DATA ANALYSIS

Data Analysis

- Is there a source there?
- What are its properties? (spectral, spatial, temporal*)
- How to present results
- Combining data from different instruments
- Sources of uncertainty

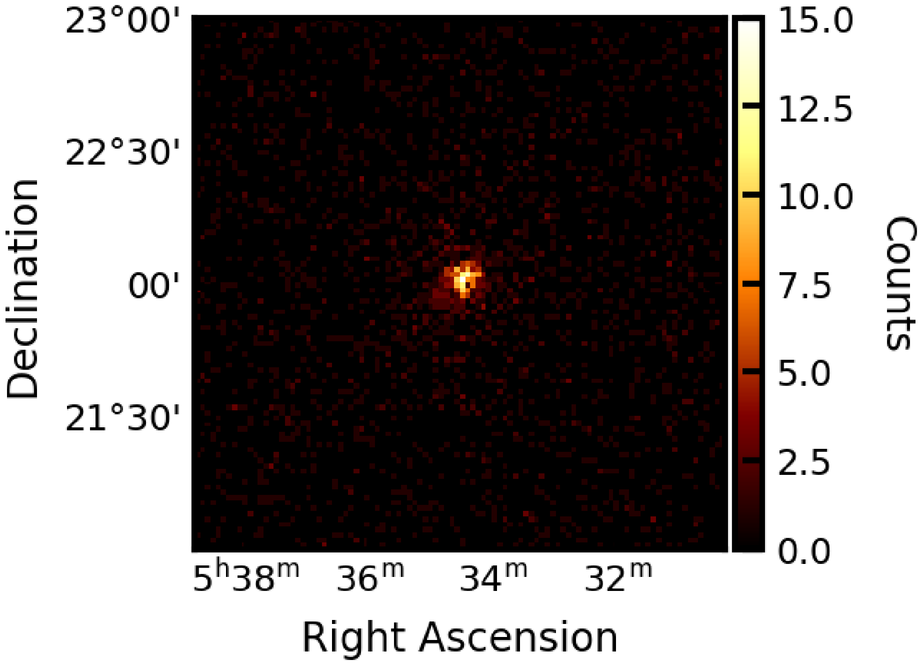
*if we have time

Data Analysis

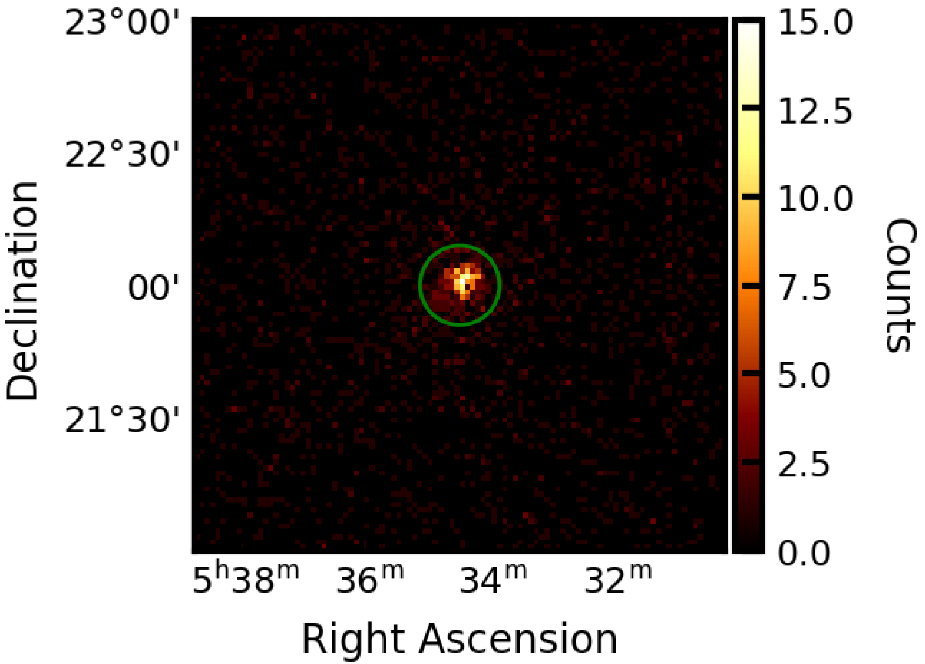
- **Is there a source there?**
- What are its properties? (spectral, spatial, temporal*)
- How to present results
- Combining data from different instruments
- Sources of uncertainty

*if we have time

Simple approach – let's take a cutout



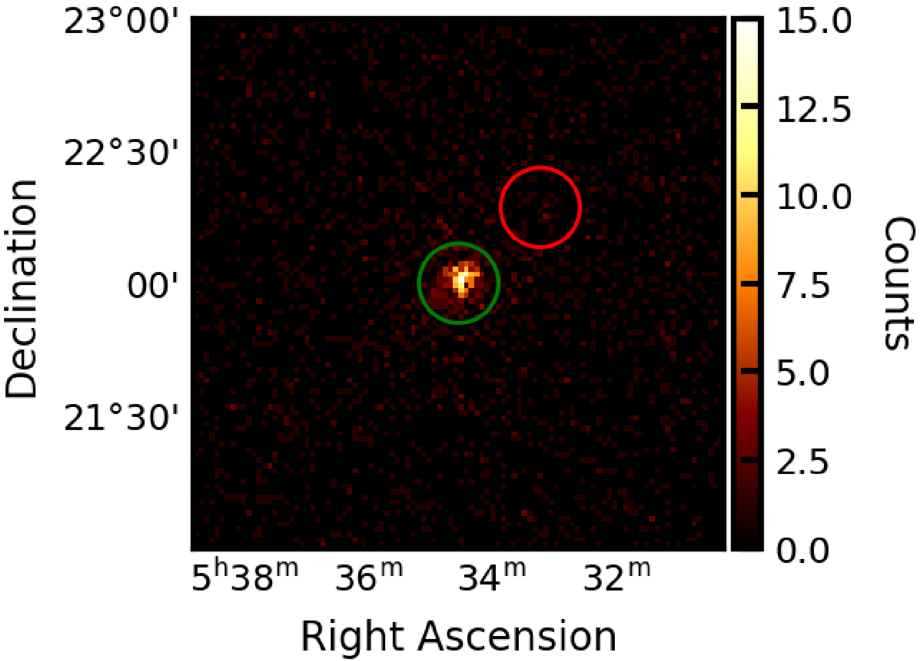
Simple approach – let's take a cutout



```
spec =stacked.to_spectrum_dataset(on_region)
```

- Green: SIGNAL region, 412 counts

Simple approach – let's take a cutout



- Green: SIGNAL region, 412 counts
- Red: BKG region, 40 counts*

*in reality one would use e.g. "reflected regions background", see talk by J. Holder

Brief aside – Hypothesis testing

A model or an excess of counts $(\mathcal{H}_1)$ is tested against a null hypothesis $(\mathcal{H}_0)$ where no source is present.

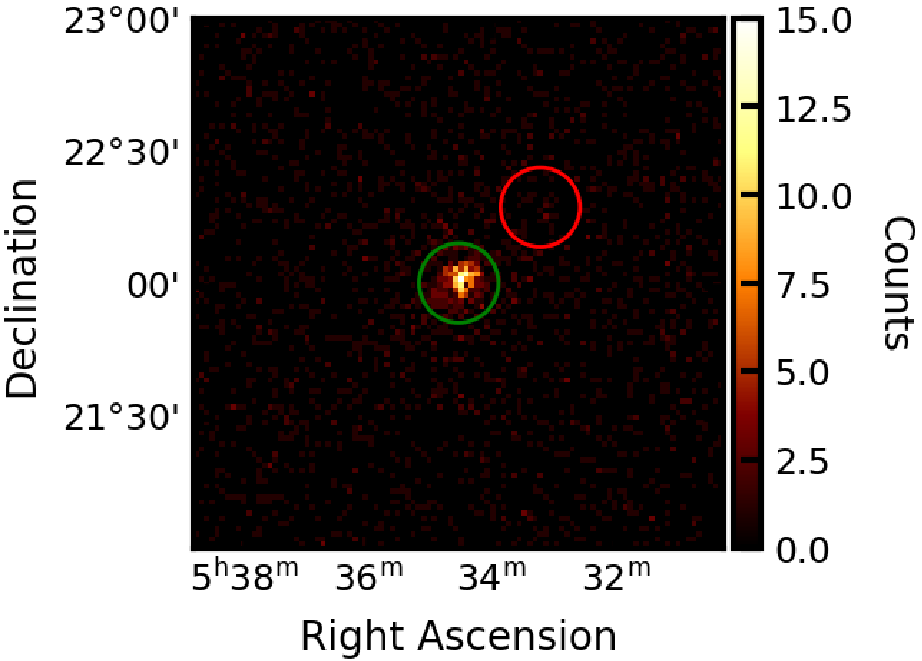
We use the difference in test statistic (TS), **the (Poisson) likelihood ratio**

$$TS = -2\log\left(\frac{\mathcal{L}(\mathcal{H}_0)}{\mathcal{L}(\mathcal{H}_1)}\right)$$

where $\mathcal{L}(\mathcal{H})$ is the maximum likelihood of a hypothesis.

When only 1 deg of freedom: $\sigma = \sqrt{TS}$, and we usually require 5σ

Simple approach – let's take a cutout



- Green: SIGNAL region, 412 counts
- Red: BKG region, 40 counts*
- From Li&MA 1983 (read this paper!)

$$S = \sqrt{-2\ln\lambda} = \sqrt{2}\left\{N_{\mathrm{on}}\ln\left[\frac{1+\alpha}{\alpha}\left(\frac{N_{\mathrm{on}}}{N_{\mathrm{on}}+N_{\mathrm{off}}}\right)\right] + N_{\mathrm{off}}\ln\left[(1+\alpha)\left(\frac{N_{\mathrm{off}}}{N_{\mathrm{on}}+N_{\mathrm{off}}}\right)\right]\right\}^{1/2} .$$

- Significance: ~35σ

*in reality one would use e.g. "reflected regions background", see talk by J. Holder

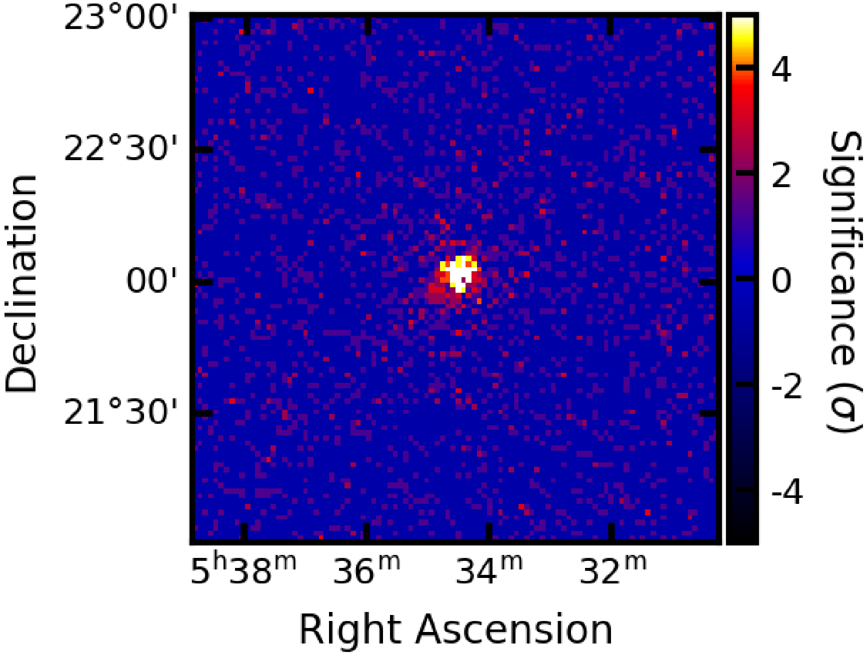
That is kind of unsatisfying – what about a map?

Here we have two options:

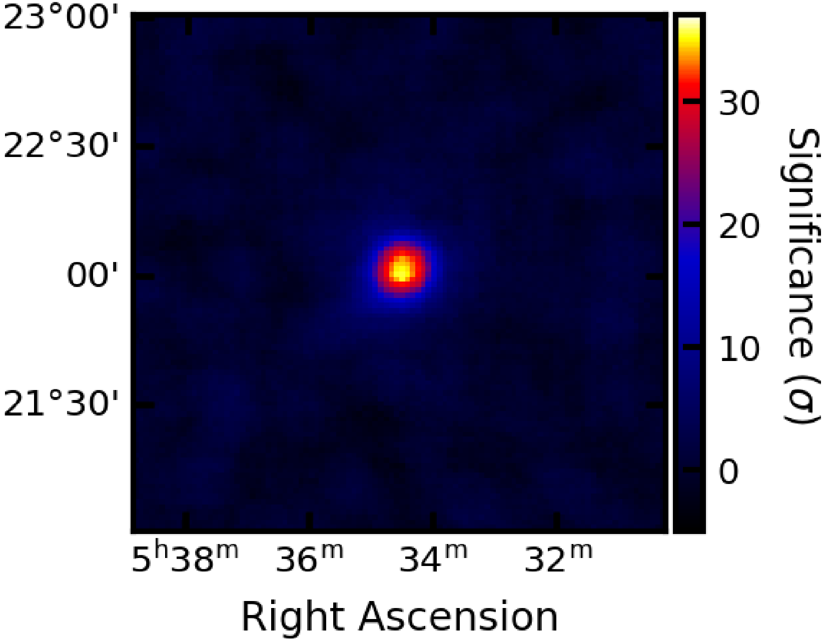
- **Based on excess:** for each pixel of the map, compare the measured counts with the expected background using Li&Ma. Not model dependent but usually done with some smoothing, which impacts scale of visible structures → **common in IACTs**
- **Based on model:** For each pixel compare the likelihood of the measured counts given a model vs the absence of it. Requires an assumption of spectral and spatial properties → **Fermi, HAWC..**

Significance maps

No smoothing



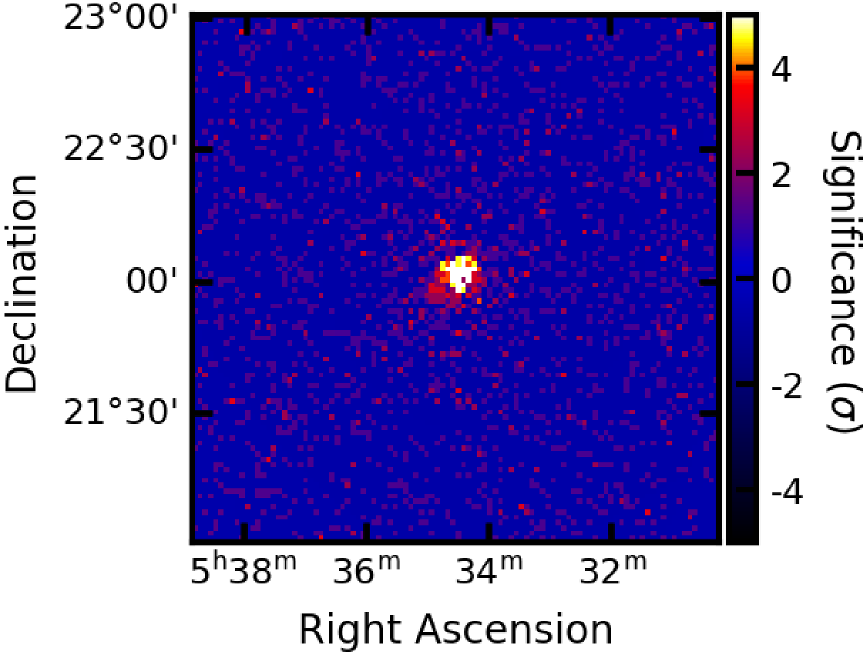
Reasonable smoothing: similar to PSF



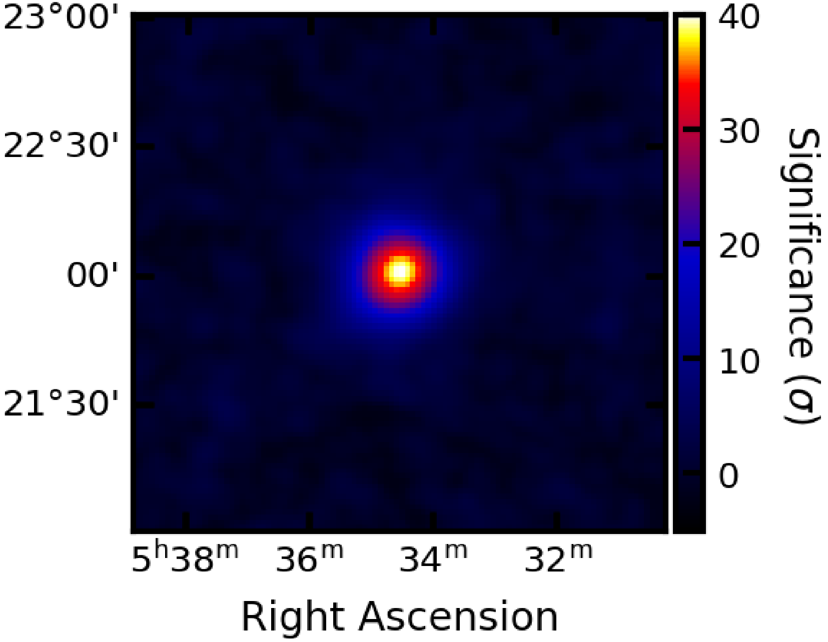
```
estimator = ExcessMapEstimator("0.1deg")
rresult = estimator.run(stacked)
```

Significance maps

NO SMOOTHING



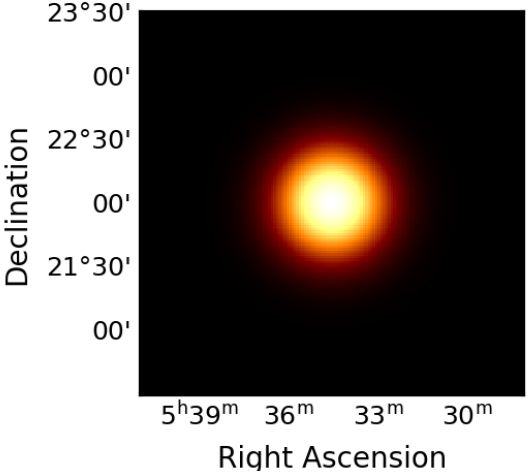
Using a point source model with E^{-2}



```
estimator = TSMapEstimator()
rresult = estimator.run(stacked)
```

Beware!

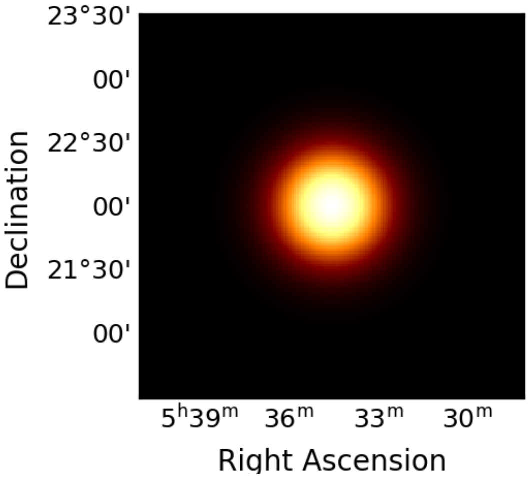
IF SPATIALLY EXTENDED SOURCE



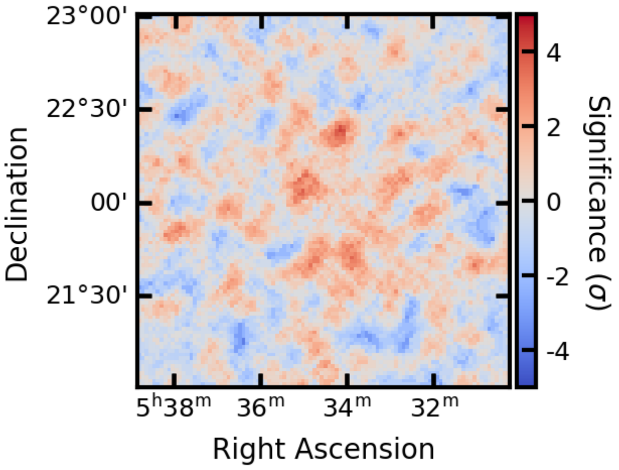
Simulate a source by giving a dataset a model and doing dataset.fake()

Beware!

