Alphaplots:
#

target_obs_time = obs_time # = the same for which the cut optimization was done

# target_obs_time = 0.5 * u.h # CHANGE ONLY IN CASE YOU WANT TO CALCULATE SENSITIVITY
                              # FOR DIFFERENT T_OBS, BUT *WITHOUT* RE-OPTIMIZING CUTS!!



norm_bins = np.array([[np.where(theta2bins>norm_range[0][0])[0][0],
                       np.where(theta2bins>norm_range[0][1])[0][0]],
                      [np.where(alphabins>norm_range[1][0])[0][0],
                       np.where(alphabins>norm_range[1][1])[0][0]]
                      ])



rebin_factor = np.array((len(logenergy_bins)-1)*[20])  # join bins in groups of rebin_factor, to make plots les
#rebin_factor = np.array((len(logenergy_bins)-1)*[60])  # join bins in groups of rebin_factor, to make plots le

rebin_factor[:3] = 120
rebin_factor[3:5] = 60
rebin_factor[14:] = 60

sensitivity = np.empty((2, len(logenergy_bins)-1))
sensitivity[:] = np.nan

delta_sensitivity = np.empty((2, 2, len(logenergy_bins)-1)) # separate upper and lower error bars
delta_sensitivity[:] = np.nan

# For 5-sigma condition only:
# (NOTE! This is with the cuts optimized using all 3 conditions! In order to obtain the best sensitivity
# for just the 5 sigma condition, one has to set min_n_gammas=0 and min_backg_percent=0 BEFORE the cut
# optimization!)
sensitivity_5s = np.zeros_like(sensitivity)
delta_sensitivity_5s = np.zeros_like(delta_sensitivity)
sensitivity_5s[:] = np.nan
delta_sensitivity_5s[:] = np.nan

reco_energy = np.zeros_like(sensitivity)
num_excess_events = np.zeros_like(sensitivity)
num_off_events = np.zeros_like(sensitivity)
reco_energy[:] = np.nan
num_excess_events[:] = np.nan
num_off_events[:] = np.nan

angular_cut = np.empty((2, 2, len(logenergy_bins)-1))
gammaness_cut = np.empty((2, 2, len(logenergy_bins)-1)) # analysis_type, odd_or_even, energy
angular_cut[:] = np.nan
gammaness_cut[:] = np.nan


for iebin in range(len(logenergy_bins)-1):

    recoE = 10**(0.5*(logenergy_bins[iebin]+logenergy_bins[iebin+1]))
    reco_energy[0][iebin] = recoE
    reco_energy[1][iebin] = recoE


    if (recoE < 0.016) | (recoE > 16):
        continue

    print(f'Energy: {recoE:.4f} Tev')

    fig = plt.figure(figsize=(16, 5))

    for analysis_type in range(2):

        indices0 = cut_indices[analysis_type][0][iebin]
        indices1 = cut_indices[analysis_type][1][iebin]
        # indices0 and indices1 have 2 elements each: [0] is the gammaness cut, [1] is the angular cut

        if (indices0>0).all() and (indices1>0).all():
```

```
        # Valid gammanness & angular cuts (cut indices ==0 means no valid cuts could be determined!)
        nevts_on = (np.sum(on_events[analysis_type][0, iebin, indices0[0]:], axis=0) +
                    np.sum(on_events[analysis_type][1, iebin, indices1[0]:], axis=0))

        nevts_off = (np.sum(off_events[analysis_type][0, iebin, indices0[0]:], axis=0) +
                     np.sum(off_events[analysis_type][1, iebin, indices1[0]:], axis=0))
    else:
        nevts_on = None
        nevts_off = None

    if analysis_type == 0:
        fig.add_subplot(1, 2, 1+analysis_type)

        if nevts_on is None:
            print('No valid cuts found for source-independent analysis!')
            continue
        else:
            print(f'Gammaness cuts: {theta2bins[indices0[0]+1]:.4f}, {theta2bins[indices1[0]+1]:.4f}')
            print(f'Theta2 cuts: {theta2bins[indices0[1]+1]:.4f}, {theta2bins[indices1[1]+1]:.4f}')

        xx, yy = plot_rebinned(theta2bins, nevts_on, nevts_on**0.5, rebin_factor[iebin], '')
        xxoff, yyoff = plot_rebinned(theta2bins, nevts_off, nevts_off**0.5, rebin_factor[iebin], '')

        plt.plot([theta2bins[indices0[1]+1], theta2bins[indices0[1]+1]],
                 [0, yy[int((indices0[1]+1)/rebin_factor[iebin])]], '--', color='tab:green')
        plt.plot([theta2bins[indices1[1]+1], theta2bins[indices1[1]+1]],
                 [0, yy[int((indices1[1]+1)/rebin_factor[iebin])]], '--', color='tab:green')

        plt.xlim(0, 0.2)
        plt.ylim(yyoff.min()*0.9, yy.max()*1.1)
        plt.xlabel('Theta2 (deg2)')
        plt.ylabel('Events')

        angular_cut[analysis_type][0][iebin] = theta2bins[indices0[1]+1]
        angular_cut[analysis_type][1][iebin] = theta2bins[indices1[1]+1]

    else:
        fig.add_subplot(1, 2, 1+analysis_type)

        if nevts_on is None:
            print('No valid cuts found for source-dependent analysis!')
            continue
        else:
            print(f'Alpha cuts: {alphabins[indices0[1]+1]:.2f}, {alphabins[indices1[1]+1]:.2f}')

        xx, yy = plot_rebinned(alphabins, nevts_on, nevts_on**0.5, rebin_factor[iebin], '')
        xxoff, yyoff = plot_rebinned(alphabins, nevts_off, nevts_off**0.5, rebin_factor[iebin], '')

        plt.plot([alphabins[indices0[1]+1], alphabins[indices0[1]+1]],
                 [0, yy[int((indices0[1]+1)/rebin_factor[iebin])]], '--', color='tab:green')
        plt.plot([alphabins[indices1[1]+1], alphabins[indices1[1]+1]],
                 [0, yy[int((indices1[1]+1)/rebin_factor[iebin])]], '--', color='tab:green')
        plt.xlim(0, 60)
        plt.ylim(yyoff.min()*0.9, yy.max()*1.1)
        plt.xlabel('Alpha (deg)')
        plt.ylabel('Events')

        angular_cut[analysis_type][0][iebin] = alphabins[indices0[1]+1]
        angular_cut[analysis_type][1][iebin] = alphabins[indices1[1]+1]

    # Add up the backg numbers (odd and even events) in the normalization region, and the excess:
    off_in_norm_region = (cum_off_events[analysis_type][0, iebin, indices0[0], norm_bins[analysis_type][1]]
                          cum_off_events[analysis_type][1, iebin, indices1[0], norm_bins[analysis_type][1]]
                          cum_off_events[analysis_type][0, iebin, indices0[0], norm_bins[analysis_type][0]]
                          cum_off_events[analysis_type][1, iebin, indices1[0], norm_bins[analysis_type][0]]
                          )
    excess_in_norm_region = (cum_excess_events[analysis_type][0, iebin, indices0[0], norm_bins[analysis_typ
                             cum_excess_events[analysis_type][1, iebin, indices1[0], norm_bins[analysis_typ
                             cum_excess_events[analysis_type][0, iebin, indices0[0], norm_bins[analysis_typ
                             cum_excess_events[analysis_type][1, iebin, indices1[0], norm_bins[analysis_typ
                            )
    off_norm_factor = 1
    if backg_normalization:
        off_norm_factor = (off_in_norm_region + excess_in_norm_region) / off_in_norm_region
        print('Off normalization for analysis type', analysis_type, ':', off_norm_factor)

        norm_min = (((off_in_norm_region + excess_in_norm_region) -
                     (off_in_norm_region + excess_in_norm_region)**0.5) /
                    (off_in_norm_region + off_in_norm_region**0.5))
        norm_max = (((off_in_norm_region + excess_in_norm_region) +
                     (off_in_norm_region + excess_in_norm_region)**0.5) /
                    (off_in_norm_region - off_in_norm_region**0.5))
        print(f'     {norm_min:.4f} to {norm_max:.4f}')


    gammaness_cut[analysis_type][0][iebin] = gammaness_bins[indices0[0]]
```

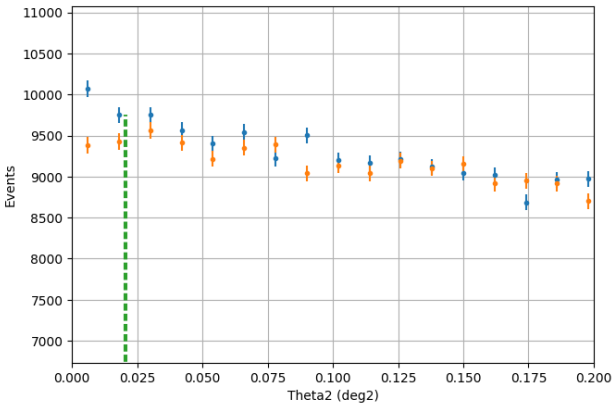
```
        gammaness_cut[analysis_type][1][iebin] = gammaness_bins[indices1[0]]

        # Add up the excess (and the off) for odd and even event_id's
        nexcess = (cum_excess_events[analysis_type][0, iebin, indices0[0], indices0[1]] +
                   cum_excess_events[analysis_type][1, iebin, indices1[0], indices1[1]])
        noff = (cum_off_events[analysis_type][0, iebin, indices0[0], indices0[1]] +
                cum_off_events[analysis_type][1, iebin, indices1[0], indices1[1]])


        nexcess = nexcess + noff * (1 - off_norm_factor)
        noff = noff * off_norm_factor


        flux = calc_flux_3conditions(np.array([nexcess]), np.array([noff]),
                                     min_signi, min_exc, min_off_events, alpha,
                                     target_obs_time, livetimes[analysis_type],
                                     min_n_gammas, min_backg_percent)

        if analysis_type == 0:
            print(f'Results (source-indep): Nexc={nexcess}, Noff={noff},  Flux/CU={flux[0]:.4f}')
        else:
            print(f'Results (source-dep): Nexc={nexcess}, Noff={noff},  Flux/CU={flux[0]:.4f}')

        sensitivity[analysis_type][iebin] = flux[0]

