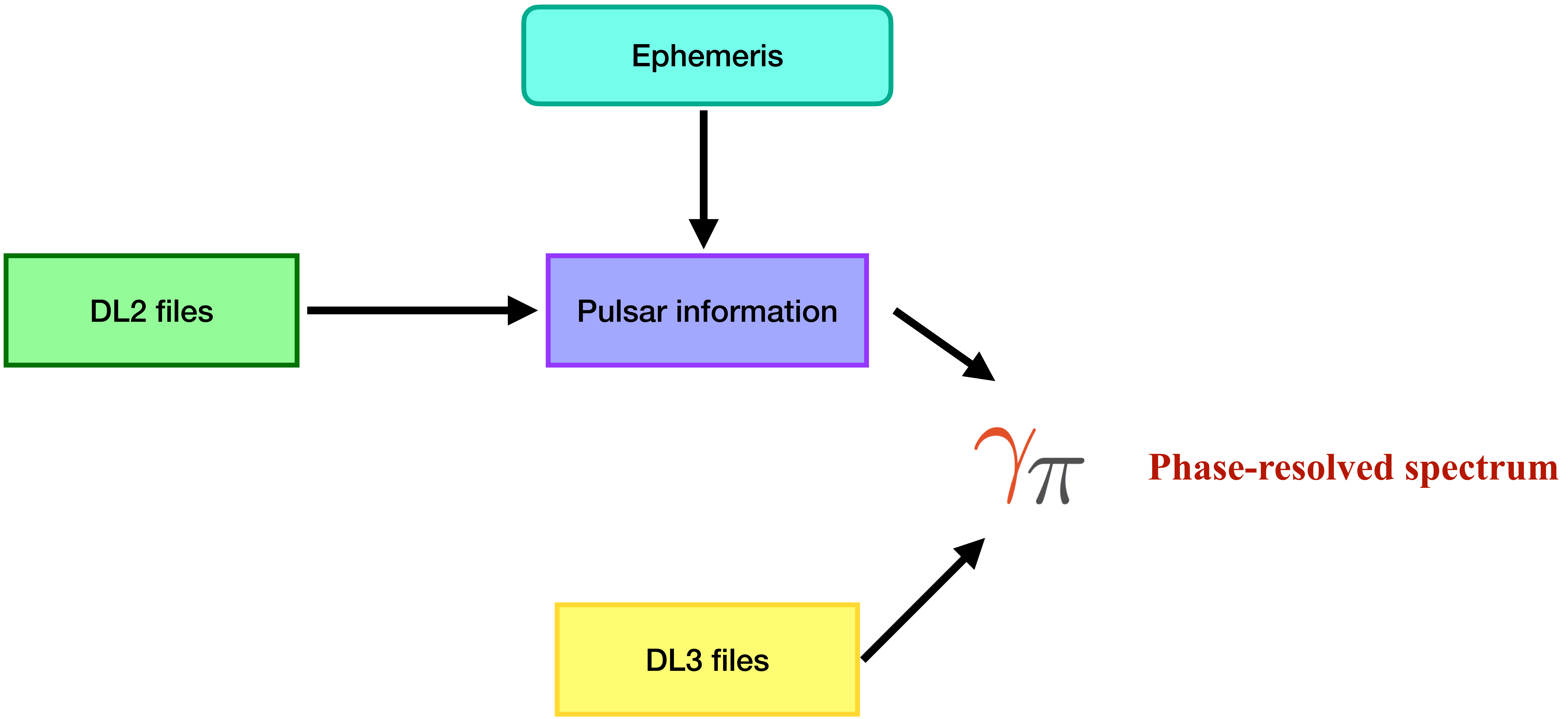
LST-1 pulsar analysis and gammapy

Alvaro Mas Aguilar 30/06/2022
Universidad Complutense de Madrid (alvmas@ucm.es)