If spatially extended source



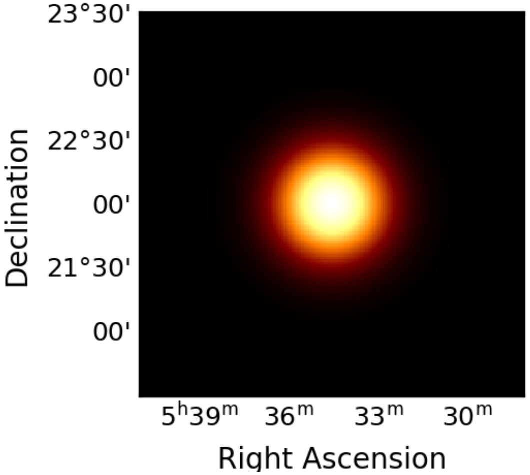
Simulate a source by giving a dataset a model and doing dataset.fake()



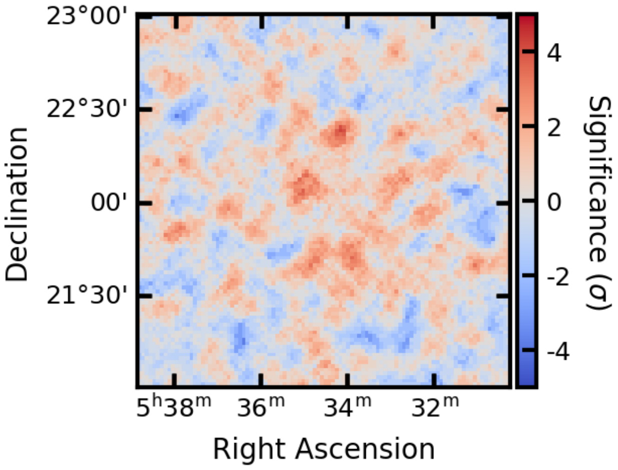
Using correlation radius like the PSF: we see nothing!

Beware!

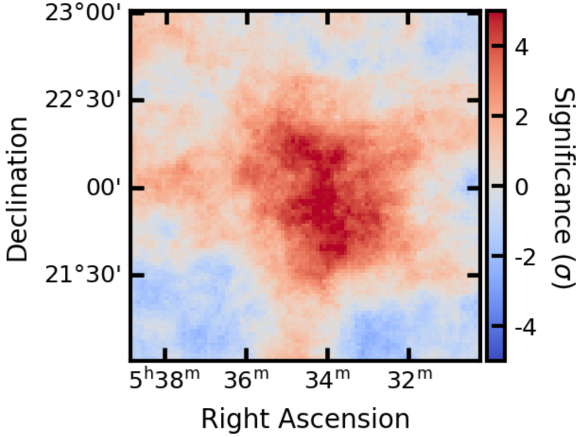
If spatially extended source



Simulate a source by giving a dataset a model and doing dataset.fake()



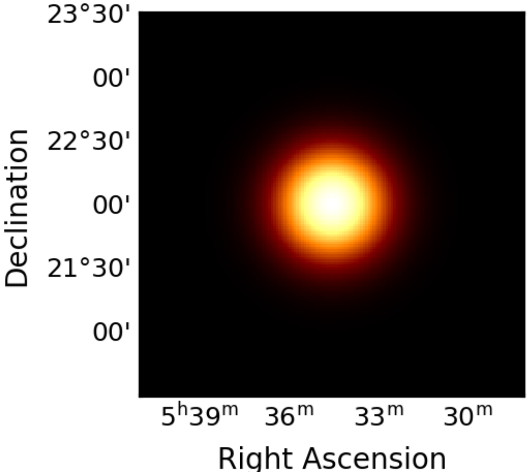
Using correlation radius like the PSF: we see nothing!



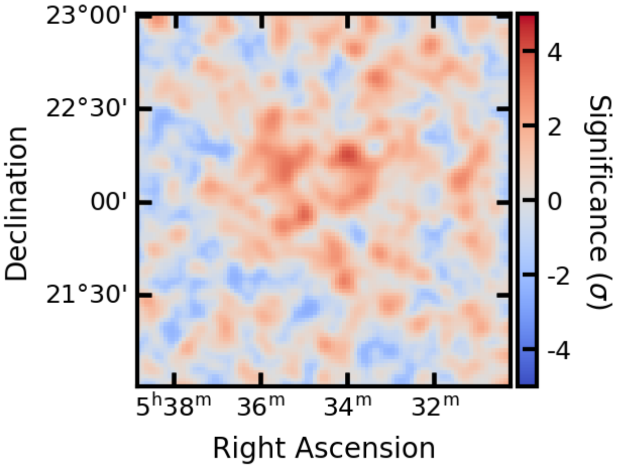
With larger radius it shows up – but background systematics are dangerous!

Beware!

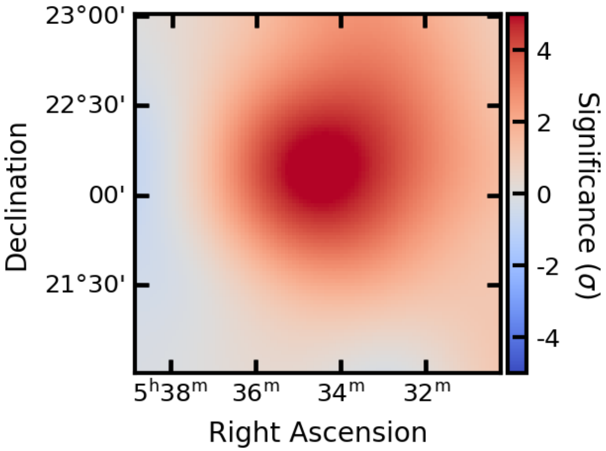
IF SPATIALLY EXTENDED SOURCE



SIMULATE A SOURCE BY GIVING A DATASET A MODEL AND DOING DATASET.FAKE()



USING CORRELATION RADIUS LIKE THE PSF: WE SEE NOTHING!



WITH LARGER RADIUS IT SHOWS UP – BUT BACKGROUND SYSTEMATICS ARE DANGEROUS!

Background normalization

When we look at a significance map we want to know:

1) Are there any significant excesses?

2) Is the background well normalized?

How can we tell?

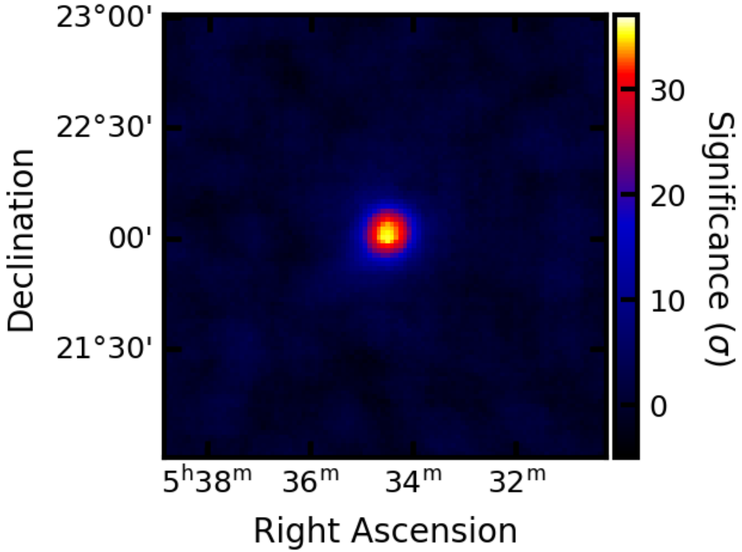
In the absence of signal, the significance distribution should be well described by a **Gaussian distribution with mean=0 and width=1**. This = as many +ve fluctuations as -ve!

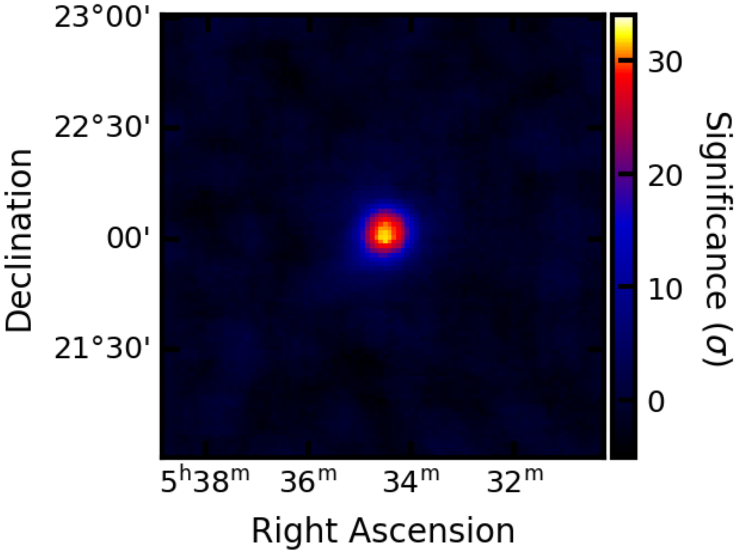
Fluctuations should be randomly distributed so that map has no structures such as gradients or other

Background normalization

The background of one of these two maps is poorly normalized.

Can you tell me which one?

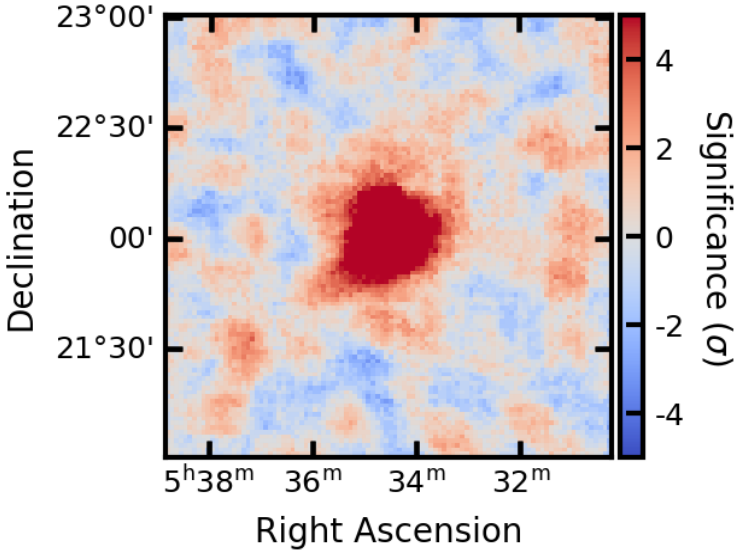


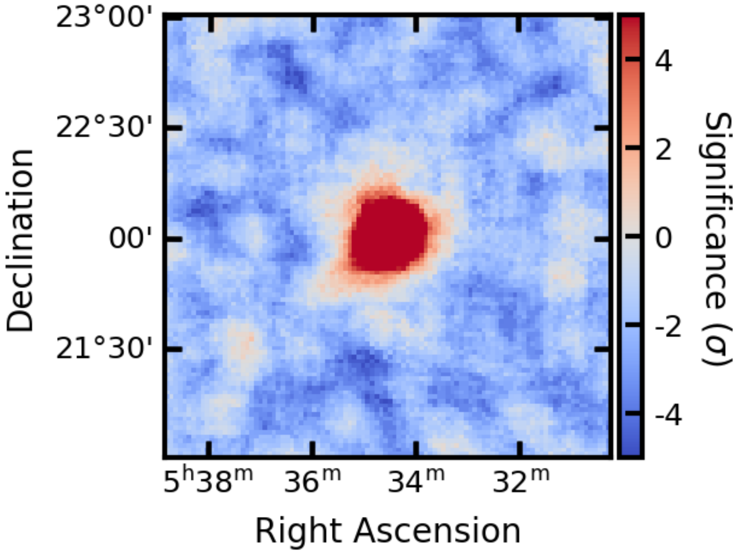


Background normalization

The background of one of these two maps is poorly normalized.

Can you tell me which one?

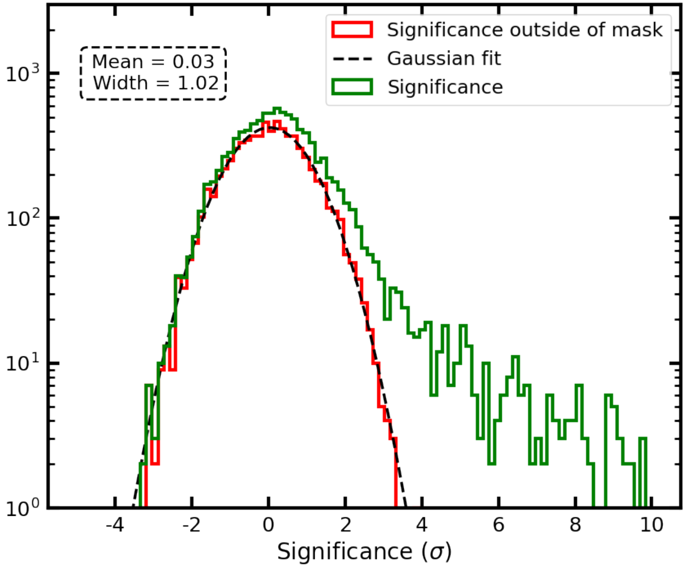


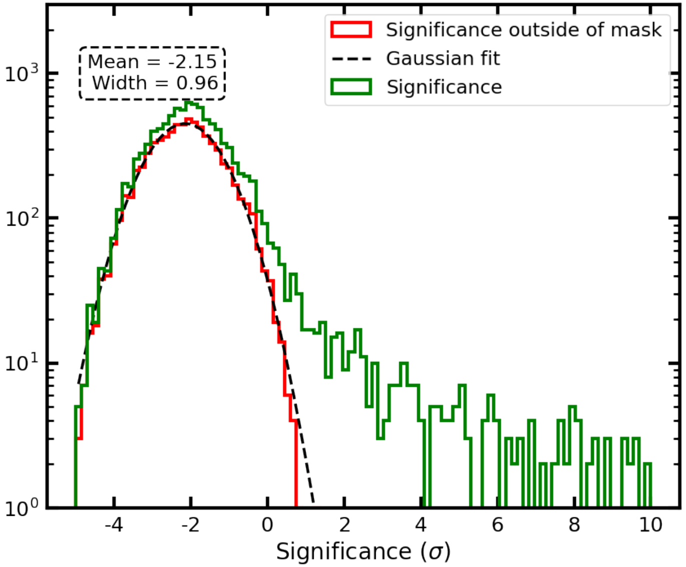


Background normalization

The background of one of these two maps is poorly normalized.

Can you tell me which one?



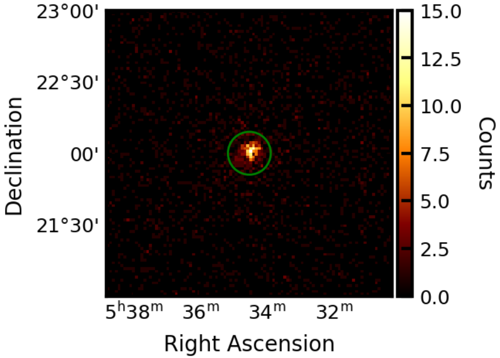


Data Analysis

- Is there a source there?
- What are its properties? (spectral, spatial, temporal*)
- How to present results
- Combining data from different instruments
- Sources of uncertainty

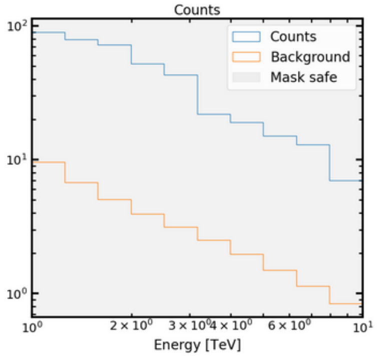
*if we have time

Modeling our source – 1D analysis



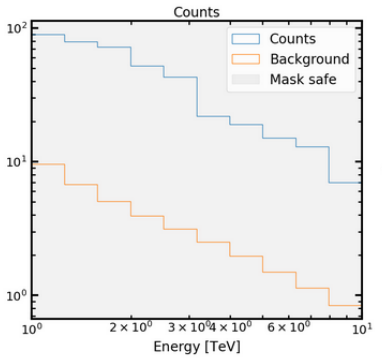
```
SPEC =STACKED.TO_SPECTRUM_DATASET(ON_REGION)
```

Modeling our source – 1D analysis

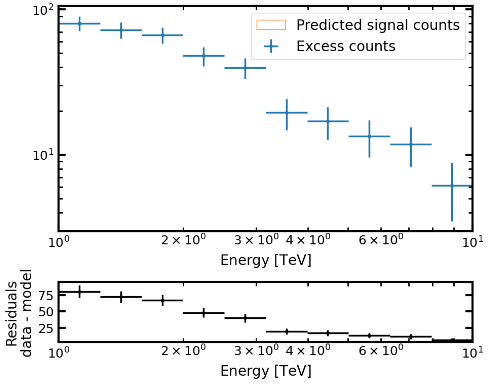


```
SPEC =STACKED.TO_SPECTRUM_DATASET(ON_REGION)
```

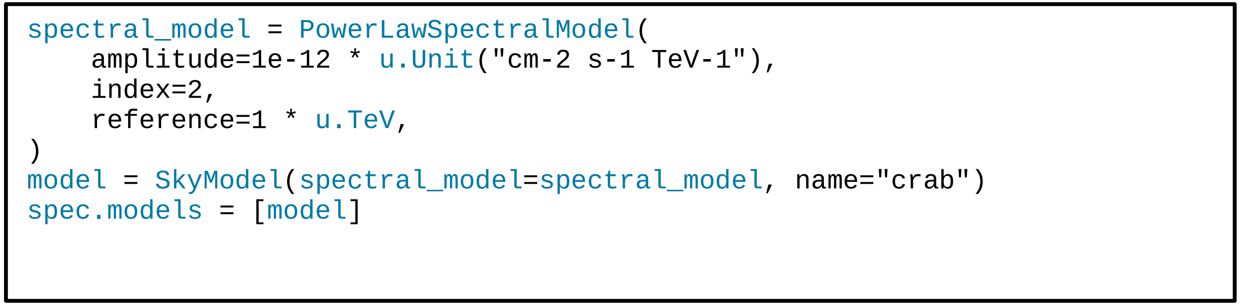
Modeling our source – 1D analysis



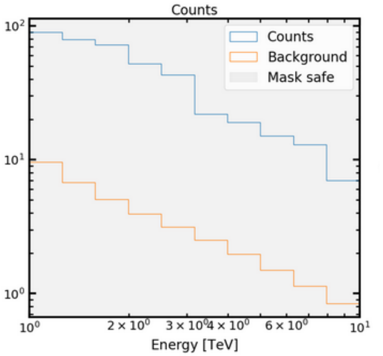
EXCESS= COUNTS-BKG



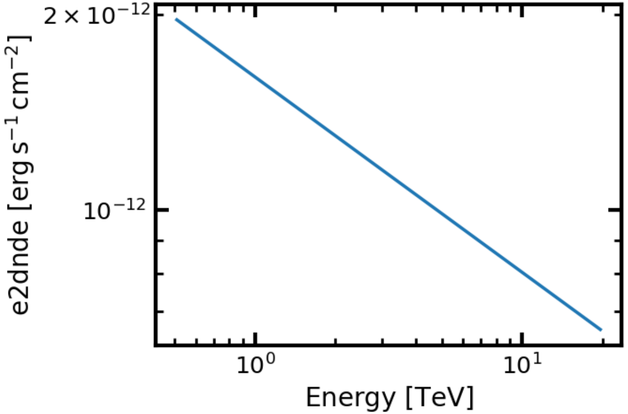
Modeling our source – 1D analysis

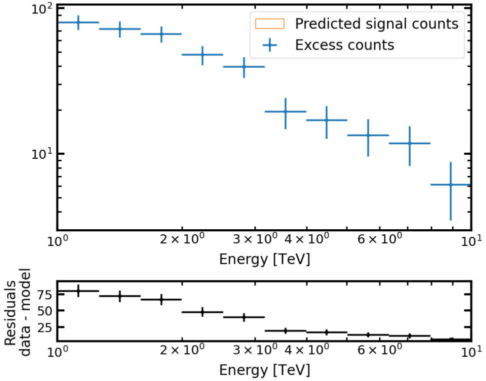


```
spectral_model = PowerLawSpectralModel(
    amplitude=1e-12 * u.Unit("cm-2 s-1 TeV-1"),
    index=2,
    reference=1 * u.TeV,
)
model = SkyModel(spectral_model=spectral_model, name="crab")
spec.models = [model]
```



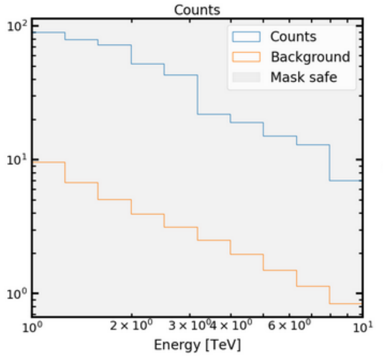
EXCESS= COUNTS-BKG



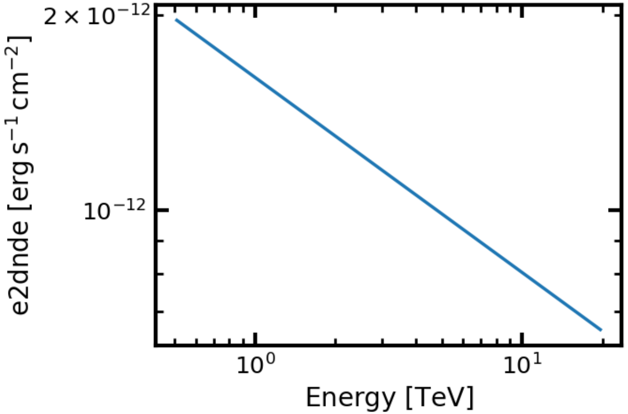


Modeling our source - 1D analysis

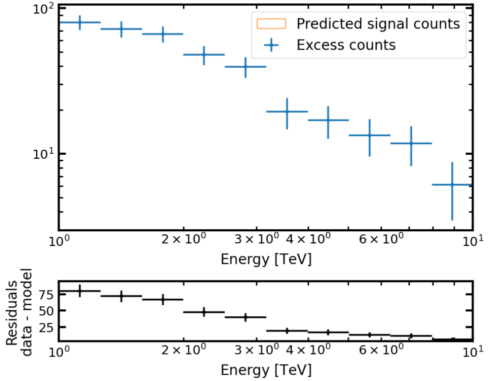
```
spectral_model = PowerLawSpectralModel(
    amplitude=1e-12 * u.Unit("cm-2 s-1 TeV-1"),
    index=2,
    reference=1 * u.TeV,
)
model = SkyModel(spectral_model=spectral_model, name="crab")
spec.models = [model]
```



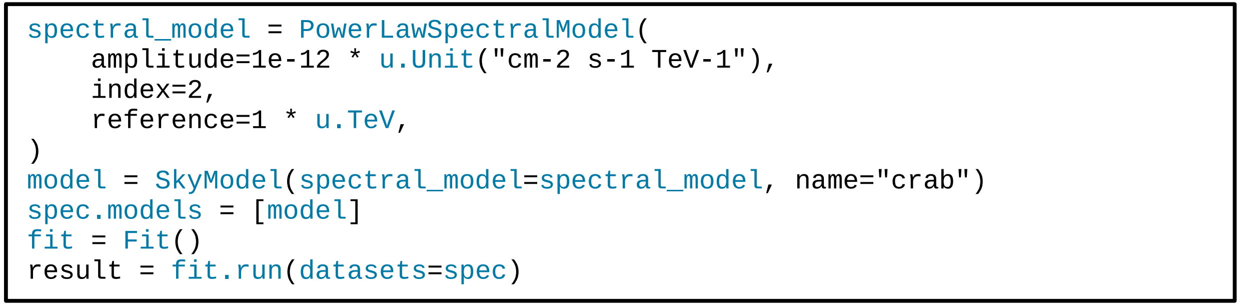
EXCESS= COUNTS-BKG



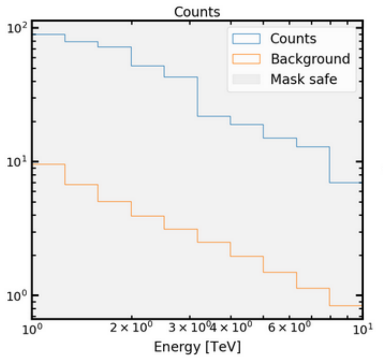
USING IRFs, CALCULATE PREDICTED EXCESS FOR EACH SET OF MODEL PARAMETERS



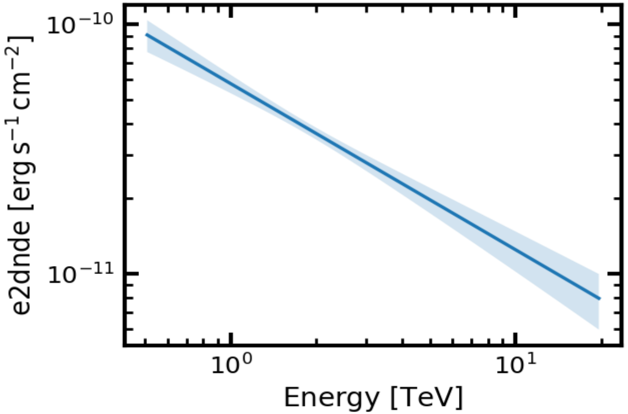
Modeling our source – 1D analysis



```
spectral_model = PowerLawSpectralModel(
    amplitude=1e-12 * u.Unit("cm-2 s-1 TeV-1"),
    index=2,
    reference=1 * u.TeV,
)
model = SkyModel(spectral_model=spectral_model, name="crab")
spec.models = [model]
fit = Fit()
result = fit.run(datasets=spec)
```

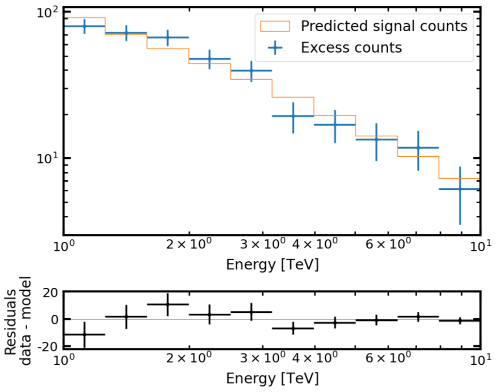


EXCESS= COUNTS-BKG



USING IRFs, CALCULATE PREDICTED EXCESS FOR EACH SET OF MODEL PARAMETERS

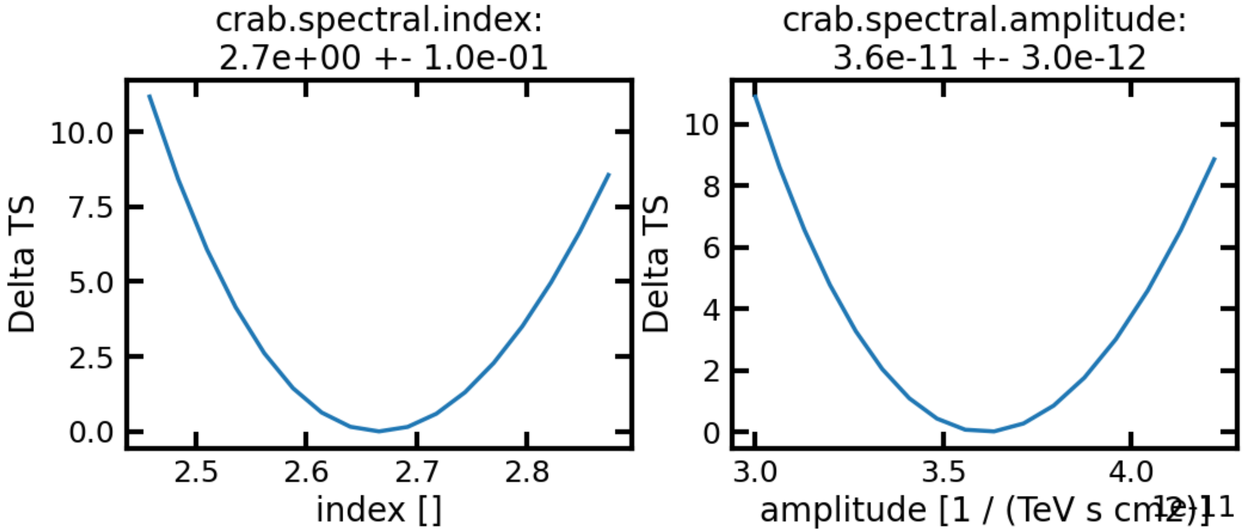
FIT: FIND SET OF PARAMETERS WHICH MAXIMIZES THE LIKELIHOOD (WHICH MAKES THE PREDICTION CLOSEST TO THE OBSERVED EXCESS)



How to determine fit quality?

$$TS = -2\log\left(\frac{\mathcal{L}(\mathcal{H}_0)}{\mathcal{L}(\mathcal{H}_1)}\right)$$

- Maximum likelihood = minimum test statistic
- Is the minimum well defined?



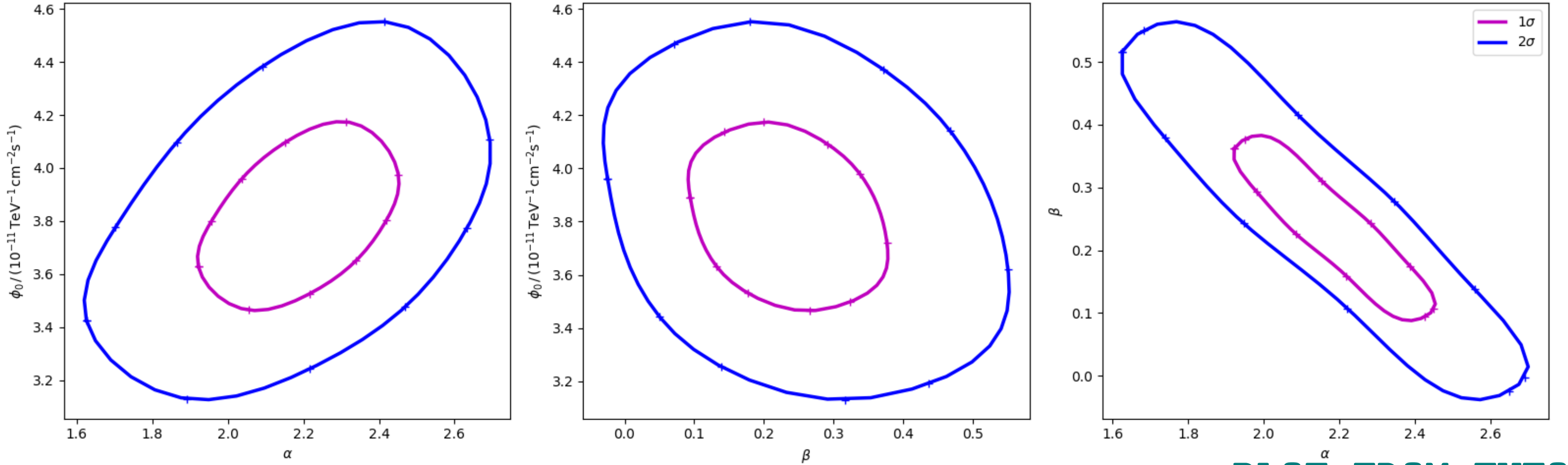
LINK TO TUTORIAL

How to determine fit quality?

$$TS = -2\log\left(\frac{\mathcal{L}(\mathcal{H}_0)}{\mathcal{L}(\mathcal{H}_1)}\right)$$

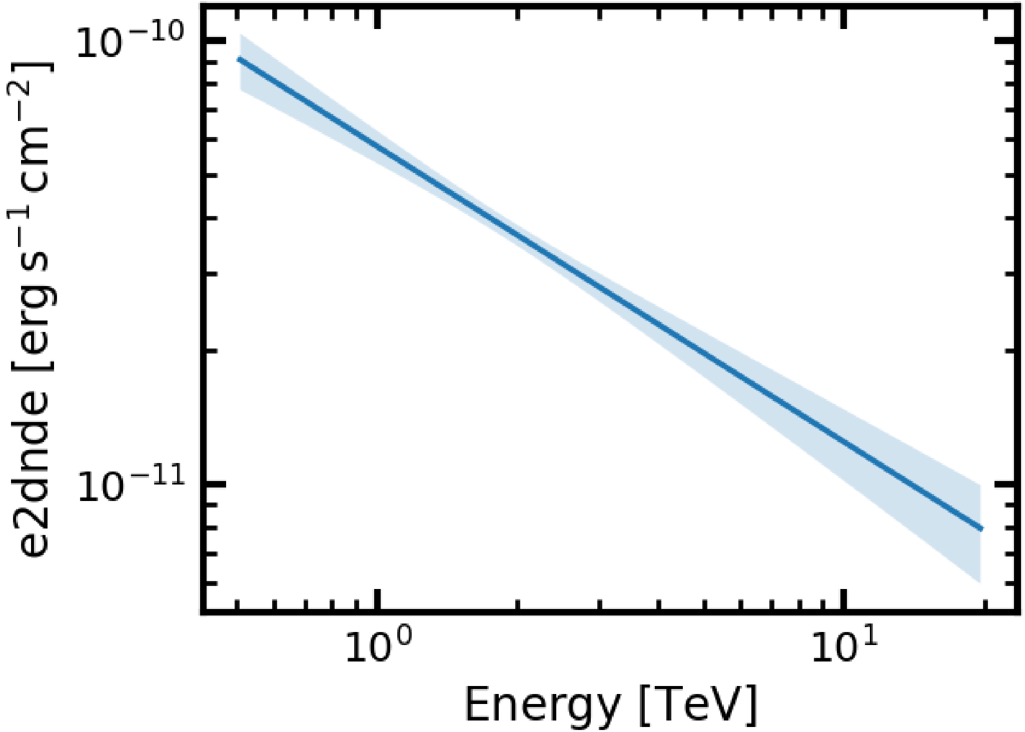
- Maximum likelihood = minimum test statistic
- Is the minimum well defined?

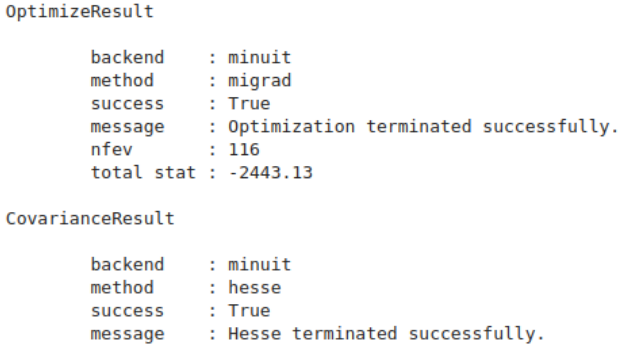
HOW TO



PLOT FROM TUTORIAL

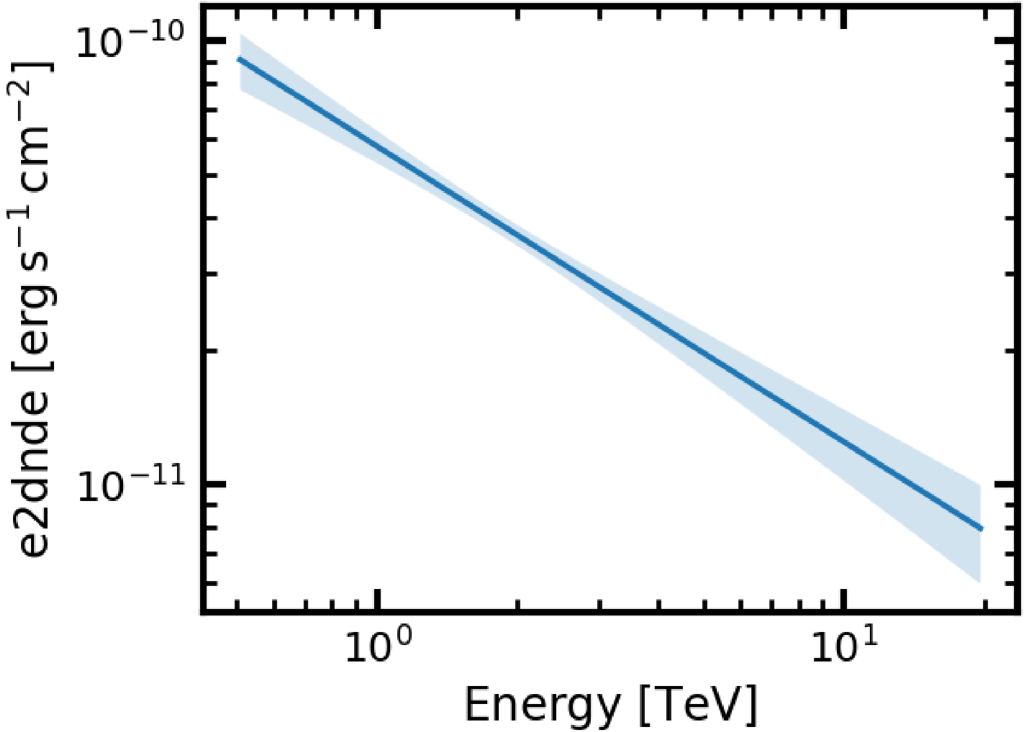
Modeling our source – 1D analysis

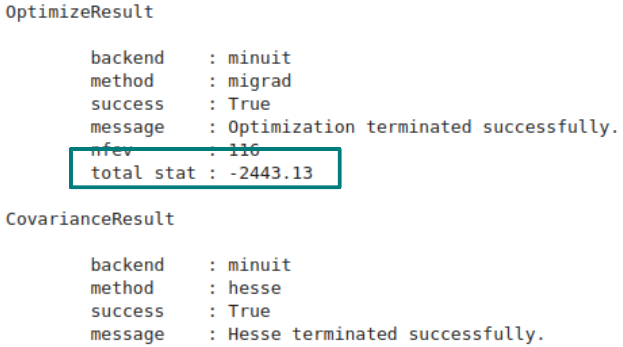




```
OptimizeResult

        backend     : minuit
        method      : migrad
        success     : True
        message     : Optimization terminated successfully.
        nfev        : 116
        total stat : -2443.13

CovarianceResult

        backend     : minuit
        method      : hesse
        success     : True
        message     : Hesse terminated successfully.
```

Modeling our source – 1D analysis

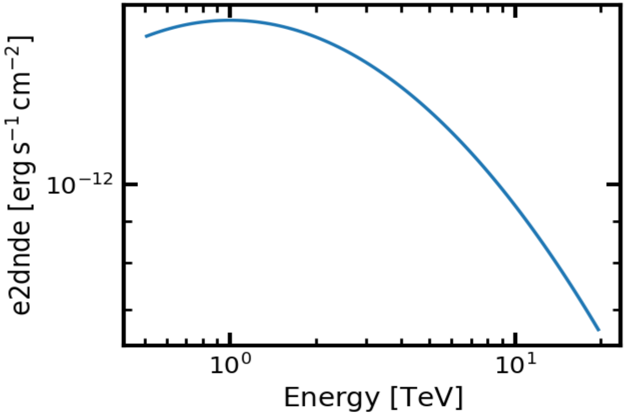


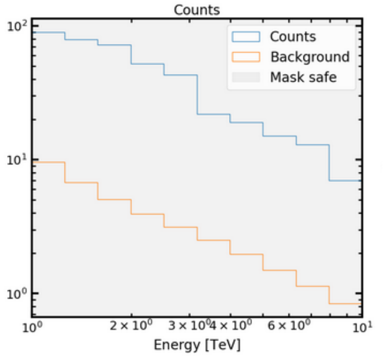


```
OptimizeResult

        backend    : minuit
        method     : migrad
        success    : True
        message    : Optimization terminated successfully.
        nfev       : 116
        total stat : -2443.13

CovarianceResult

        backend    : minuit
        method     : hesse
        success    : True
        message    : Hesse terminated successfully.
```

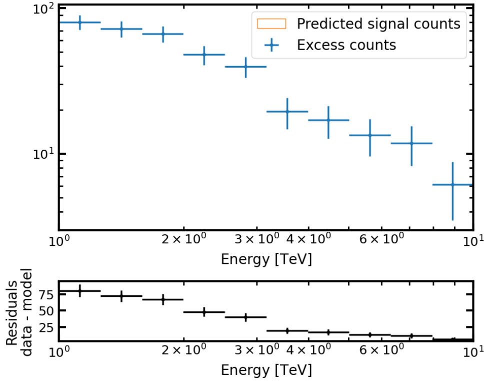
Was that the best model?

```
spectral_model = LogParabolaSpectralModel(
    amplitude=1e-12 * u.Unit("cm-2 s-1 TeV-1"),
    index=2,
    beta=0.01,
    reference=1 * u.TeV,
)
model = SkyModel(spectral_model=spectral_model, name="crab-lp")
spec.models = [model]
fit = Fit()
result = fit.run(datasets=spec)
```



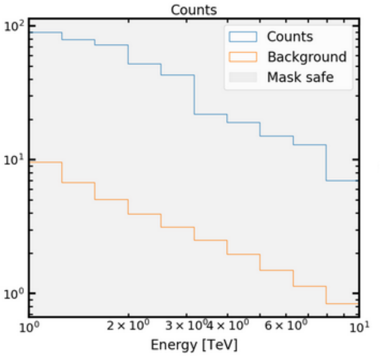


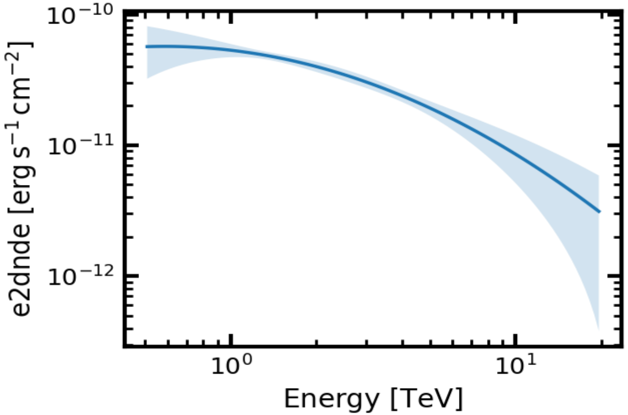
EXCESS= COUNTS-BKG

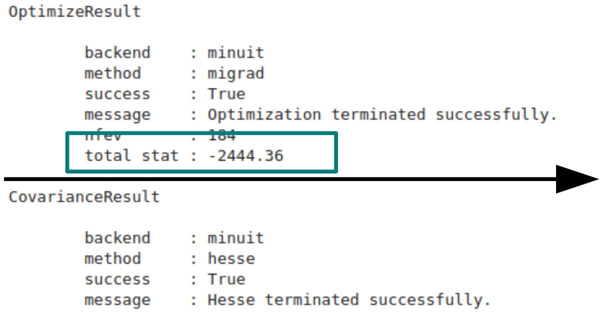


Was that the best model?

```
spectral_model = LogParabolaSpectralModel(
    amplitude=1e-12 * u.Unit("cm-2 s-1 TeV-1"),
    index=2,
    beta=0.01,
    reference=1 * u.TeV,
)
model = SkyModel(spectral_model=spectral_model, name="crab-lp")
spec.models = [model]
fit = Fit()
result = fit.run(datasets=spec)
```

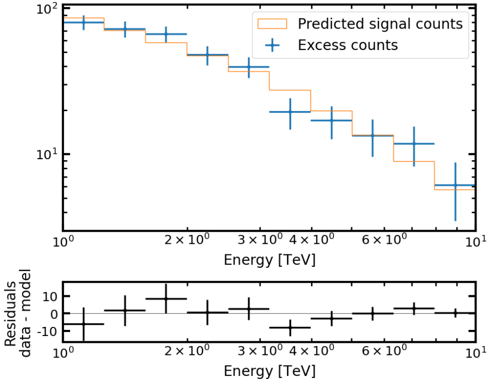




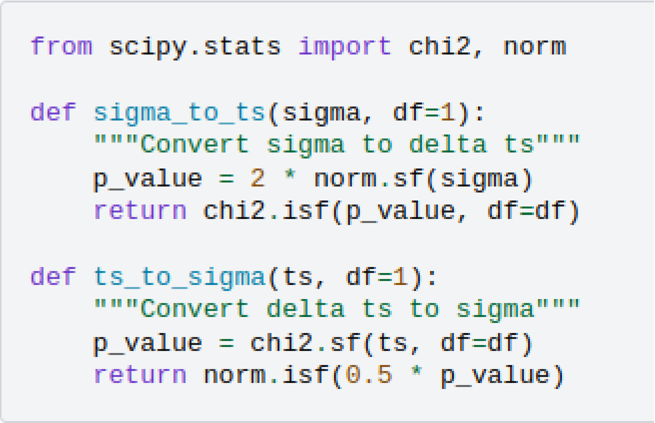


```
OptimizeResult

        backend    : minuit
        method     : migrad
        success    : True
        message    : Optimization terminated successfully.
        nfev       : 184
        total stat : -2444.36

CovarianceResult

        backend    : minuit
        method     : hesse
        success    : True
        message    : Hesse terminated successfully.
```

EXCESS= COUNTS-BKG



Was that the best model?



```python
from scipy.stats import chi2, norm

def sigma_to_ts(sigma, df=1):
    """Convert sigma to delta ts"""
    p_value = 2 * norm.sf(sigma)
    return chi2.isf(p_value, df=df)

def ts_to_sigma(ts, df=1):
    """Convert delta ts to sigma"""
    p_value = chi2.sf(ts, df=df)
    return norm.isf(0.5 * p_value)
```

- Again we are testing two hypothesis, which are **nested**

$$TS = -2 \log \left(\frac{\mathcal{L}(\mathcal{H}_0)}{\mathcal{L}(\mathcal{H}_1)} \right)$$

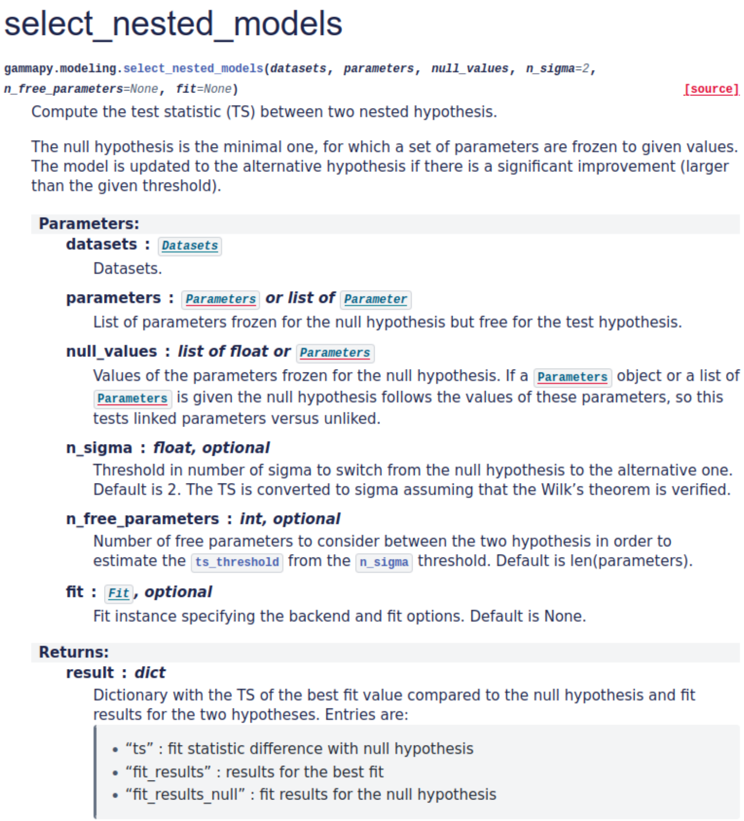
- **Wilks theorem** shows that the difference of the test statistic values for the two hypotheses asymptotically follows a X^2 distribution with N_{DOF} degrees of freedom, where N_{DOF} is the difference of free parameters between the two hypothesis <u>as long as they are nested</u>

- if N_{DOF}=1 then we can simply take it as $\sigma = \sqrt{TS}$. Which is what we did for the maps, remember?

Was that the best model?

- The log-parabola model is equivalent to the powerlaw model with one extra parameter
- In that simple case we can just do TS_{PL}-Ts_{LogP} to determine whether the description with one more parameter is more likely given the data
- $TS_{PL}-Ts_{LogP}$ = -2443.13-(-2444.36) = 1.23 → not really!
- In reality the Crab spectrum is curved – but the H.E.S.S. public data is not sensitive enough!

Was that the best model?



select_nested_models

`gammapy.modeling.select_nested_models(datasets, parameters, null_values, n_sigma=2, n_free_parameters=None, fit=None)` [source]

Compute the test statistic (TS) between two nested hypothesis.

The null hypothesis is the minimal one, for which a set of parameters are frozen to given values. The model is updated to the alternative hypothesis if there is a significant improvement (larger than the given threshold).

Parameters:

datasets : `Datasets`
Datasets.

parameters : `Parameters` ***or list of*** `Parameter`
List of parameters frozen for the null hypothesis but free for the test hypothesis.

null_values : ***list of float or*** `Parameters`
Values of the parameters frozen for the null hypothesis. If a `Parameters` object or a list of `Parameters` is given the null hypothesis follows the values of these parameters, so this tests linked parameters versus unliked.

n_sigma : ***float, optional***
Threshold in number of sigma to switch from the null hypothesis to the alternative one. Default is 2. The TS is converted to sigma assuming that the Wilk’s theorem is verified.

n_free_parameters : ***int, optional***
Number of free parameters to consider between the two hypothesis in order to estimate the `ts_threshold` from the `n_sigma` threshold. Default is len(parameters).

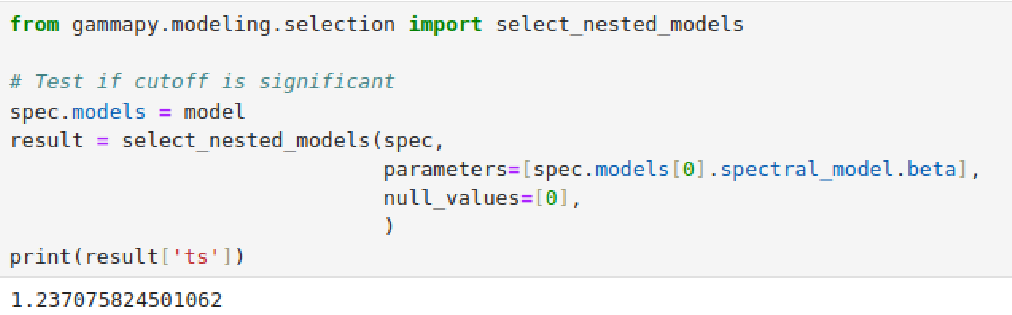
fit : `Fit`, ***optional***
Fit instance specifying the backend and fit options. Default is None.

Returns:

result : ***dict***
Dictionary with the TS of the best fit value compared to the null hypothesis and fit results for the two hypotheses. Entries are:

- “ts” : fit statistic difference with null hypothesis
- “fit_results” : results for the best fit
- “fit_results_null” : fit results for the null hypothesis

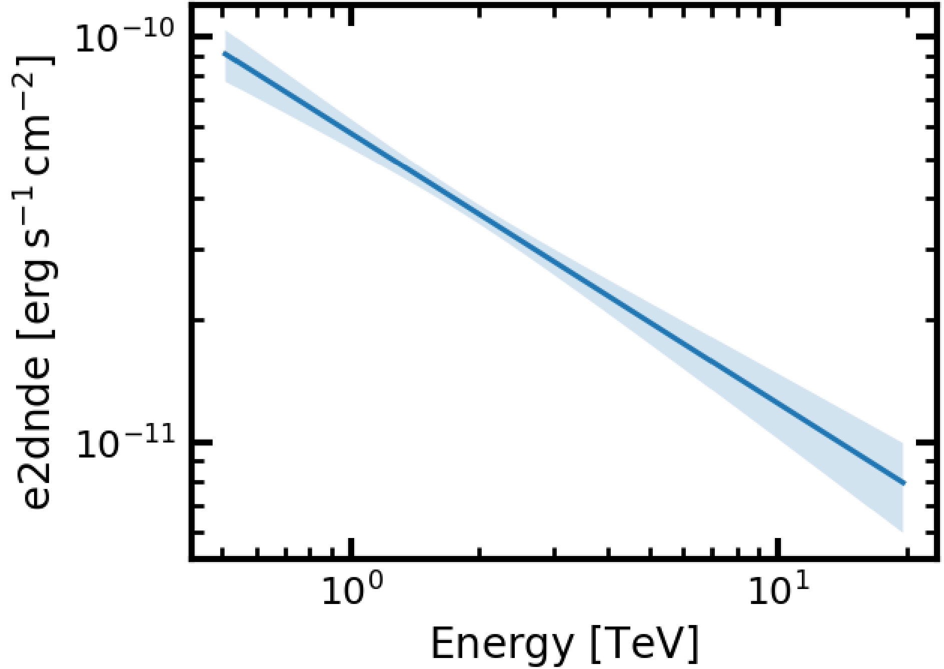
```
FROM GAMMAPY.MODELING.SELECTION IMPORT SELECT_NESTED_MODELS


# TEST IF CURVATURE IS SIGNIFICANT
SPEC.MODELS = MODEL
RESULT = SELECT_NESTED_MODELS(SPEC,
                              PARAMETERS=[SPEC.MODELS[0].SPECTRAL_MODEL.BETA],
                              NULL_VALUES=[0],
                              )
PRINT(RESULT['TS'])
```



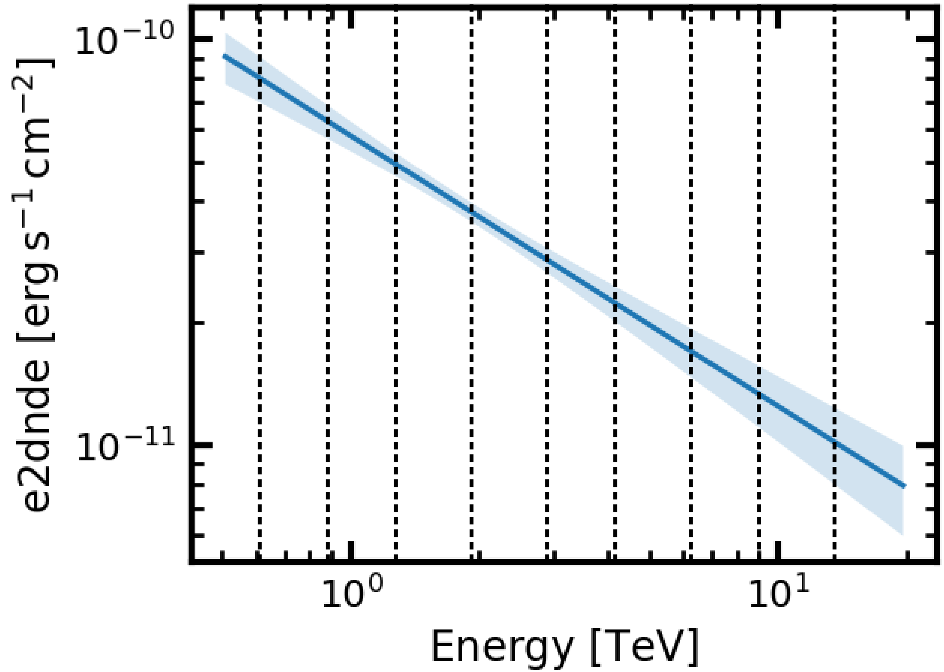
```
from gammapy.modeling.selection import select_nested_models

# Test if cutoff is significant
spec.models = model
result = select_nested_models(spec,
                              parameters=[spec.models[0].spectral_model.beta],
                              null_values=[0],
                              )
print(result['ts'])
```

```
1.237075824501062
```

Modeling our source – 1D analysis

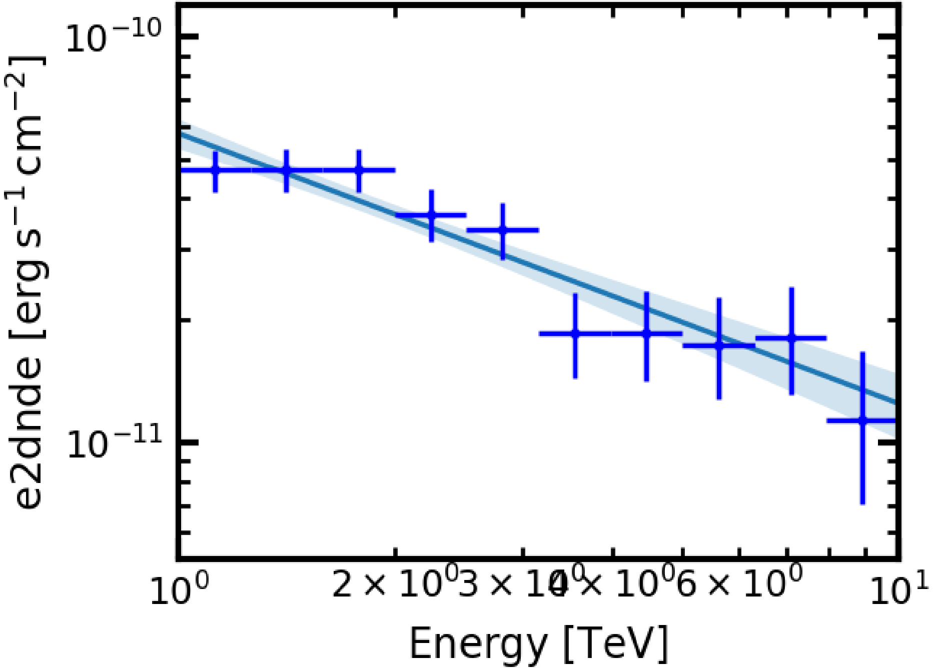


Modeling our source – 1D analysis



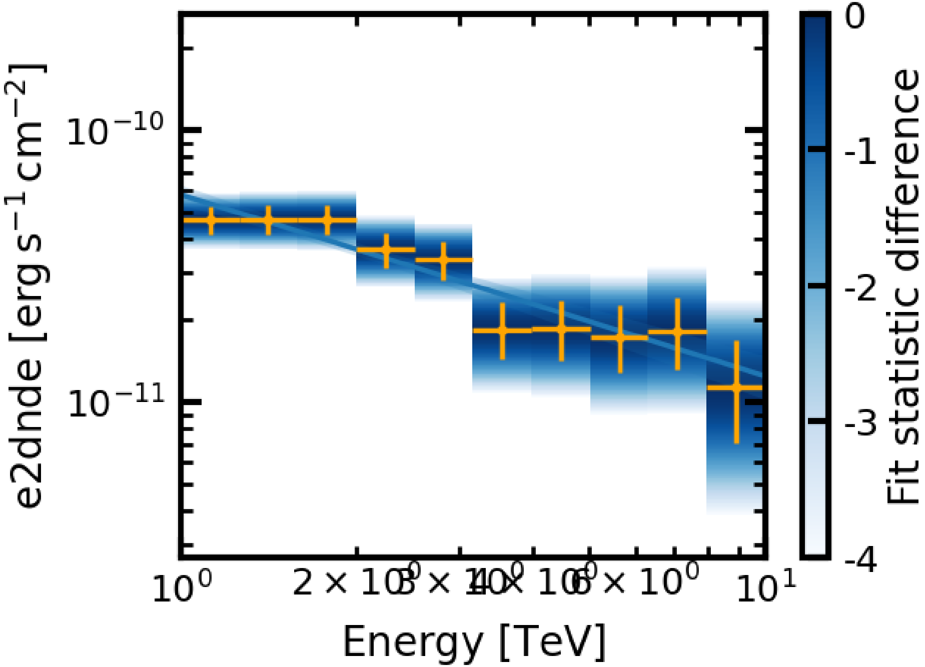
Fit again in each small bin assuming the overall shape but letting the normalization free

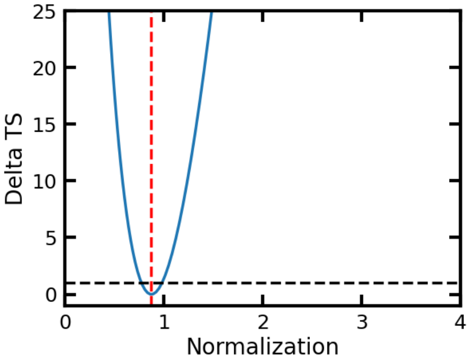
Modeling our source – 1D analysis



Fit again in each small bin assuming the overall shape but letting the normalization free

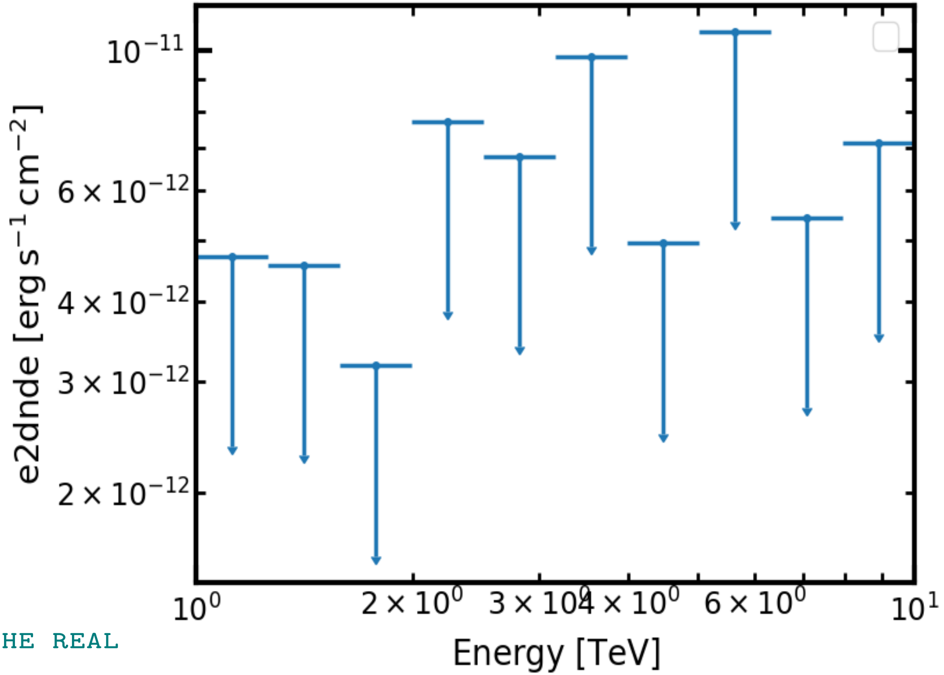
Modeling our source – 1D analysis





```
FLUX_POINTS.PLOT_TS_PROFILES(SED_TYPE="E2DNDE")
```

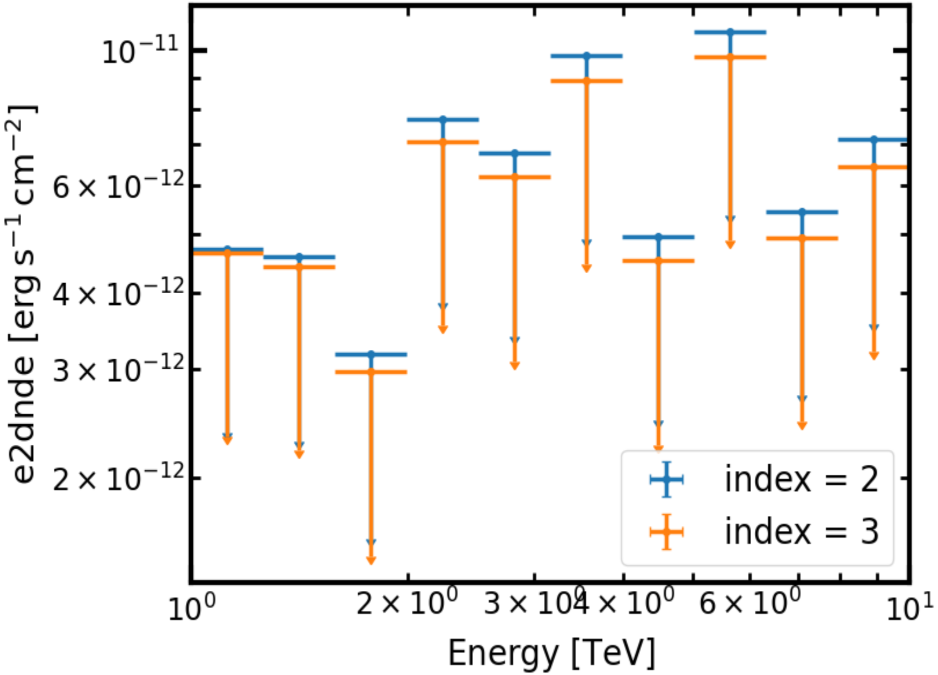
What if nothing is detected? upper limits



You need to assume a spectral shape!

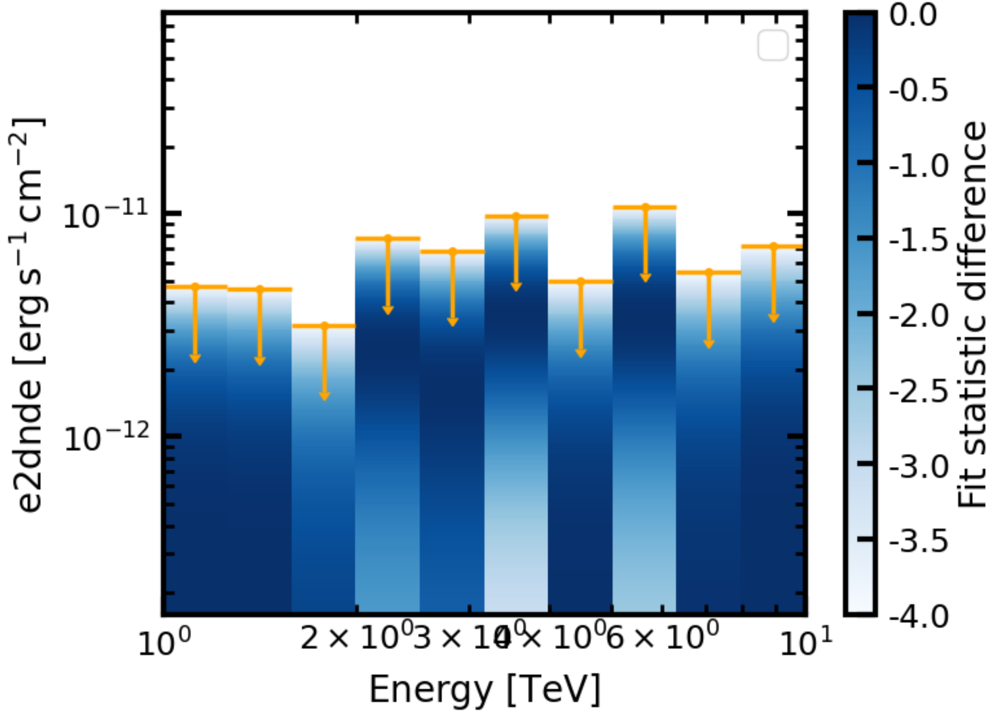
Here I removed the real counts, gave the dataset no model and did dataset.fake()

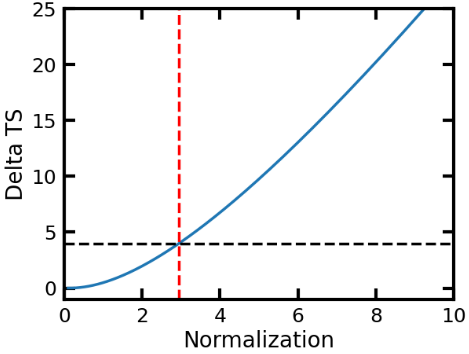
What if nothing is detected? upper limits



You need to assume a spectral shape!

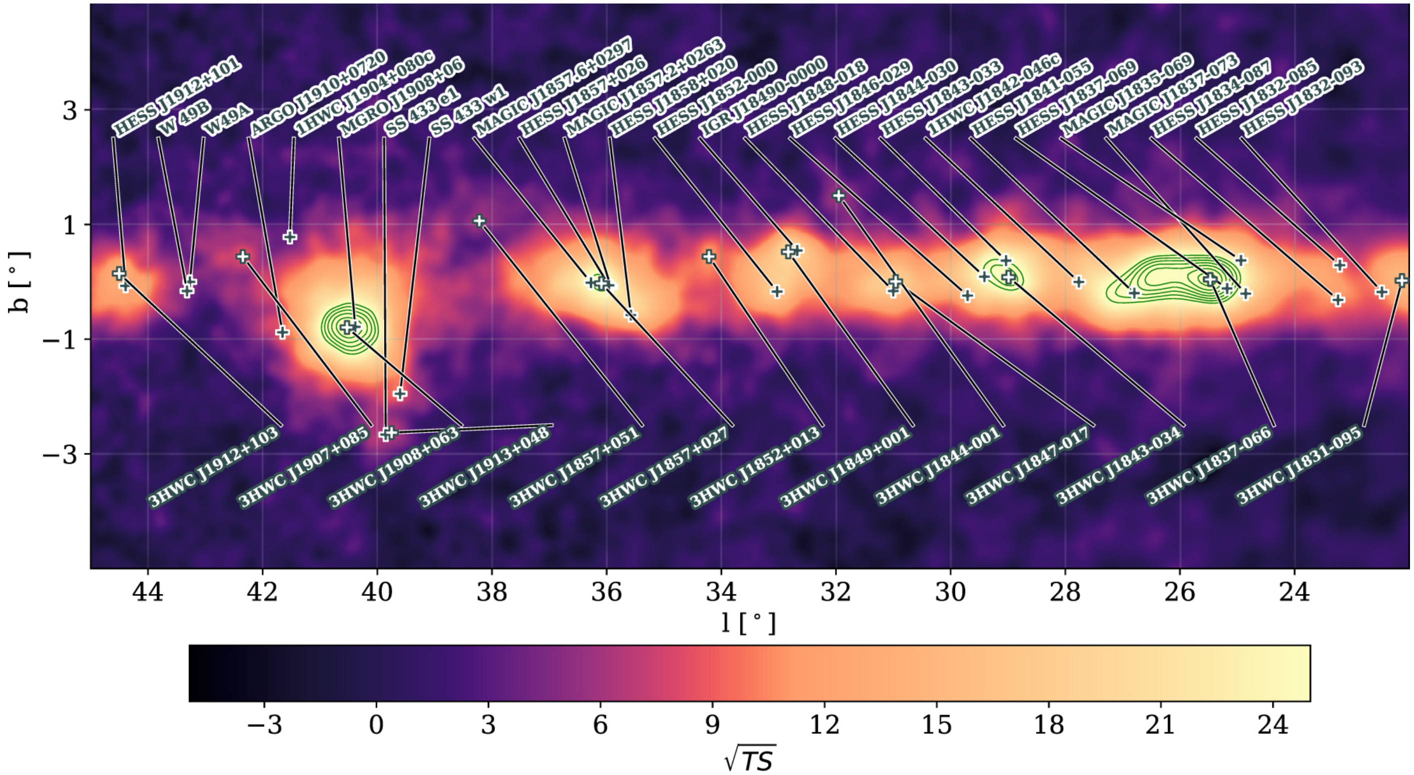
What if nothing is detected? upper limits



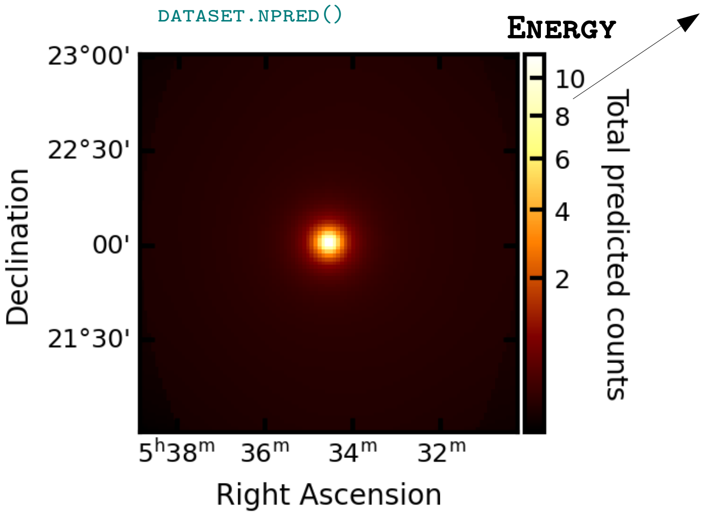


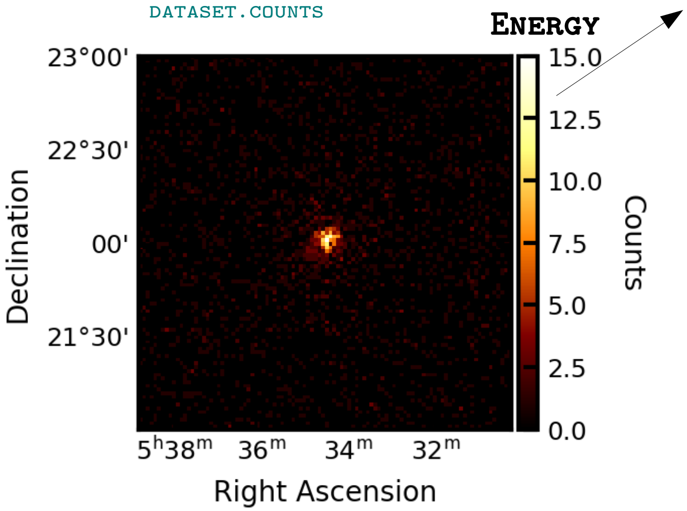
What if the region is complicated?

3HAWC Catalog



Fitting spectra and morphology at once – 3D analysis

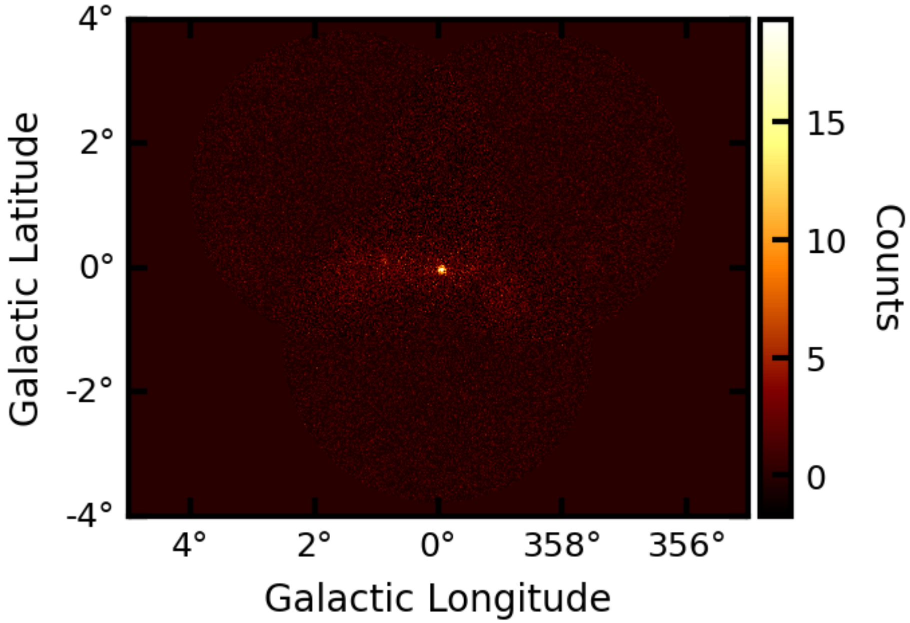


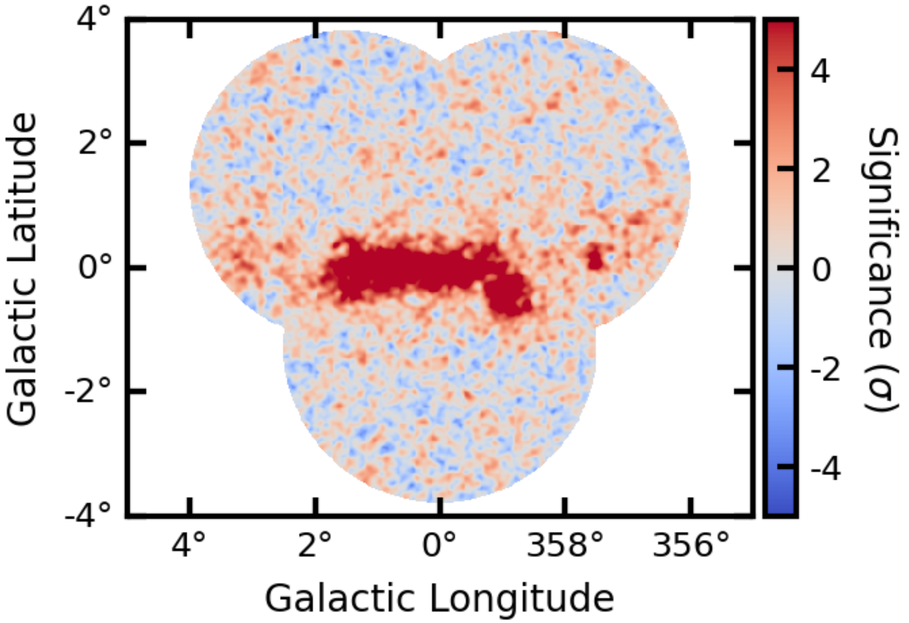


Now we can fit a model to the data by comparing prediction to observation!

Fitting a complicated region

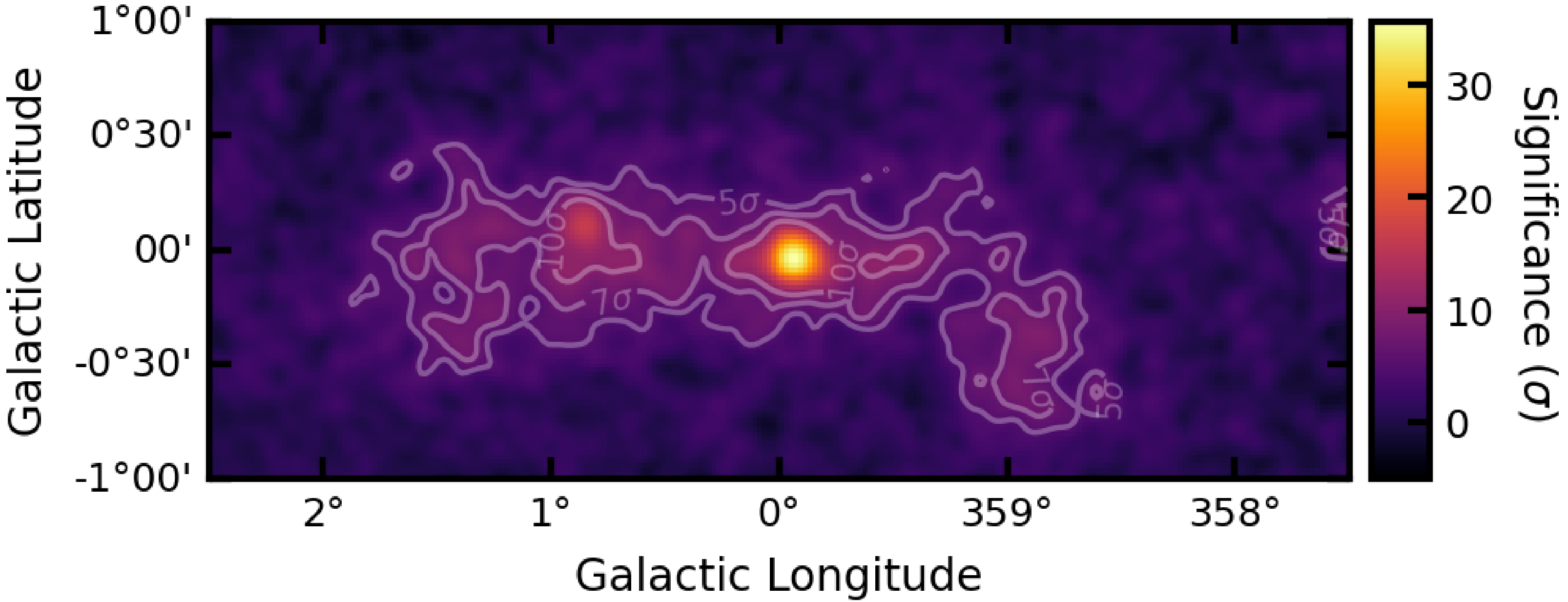
Using simulated CTA observations of the Galactic center





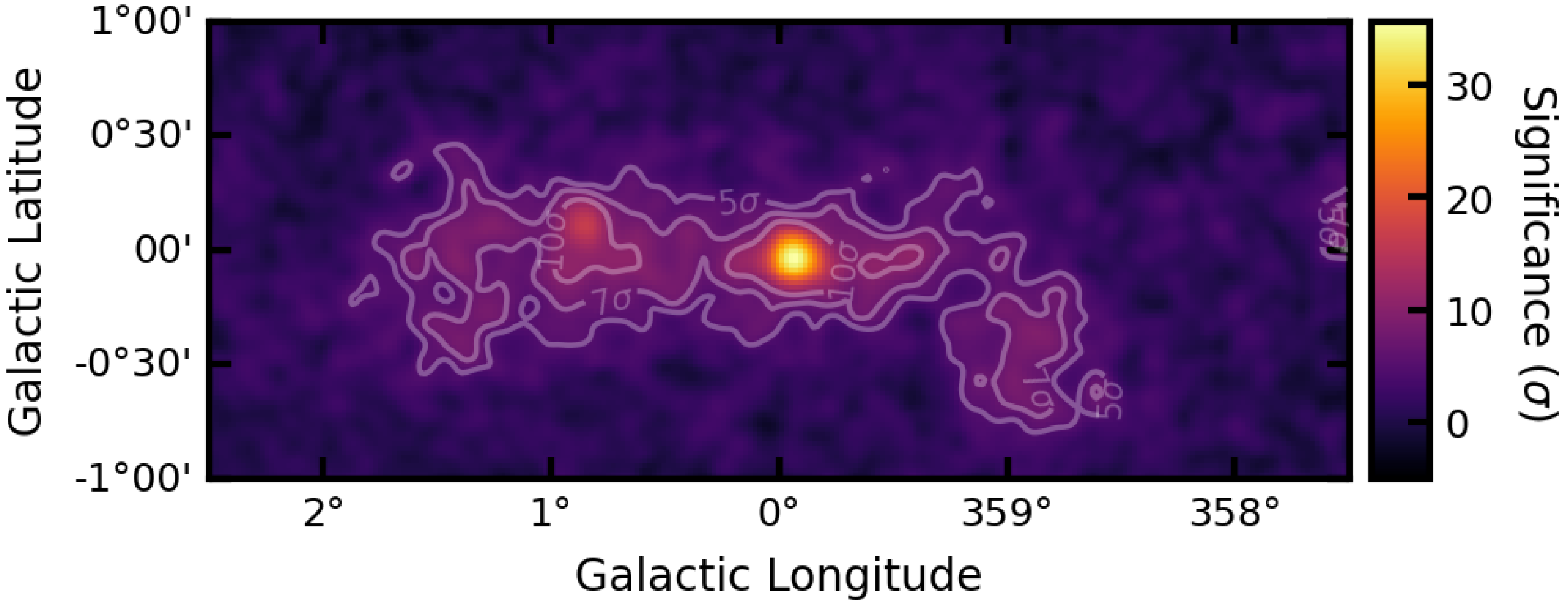
Fitting a complicated region

In theory: start fitting 1 point source and keep adding more sources until not significant anymore. Then test e.g. extension, curvature...



Fitting a complicated region

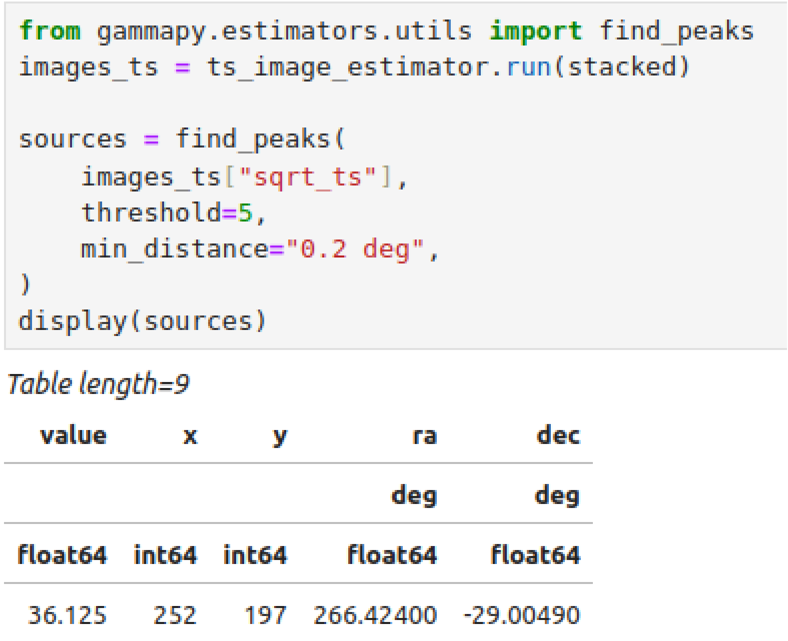
Alternative: Find peaks in TS map and start with already a handful of sources, and then iteratively make the model more complex



Fitting a complicated region

[LINK TO TUTORIAL]

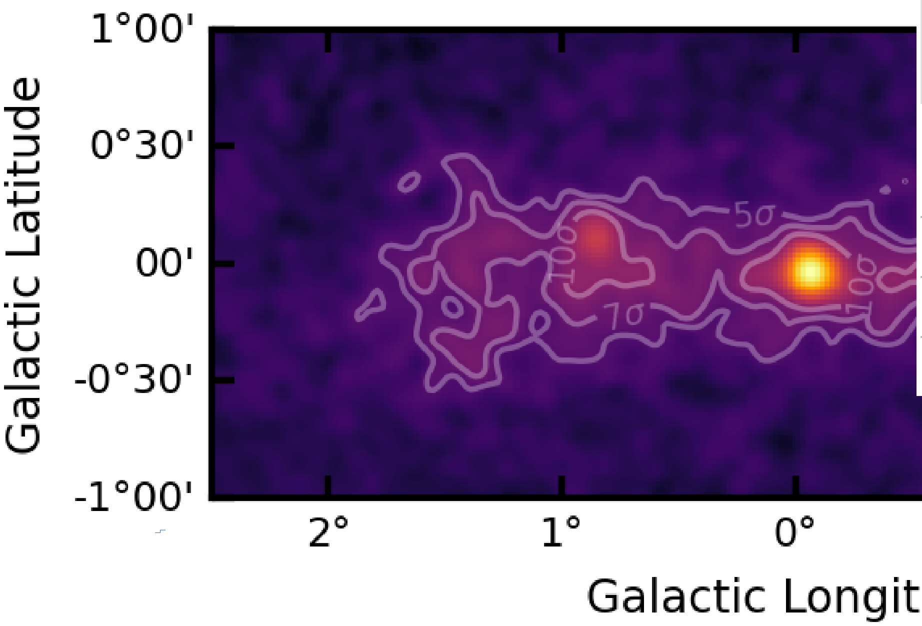
Alternative: Find peaks in TS map and start
sources, and then iteratively make the mode



```python
from gammapy.estimators.utils import find_peaks
images_ts = ts_image_estimator.run(stacked)

sources = find_peaks(
    images_ts["sqrt_ts"],
    threshold=5,
    min_distance="0.2 deg",
)
display(sources)
```

Table length=9

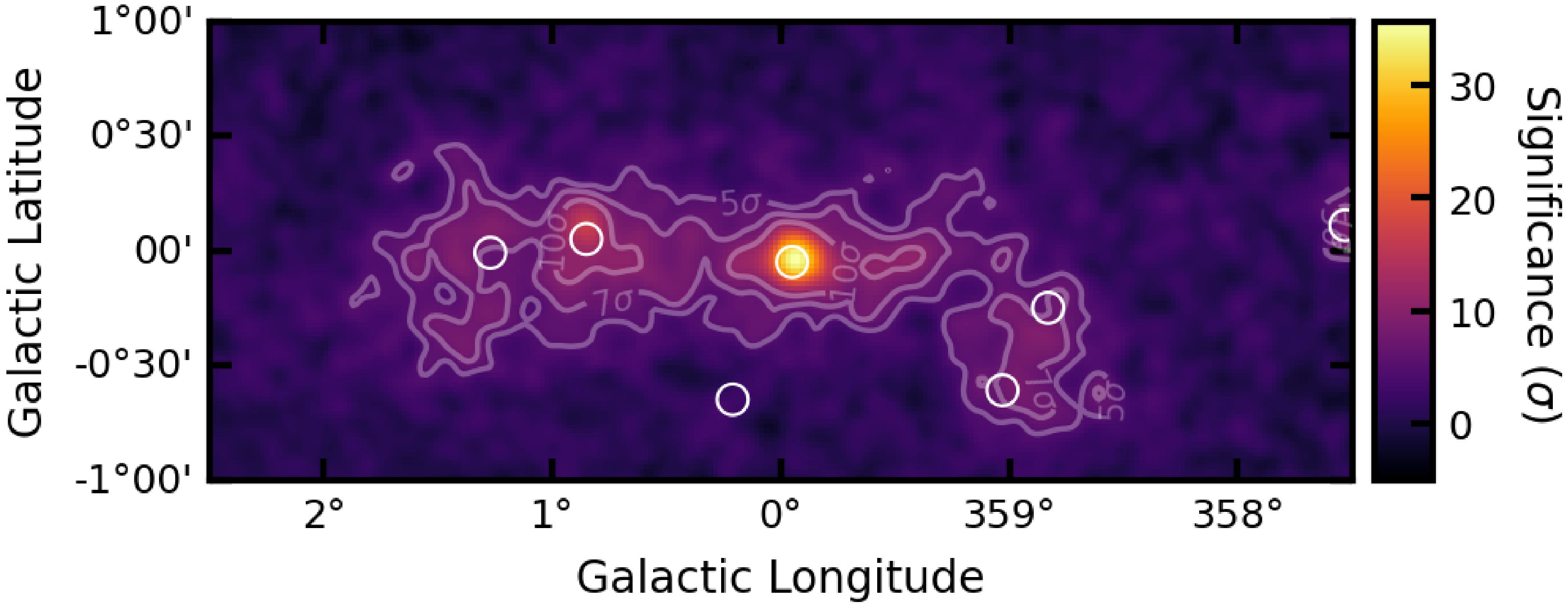
value	x	y	ra	dec
			deg	deg
float64	int64	int64	float64	float64
36.125	252	197	266.42400	-29.00490



Fitting a complicated region

LINK TO TUTORIAL

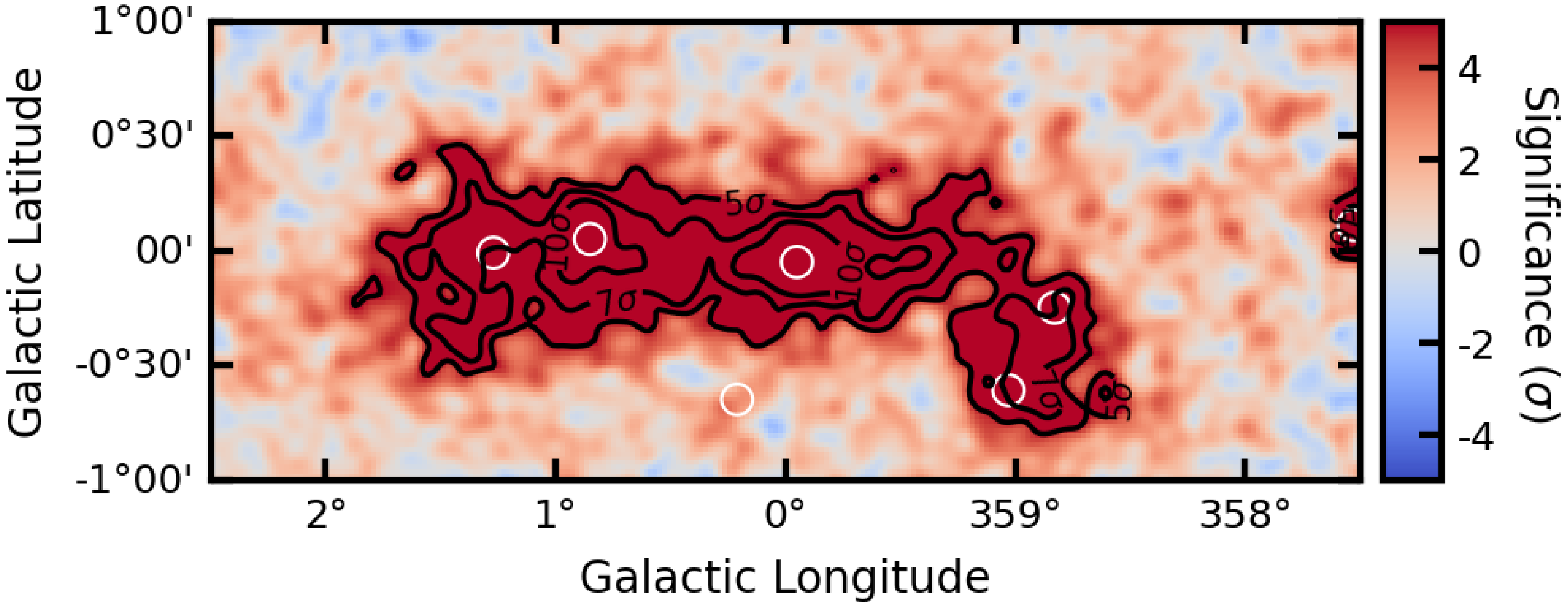
Alternative: Find peaks in TS map and start with already a handful of sources, and then iteratively make the model more complex



Fitting a complicated region

LINK TO TUTORIAL

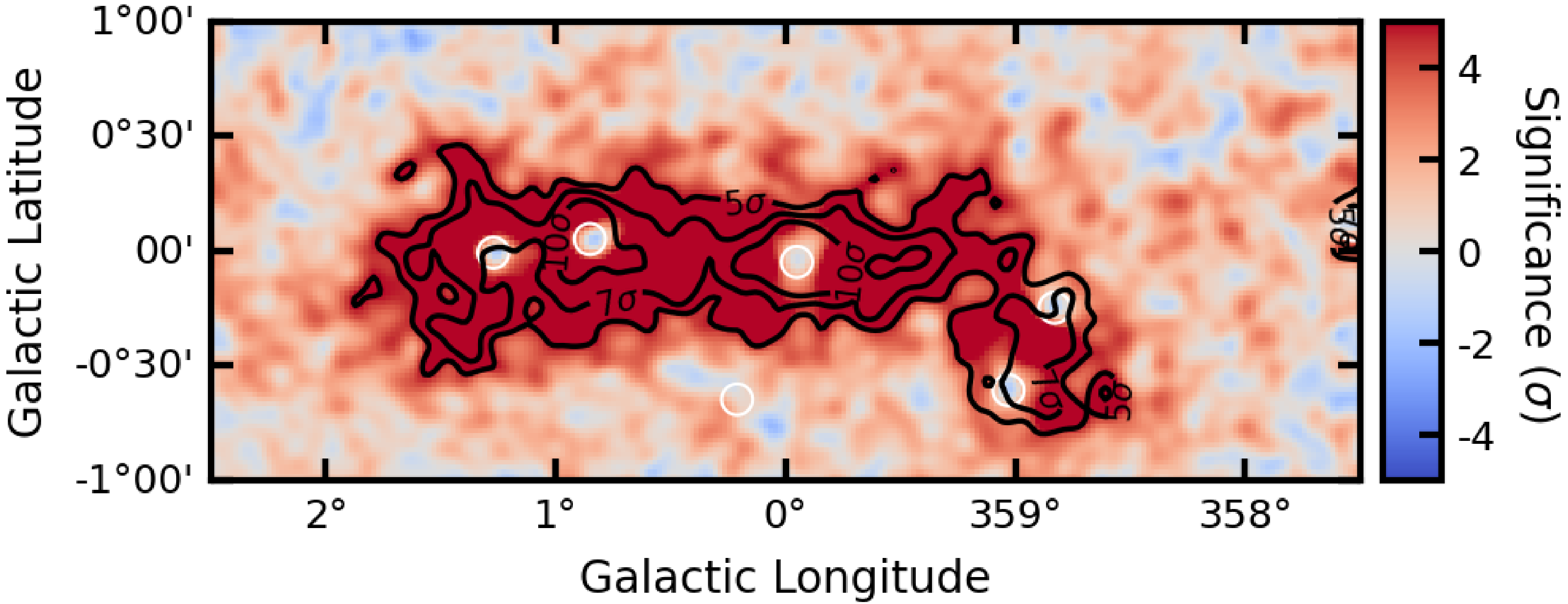
Alternative: Find peaks in TS map and start with already a handful of sources, and then iteratively make the model more complex



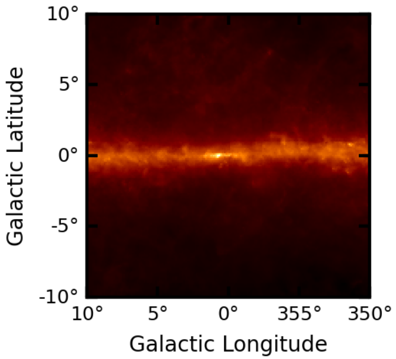
Fitting a complicated region

LINK TO TUTORIAL

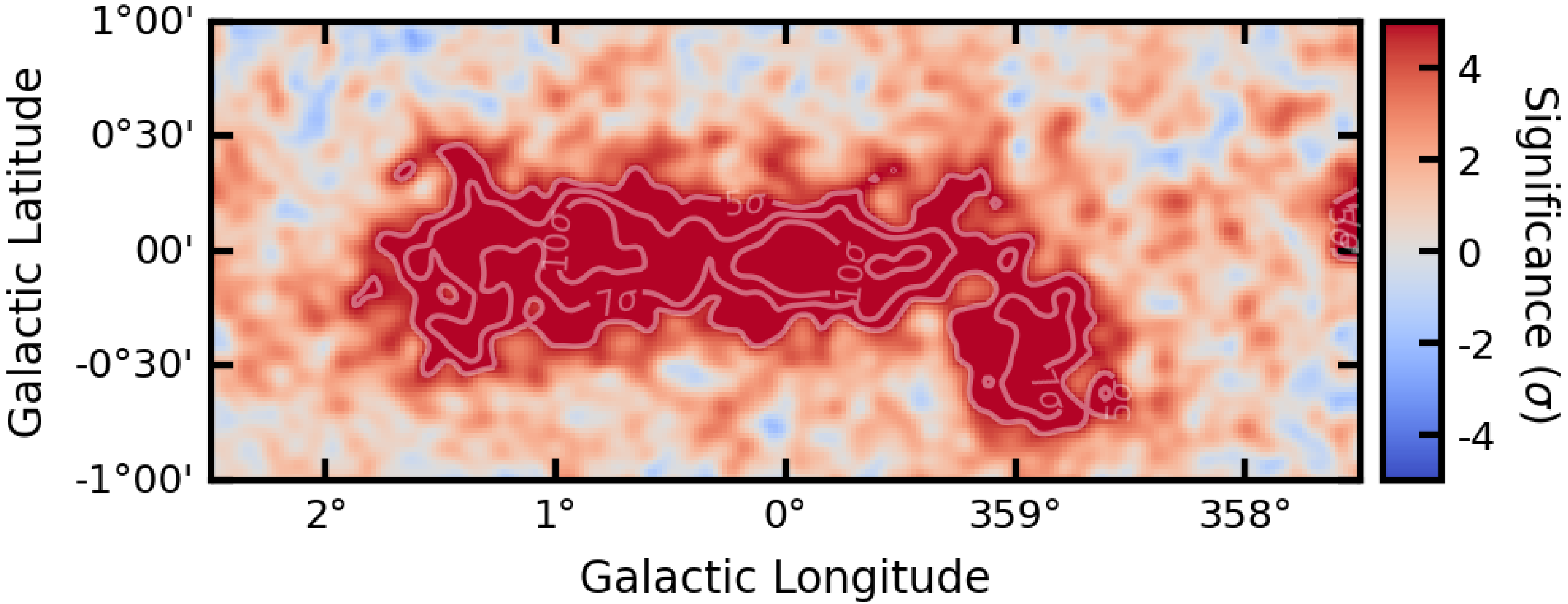
Alternative: Find peaks in TS map and start with already a handful of sources, and then iteratively make the model more complex



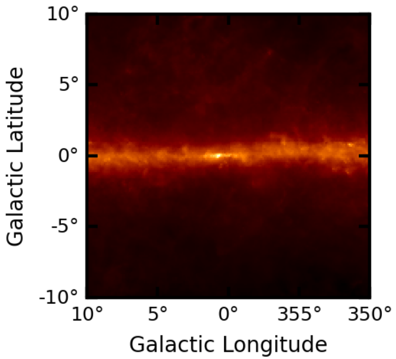
Fitting a complicated region



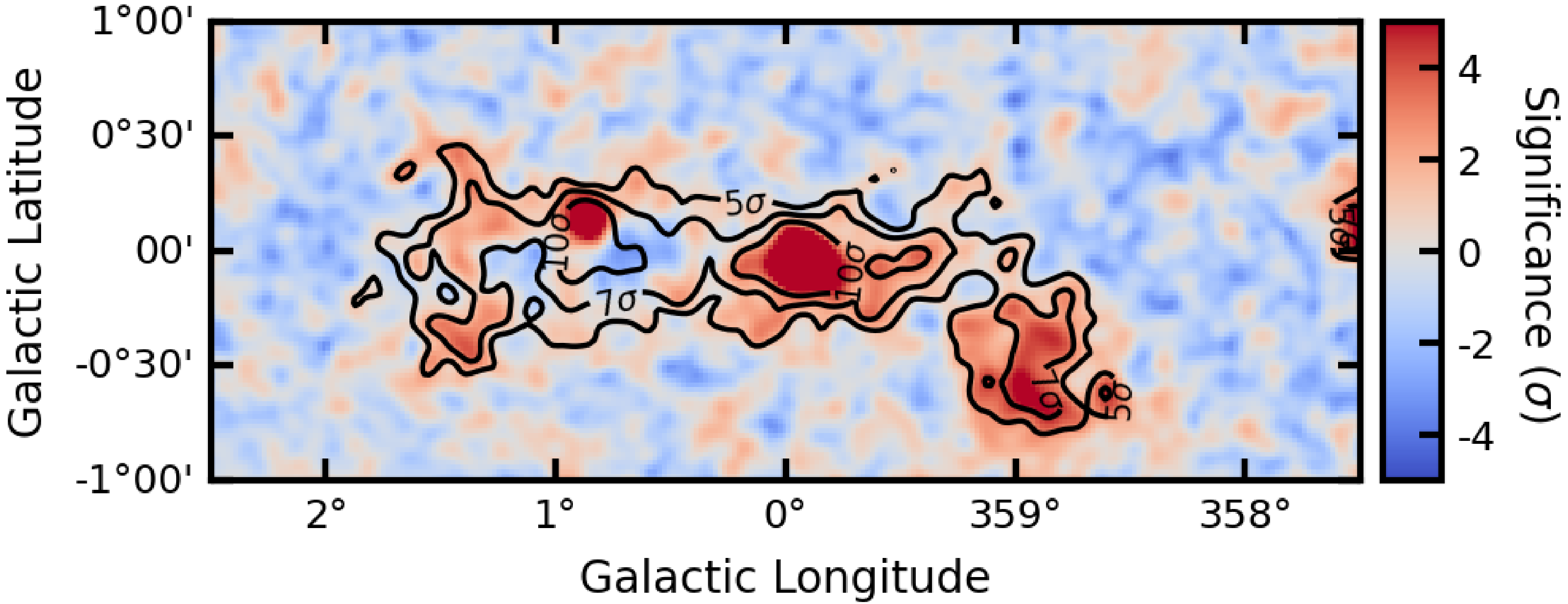
Models don't need to be analytical!



Fitting a complicated region

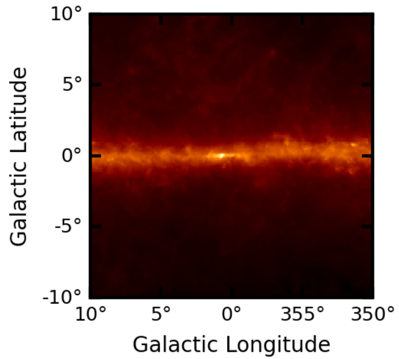


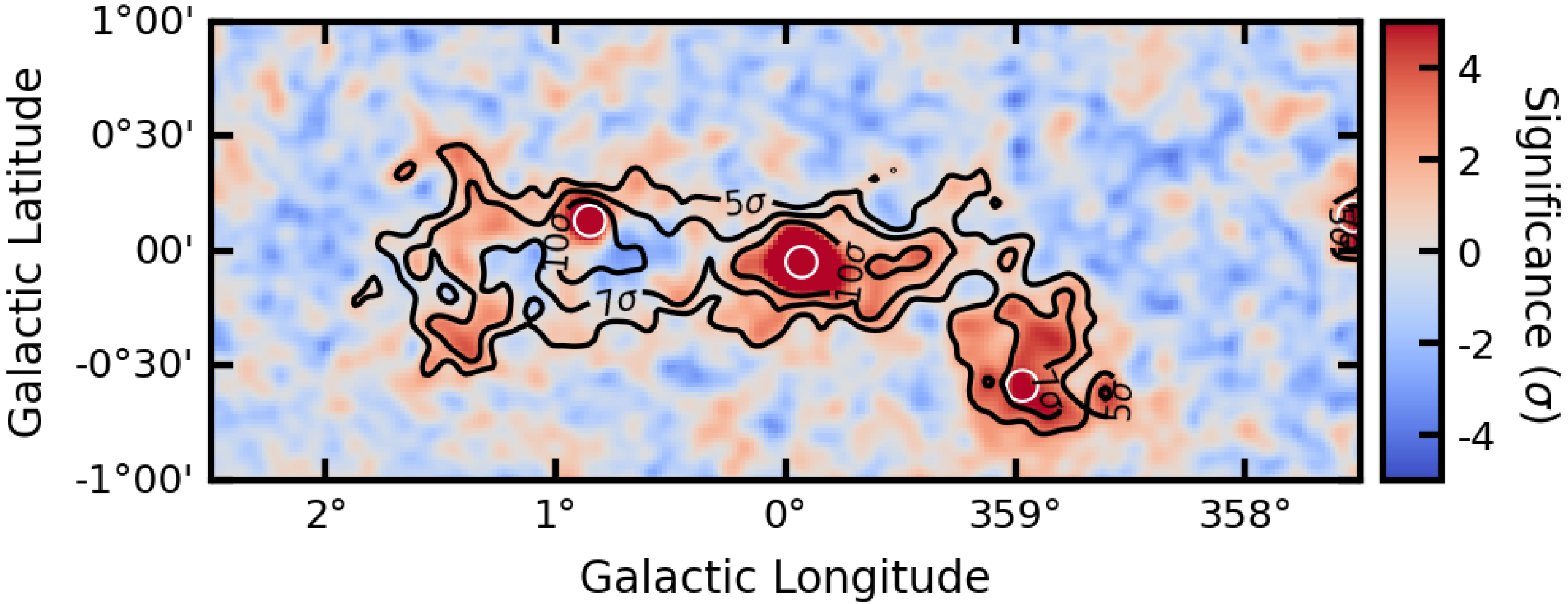
Models don't need to be analytical!



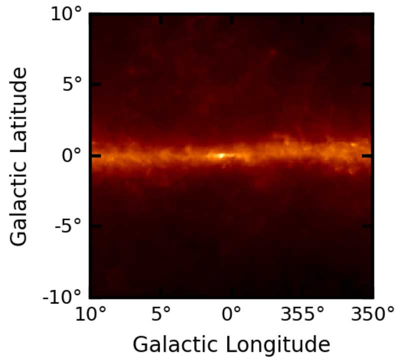
Fitting a complicated region

Models don't need to be analytical!

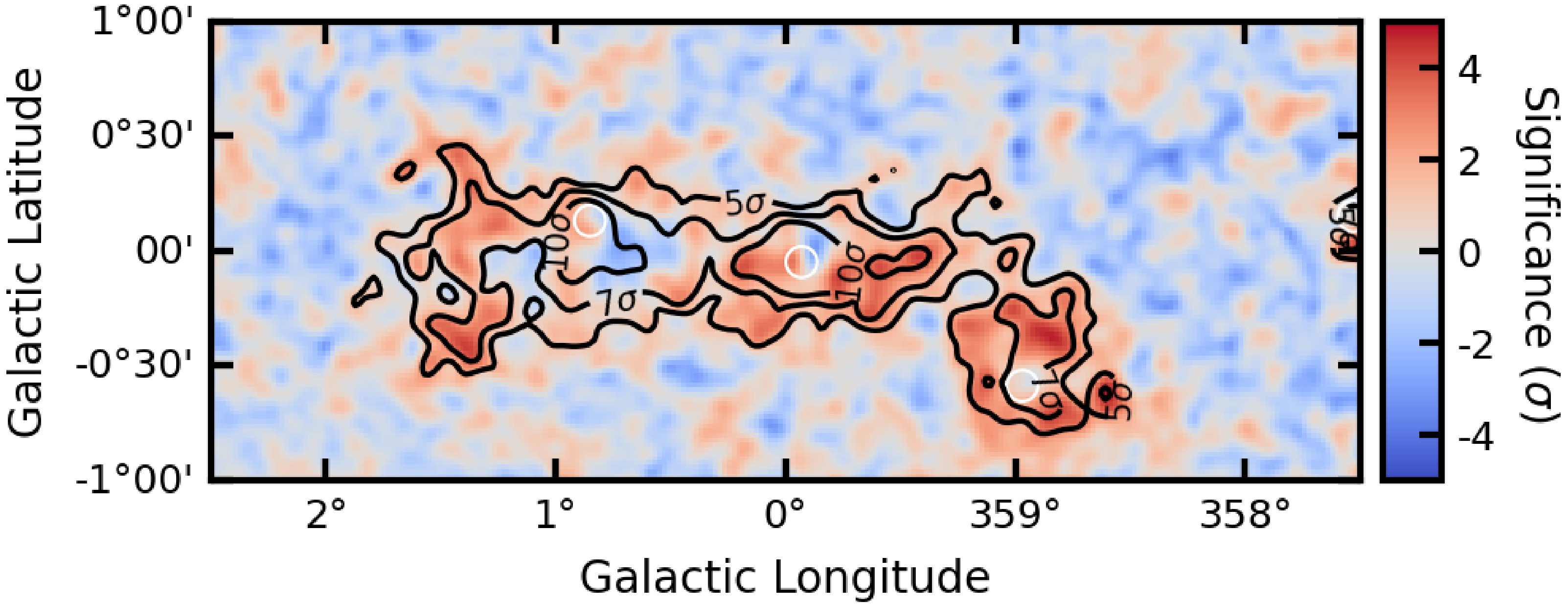




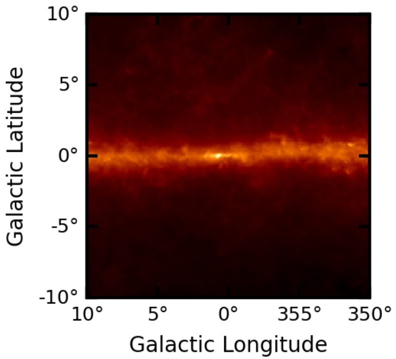
Fitting a complicated region



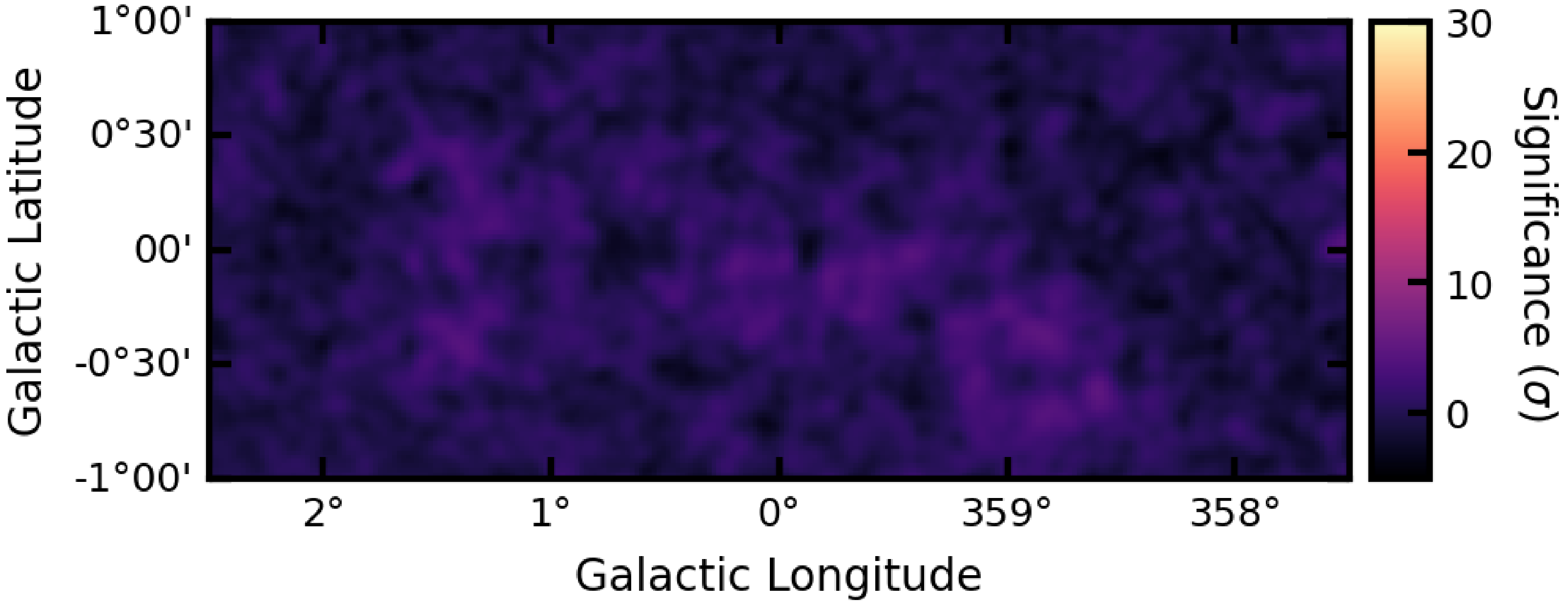
Models don't need to be analytical!



Fitting a complicated region



BE CAREFUL WITH YOUR COLORMAPS!



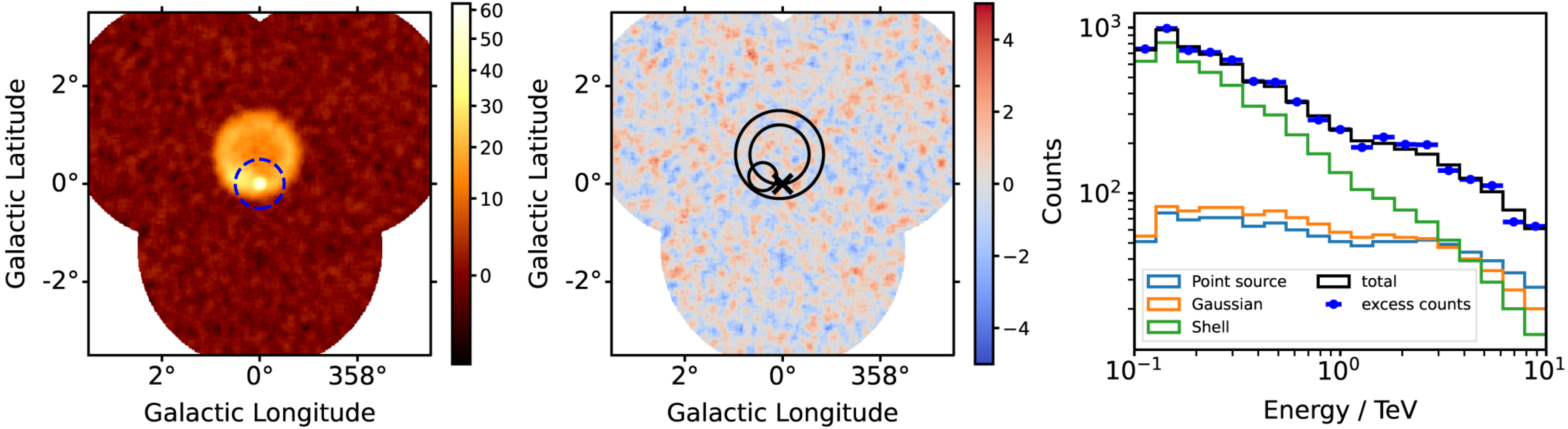
Power of 3D analysis

Can disentangle contributions of overlapping sources!

in this example there is a point source with power law spectrum

a Gaussian surce with log-parabola spectrum

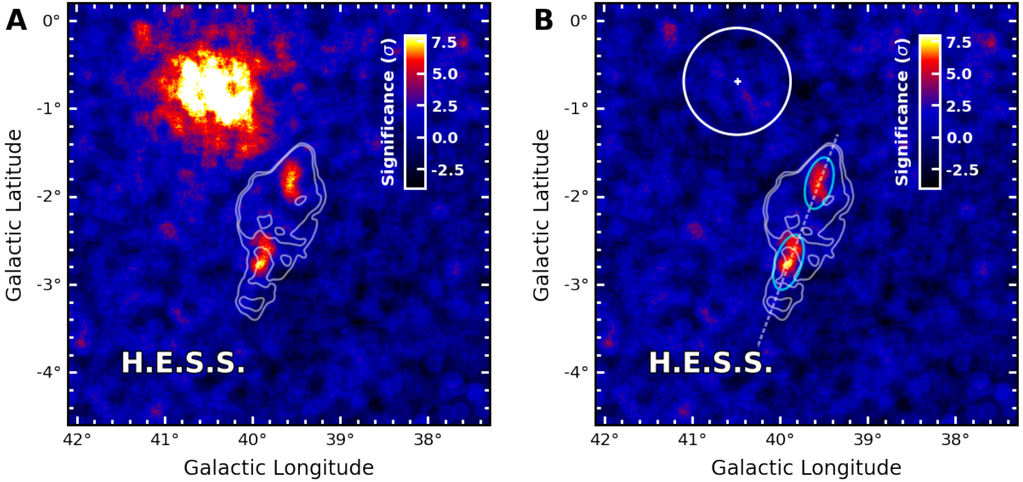
A shell with power law spectrumm

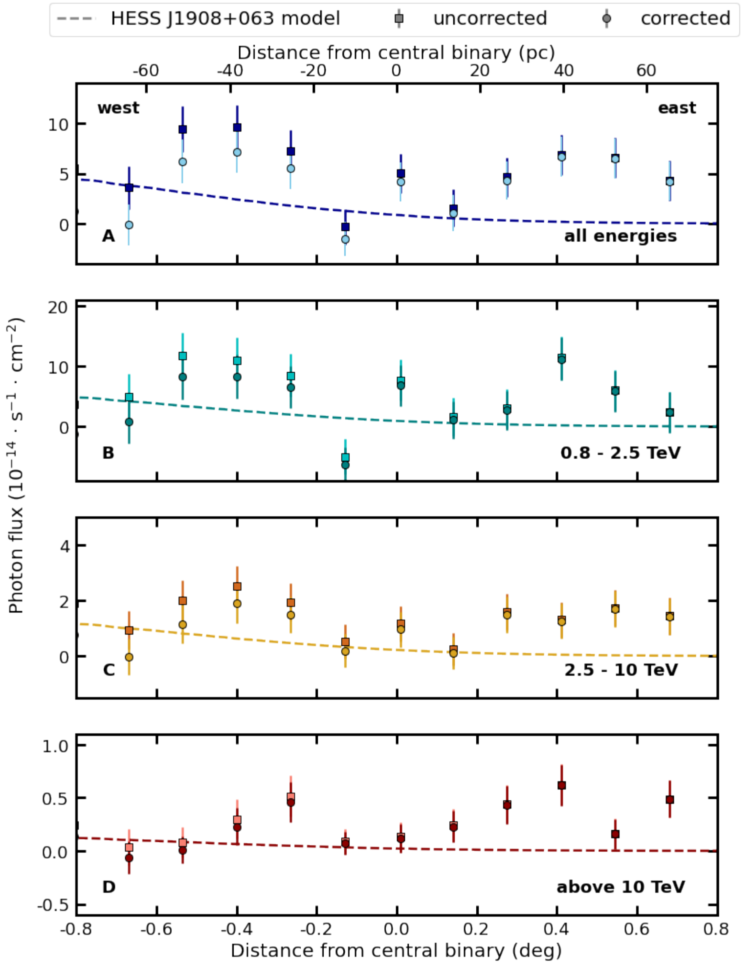


A. Donath et al

Also spatially!

H.E.S.S. Coll. 2024





LINK TO TUTORIAL

Data Analysis

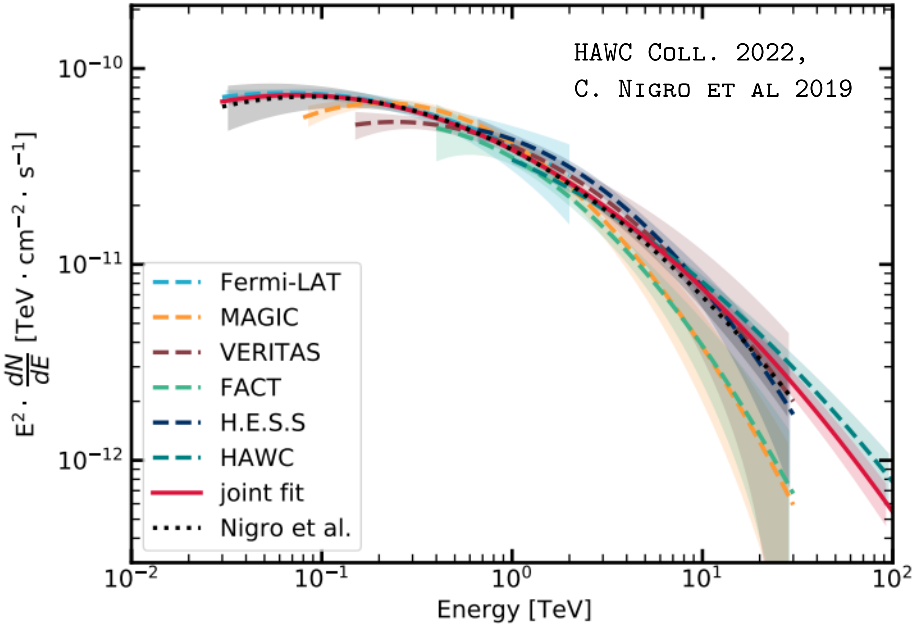
- Is there a source there?
- What are its properties? (spectral, spatial, temporal*)
- How to present results
- **Combining data from different instruments**
- Sources of uncertainty

*if we have time

Joint Analysis

Everything I showed you so far with one dataset can be done with a list of datasets

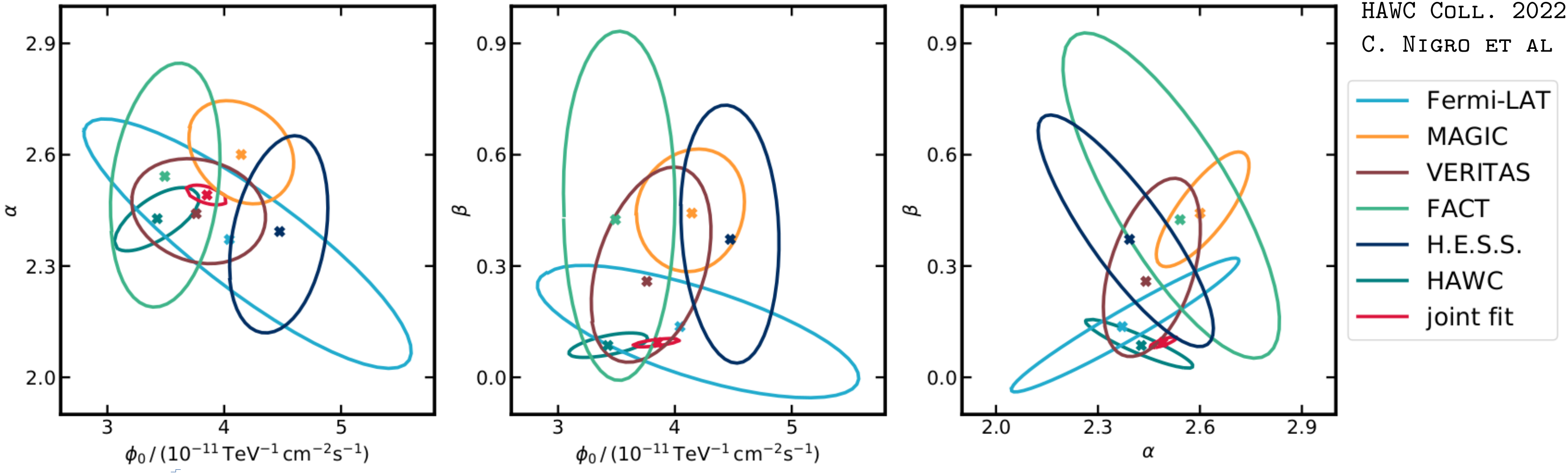
Gammapy doesn't care which instrument took the data in your dataset



Joint Analysis

<u>Everything</u> I showed you so far with one dataset can be done with a list of datasets

Gammapy doesn’t care which instrument took the data in your dataset



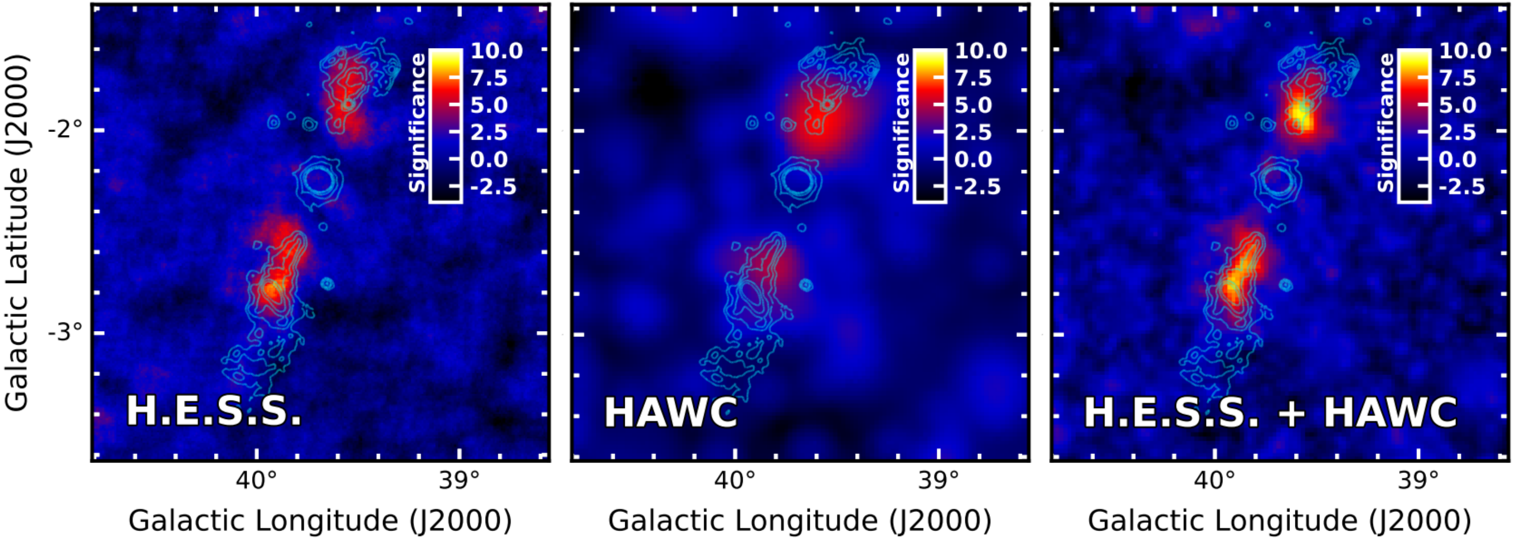
HAWC Coll. 2022,
C. Nigro et al 2019

Joint Analysis

Everything I showed you so far with one dataset can be done with a list of datasets

Gammapy doesn't care which instrument took the data in your dataset

From my PhD Thesis, preliminary



Data Analysis

- Is there a source there?
- What are its properties? (spectral, spatial, temporal*)
- How to present results
- Combining data from different instruments
- **Sources of uncertainty**

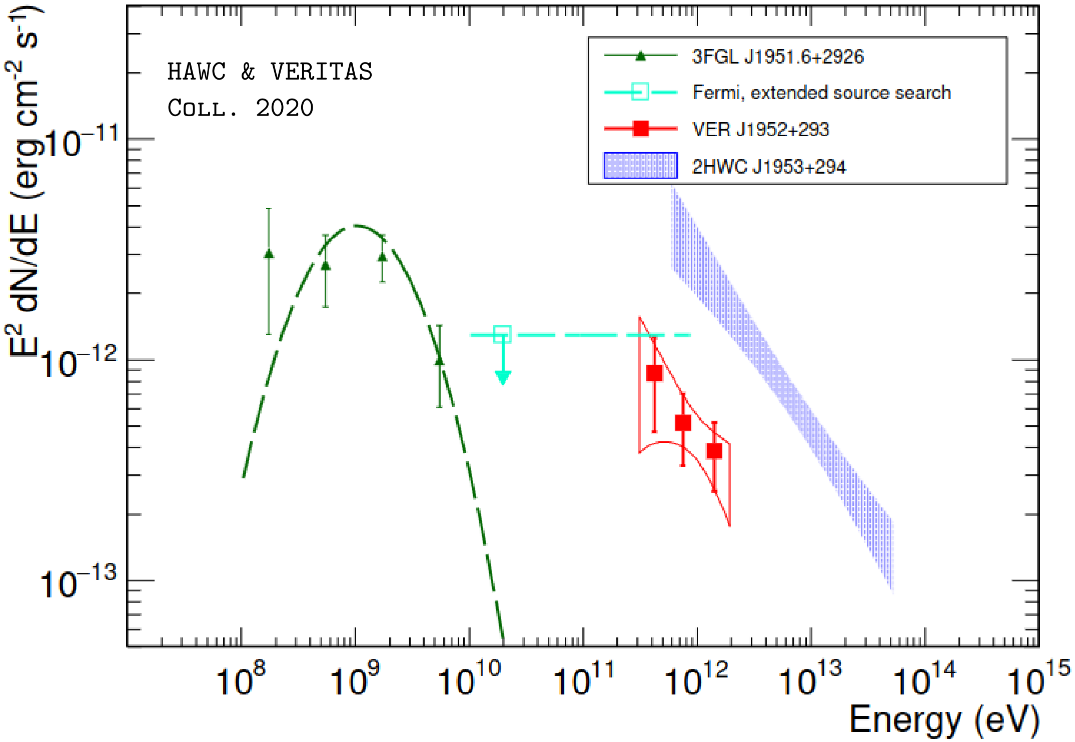
*if we have time

Systematic sources of uncertainty

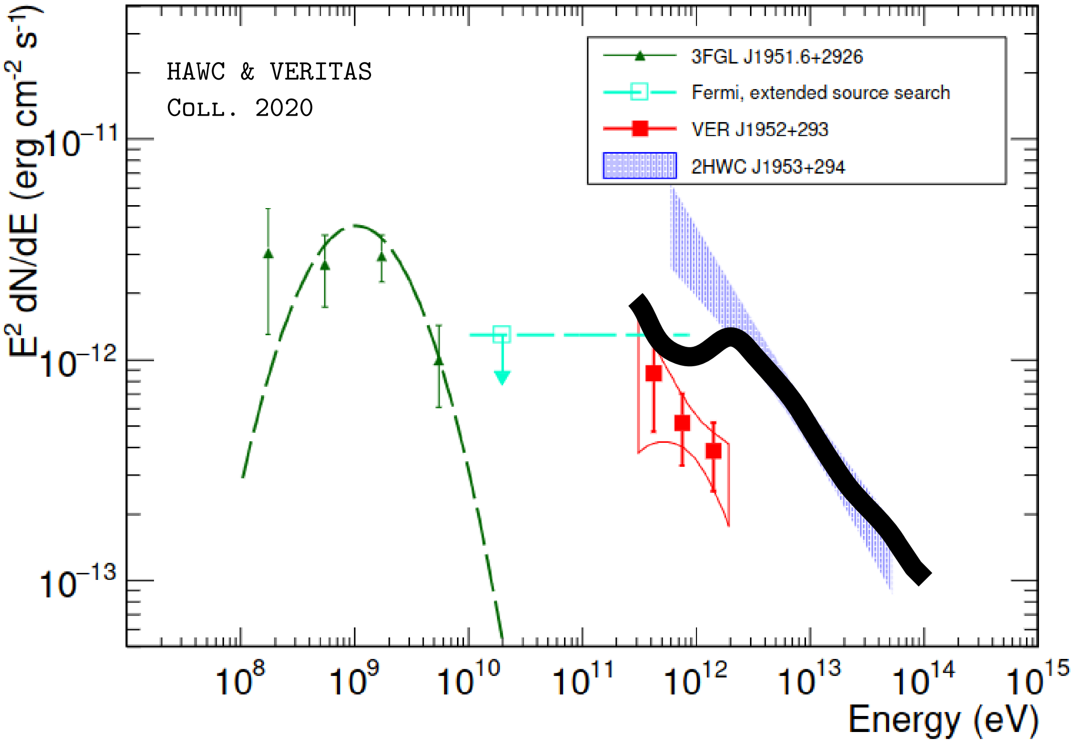
- Statistical uncertainty is reduced by taking more data
- Systematic uncertainty is **not**!
- One needs to estimate it and account for it → "reasonable guess"
- Different approaches, more instrument-dependent
- Think about things you might be getting wrong. What's their impact?
- E.g. "What if our IRFs are not right for the data?" → modify the IRFs randomly, repeat analysis and see how resulting parameters change
- Comparison with other instruments, previous results…

INCLUDE THEM IN YOUR ERROR BARS BEFORE MAKING CONCLUSIONS

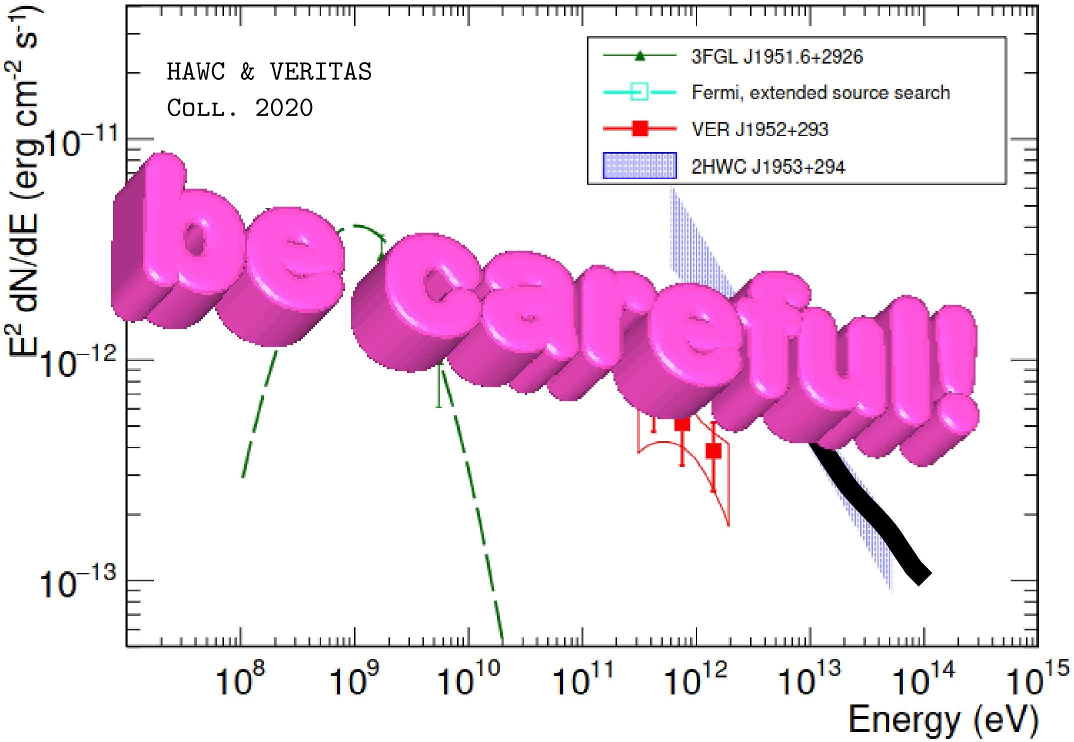
Systematic sources of uncertainty & Joint analyses



Systematic sources of uncertainty & Joint analyses



Systematic sources of uncertainty & Joint analyses

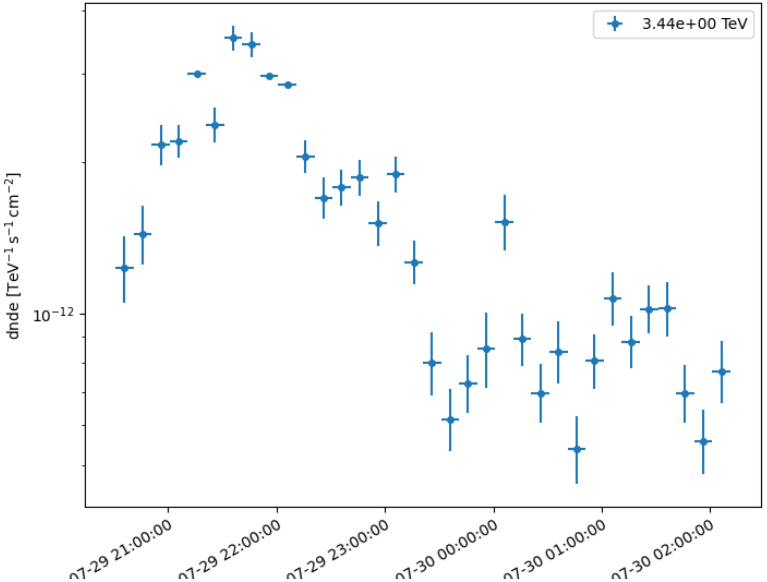


QUESTIONS

Conclusion (=random thoughts)

- A common data format allows for efficient data sharing, common tools and joint analysis
- Data/simulation consistency is the basis on which all of our analyses rest on
- Almost every high-level data product is produced with assumptions.
- Do not ignore systematics!!!!
- make as many sanity checks and diagnostic plots as you can, be careful with visualization!

Extra – Lightcurves



LINK TO TUTORIAL

- Basically the same thing except you can bin your data in time
- Fit normalizations to get flux variations
- Gammapy allows binning in times smaller than an observation run!

Subtleties – Stacked vs joint

"stacking": adding up counts, background, combining weighted IRFs of multiple observations into one Gammapy dataset

"joint": fitting a list of datasets containing 1 per observation

IACT analysis with 100s of runs need to stack somehow (too slow otherwise)

HAWC datasets with the different image size bins should not be combined

In short: only do it if the IRFs are similar enough