        # Assume background systematics and statistical fluctuation of excess go in same direction:
        max_excess = nexcess + backg_syst * noff + (nexcess + 2*noff)**0.5
        min_excess = nexcess - backg_syst * noff - (nexcess + 2*noff)**0.5

        flux_minus = calc_flux_3conditions(np.array([max_excess]),
                                           np.array([noff]),
                                           0, 0, 0, alpha,
                                           target_obs_time, livetimes[analysis_type],
                                           min_n_gammas, min_backg_percent)
        flux_plus = calc_flux_3conditions(np.array([min_excess]),
                                          np.array([noff]),
                                          0, 0, 0, alpha,
                                          target_obs_time, livetimes[analysis_type],
                                          min_n_gammas, min_backg_percent)

        delta_sensitivity[analysis_type][1][iebin] = flux_plus[0] - flux[0]
        delta_sensitivity[analysis_type][0][iebin] = flux[0] - flux_minus[0]


        # Now only with the 5-sigma condition (remove req for 10 excess events & 5% of backg):

        flux_5s = calc_flux_3conditions(np.array([nexcess]), np.array([noff]),
                                        min_signi, min_exc, min_off_events, alpha,
                                        target_obs_time, livetimes[analysis_type],
                                        0, 0)
        sensitivity_5s[analysis_type][iebin] = flux_5s[0]
        flux_5s_minus = calc_flux_3conditions(np.array([max_excess]),
                                              np.array([noff]),
                                              0, 0, 0, alpha,
                                              target_obs_time, livetimes[analysis_type],
                                              0, 0)
        flux_5s_plus = calc_flux_3conditions(np.array([min_excess]),
                                             np.array([noff]),
                                             0, 0, 0, alpha,
                                             target_obs_time, livetimes[analysis_type],
                                             0, 0)

        delta_sensitivity_5s[analysis_type][1][iebin] = flux_5s_plus[0] - flux_5s[0]
        delta_sensitivity_5s[analysis_type][0][iebin] = flux_5s[0] - flux_5s_minus[0]


        num_excess_events[analysis_type][iebin] = nexcess
        num_off_events[analysis_type][iebin] = noff

        # plt.ylim(0, 250)

        plt.grid()
        # plt.legend()