cherenkov
telescope
array

Gammapy and Pulsar analysis with LST: current scheme



1. Phase computation and analysis

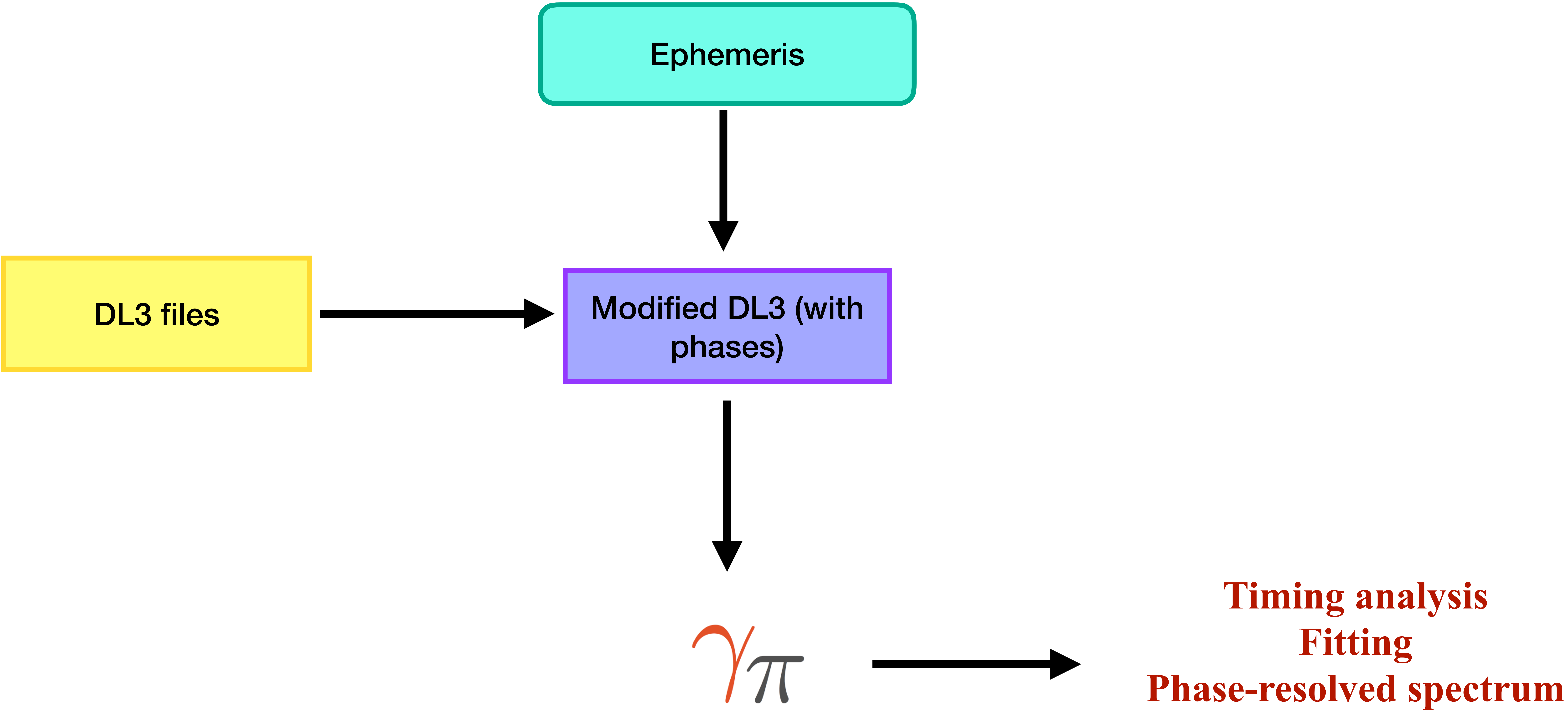
Right now, the phase-folding is done at DL2 level with PINT-pulsar but it could be a good idea to do it at DL3 level for the future.

1. **First problem**: PINT-pulsar works slow for large amount of data. Almost solved.

2. **Second Problem:** DL3 format need to be changed to include the phase information. Is this possible? Could be this a problem for gammapy?

Conclusion: We want to apply the phase-folding tool to the DL3 files and save the result into the same file just before using gammapy. Or maybe there could be some tool to add extra information to the EventLists in gammapy.

Gammapy and Pulsar analysis with LST: desired scheme



2. Pulsar analysis with gammapy

Right now in gammapy there is a tutorial (*https://docs.gammapy.org/0.20/tutorials/analysis/time/pulsar_analysis.html*) for pulsar analysis that includes:

- **Phaseogram:**
    - Could also be interested to implement some tools for the computation of statistics (H-test, Chi2 test, etc.) in gammapy.
    - It would be also very useful to include some fitting tools for the Phaseogram to study the morphology of the peaks.
- **Phase-resolved spectrum and phase-resolved map**
    - I have followed the tutorial for the phase-resolved spectrum and seems to be working good for the case.

Need for both again the DL3 files with the PHASE information!! In LST right now we merge the information of the DL2 and the DL3 to add a new column with the pulsar information.

2.1 Timing analysis: phaseogram and pulse searching

I have designed some tools in Python that allows to calculate the Phaseogram and the statistics on a very user-friendly way. Right now, to use it we need as inputs LST-1 DL2 files or FITS files but could be generalized to read DL3 files.

Main tasks:

1. Reads different type of files, extract and filter the information for the Pulsar analysis.

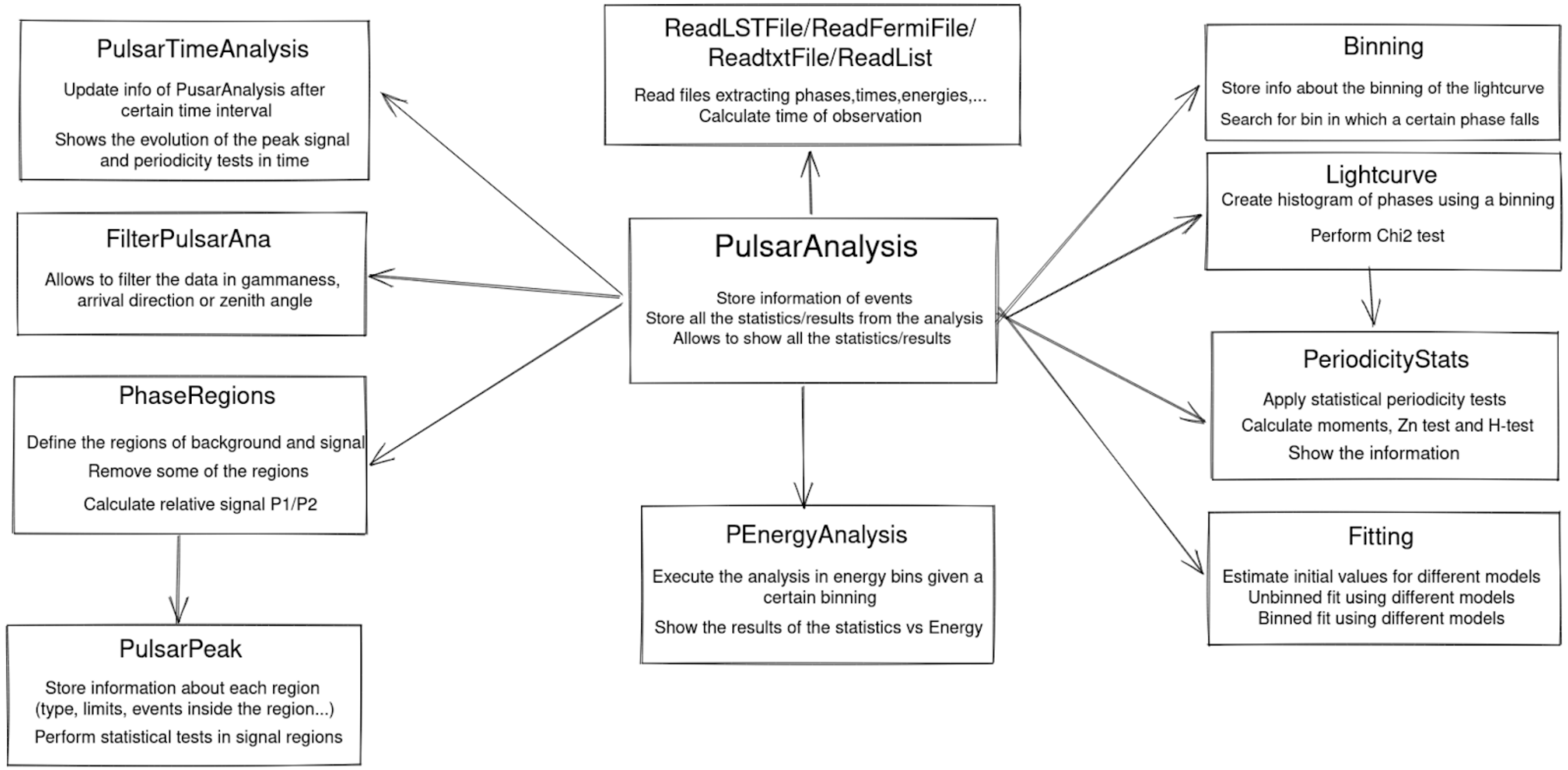
2. Creates the phaseogram using a certain binning.

3. Performs the different statistical tests.

4. Shows the evolution of the signal in time and energy.

5. Fits the data to different models

https://github.com/alvmas/PulsarTimingAnalysis/ptiming_ana/phaseogram



2.1 Timing analysis: phaseogram and pulse searching

https://github.com/alvmas/PulsarTimingAnalysis/ptiming_ana/phaseogram

Example of use:

In [2]:
```
h=PulsarAnalysis()
```

Creation of the main object

Setting the input file. If we use a LST1 DL2-file:

In [17]:
```
h.setLSTInputFile('psample_nov2020march2021.h5',src_dep=False)
```

Defining the input file. **The idea is to be able to give as an input DL3 files**

We need to set the phase limits of the background and the peaks (signal region). We can define one,two or three signal regions.

In [18]:
```
h.setBackgroundLimits([0.52,0.87])
h.setPeaklimits(P1_limits=[0,0.026,0.983,1],P2_limits=[0.377,0.422],P3_limits=None)
```

Defining background and signal phase regions.

We can also set the binning that we are going to use for the construction of the lightcurve:

In [19]:
```
h.setBinning(50,xmin=0,xmax=1)
```

Set the binning. If not set, default binning is N=50

Additional cuts in gammaness or arrival direction can be set:

- Gammaness_cut
- Alpha_cut
- Theta2_cut
- Zd_cut

In [20]:
```
h.setParamCuts(gammaness_cut=0.5,alpha_cut=12,zd_cut=35)
```

Set fixed cuts to filter

We can also define the time interval (in seconds) in which we update the statistics. For instance, if we set tint=3600, the statistics will be calculated every hour of accumulated time of observation.

In [22]:
```
h.setTimeInterval(tint=3600*3)
```

Set the interval of time to update the statisti

2.1 Timing analysis: phaseogram and pulse searching

https://github.com/alvmas/PulsarTimingAnalysis/ptiming_ana/phaseogram

Example of use:

If we want to do the fitting to the peaks to a model we can define it as following:

Available models:

- Single gaussian ('gaussian')
- Double gausssian ('dgaussian')
- Assymetric doble gaussian ('asym_dgaussian')
- Double lorentzian ('lorentzian')

In [8]:
```
h.setFittingParams(model='dgaussian',binned=False)
```

Set the parameters for the fit

We can also do the statistics in certain energy bins. To set this binning:

In [38]:
```
h.setEnergybinning([0.03,0.06,0.15])
```

Set the energy binning for energy-dependent analysis

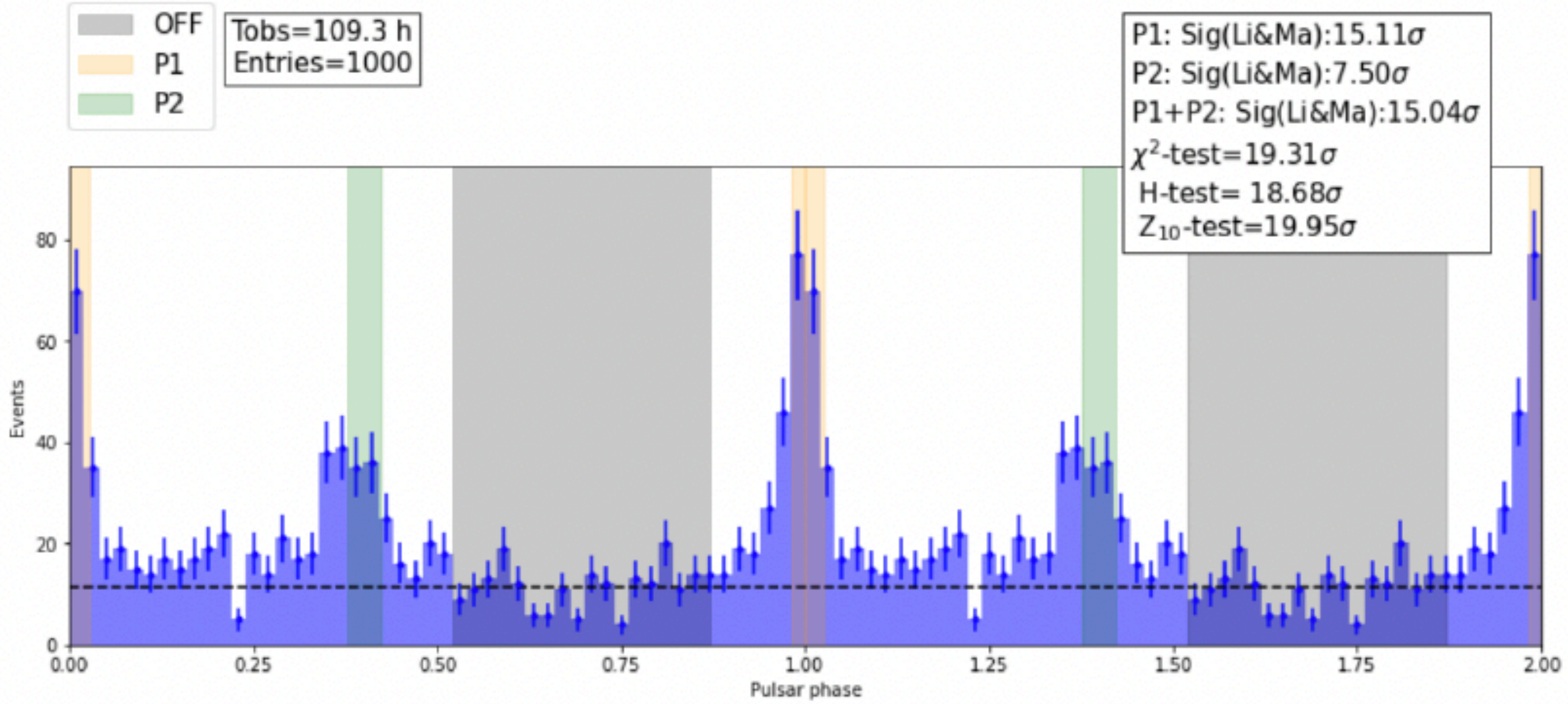
In [15]:
```
h.run()
```

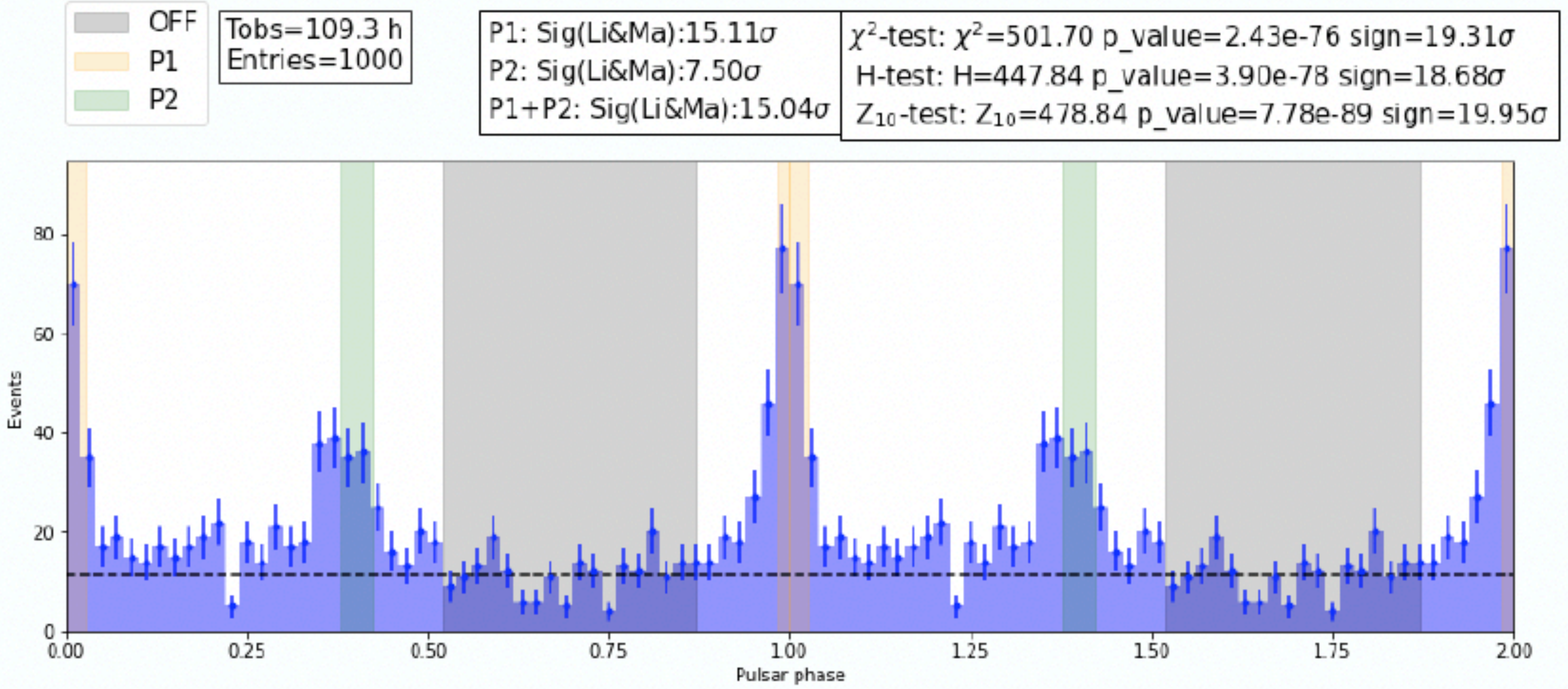
Finally, execute the analysis

2.1 Timing analysis: phaseogram and pulse searching

https://github.com/alvmas/PulsarTimingAnalysis/ptiming_ana/phaseogram/

[15]:
```
phaseogram=h.draw_phaseogram(phase_limits=[0,2],colorhist='blue')
```





[17]:
```
results=h.show_Presults()
```

Li&Ma excess in each peak

```
RESULTS FOR THE PEAK STATISTICS:

                      P1          P2        P1+P2
Significance   15.113320    7.503019    15.044473
