A Python package for gamma-ray astronomy

ERLANGEN CENTRE FOR ASTROPARTICLE PHYSICS

Prospects for an unbinned analysis framework in Gammapy

Tim Unbehaun, Giacomo D'Amico, Julia Djuvsland - Gammapy Coding Sprint – June 2022

Binned vs Unbinned

- Converging results in case of fine bins (finer than the telescope resolution)
  → mostly beneficial in terms of sensitivity for time analysis?
- Possible computational benefits in case of low event numbers (<< pixel number)
- Ideally we don't need to fix a reconstructed binning which might lead to biases in the analysis

Binned likelihood

$$\ln \mathcal{L}(\xi) = \sum_{i=1}^{N} \ln \left[\frac{\nu_i(\xi)^{n_i}}{n_i!} \times \exp(-\nu_i(\xi)) \right]$$

sum over pixels

Poisson probability to find n counts when ν are predicted

Unbinned likelihood

$$\ln \mathcal{L}(\xi) = \sum_{e=1}^{N} \ln \left[\frac{dN_i(\xi)}{d\Omega \times dE_{reco}} \right] - N^{tot}(\xi)$$

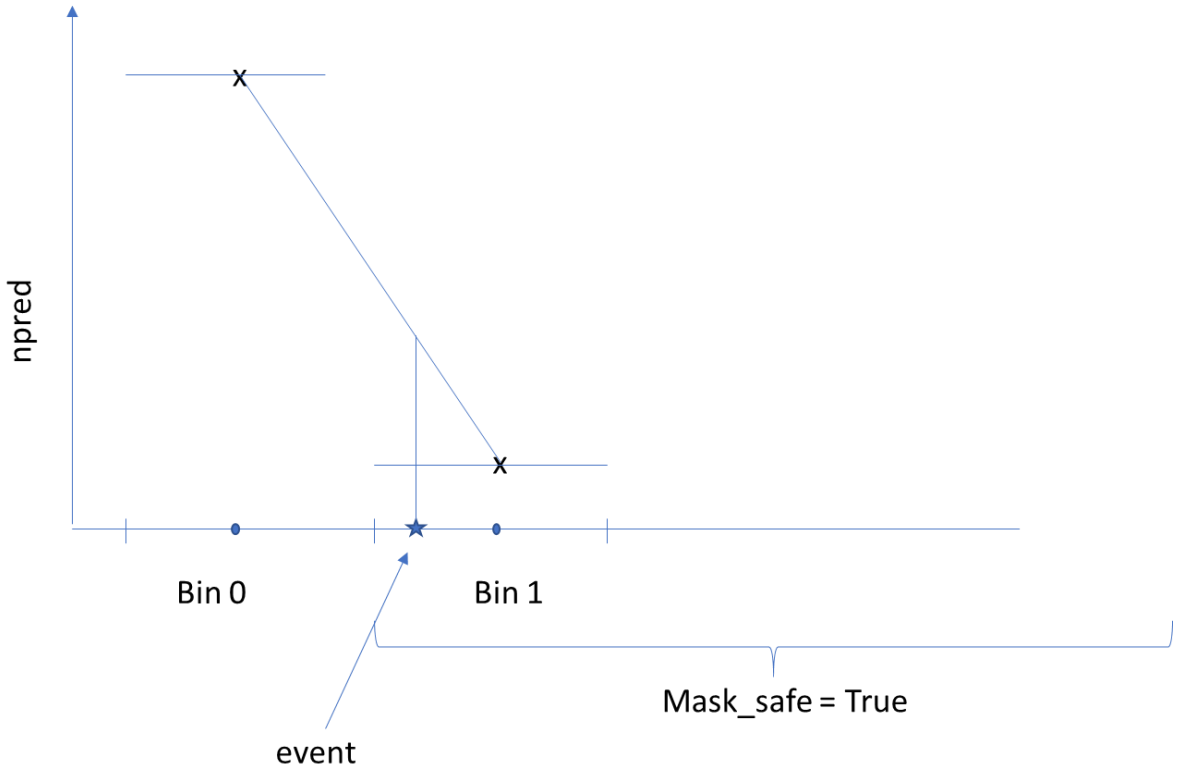
total number of predicted counts

sum over events

Probability of event to belong to model
(≙npred at events coordinates)

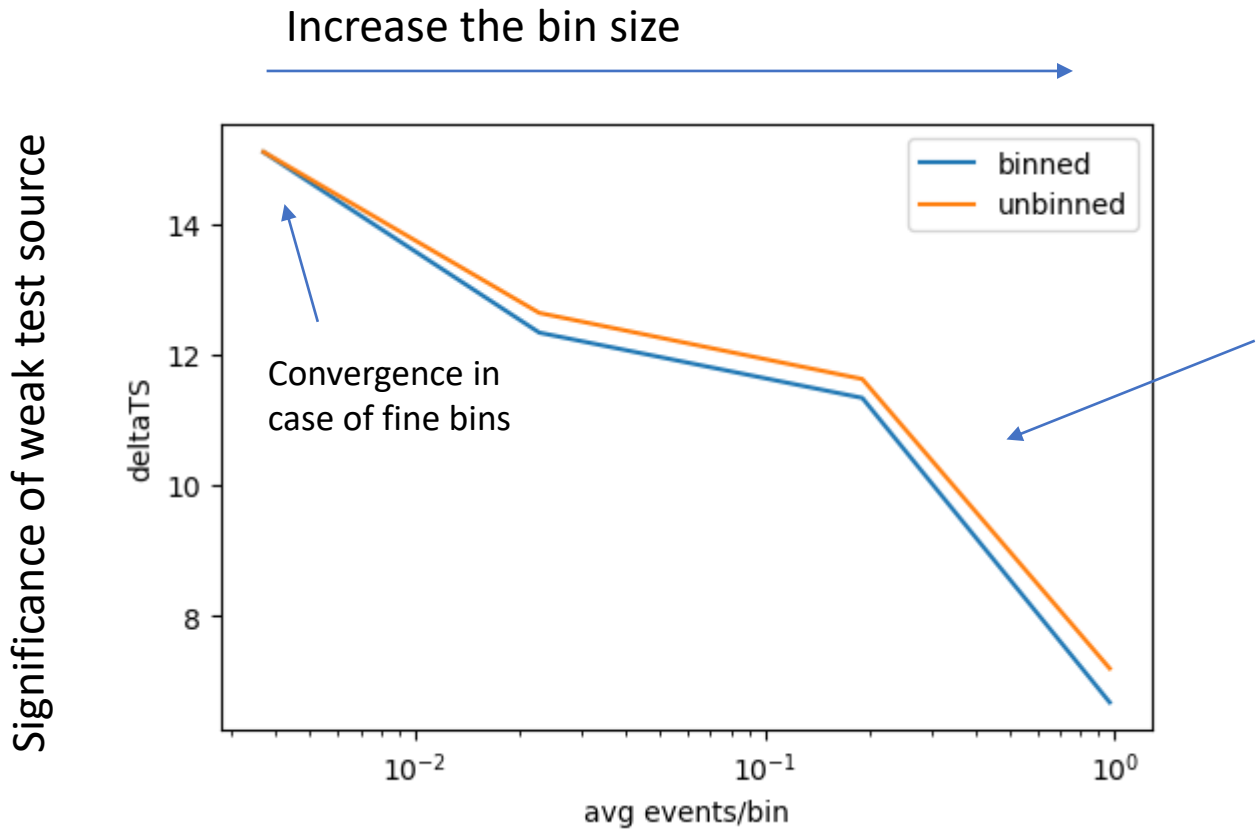
Unbinned – interpolating npred cube

- Use the existing binned datasets with implementation of new unbinned likelihood
- Interpolate the npred cube at the coordinates of the events – sum of npred cube
- Problem with extrapolation in masked regions
  → linear interpolation inside the bin-center-cube
  → nearest interpolation for events beyond the bin centers but still within the mask
- We need two masks and two interpolations → no computational benefit



Unbinned – interpolating npred cube

- Use the existing binned datasets with implementation of new unbinned likelihood
- Interpolate the npred cube at the coordinates of the events – sum of npred cube
  - we still need compute npred for a large number of pixels (>> number of events)



Both methods are dependent on reco binning

Unbinned – at events' coordinates

A Python package for gamma-ray astronomy

ERLANGEN CENTRE FOR ASTROPARTICLE PHYSICS

- Directly compute the npred at the events' coordinates
- No reco binning necessary, no interpolation with all its problems

- Method (firsts thoughts):

1. Set up true geometry with fine enough binning to represent all features of the model
2. Compute static IRF cube for each event (contribution of each pixel of the true geometry)
3. Multiplication of model integrated true geometry and IRF cube will give npred for each event
4. "acceptance" matrix multiplied to model integrated true geometry will give the sum of npred in reco coordinates

- IRF cube of dimensions (N_{events}, energy, lon, lat) might be too large for larger events numbers
- We need a new IRF cube each time the model drifts out of the true geometry for which the cube was computed
- Point sources (with fixed position) only need (N_{events}, energy) IRF cube

Unbinned – at events' coordinates

A Python package for gamma-ray astronomy

ERLANGEN CENTRE FOR ASTROPARTICLE PHYSICS

- Directly compute the npred at the events' coordinates
- No reco binning necessary, no interpolation with all its problems

- Method (firsts thoughts):

1. Set up true geometry with fine enough binning to represent all features of the model
2. Compute static IRF cube for each event (contribution of each pixel of the true geometry)
3. Multiplication of model integrated true geometry and IRF cube will give npred for each event
4. "acceptance" matrix multiplied to model integrated true geometry will give the sum of npred in reco coordinates

- Other option: do the forward folding for each event on the fly
- Evaluating IRFs/ setting up the kernel is slow
- For loop over all events, also slow