    plt.show()
    print()
```

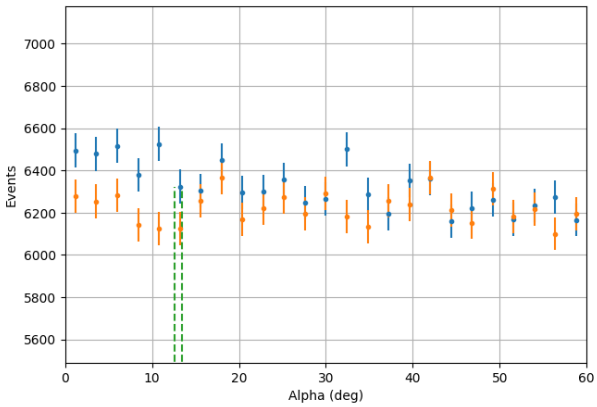
```
Energy: 0.0200 Tev
No valid cuts found for source-independent analysis!
Alpha cuts: 5.52, 5.50
Results (source-dep): Nexc=1168.0, Noff=31007.0,  Flux/CU=1.3274
```



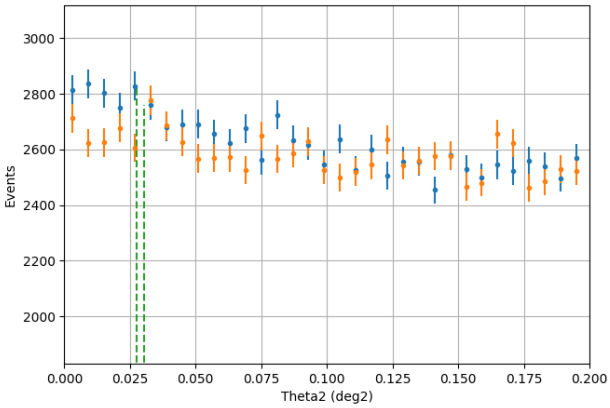


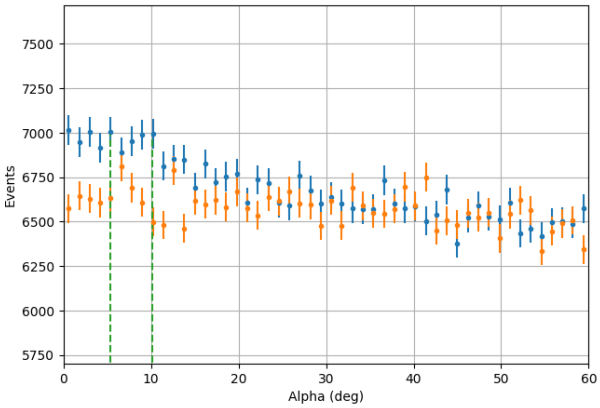
Energy: 0.0316 Tev
Gammaness cuts: 0.1223, 0.1235
Theta2 cuts: 0.0209, 0.0201
Results (source-indep): Nexc=890.0, Noff=16151.0, Flux/CU=0.9074
Alpha cuts: 12.56, 13.50
Results (source-dep): Nexc=1412.0, Noff=33270.0, Flux/CU=1.1781



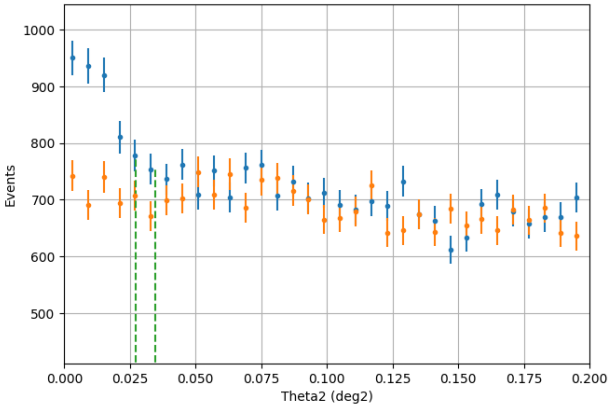


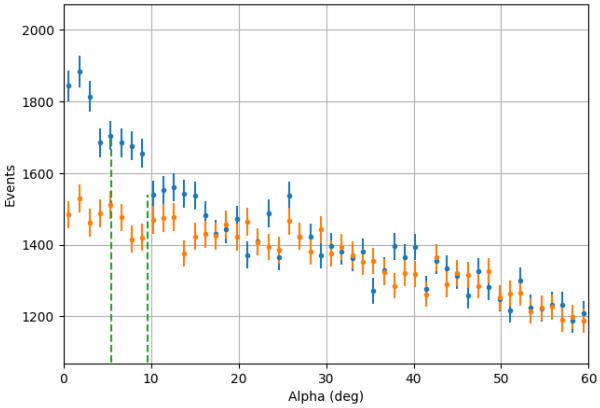
Energy: 0.0501 Tev
Gammaness cuts: 0.1417, 0.1416
Theta2 cuts: 0.0304, 0.0277
Results (source-indep): Nexc=704.0, Noff=12852.0, Flux/CU=0.9128
Alpha cuts: 10.18, 5.40
Results (source-dep): Nexc=1819.0, Noff=33475.0, Flux/CU=0.9201



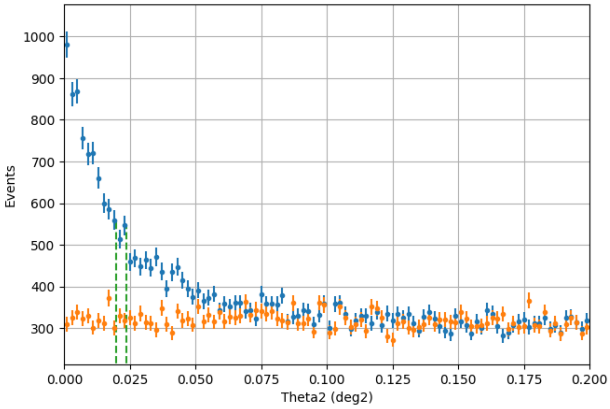


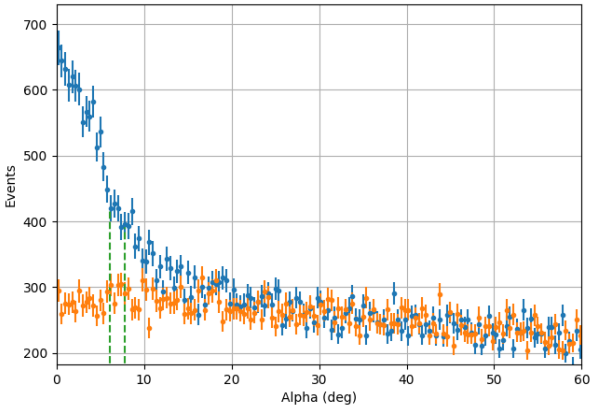
Energy: 0.0794 Tev
Gammaness cuts: 0.1652, 0.1820
Theta2 cuts: 0.0347, 0.0271
Results (source-indep): Nexc=876.0, Noff=4087.0, Flux/CU=0.3351
Alpha cuts: 5.48, 9.60
Results (source-dep): Nexc=1746.0, Noff=8750.0, Flux/CU=0.2506



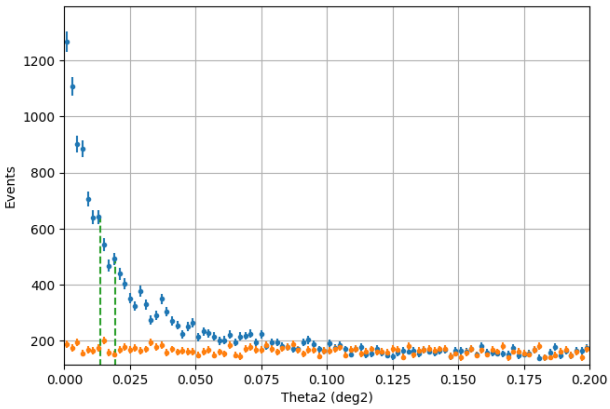


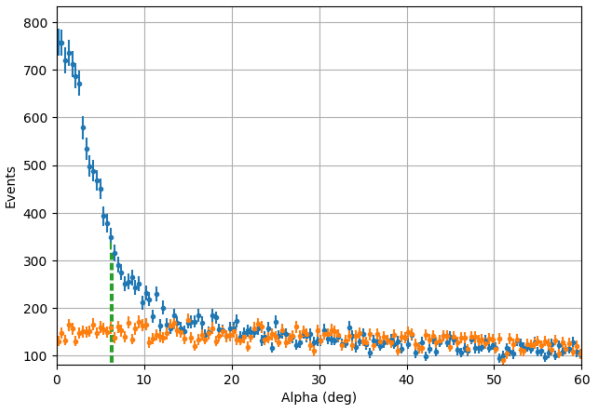
Energy: 0.1259 Tev
Gammaness cuts: 0.1774, 0.1760
Theta2 cuts: 0.0235, 0.0199
Results (source-indep): Nexc=4254.0, Noff=3457.0, Flux/CU=0.0635
Alpha cuts: 7.84, 6.04
Results (source-dep): Nexc=4766.0, Noff=5116.0, Flux/CU=0.0688



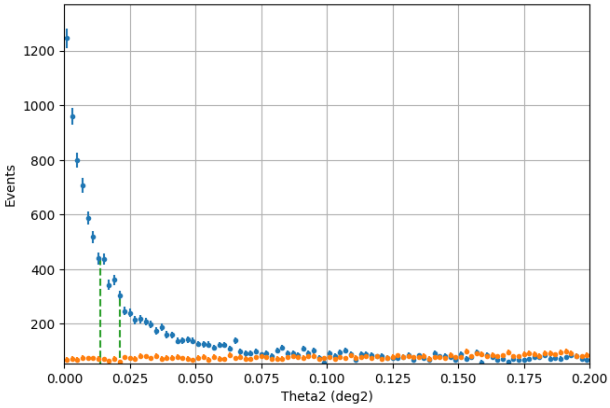


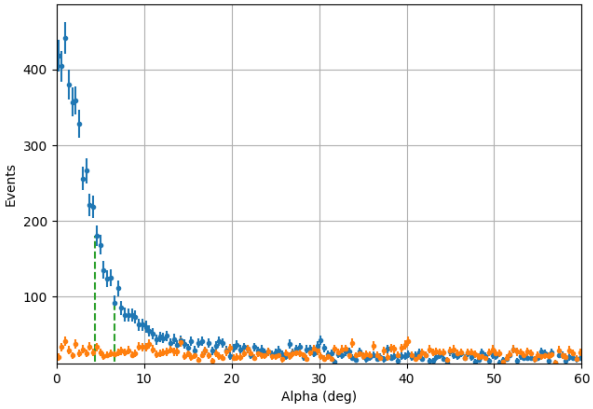
Energy: 0.1995 Tev
Gammaness cuts: 0.1804, 0.1824
Theta2 cuts: 0.0195, 0.0138
Results (source-indep): Nexc=5357.0, Noff=1513.0, Flux/CU=0.0336
Alpha cuts: 6.22, 6.44
Results (source-dep): Nexc=6746.0, Noff=2395.0, Flux/CU=0.0335



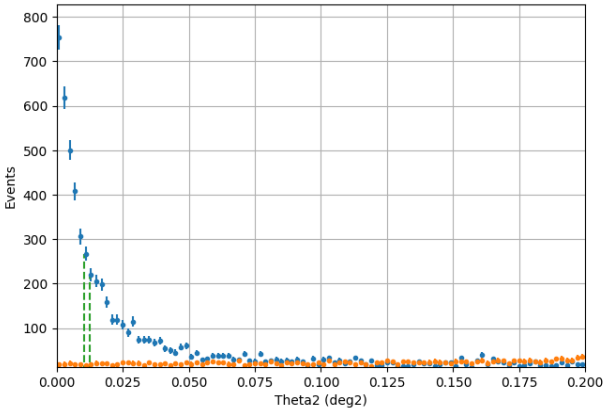


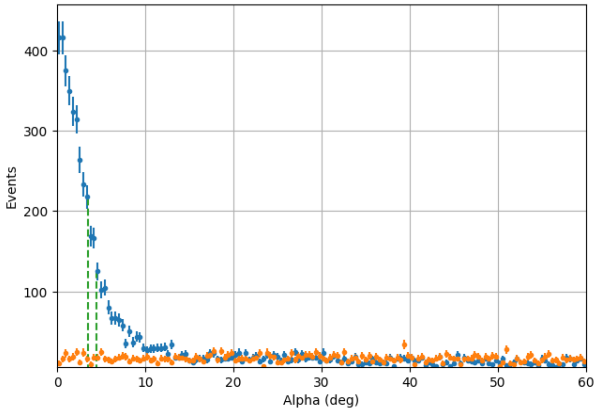
Energy: 0.3162 Tev
Gammaness cuts: 0.1855, 0.1841
Theta2 cuts: 0.0211, 0.0138
Results (source-indep): Nexc=5197.6669921875, Noff=612.333251953125, Flux/CU=0.0223
Alpha cuts: 6.64, 4.42