Nex           123.674286   51.542857   175.217143
Nex_error      12.287741    8.959523    15.411581
Number        148.000000   77.000000   225.000000
noff           24.325714   25.457143    49.782857
sign_t_ratio    1.445504    0.717621     1.438919
s/n ratio      25.075327   10.215595    24.833429

P1/P2 ratio=2.40+/-0.48
```

Periodicity tests to search pulsations

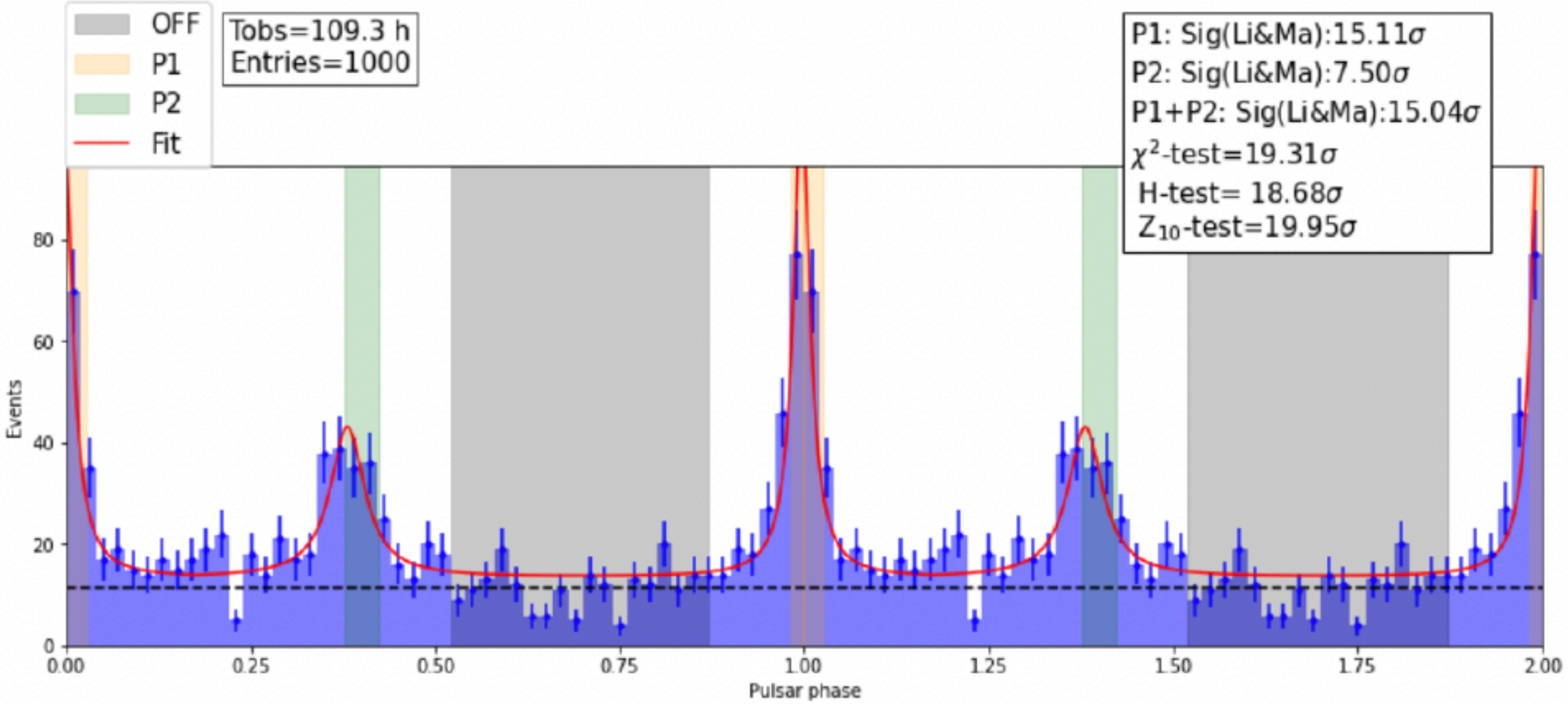
```
RESULTS FOR THE PERIODICITY SEARCH:

                   Chi_square_test      Zn_test        H_test
Statistic             5.017000e+02  4.788381e+02  4.478429e+02
p-value               2.430223e-76  7.777830e-89  3.896752e-78
Number of $\sigma$    1.930516e+01  1.994814e+01  1.867579e+01
```

2.1 Timing analysis: phaseogram and pulse searching

https://github.com/alvmas/PulsarTimingAnalysis/ptiming_ana/phaseogram/

Fitting:

```
In [20]: phaseogram=h.draw_phaseogram(phase_limits=[0,2],colorhist='blue',fit=True)
```



Fermi-LAT data sample for this example.

```
In [21]: h.fit_model
Out[21]: 'lorentzian'

In [22]: h.binned
Out[22]: False

In [23]: res=h.show_fit_results()
```

Unbinned

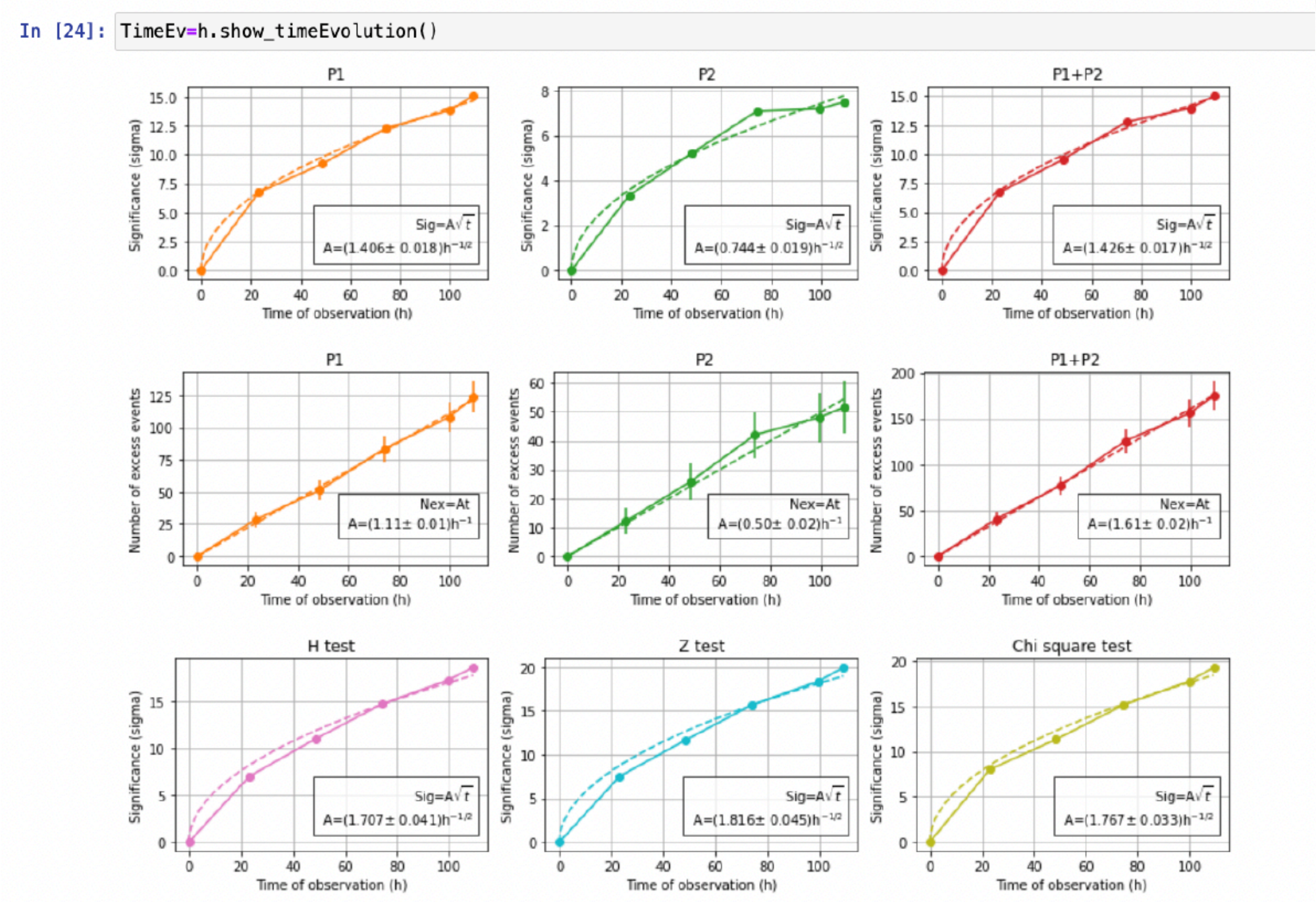
	Name	Value	Error
0	mu_1	0.996350	0.001760
1	gamma_1	0.013603	0.001779
2	mu_2	0.380775	0.006582
3	gamma_2	0.028170	0.004942
4	A	1.283304	4.094031
5	B	0.379836	1.211799
6	C	0.251206	0.802412

A,B,C are the background, P1 and P2 relative intensity over the total signal

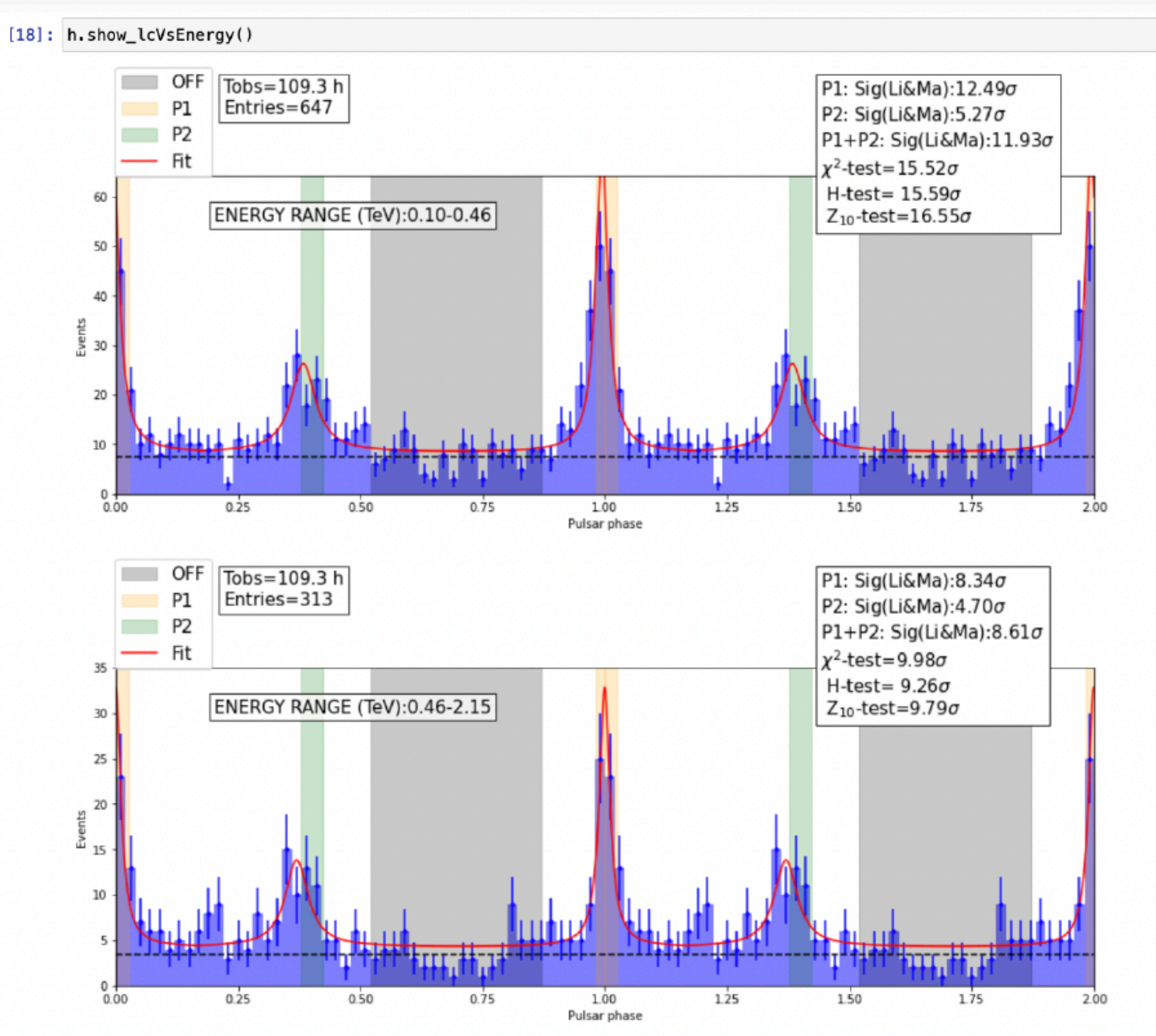
2.1 Timing analysis: phaseogram and pulse searching

https://github.com/alvmas/PulsarTimingAnalysis/ptiming_ana/phaseogram

Statistics vs Time:



Statistics vs Energy:



2.1 Timing analysis: phaseogram and pulse searching

https://github.com/alvmas/PulsarTimingAnalysis/ptiming_ana/phaseogram

Some of this code could be implemented as an specific class in gammapy to make easier the timing analysis of pulsars!

2.2 Phase-resolved spectrum with gammapy

The only difference with respect to the standard analysis is to use the off/on region in phases to compute the background and signal. This is donde by the PhaseBackgroundMaker class.

- It seems to be working fine for me. The overall significance of the stacked sample is similar to the one that I obtain using my tools.

- The phase-resolved spectrum is produced correctly, not showing strange features.

- **But maybe there is some problem I did not notice. Any known problems with this class?**

2.2 Phase-resolved spectrum with gammapy

https://docs.gammapy.org/dev/tutorials/analysis/time/pulsar_analysis.html?highlight=pulsar%20analysis

```
obs_ids=wob_obs_list
observation_list=[]

for j in range(0,5):
    print(obs_ids[j])

    #Read the DL2 files that will be used
    filelist=[]
    dl2_dir='/fefs/aswg/workspace/alvaro.mas/data/real/Crab/src_dependent/v0.9_crab/DL2_pulsar/'
    for x in os.listdir(dl2_dir):
            p_dir = os.path.relpath(dl2_dir)
            p_file = os.path.join(p_dir, x)
            if str(obs_ids[j]) in p_file:
                filelist.append(p_file)

    #Read the DL3-table and extract the times
    table=unique(observations[j].events.select_region(on_region).table,'TIME')
    times=table['TIME']

    df_list=[]
    #Read the Dl2 events and filter by times to obtain the DL3-event phases
    for file in filelist:
        df_i=pd.read_hdf(file,key=dl2_params_lstcam_key)
        df_i=df_i[['trigger_time','pulsar_phase']]
        phases=df_i.pulsar_phase.to_list()

        #Shift the negative phases
        for i in range(0,len(phases)):
            if phases[i]<0:
                phases[i]=phases[i]+1
        df_i['pulsar_phase']=phases
        df_list.append(df_i)

    df = pd.concat(df_list)
    df=df[df['trigger_time'].isin(times)]
    df=df.sort_values('trigger_time')

    #Create the new table with the phases
    table['PHASE']=df.pulsar_phase
    Events=EventList(table=table)

    #Create an observation for object with this new table and set the same IRFs
    obs=Observation(obs_id=observations[j].obs_id,obs_info=observations[j].obs_info,events=Events,
                    gti=observations[j].gti,aeff=observations[j].aeff,edisp= observations[j].edisp)

    observation_list.append(obs)
print('Finished reading and adding phases')
```

Only need the 'PHASE' column in the DL3 file. Right now, I join the DL3 EventList table with the PHASE information from the DL2 files to create a new Observation object by scratch.

Read the DL3 file

Read phases and times from DL2

Filter the DL2 events to select only those in the DL3 files using the trigger times

Create new EventList table with the information of the phases. The column must be named 'PHASE'

Create a new Observation object with the same data as the original one but with the new EventList object

3 Pulsar observation simulation

I also tried to create a pulsar observation simulation using the IRFs and the predicted model but it seems more tricky. There is no function to fake events taking into account the duty cycle of the pulsar (ON region treated as a region in the Phaseogram).

1. In gammapy the background rate should be estimated using the background model. Now with LST-1 I am using real OFF data to estimate this rate.

2. This OFF rate should be scaled by the duty cycle so the significance of the fake sample should scale as α being α the width of the ON region in phase

I tried to use SpectrumDatasetOnOff class with an acceptance=α but this seems not to work well so I implemented it by hand. Also not consistent with expectations from real observations.

3 Pulsar observation simulation

In [ ]:
```
def simulate_OnOffpulse_model(t,model_simu,irf_file,acceptance,pointing,emin):
    # Define simulation parameters parameters
    livetime = t * u.h

    # Reconstructed and true energy axis
    energy_axis = MapAxis.from_edges(np.logspace(np.log10(emin), np.log10(10),30), unit="TeV", name="energy", interp
    energy_axis_true = MapAxis.from_edges(np.logspace(np.log10(0.003), np.log10(50), 50), unit="TeV", name="energy_t

    #Define On region
    on_region_radius = Angle("0.3 deg")
    center = pointing.directional_offset_by(position_angle=0*u.deg, separation=0.4*u.deg)
    on_region = CircleSkyRegion(center=center, radius=on_region_radius)

    #Define model
    model = SkyModel(spectral_model=model_simu, name="source")

    #Define IRFs
    aeff = EffectiveAreaTable2D.read(irf_file)
    edisp = EnergyDispersion2D.read(irf_file, hdu="ENERGY DISPERSION")
    irf=dict(aeff=aeff,edisp=edisp, psf=None)

    #Create observation
    obs = Observation.create(pointing=pointing, livetime=livetime, irfs=irf)

    # Make the SpectrumDataset
    geom = RegionGeom.create(region=on_region, axes=[energy_axis])
    dataset_empty = SpectrumDataset.create(geom=geom, energy_axis_true=energy_axis_true, name="obs-0")
    maker = SpectrumDatasetMaker(selection=["exposure", "edisp"])
    dataset = maker.run(dataset_empty, obs)


    # Set the model on the dataset, and fake
    dataset.models = model
    dataset.fake(random_state=9)
    on_rate=dataset.npred_signal("source").data.sum()/(t*3600)


    #Define the off rate
    off_rate=0.01 #Taken from real data sample of the Crab pulsar analysis (Range off r
    off_rate_scaled=off_rate*acceptance

    significance=(on_rate/(off_rate_scaled**0.5))

    return(significance*t**0.5)
```

```
#Estimate rate of background.
noff_est=247008/(32*u.h)*t*0.39/acceptance

#Estimate bkg counts in on region and fake
dataset_on_off = SpectrumDatasetOnOff.from_spectrum_dataset(dataset=dataset,acceptance=acceptance,a
dataset_on_off.fake(npred_background=Map(geom=geom,data=noff_est))
dic=dataset_on_off.info_dict()

return(dic['sqrt_ts'])
'''
```

Arguments:

t: time of observation

model_simu: model given for the source

irf_file: IRFs file

acceptance: phase width of the ON region in phaseogram

pointing: pointing of the telescope

emin: minimum of reco energy axis