Results (source-dep): Nexc=3687.0, Noff=434.0, Flux/CU=0.0267



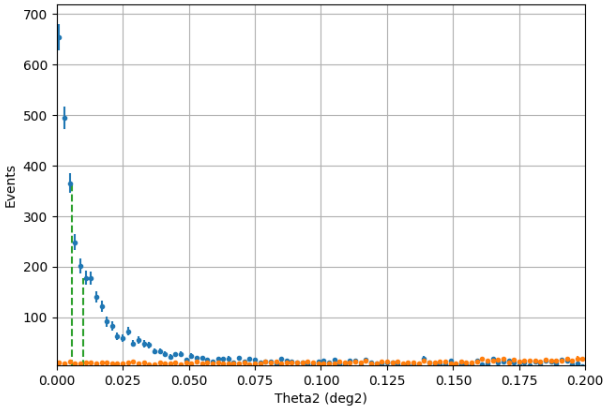


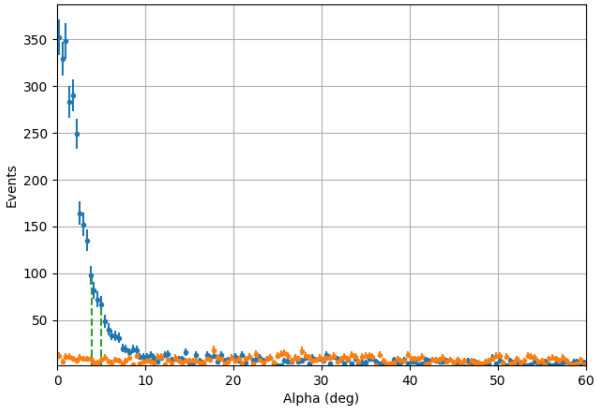
Energy: 0.5012 Tev
Gammaness cuts: 0.1934, 0.1888
Theta2 cuts: 0.0103, 0.0126
Results (source-indep): Nexc=2693.0, Noff=114.00004577636719, Flux/CU=0.0194
Alpha cuts: 4.42, 3.48
Results (source-dep): Nexc=2957.0, Noff=188.0, Flux/CU=0.0223



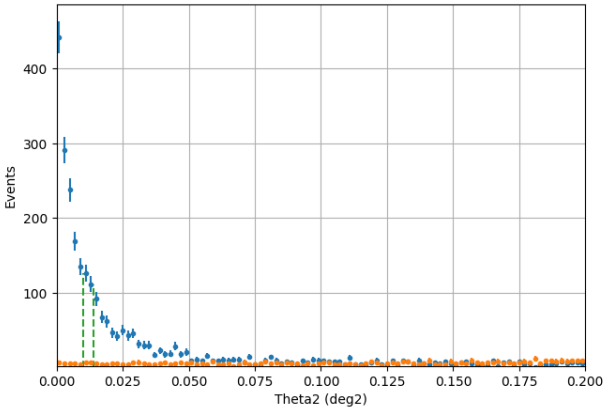


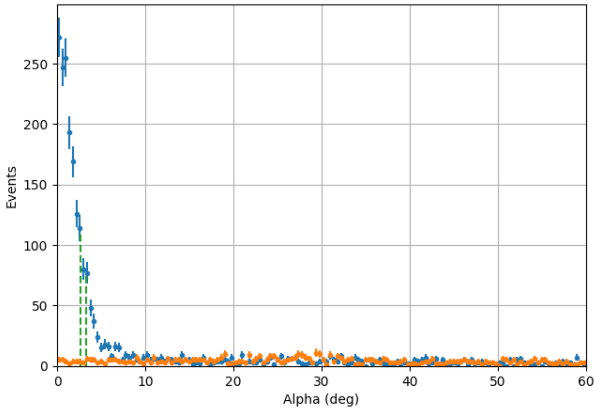
Energy: 0.7943 Tev
Gammaness cuts: 0.1925, 0.1928
Theta2 cuts: 0.0100, 0.0057
Results (source-indep): Nexc=1677.333251953125, Noff=38.66666793823242, Flux/CU=0.0192
Alpha cuts: 3.96, 4.98
Results (source-dep): Nexc=2371.0, Noff=97.0, Flux/CU=0.0205



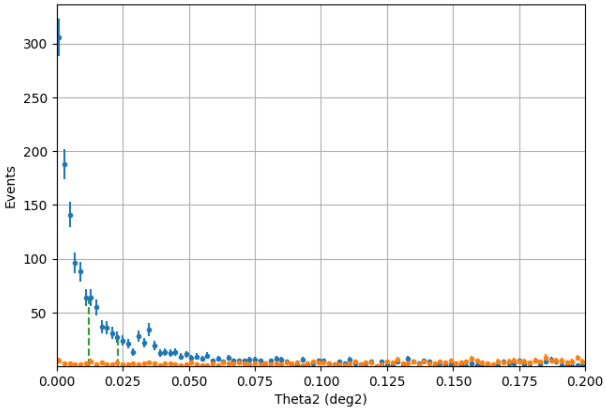


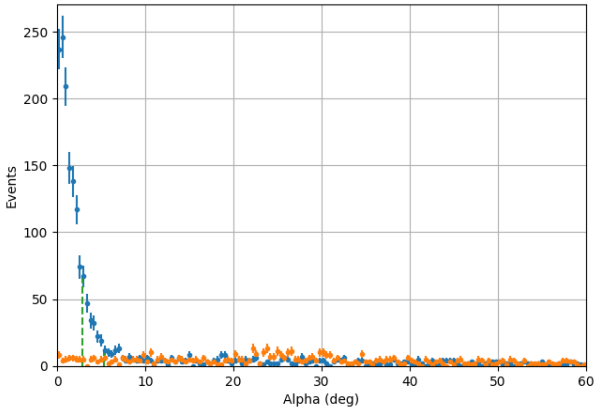
Energy: 1.2589 Tev
Gammaness cuts: 0.1910, 0.1941
Theta2 cuts: 0.0102, 0.0139
Results (source-indep): Nexc=1342.333251953125, Noff=33.666664123535156, Flux/CU=0.0226
Alpha cuts: 3.30, 2.64
Results (source-dep): Nexc=1370.0, Noff=32.0, Flux/CU=0.0216



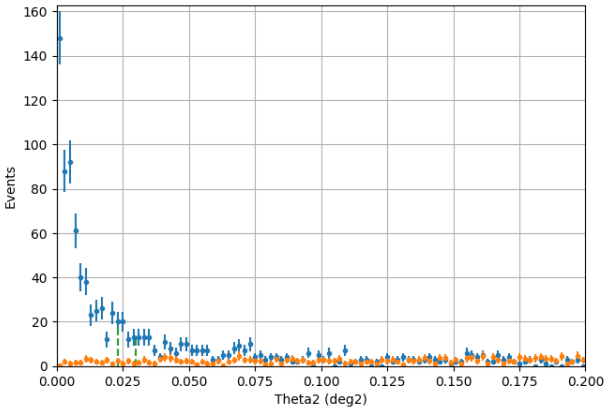


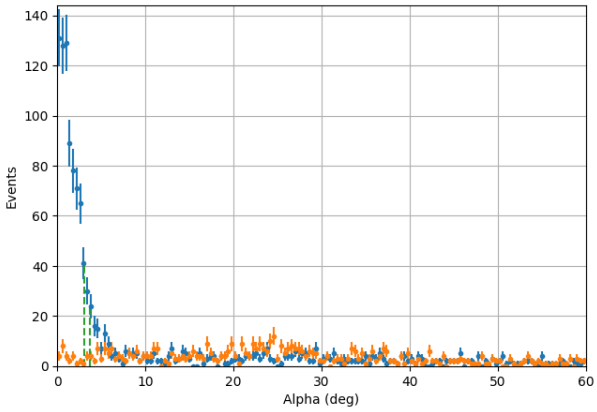
Energy: 1.9953 Tev
Gammaness cuts: 0.1944, 0.1925
Theta2 cuts: 0.0232, 0.0122
Results (source-indep): Nexc=970.3333740234375, Noff=26.66666030883789, Flux/CU=0.0282
Alpha cuts: 2.86, 5.34
Results (source-dep): Nexc=1216.0, Noff=48.0, Flux/CU=0.0291



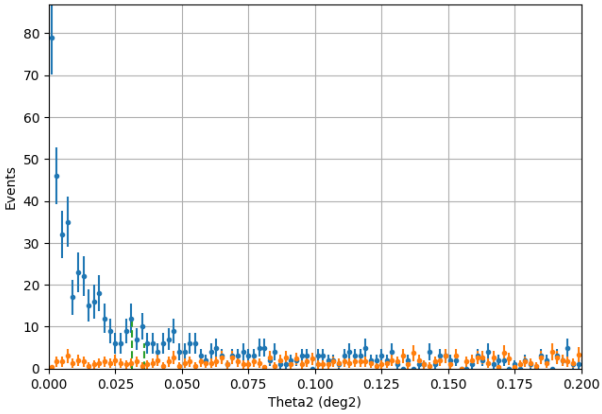


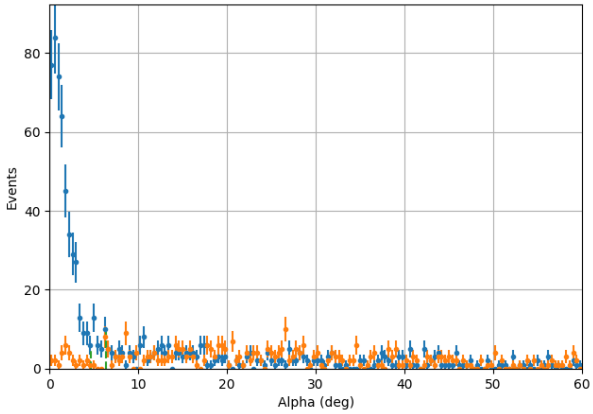
Energy: 3.1623 Tev
Gammaness cuts: 0.1909, 0.1921
Theta2 cuts: 0.0232, 0.0300
Results (source-indep): Nexc=584.6666870117188, Noff=25.333328247070312, Flux/CU=0.0458
Alpha cuts: 3.68, 3.06
Results (source-dep): Nexc=715.0, Noff=31.0, Flux/CU=0.0409



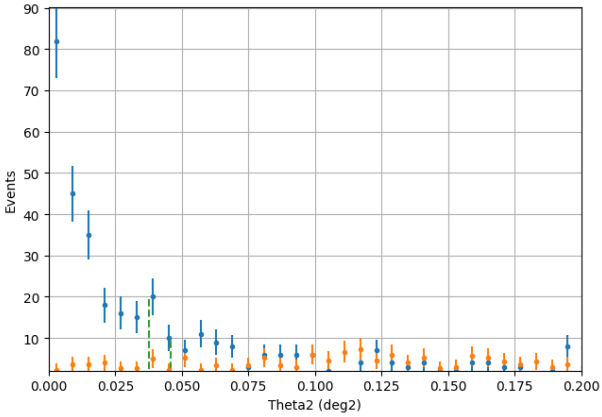


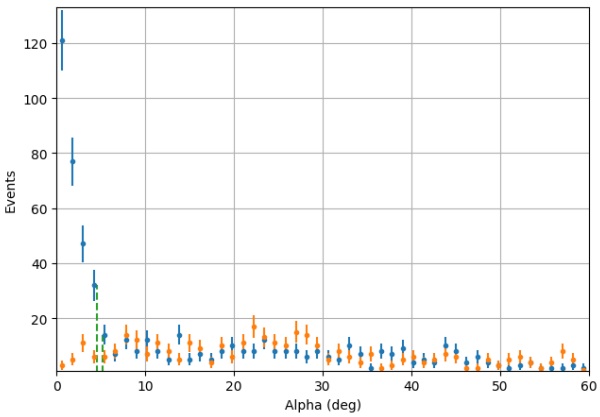
Energy: 5.0119 Tev
Gammaness cuts: 0.1877, 0.1898
Theta2 cuts: 0.0313, 0.0360
Results (source-indep): Nexc=331.0, Noff=24.999996185302734, Flux/CU=0.0804
Alpha cuts: 4.70, 6.36
Results (source-dep): Nexc=442.0, Noff=34.0, Flux/CU=0.0687



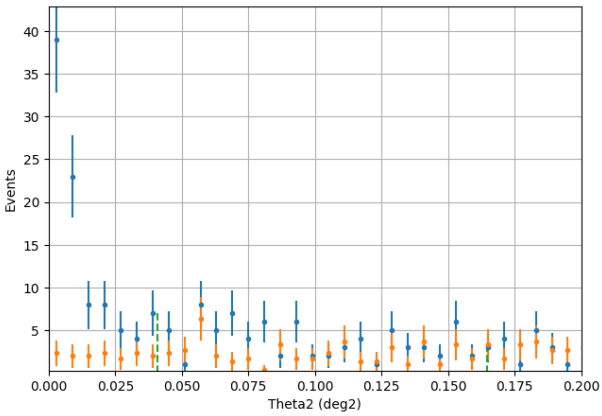


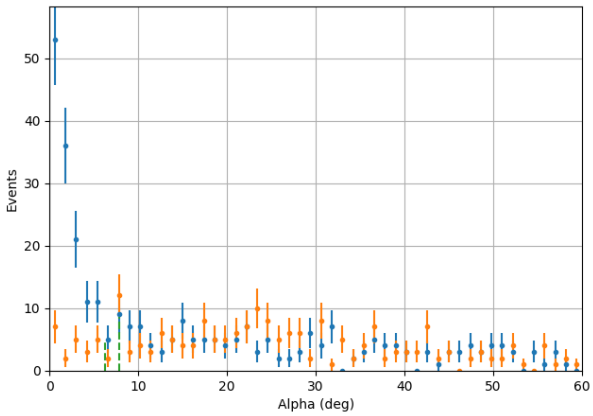

```
Energy: 7.9433 Tev
Gammaness cuts: 0.1790, 0.1839
Theta2 cuts: 0.0377, 0.0458
Results (source-indep): Nexc=205.0, Noff=21.999996185302734,  Flux/CU=0.1230
Alpha cuts: 4.54, 5.20
Results (source-dep): Nexc=250.0, Noff=26.0,  Flux/CU=0.1082
```






```
Energy: 12.5893 Tev
Gammaness cuts: 0.1872, 0.1665
Theta2 cuts: 0.1646, 0.0408
Results (source-indep): Nexc=86.00000762939453, Noff=23.9999942779541,  Flux/CU=0.3039
Alpha cuts: 7.86, 6.22
Results (source-dep): Nexc=111.0, Noff=27.0,  Flux/CU=0.2476
```

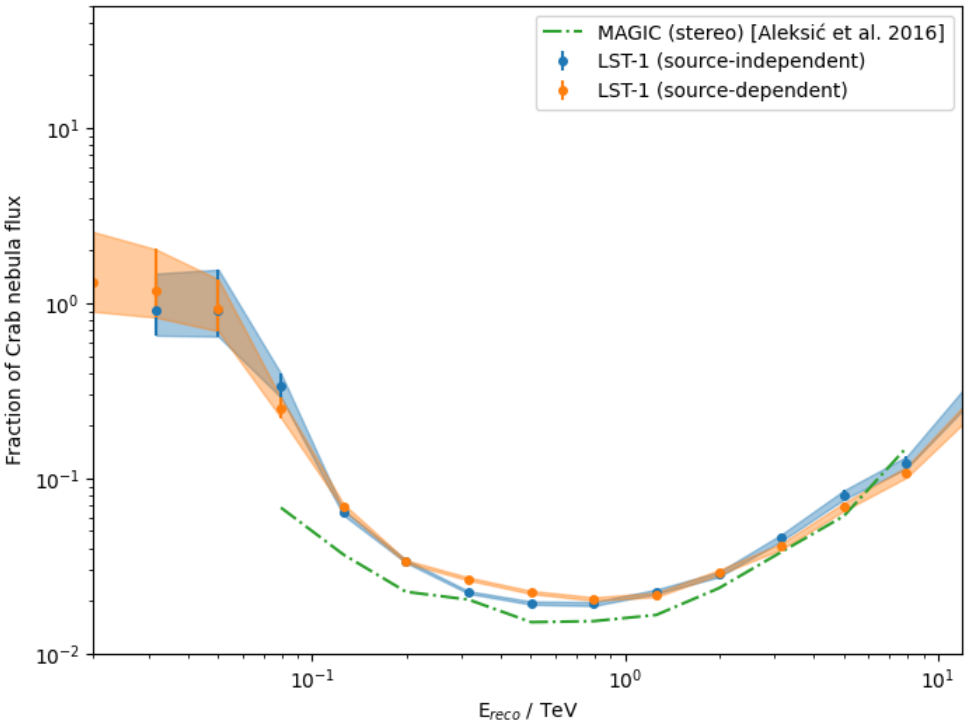




In [191…

```
#
# FINAL SENSITIVITY PLOTS
#
plt.figure(figsize=(8,6))


plt.fill_between(reco_energy[0][:-1],
                 (sensitivity[0]-delta_sensitivity[0][0])[:-1],
                 (sensitivity[0]+delta_sensitivity[0][1])[:-1], alpha=0.4, color='tab:blue')


plt.fill_between(reco_energy[1][:-1],
                 (sensitivity[1]-delta_sensitivity[1][0])[:-1],
                 (sensitivity[1]+delta_sensitivity[1][1])[:-1], alpha=0.4, color='tab:orange')


plt.errorbar(reco_energy[0], sensitivity[0],
             yerr=delta_sensitivity[0], marker='o', color='tab:blue', ls='none', markersize=4,
             label='LST-1 (source-independent)')
```

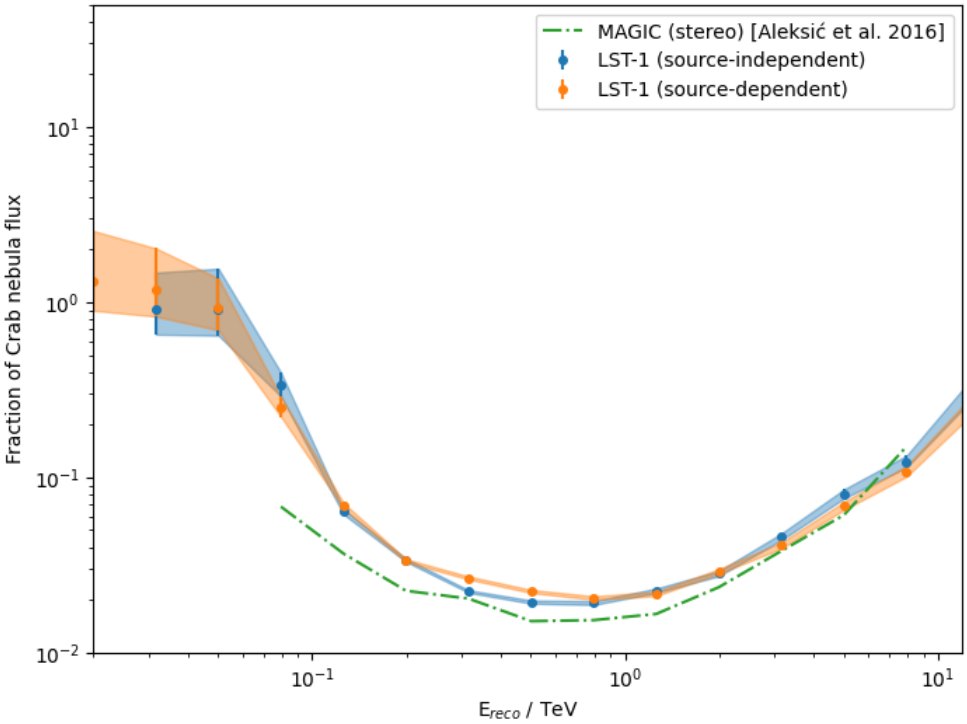
```
plt.errorbar(reco_energy[1], sensitivity[1],
             yerr=delta_sensitivity[1], marker='o', color='tab:orange', ls='none', markersize=4,
             label='LST-1 (source-dependent)')


plot_MAGIC_sensitivity_fraction()

plt.xscale('log')
plt.yscale('log')
plt.xlabel('E$_{reco}$ / TeV')
plt.ylabel('Fraction of Crab nebula flux')
plt.ylim(0.01, 50)
plt.xlim(0.02, 12)
plt.legend()
plt.show()
```



In [192…
```
# Write out the results:
Output_filename = "Sensitivity_output.csv"
np.savetxt(Output_filename, [reco_energy[0],
                             sensitivity[0],
                             delta_sensitivity[0][0],
                             delta_sensitivity[0][1],
                             sensitivity[1],
                             delta_sensitivity[1][0],
                             delta_sensitivity[1][1],
                             sensitivity_5s[0],
                             delta_sensitivity_5s[0][0],
                             delta_sensitivity_5s[0][1],
                             sensitivity_5s[1],
                             delta_sensitivity_5s[1][0],
                             delta_sensitivity_5s[1][1]],
           fmt='%.5e', delimiter=',',
           header = 'E(TeV)  SI_sensitivity, SI_delta_sensitivity_low, SI_delta_sensitivity_high, '+
           'SD_sensitivity, SD_delta_sensitivity_low, SD_delta_sensitivity_high, '+
           'SI_sensitivity_5s, SI_delta_sensitivity_5s_low, SI_delta_sensitivity_5s_high, '+
           'SD_sensitivity_5s, SD_delta_sensitivity_5s_low, SD_delta_sensitivity_5s_high')
```

In [193…
```
# REDO PLOT FROM CSV FILE:

plt.figure(figsize=(8,6))

data = np.loadtxt(Output_filename, delimiter=',')
plt.fill_between(data[0][:-1],
                 (data[1]-data[2])[:-1],
                 (data[1]+data[3])[:-1], alpha=0.4, color='tab:blue')
plt.fill_between(data[0][:-1],
                 (data[4]-data[5])[:-1],
                 (data[4]+data[6])[:-1], alpha=0.4, color='tab:orange')

plt.errorbar(data[0], data[1],
             yerr=[data[2], data[3]], marker='o', color='tab:blue', ls='none', markersize=4,
             label='LST-1 (source-independent)')
```

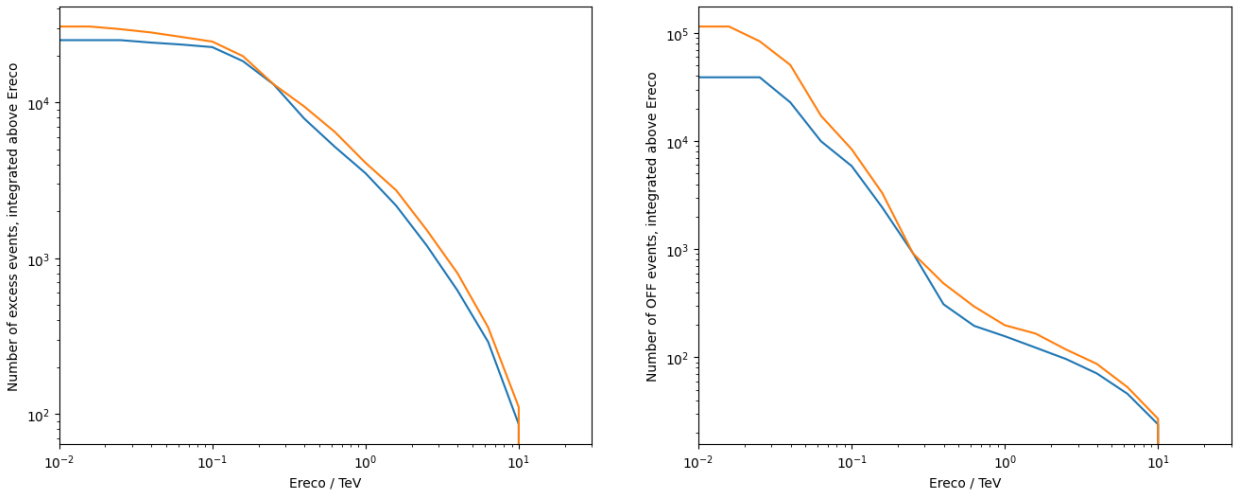
```
plt.errorbar(data[0], data[4],
             yerr=[data[5], data[6]], marker='o', color='tab:orange', ls='none', markersize=4,
             label='LST-1 (source-dependent)')

plot_MAGIC_sensitivity_fraction()

plt.xscale('log')
plt.yscale('log')
plt.xlabel('E$_{reco}$ / TeV')
plt.ylabel('Fraction of Crab nebula flux')
plt.xlim(0.02, 12)
plt.ylim(0.01, 50)
plt.legend()
plt.show()
```



In [194…
```
np.nancumsum(num_excess_events[0][::-1])[::-1]
```

Out[194]:
```
array([25168.3336, 25168.3336, 25168.3336, 24278.3336, 23574.3336,
       22698.3336, 18444.3336, 13087.3336,  7889.6666,  5196.6666,
        3519.3333,  2177.0001,  1206.6667,   622.    ,   291.    ,
          86.    ,     0.    ,     0.    ,     0.    ,     0.    ,
           0.    ])
```

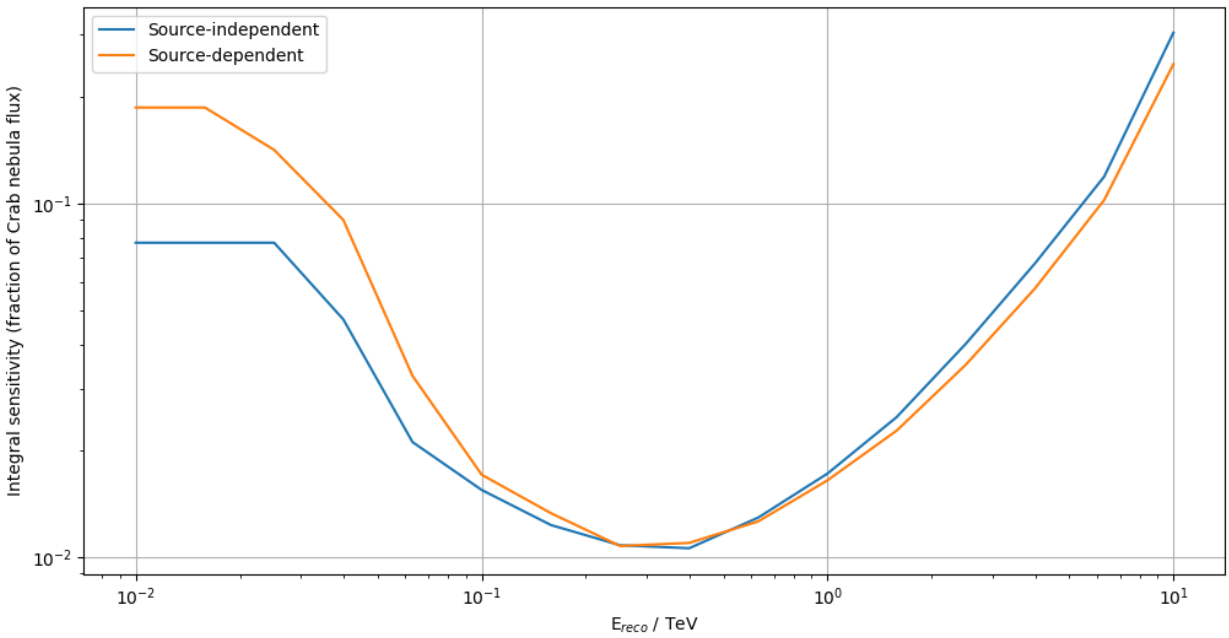
In [195…
```
# Integral numbers of events
fig = plt.figure(figsize=(16, 6))

fig.add_subplot(1, 2, 1)
plt.plot(10**logenergy_bins[:-1], np.nancumsum(num_excess_events[0][::-1])[::-1])
plt.plot(10**logenergy_bins[:-1], np.nancumsum(num_excess_events[1][::-1])[::-1])
plt.xscale('log')
plt.yscale('log')
plt.xlabel('Ereco / TeV')
plt.ylabel('Number of excess events, integrated above Ereco')
plt.xlim(0.01, 30)

fig.add_subplot(1, 2, 2)
plt.plot(10**logenergy_bins[:-1], np.nancumsum(num_off_events[0][::-1])[::-1])
plt.plot(10**logenergy_bins[:-1], np.nancumsum(num_off_events[1][::-1])[::-1])
plt.xscale('log')
plt.yscale('log')
plt.xlabel('Ereco / TeV')
plt.ylabel('Number of OFF events, integrated above Ereco')
plt.xlim(0.01, 30)
plt.show()
```



In [196…
```
#
# Calculate integral sensitivity, using the same optimized cuts in each bin of Ereco.
# NOTE: at low E differential sensitivities may be similar, yet the integral ones differ, because of
# the different rates of excess and background (resulting in different conditions determining the
# flux sensitivity value). Cuts are not re-optimized!
# NOTE anyway that only the best integral sensitivity (in terms of Crab fraction) is relevant!
#

integral_sensitivity = np.zeros_like(sensitivity)
integral_sensitivity[:] = np.nan

for analysis_type in range(2):

    for iebin in range(len(logenergy_bins)-1):
        total_excess = np.nansum(num_excess_events[analysis_type][iebin:])
        total_off = np.nansum(num_off_events[analysis_type][iebin:])

        flux = calc_flux_3conditions(np.array([total_excess]), np.array([total_off]),
                                     min_signi, min_exc, min_off_events, alpha,
                                     target_obs_time, livetimes[analysis_type],
                                     min_n_gammas, min_backg_percent)

        integral_sensitivity[analysis_type][iebin] = flux[0]
```

```
/tmp/ipykernel_10179/3511943101.py:6: RuntimeWarning: divide by zero encountered in true_divide
  return (N_excess/(cumul_excess*time_factor))
/tmp/ipykernel_10179/3629251318.py:7: RuntimeWarning: invalid value encountered in true_divide
  return percent/100*cumul_off/cumul_excess
```

In [197…
```
plt.figure(figsize=(12,6))

plt.plot(10**logenergy_bins[:-1], integral_sensitivity[0], label='Source-independent')
plt.plot(10**logenergy_bins[:-1], integral_sensitivity[1], label='Source-dependent')

plt.xscale('log')
plt.yscale('log')
#plt.ylim(0.01, 0.5)
plt.xlabel('E$_{reco}$ / TeV')
plt.ylabel('Integral sensitivity (fraction of Crab nebula flux)')
plt.grid()
plt.legend()
plt.show()

print(f'Best integral sensitivity, source-indep: {np.nanmin(integral_sensitivity[0]):.4f} C.U.')
print(f'Best integral sensitivity, source-dep: {np.nanmin(integral_sensitivity[1]):.4f} C.U.')
```




```
Best integral sensitivity, source-indep: 0.0106 C.U.
Best integral sensitivity, source-dep: 0.0108 C.U.
```

In [198…

```
np.set_printoptions(precision=4)
print("Energies (TeV):", np.array(reco_energy[0]))

print()
for analysis_type in range(2):
    if analysis_type == 0:
        print("Values for source-independent analysis:")
    else:
        print("Values for source-dependent analysis:")
    print("Sensitivity:", sensitivity[analysis_type])
    print()
    print("Nexcess:", num_excess_events[analysis_type])
    print()
    print("Noff:", num_off_events[analysis_type])

    print()
    print("Gammaness cut (even):", np.where(np.isnan(sensitivity[analysis_type]), np.nan,
                                            gammaness_cut[analysis_type][0]))
    print()
    print("Gammaness cut (odd):", np.where(np.isnan(sensitivity[analysis_type]), np.nan,
                                           gammaness_cut[analysis_type][1]))
    print()
    print("Angular cut (even):", np.where(np.isnan(sensitivity[analysis_type]), np.nan,
                                          angular_cut[analysis_type][0]))
    print()
    print("Angular cut (odd):", np.where(np.isnan(sensitivity[analysis_type]), np.nan,
                                         angular_cut[analysis_type][1]))

    print('\n'*3)
```

```
Energies (TeV): [1.2589e-02 1.9953e-02 3.1623e-02 5.0119e-02 7.9433e-02 1.2589e-01
 1.9953e-01 3.1623e-01 5.0119e-01 7.9433e-01 1.2589e+00 1.9953e+00
 3.1623e+00 5.0119e+00 7.9433e+00 1.2589e+01 1.9953e+01 3.1623e+01
 5.0119e+01 7.9433e+01 1.2589e+02]

Values for source-independent analysis:
Sensitivity: [   nan    nan 0.9074 0.9128 0.3351 0.0635 0.0336 0.0223 0.0194 0.0192
 0.0226 0.0282 0.0458 0.0804 0.123  0.3039    nan    nan    nan    nan
    nan]

Nexcess: [      nan       nan  890.      704.      876.     4254.     5357.
 5197.667  2693.     1677.3333 1342.3333  970.3334  584.6667  331.
  205.       86.           nan       nan       nan       nan       nan]

Noff: [       nan        nan 16151.     12852.      4087.      3457.
  1513.       612.3333   114.        38.6667    33.6667    26.6667
    25.3333    25.        22.        24.            nan        nan
        nan        nan        nan]

Gammaness cut (even): [   nan    nan 0.611  0.708  0.8255 0.8865 0.9015 0.927  0.9665 0.962
 0.9545 0.9715 0.954  0.938  0.8945 0.9355    nan    nan    nan    nan
    nan]

Gammaness cut (odd): [   nan    nan 0.617  0.7075 0.9095 0.8795 0.9115 0.92   0.9435 0.9635
 0.97   0.962  0.96   0.9485 0.919  0.832     nan    nan    nan    nan
    nan]

Angular cut (even): [   nan    nan 0.0209 0.0304 0.0347 0.0235 0.0195 0.0211 0.0103 0.01
 0.0102 0.0232 0.0232 0.0313 0.0377 0.1646    nan    nan    nan    nan
    nan]

Angular cut (odd): [   nan    nan 0.0201 0.0277 0.0271 0.0199 0.0138 0.0138 0.0126 0.0057
 0.0139 0.0122 0.03   0.036  0.0458 0.0408    nan    nan    nan    nan
    nan]



Values for source-dependent analysis:
Sensitivity: [   nan 1.3274 1.1781 0.9201 0.2506 0.0688 0.0335 0.0267 0.0223 0.0205
 0.0216 0.0291 0.0409 0.0687 0.1082 0.2476    nan    nan    nan    nan
    nan]

Nexcess: [  nan 1168. 1412. 1819. 1746. 4766. 6746. 3687. 2957. 2371. 1370. 1216.
  715.  442.  250.  111.   nan   nan   nan   nan   nan]

Noff: [       nan 3.1007e+04 3.3270e+04 3.3475e+04 8.7500e+03 5.1160e+03
 2.3950e+03 4.3400e+02 1.8800e+02 9.7000e+01 3.2000e+01 4.8000e+01
 3.1000e+01 3.4000e+01 2.6000e+01 2.7000e+01        nan        nan
        nan        nan        nan]

Gammaness cut (even): [   nan 0.631  0.727  0.816  0.87   0.91   0.9555 0.968  0.955  0.9685
 0.974  0.9455 0.932  0.887  0.8475 0.844     nan    nan    nan    nan
    nan]

Gammaness cut (odd): [   nan 0.631  0.7595 0.7145 0.887  0.9365 0.928  0.9815 0.989  0.982
 0.973  0.965  0.9335 0.9325 0.85   0.7975    nan    nan    nan    nan
    nan]

Angular cut (even): [  nan  5.52 12.56 10.18  5.48  7.84  6.22  6.64  4.42  3.96  3.3   2.86
  3.68  4.7   4.54  7.86   nan   nan   nan   nan   nan]

Angular cut (odd): [  nan  5.5  13.5   5.4   9.6   6.04  6.44  4.42  3.48  4.98  2.64  5.34
  3.06  6.36  5.2   6.22   nan   nan   nan   nan   nan]
```

In [199…
```
[angular_cut[0][1], angular_cut[1][1]]
```

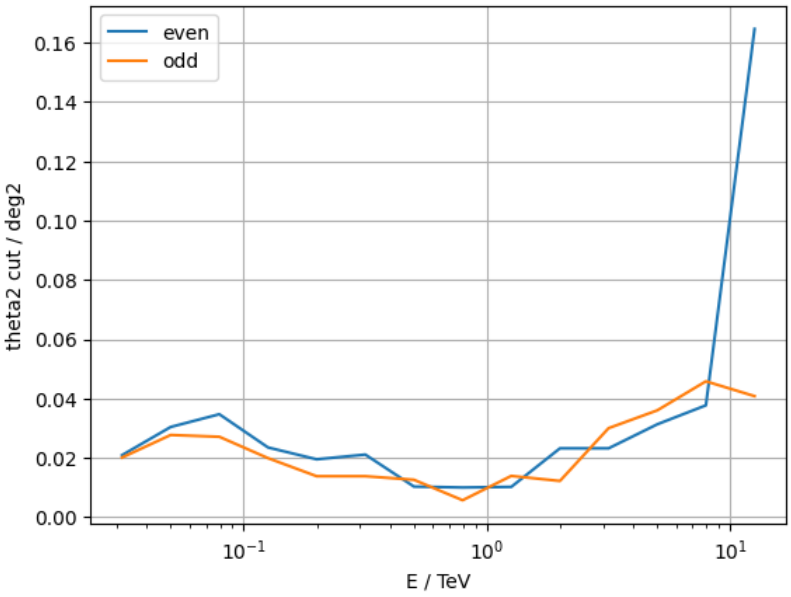
Out[199]:
```
[array([   nan,    nan, 0.0201, 0.0277, 0.0271, 0.0199, 0.0138, 0.0138,
        0.0126, 0.0057, 0.0139, 0.0122, 0.03  , 0.036 , 0.0458, 0.0408,
           nan,    nan,    nan,    nan,    nan]),
 array([  nan,  5.5 , 13.5 ,  5.4 ,  9.6 ,  6.04,  6.44,  4.42,  3.48,
         4.98,  2.64,  5.34,  3.06,  6.36,  5.2 ,  6.22,   nan,   nan,
          nan,   nan,   nan])]
```

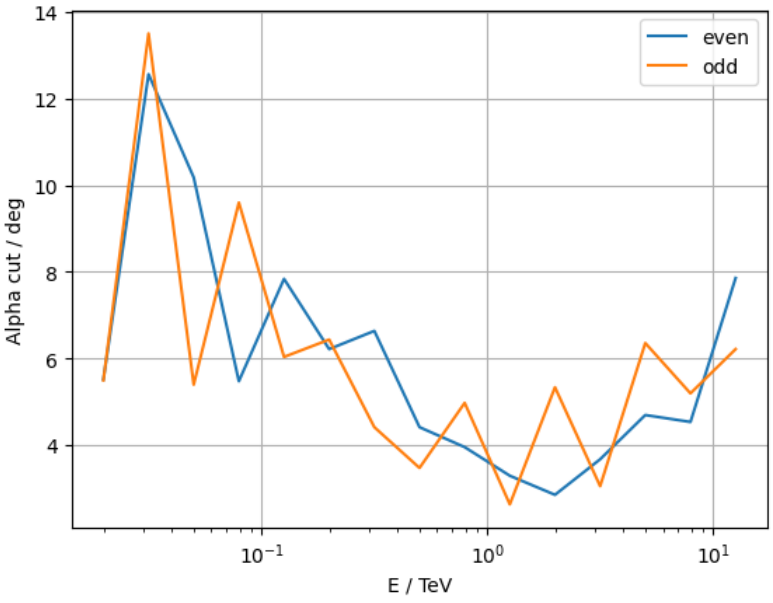
In [200…
```
[gammaness_cut[0][1], gammaness_cut[1][1]]
```

Out[200]:
```
[array([   nan,    nan, 0.617 , 0.7075, 0.9095, 0.8795, 0.9115, 0.92  ,
        0.9435, 0.9635, 0.97  , 0.962 , 0.96  , 0.9485, 0.919 , 0.832 ,
           nan,    nan,    nan,    nan,    nan]),
 array([   nan, 0.631 , 0.7595, 0.7145, 0.887 , 0.9365, 0.928 , 0.9815,
        0.989 , 0.982 , 0.973 , 0.965 , 0.9335, 0.9325, 0.85  , 0.7975,
           nan,    nan,    nan,    nan,    nan])]
```

In [201…

```
plt.plot(reco_energy[0], angular_cut[0][0], label='even')
plt.plot(reco_energy[1], angular_cut[0][1], label='odd')
plt.xscale('log')
plt.grid()
plt.ylabel('theta2 cut / deg2')
plt.xlabel('E / TeV')
plt.legend()
plt.show()

plt.plot(reco_energy[0], angular_cut[1][0], label='even')
plt.plot(reco_energy[1], angular_cut[1][1], label='odd')
plt.xscale('log')
plt.grid()
plt.ylabel('Alpha cut / deg')
plt.xlabel('E / TeV')
plt.legend()
plt.show()
```

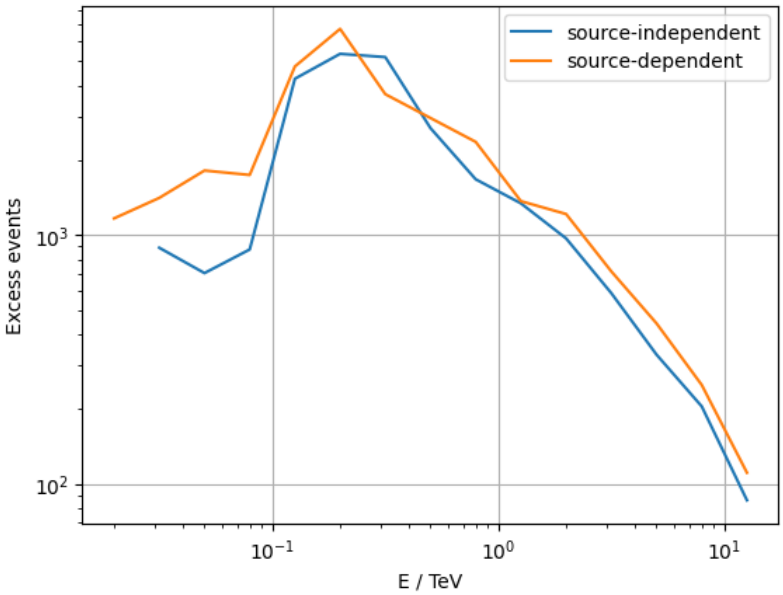


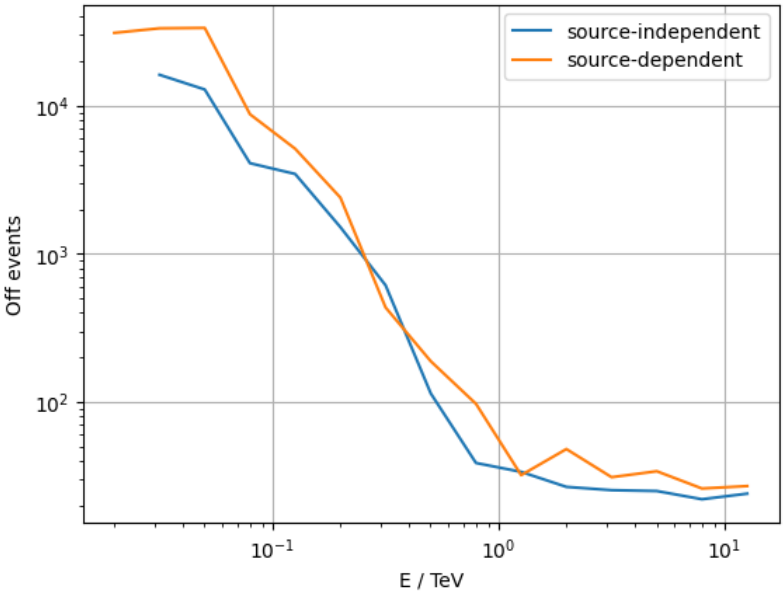


In [202…

```
plt.plot(reco_energy[0], num_excess_events[0], label='source-independent')
plt.plot(reco_energy[1], num_excess_events[1], label='source-dependent')
plt.xscale('log')
plt.yscale('log')
plt.grid()
plt.ylabel('Excess events')
plt.xlabel('E / TeV')
plt.legend()
plt.show()

plt.plot(reco_energy[0], num_off_events[0], label='source-independent')
plt.plot(reco_energy[1], num_off_events[1], label='source-dependent')
plt.xscale('log')
plt.yscale('log')
```

```
plt.grid()
plt.ylabel('Off events')
plt.xlabel('E / TeV')
plt.legend()
plt.show()
```





In [203…
```
num_off_events[0]
```

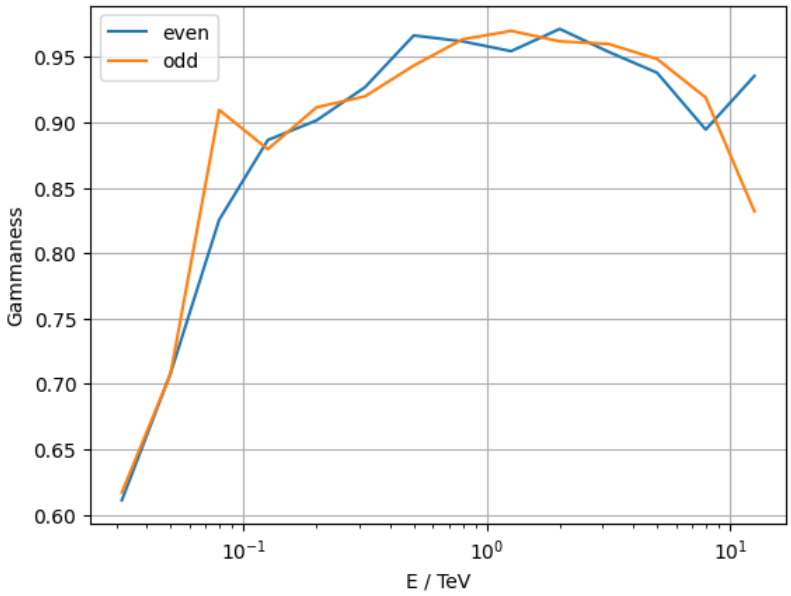
Out[203]:
```
array([       nan,        nan, 16151.    , 12852.    ,  4087.    ,
        3457.    ,  1513.    ,   612.3333,   114.    ,    38.6667,
          33.6667,    26.6667,    25.3333,    25.    ,    22.    ,
          24.    ,        nan,        nan,        nan,        nan,
              nan])
```

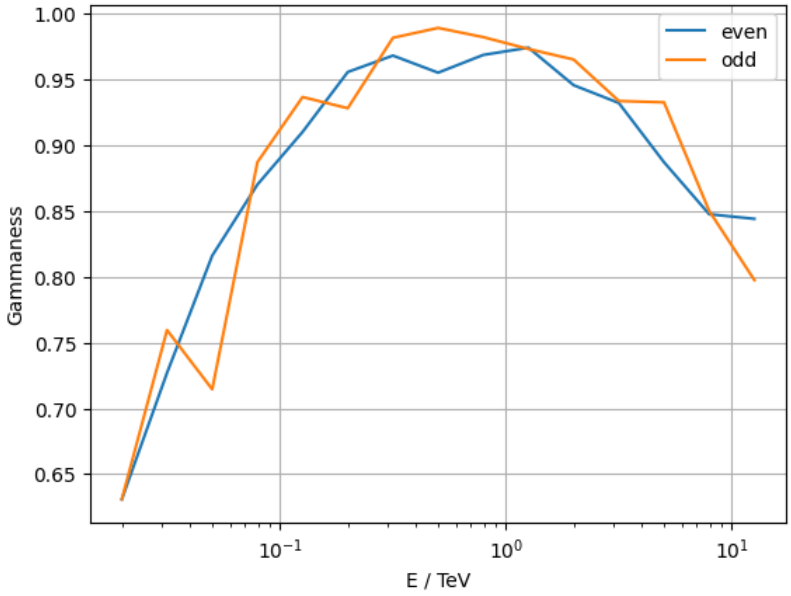
In [204…
```
num_excess_events[0]
```

Out[204]:
```
array([      nan,        nan,  890.    ,  704.    ,  876.    , 4254.    ,
       5357.    , 5197.667 , 2693.    , 1677.3333, 1342.3333,  970.3334,
        584.6667,  331.    ,  205.    ,   86.    ,        nan,       nan,
             nan,       nan,       nan])
```

In [205…
```
plt.plot(reco_energy[0], gammaness_cut[0][0], label='even')
plt.plot(reco_energy[0], gammaness_cut[0][1], label='odd')
plt.xscale('log')
plt.grid()
plt.ylabel('Gammaness')
plt.xlabel('E / TeV')
plt.legend()
plt.show()

plt.plot(reco_energy[1], gammaness_cut[1][0], label='even')
plt.plot(reco_energy[1], gammaness_cut[1][1], label='odd')
plt.xscale('log')
```

```
plt.grid()
plt.ylabel('Gammaness')
plt.xlabel('E / TeV')
#plt.ylim(0.9, 1.05)
plt.legend()
plt.show()
```



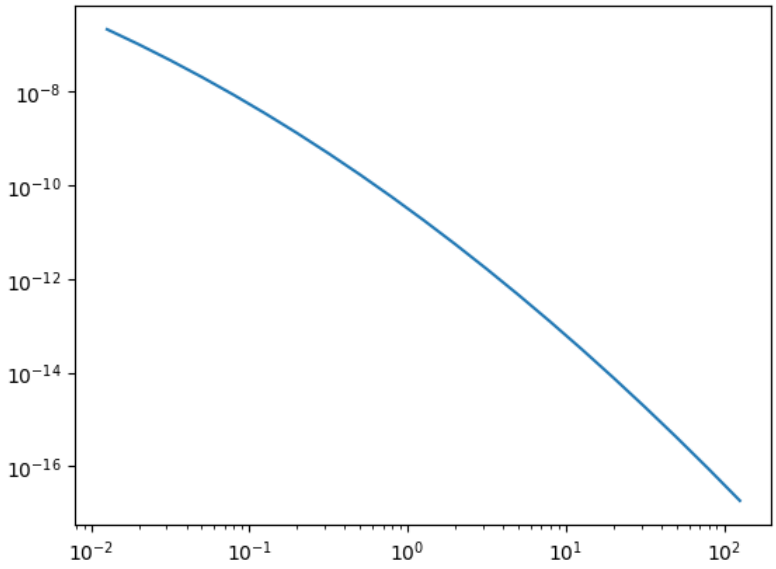


In [206…
```
import scipy.integrate as integrate
def dfde(x):
    return CRAB_MAGIC_JHEAP2015(x*u.TeV).to_value(1/(u.TeV*u.s*u.cm**2))
```

In [207…
```
dfde(1.)
```

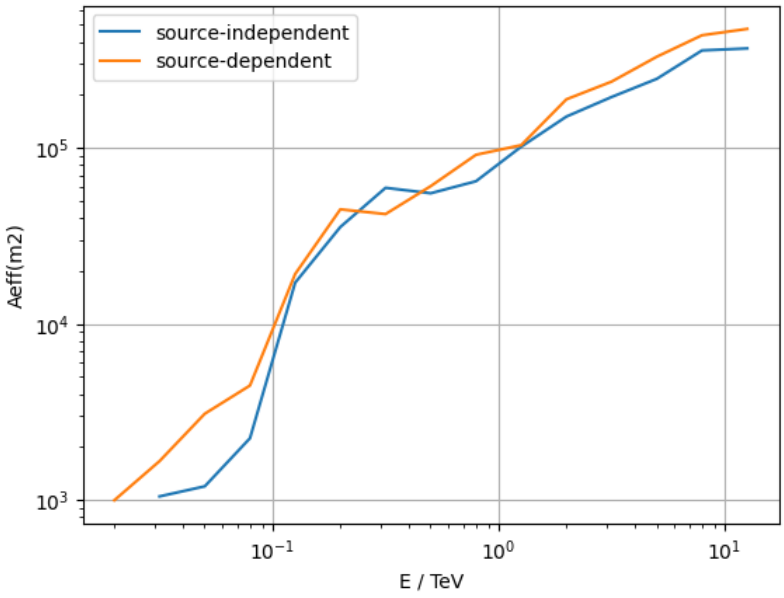
Out[207]: 3.23e-11

In [208…
```
etev = reco_energy[0] * u.TeV
plt.plot(etev, CRAB_MAGIC_JHEAP2015(etev))
plt.xscale('log')
plt.yscale('log')
plt.show()

logenergy_bins[iebin]+logenergy_bins[iebin+1]

rate_per_s_m2 = 1e4*np.array([integrate.quad(dfde, 10**a, 10**b)[0]
                              for a, b in zip(logenergy_bins[:-1], logenergy_bins[1:])])
rate_per_s_m2
```



Out[208]:
```
array([1.2573e-05, 9.5211e-06, 6.8993e-06, 4.7840e-06, 3.1743e-06,
       2.0154e-06, 1.2245e-06, 7.1192e-07, 3.9606e-07, 2.1085e-07,
       1.0741e-07, 5.2358e-08, 2.4423e-08, 1.0901e-08, 4.6561e-09,
       1.9030e-09, 7.4426e-10, 2.7853e-10, 9.9745e-11, 3.4180e-11,
       1.1208e-11])
```

In [209…
```
plt.plot(reco_energy[0], num_excess_events[0]/livetimes[0]/rate_per_s_m2, label='source-independent')
plt.plot(reco_energy[1], num_excess_events[1]/livetimes[1]/rate_per_s_m2, label='source-dependent')
plt.xscale('log')
plt.yscale('log')
plt.grid()
plt.ylabel('Aeff(m2)')
plt.xlabel('E / TeV')
plt.legend()
plt.show()
```



In [ ]: