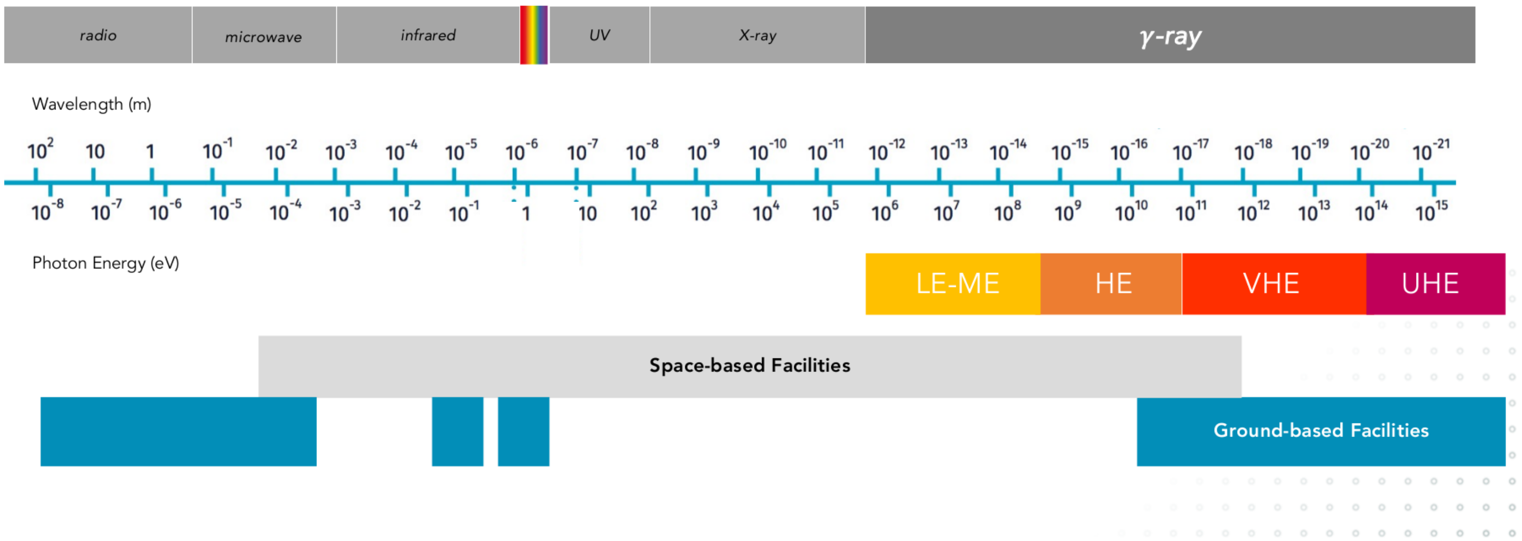
Data Analysis & Software tools for Gamma-ray astronomy

CDY-MPIK Summer school

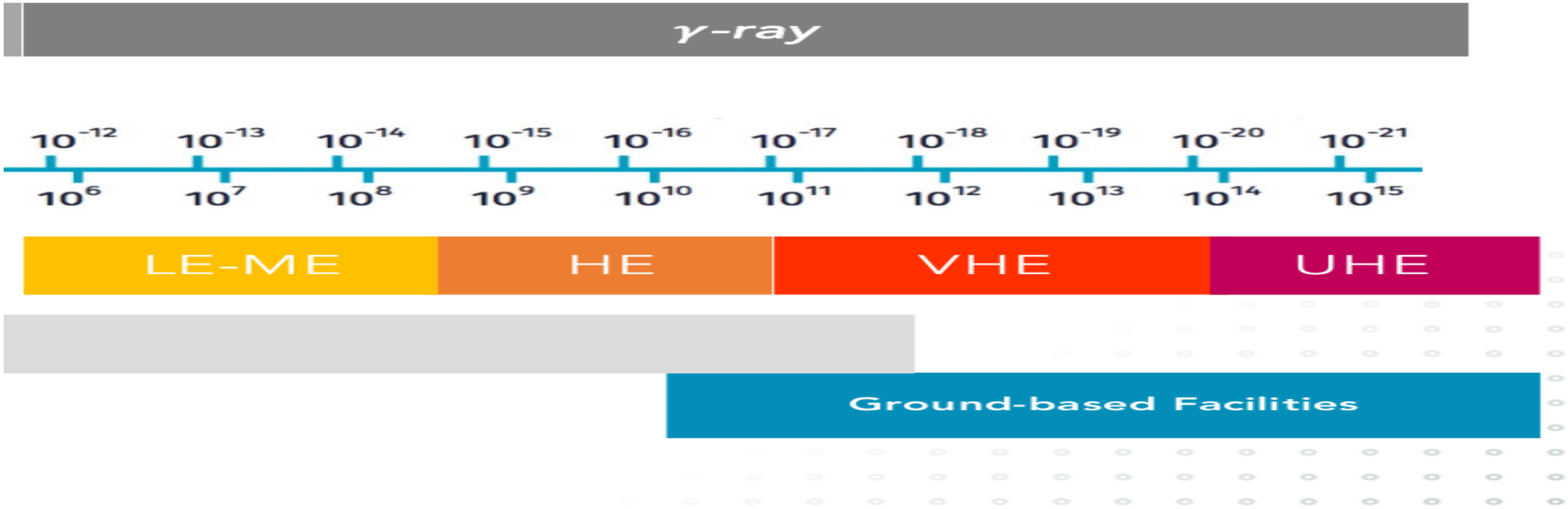
Heidelberg, June 2024

Laura Olivera-Nieto

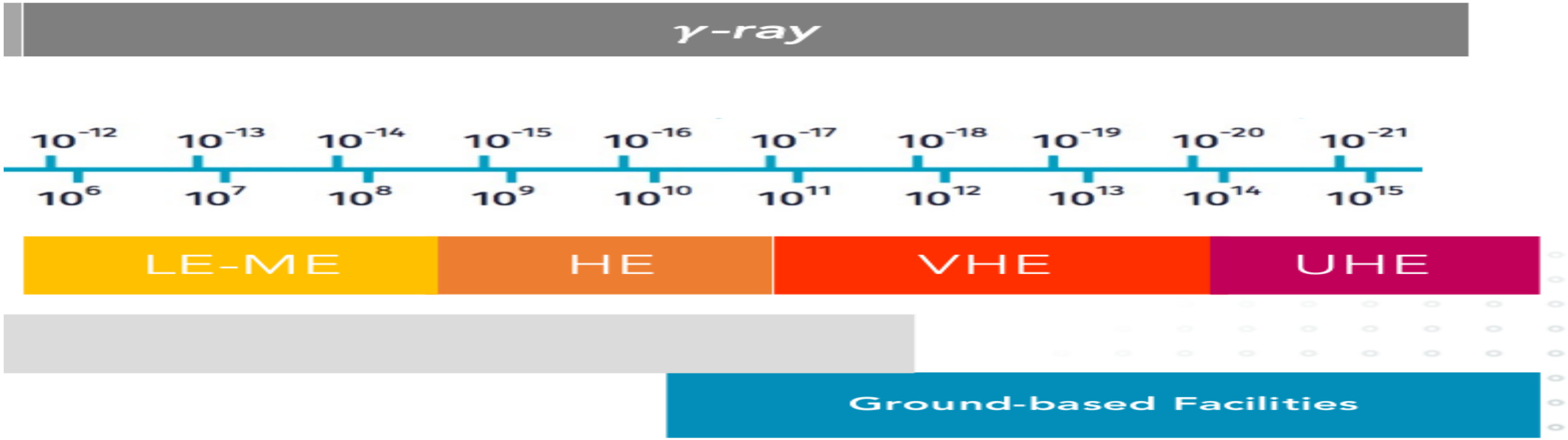
Gamma-ray astronomy and this talk

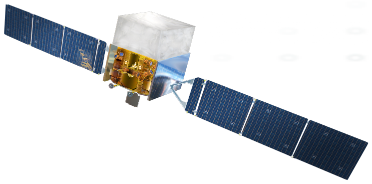


Gamma-ray astronomy and this talk



Gamma-ray astronomy and this talk





Satellites

IACTs

WCD arrays

Gamma-ray astronomy and this talk

- Very hard to cover all topics/existing software in 90 minutes
- 3 different types of instrument, subtleties and technicalities specific to each (as you have seen already!)
- In this talk I will instead try to focus on the **<u>things in common</u>**
- For that I will rely heavily on Gammapy

  A **Python** package for **gamma-ray** astronomy
- Not the only existing package of course but the only one dedicated to gamma-ray as a whole

Data Analysis

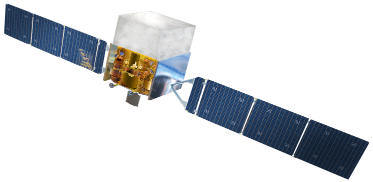
DATA ANALYSIS

DATA ANALYSIS

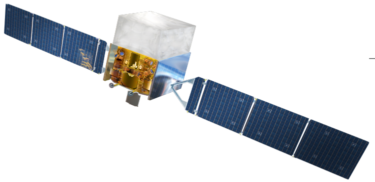
- WHAT DOES THE DATA OF GAMMA-RAY INSTRUMENTS LOOK LIKE?
- WHAT DO I NEED TO GO FROM DATA TO PHYSICAL QUANTITIES?
- HOW TO DETERMINE THE VALIDITY OF DATA?
- WHAT IS "DATA REDUCTION"?

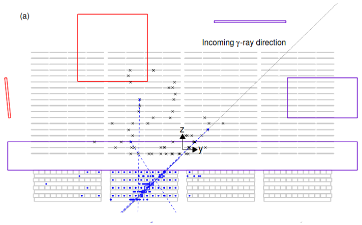
DATA ANALYSIS

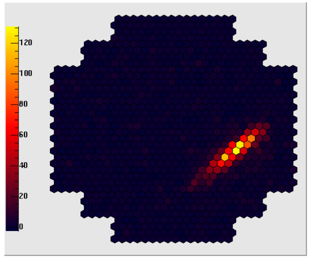
- **WHAT DOES THE DATA OF GAMMA-RAY INSTRUMENTS LOOK LIKE?**
- WHAT DO I NEED TO GO FROM DATA TO PHYSICAL QUANTITIES?
- HOW TO DETERMINE THE VALIDITY OF DATA?
- WHAT IS "DATA REDUCTION"?

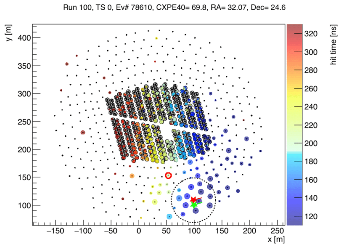


Data-taking

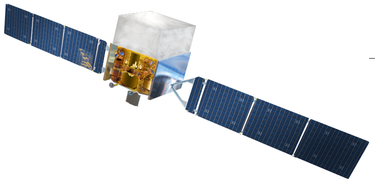


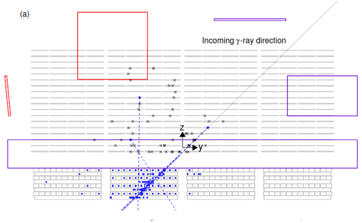

(a)
Incoming γ-ray direction
z
y

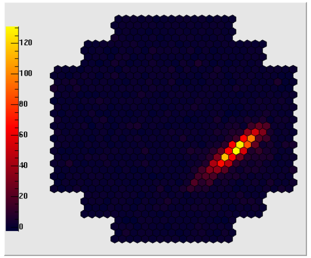

120
100
80
60
40
20
0

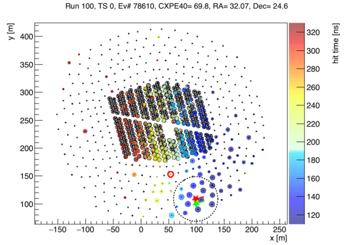

Run 100, TS 0, Ev# 78610, CXPE40= 69.8, RA= 32.07, Dec= 24.6
y [m]
x [m]
hit time [ns]

Data-taking Reconstruction









EVENT ID

ENERGY

DIRECTION

TIME

"GAMMANESS"

Real Data from a gamma-ray instrument

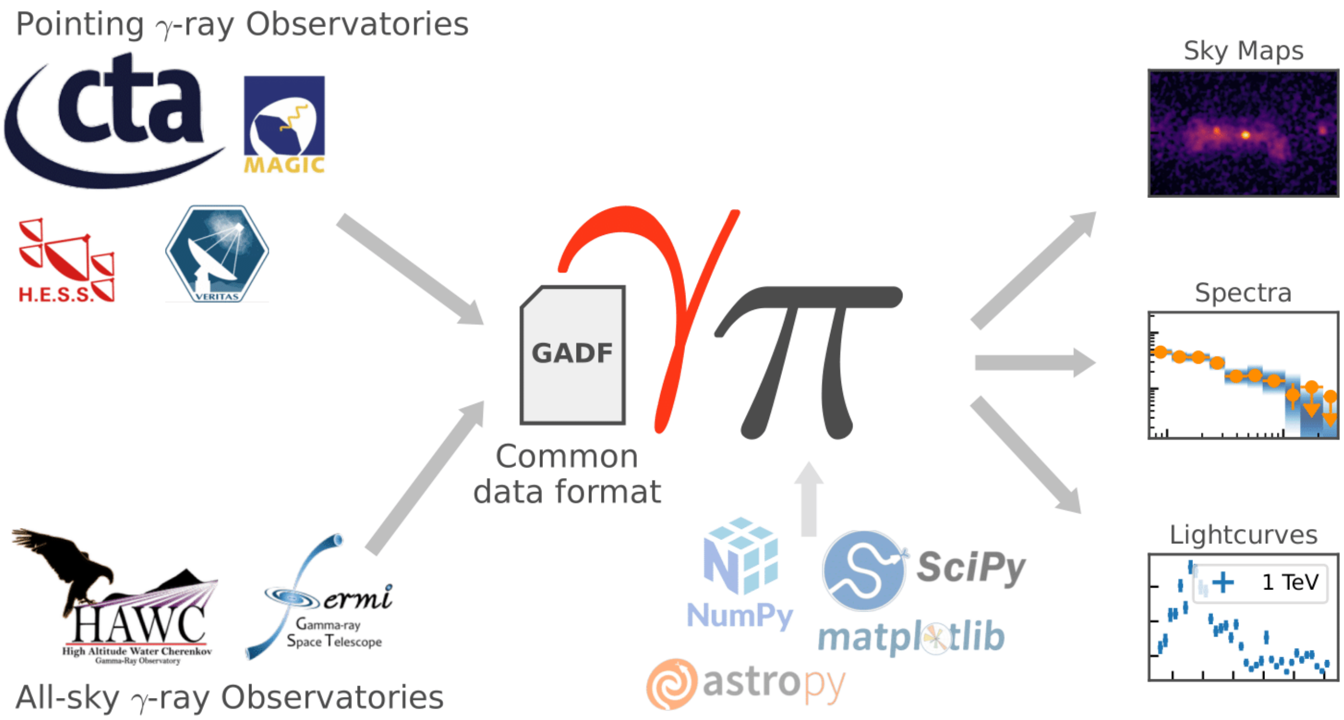
EVENT_ID	TIME	RA	DEC	ENERGY
	s	deg	deg	TeV
int64	float64	float32	float32	float32
18184891531583	139533040.89130569	284.92114	2.3147857	8.494793
18210661335308	139533064.76603198	284.68036	2.1020594	1.2389338
18214956302833	139533070.2015779	284.57544	2.641703	3.212592
18223546236968	139533075.5970521	285.67664	1.9790567	0.50208473
18223546237257	139533077.386261	286.9335	1.7020291	19.793955
18227841204705	139533082.1508355	285.97955	1.0685997	1.273117
18245021073483	139533096.00451803	284.84094	3.5419352	25.999714
18257905975463	139533108.47312307	283.92752	2.3060772	0.81107205
18262200943041	139533114.4788592	285.01868	4.477947	0.6513306
18270790877414	139533121.14376855	284.47827	4.6750755	0.5971853
18296560681602	139533147.5147109	284.61392	3.4823265	0.76477724
18300855648487	139533149.4298861	286.3041	2.4164367	0.96899956
18313740550377	139533161.2780261	284.2054	3.1820278	0.5589061
18322330485363	139533171.584491	284.23578	1.9863799	0.83051723
18335215387058	139533182.66867948	288.14926	1.5052232	1.3813052
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Can you tell me which one?

A unified format and tool for gamma-ray astronomy

- In the past decade or so, there has been a big effort to define a standard format to use when storing gamma-ray astronomy data
- By format I mean which columns/quantity names and so on
- Learned from existing standards, such as x-ray data and Fermi
- Result: [Gamma-astro-data-format]
- If all data looks the same, we can all share a tool!
- Result: Gammapy

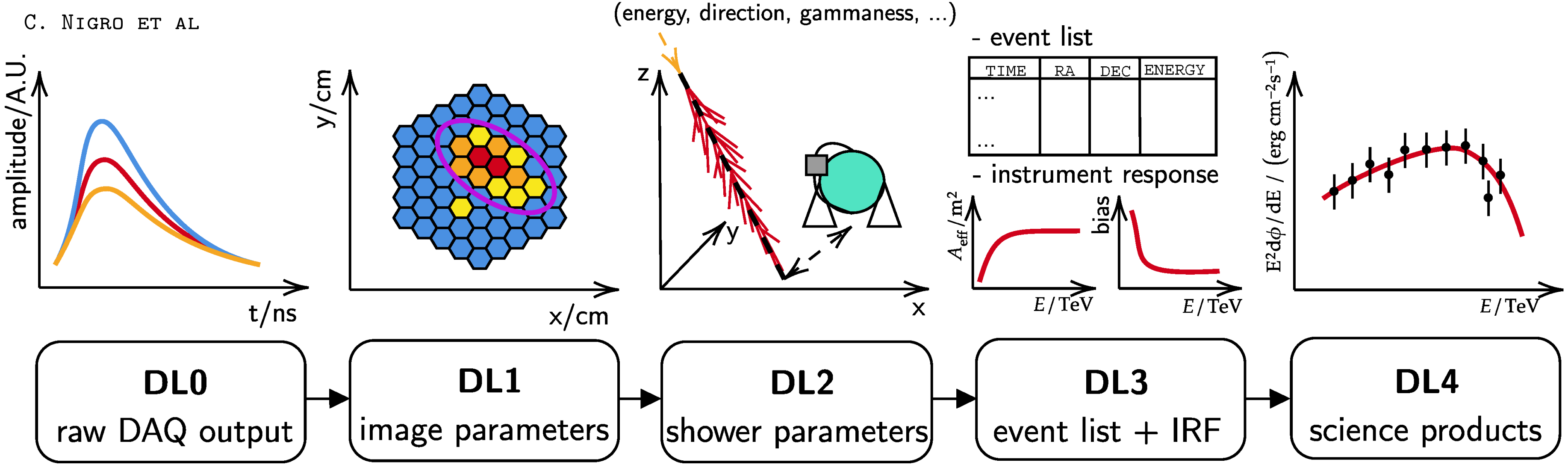
A UNIFIED FORMAT AND TOOL FOR GAMMA-RAY ASTRONOMY



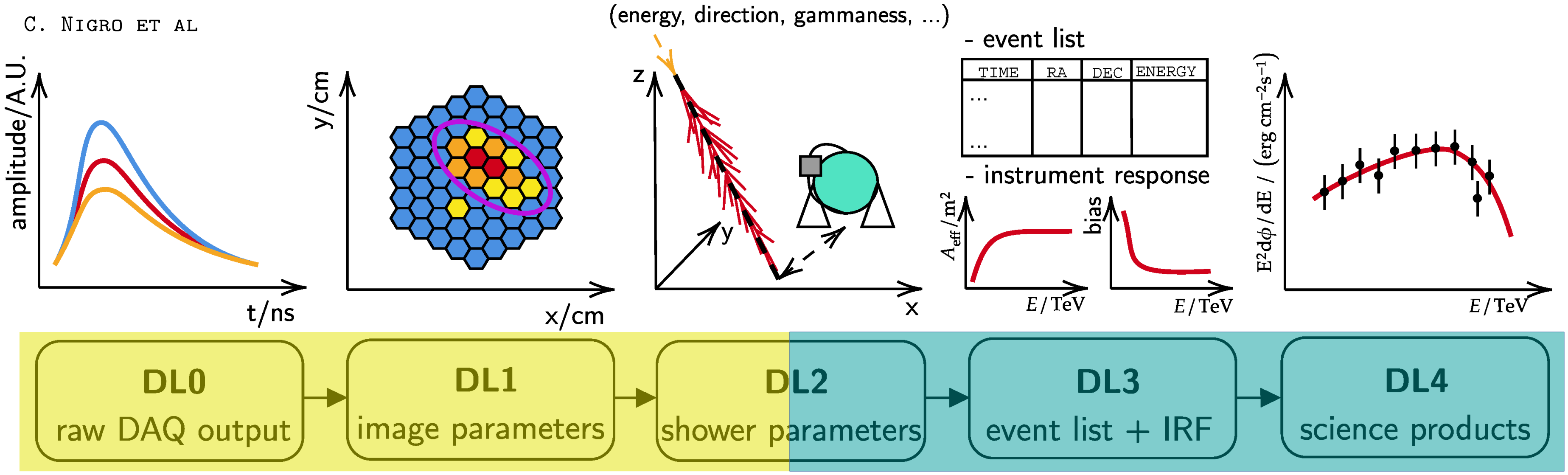
A. DONATH ET AL

A unified format and tool for gamma-ray astronomy

C. Nigro et al



A unified format and tool for gamma-ray astronomy



HIGHLY INSTRUMENT SPECIFIC **INSTRUMENT AGNOSTIC**

Event List

List of gamma-like events

EVENT_ID	TIME	RA	DEC	ENERGY
	s	deg	deg	TeV
int64	float64	float32	float32	float32
18184891531583	139533040.89130569	284.92114	2.3147857	8.494793
18210661335308	139533064.76603198	284.68036	2.1020594	1.2389338
18214956302833	139533070.2015779	284.57544	2.641703	3.212592
18223546236968	139533075.5970521	285.67664	1.9790567	0.50208473
18223546237257	139533077.386261	286.9335	1.7020291	19.793955
18227841204705	139533082.1508355	285.97955	1.0685997	1.273117
18245021073483	139533096.00451803	284.84094	3.5419352	25.999714
18257905975463	139533108.47312307	283.92752	2.3060772	0.81107205
18262200943041	139533114.4788592	285.01868	4.477947	0.6513306
18270790877414	139533121.14376855	284.47827	4.6750755	0.5971853
18296560681602	139533147.5147109	284.61392	3.4823265	0.76477724
18300855648487	139533149.4298861	286.3041	2.4164367	0.96899956
18313740550377	139533161.2780261	284.2054	3.1820278	0.5589061
18322330485363	139533171.584491	284.23578	1.9863799	0.83051723
18335215387058	139533182.66867948	288.14926	1.5052232	1.3813052

Event List

List of gamma-like events

EVENT_ID	TIME	RA	DEC	ENERGY
	s	deg	deg	TeV
int64	float64	float32	float32	float32
18184891531583	139533040.89130569	284.92114	2.3147857	8.494793
18210661335308	139533064.76603198	284.68036	2.1020594	1.2389338
18214956302833	139533070.2015779	284.57544	2.641703	3.212592
18223546236968	139533075.5970521	285.67664	1.9790567	0.50208473
18223546237257	139533077.386261	286.9335	1.7020291	19.793955
18227841204705	139533082.1508355	285.97955	1.0685997	1.273117
18245021073483	139533096.00451803	284.84094	3.5419352	25.999714
18257905975463	139533108.47312307	283.92752	2.3060772	0.81107205
18262200943041	139533114.4788592	285.01868	4.477947	0.6513306
18270790877414	139533121.14376855	284.47827	4.6750755	0.5971853
18296560681602	139533147.5147109	284.61392	3.4823265	0.76477724
18300855648487	139533149.4298861	286.3041	2.4164367	0.96899956
18313740550377	139533161.2780261	284.2054	3.1820278	0.5589061
18322330485363	139533171.584491	284.23578	1.9863799	0.83051723
18335215387058	139533182.66867948	288.14926	1.5052232	1.3813052
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Good Time Intervals (GTI)

Time ranges when the instrument was taking data

START	STOP
Time	Time
53524.96611324074	53524.985685

Event List

List of gamma-like events

EVENT_ID	TIME	RA	DEC	ENERGY
	s	deg	deg	TeV
int64	float64	float32	float32	float32
18184891531583	139533040.89130569	284.92114	2.3147857	8.494793
18210661335308	139533064.76603198	284.68036	2.1020594	1.2389338
18214956302833	139533070.2015779	284.57544	2.641703	3.212592
18223546236968	139533075.5970521	285.67664	1.9790567	0.50208473
18223546237257	139533077.386261	286.9335	1.7020291	19.793955
18227841204705	139533082.1508355	285.97955	1.0685997	1.273117
18245021073483	139533096.00451803	284.84094	3.5419352	25.999714
18257905975463	139533108.47312307	283.92752	2.3060772	0.81107205
18262200943041	139533114.4788592	285.01868	4.477947	0.6513306
18270790877414	139533121.14376855	284.47827	4.6750755	0.5971853
18296560681602	139533147.5147109	284.61392	3.4823265	0.76477724
18300855648487	139533149.4298861	286.3041	2.4164367	0.96899956
18313740550377	139533161.2780261	284.2054	3.1820278	0.5589061
18322330485363	139533171.584491	284.23578	1.9863799	0.83051723
18335215387058	139533182.66867948	288.14926	1.5052232	1.3813052

Good Time Intervals (GTI)

Time ranges when the instrument was taking data

START	STOP
Time	Time
53524.96611324074	53524.985685

These lists are prepared by the people from each instrument and represent "science-ready" data or "Data level 3"

Data Analysis

- What does the data of gamma-ray instruments look like?
- **What do I need to go from data to physical quantities?**
- How to determine the validity of data?
- What is "data reduction"?

From "events" to physics

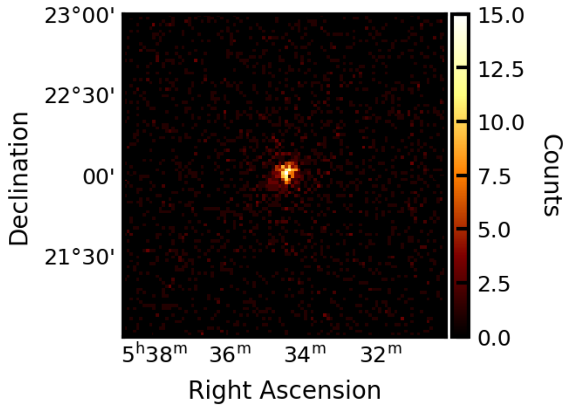
Let's say you have a list of events taken by a gamma-ray instrument when observing a source, and you want to study that source.

THIS IS DATA REDUCTION

From "events" to physics

Let's say you have a list of events taken by a gamma-ray instrument when observing a source, and you want to study that source.

You can make a map! Which is just a 2D histogram of the sky coordinates



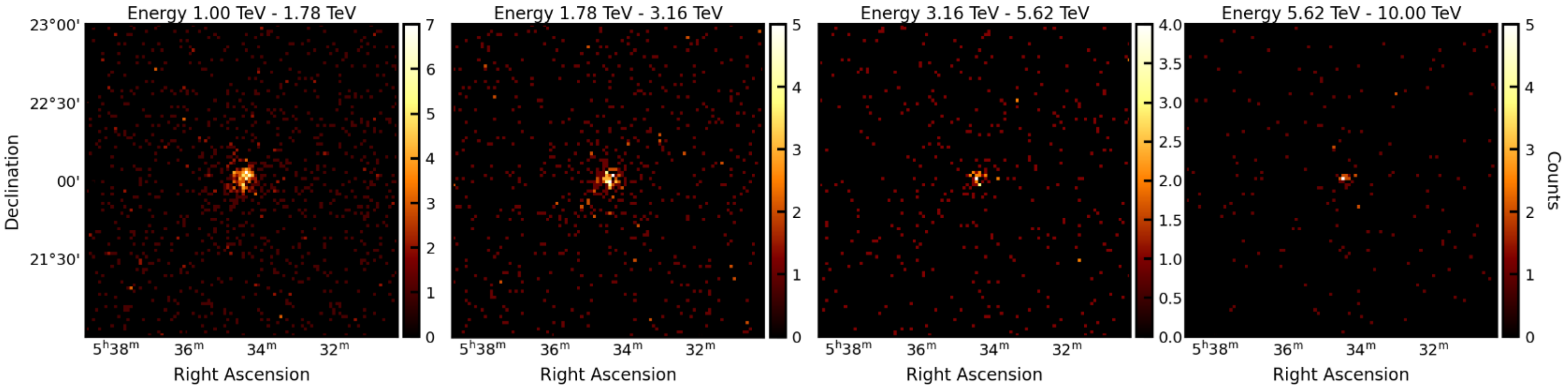

```
dataset.counts.reduce_over_axes().plot()
```

From "events" to physics

This is data reduction

Let's say you have a list of events taken by a gamma-ray instrument when observing a source, and you want to study that source.

You can make a map! Which is just a 3D histogram of the sky coordinates and energy*



`dataset.counts.plot_grid()`

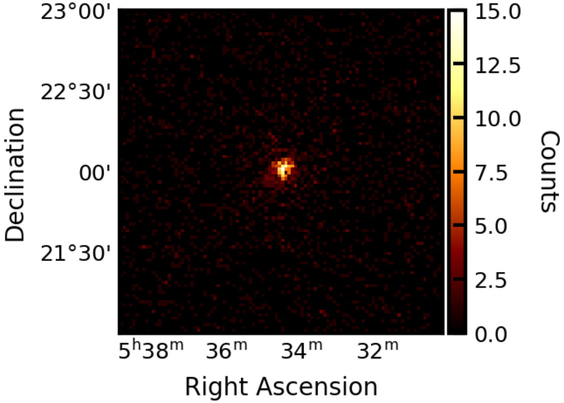
* or time, or some other quantity...

From "events" to physics

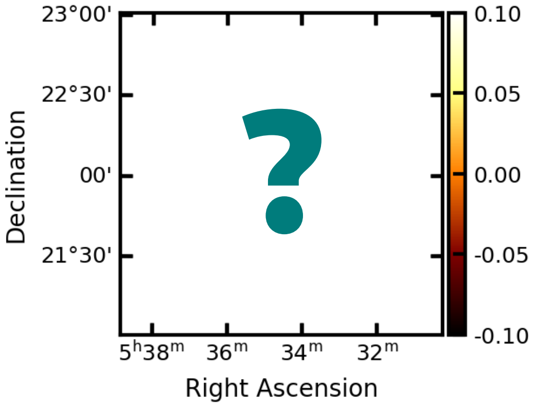
Let's say you have a list of events taken by a gamma-ray instrument when observing a source, and you want to study that source.

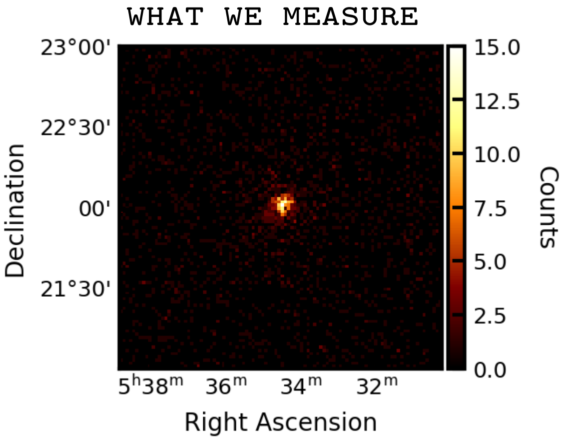
You can make a map! Which is just a 3D histogram of the sky coordinates and energy

What we measure

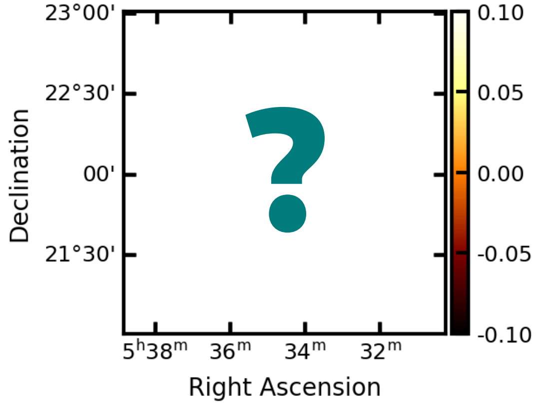


What is the "truth"?





What is the "truth"?



Measuring is not perfect! Instruments introduce biases, inaccuracies,...

- We measure X **counts** from the source → how **bright** is it?
- The counts "blob" has a spatial extent → what is the **actual size** of the source?
- We only see a source in the middle → but did we **observe** other parts of the map as much?
- The measured counts have an energy distribution → what is the **spectrum** of the source?

Events are not enough! We also need the **INSTRUMENT RESPONSE**

Instrument response functions

Gamma ray source $\frac{dN_\gamma}{dE_{true}\,dx_{true}}$ ⟷ "Gamma-like" events $\frac{dN_C}{dE_{reco}\,dx_{reco}}$

- How many of the arriving gamma-rays do we detect?
- How many do we miss-classify?
- How wrong do we get their energy?
- How wrong do we get their direction?
- How much background do we let through?

Instrument response functions

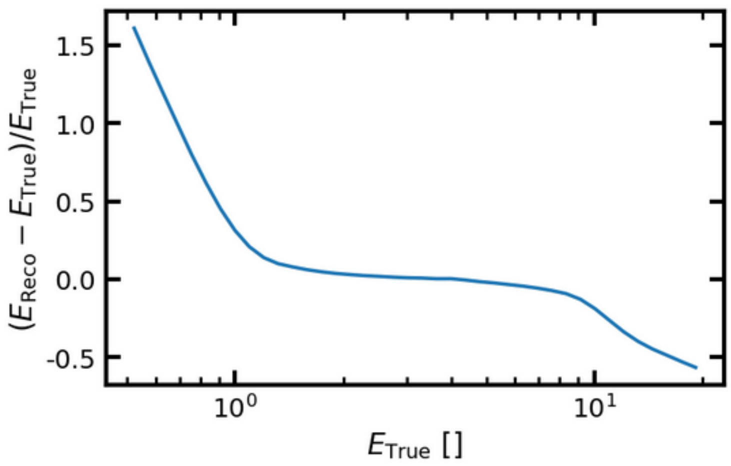
Gamma ray source $\frac{dN_\gamma}{dE_{true}\,dx_{true}}$ ⟷ "Gamma-like" events $\frac{dN_C}{dE_{reco}\,dx_{reco}}$

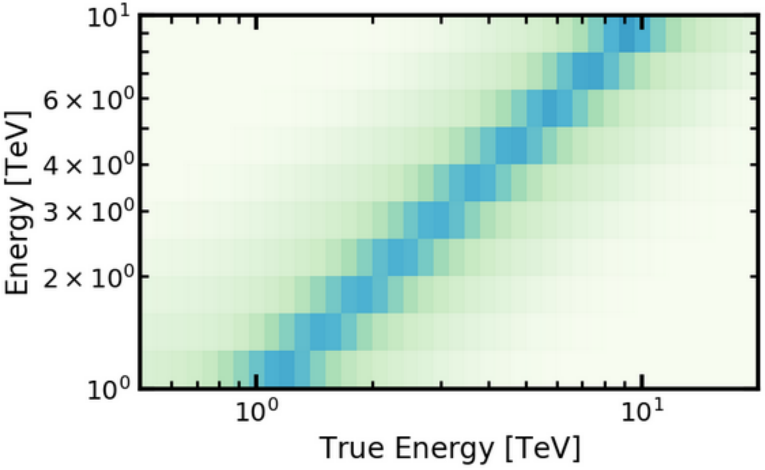
- How many of the arrivi… …tect?
- Ho…
- …rong do we get their energy?
- How wrong do we get their direction?
- How much background do we let through?

Energy dispersion

$EDISP(E_{true}, E_{reco})$

For each true gamma-ray energy, what is the probability that the event gets assigned a certain reconstructed energy?



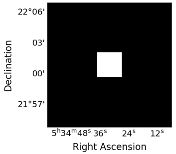


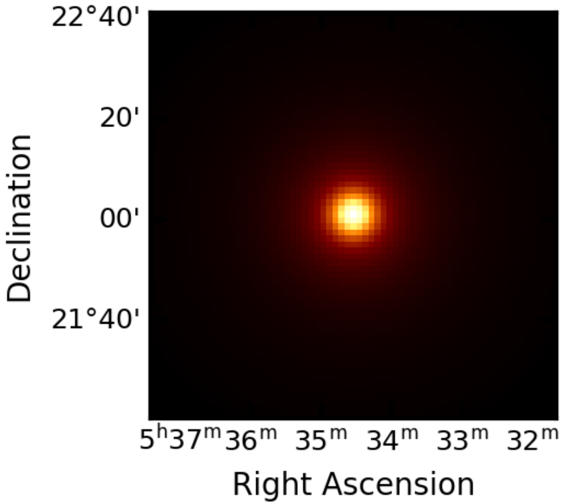
`DATASET.EDISP.PEEK()`

Point-Spread function

$$PSF(E_{true}, x_{reco}, x_{true})$$

For each true gamma-ray arriving direction, what is the probability that the event gets assigned a certain reconstructed direction ?



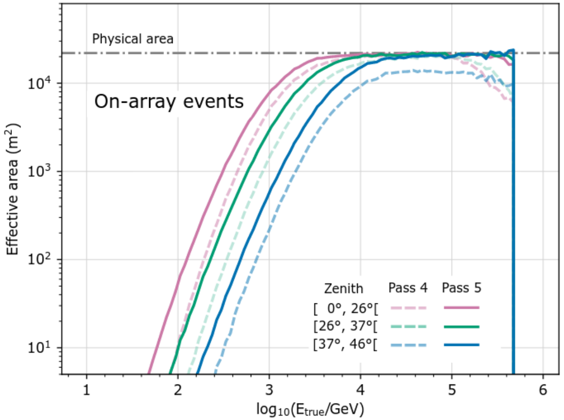


Effective Area

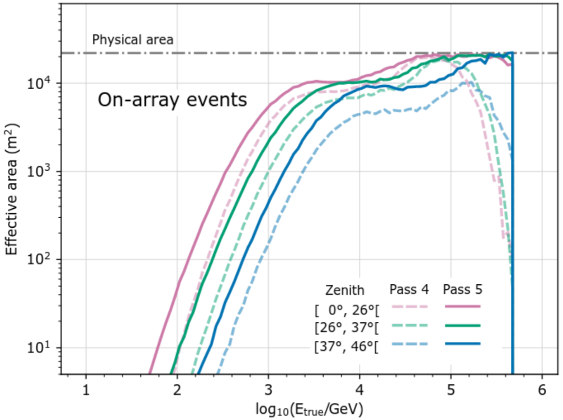
$$Aeff\left(E_{true}\right)$$

Detection probability of the gamma-ray (due to energy threshold + bad classification + instrumented area)

often multiplied by **livetime** to obtain "effective exposure" in units of m^2s



(a) Trigger conditions



(b) Gamma/hadron cuts

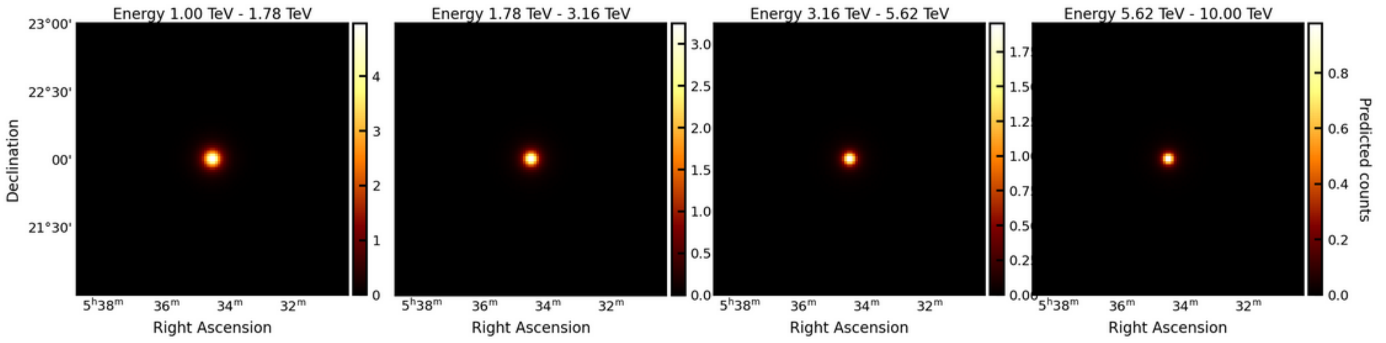
S. Yun-Cárcamo

So now we know how a source should look like!

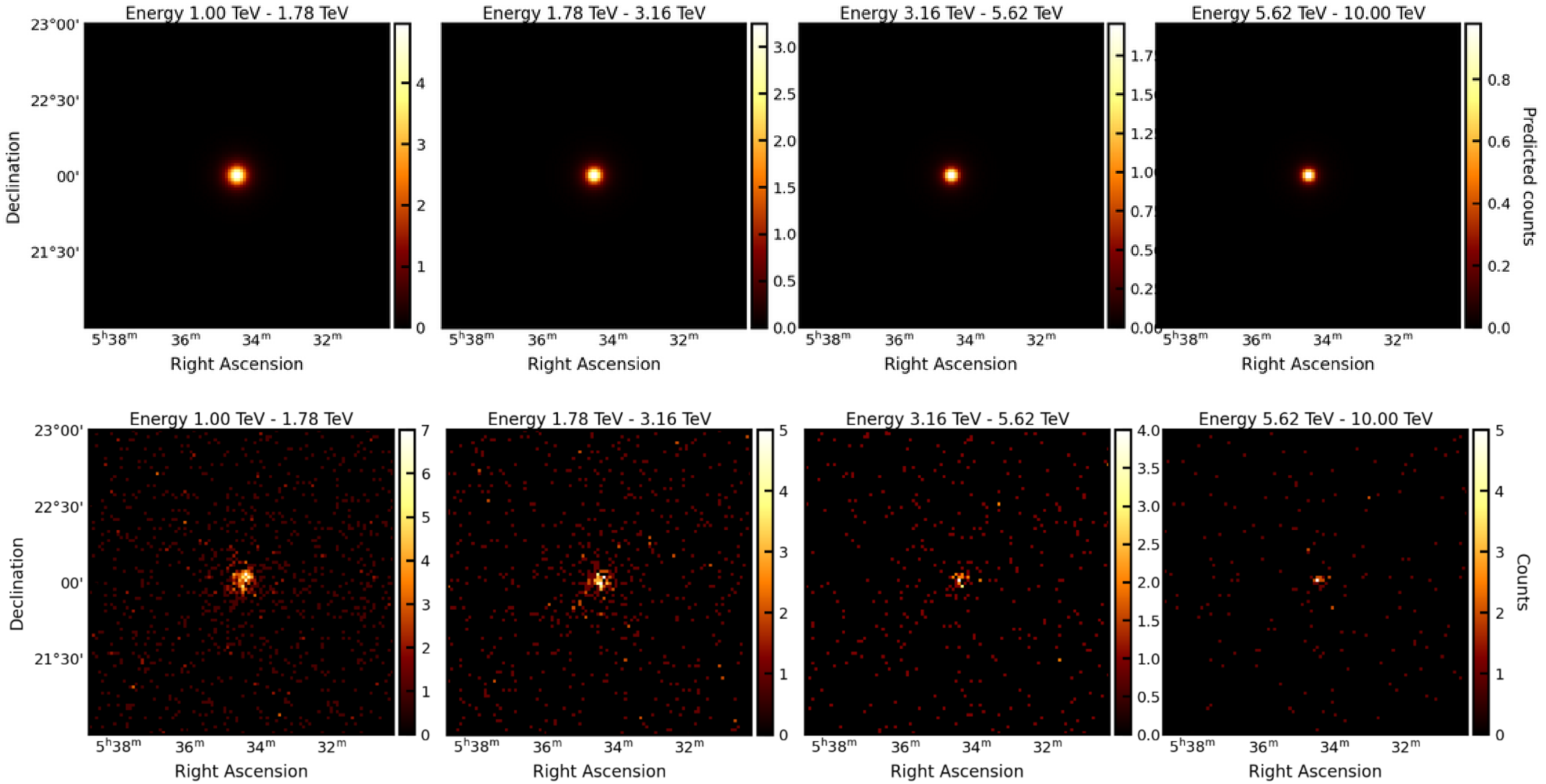
How would a **point source** with a **Crab like spectrum** look like in our detector?

- From the **spectrum** we know the flux (counts/s/TeV/cm2 in true energy!) $\left(\frac{dN_\gamma}{dE_{true}}(x_{true})\right)$
- With **PSF** we shift from true to reconstructed position
- With **AEFF** and **livetime** we go from flux to counts (counts/TeV in true energy!)
- With **EDISP** we shift from true to reconstructed energy (counts/TeV in reconstructed energy!)
- We now can predict what we would observe if the model was a good description of reality!

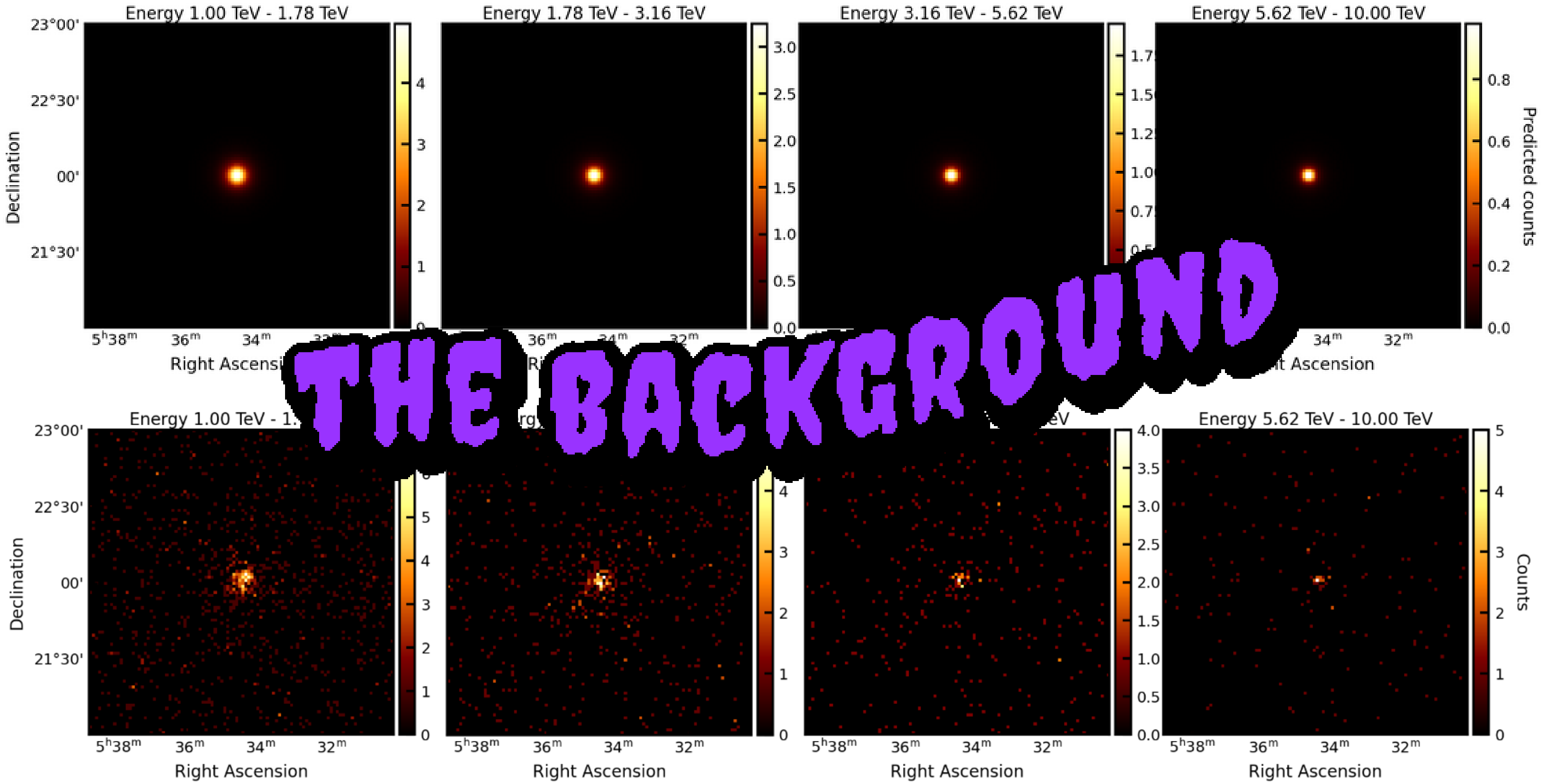
```
dataset.npred_signal().plot_grid()
```



Is that all?


Energy 1.00 TeV - 1.78 TeV
Energy 1.78 TeV - 3.16 TeV
Energy 3.16 TeV - 5.62 TeV
Energy 5.62 TeV - 10.00 TeV
Declination
Right Ascension
Predicted counts
Counts

Is that all?


Energy 1.00 TeV - 1.78 TeV
Energy 1.78 TeV - 3.16 TeV
Energy 3.16 TeV - 5.62 TeV
Energy 5.62 TeV - 10.00 TeV
Declination
Right Ascension
Predicted counts
Counts
THE BACKGROUND

Background

$N_{bkg}(E_{reco})$

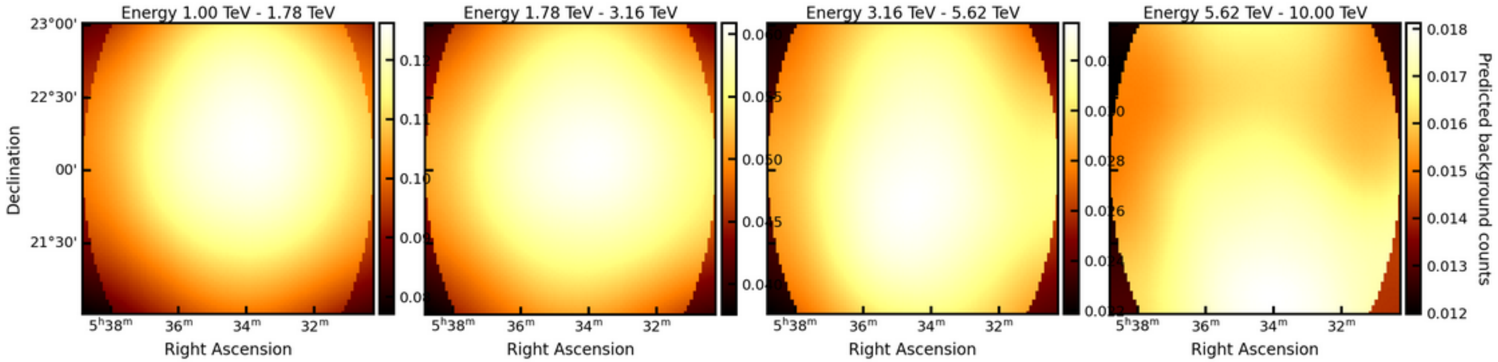
See Mohrmann et al 2019 to see how IACTs "caught up" with the other instruments ;-)

How many background events do we expect to mis-classify as signal?

A little bit different: usually derived from data of regions with no sources

- "Easy" in wide-field instruments, as the whole sky is rather empty in the TeV range
- Challenge for pointed instruments with small field of view: usually need to **assume where you expect your source to be (bias!!!)**

`dataset.background.plot_grid()`



Background

$N_{bkg}(E_{reco})$

See Mohrmann et al 2019 to see how IACTs "caught up" with the other instruments ;-)

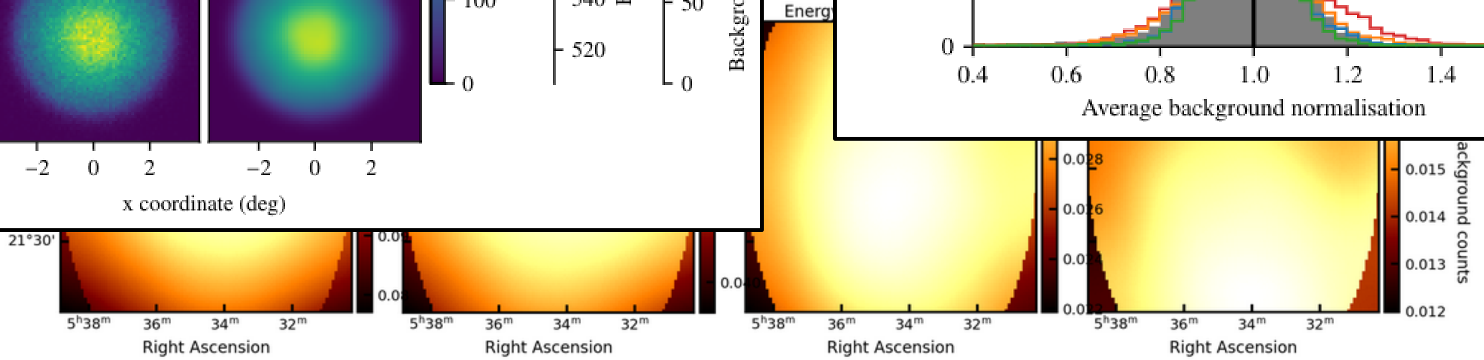
How many background events do we expect to ... y as signal?

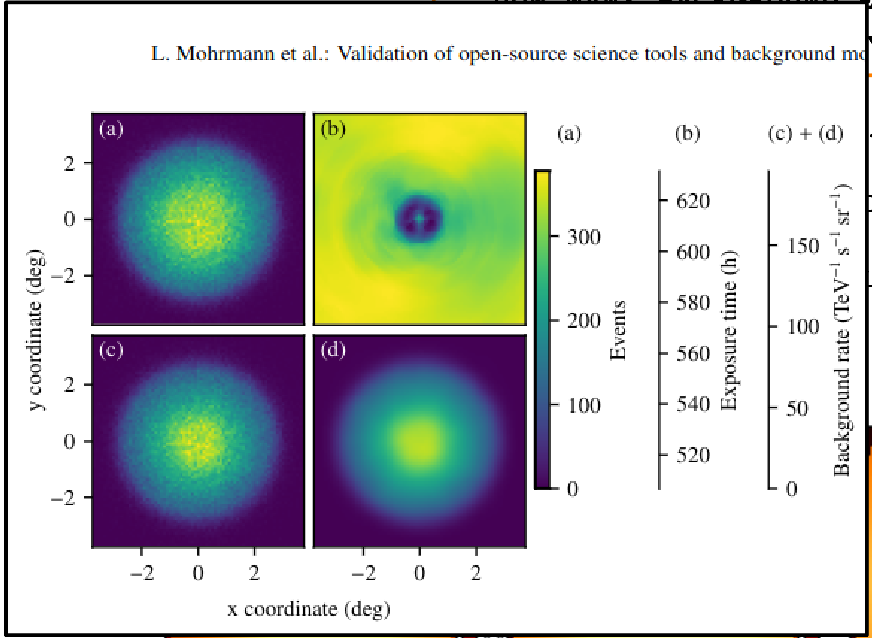
... ata

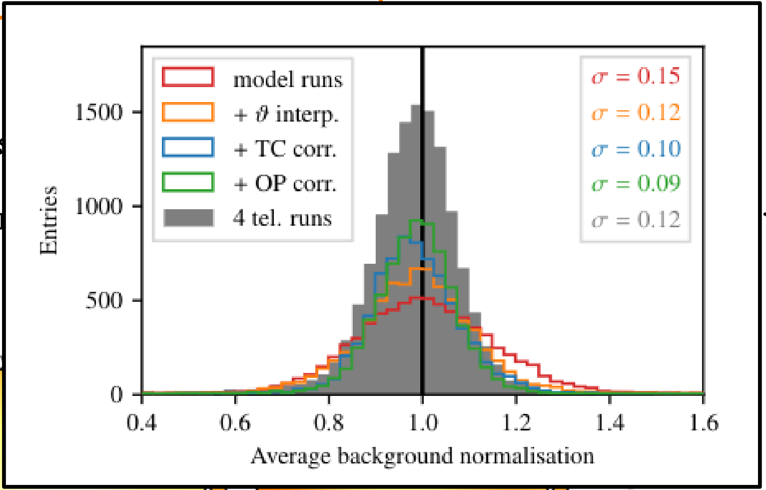
... ole ...

... l fi ...

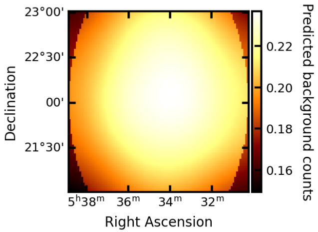
... you

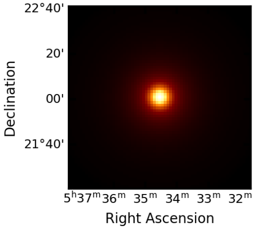


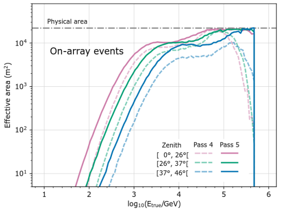




Now all together

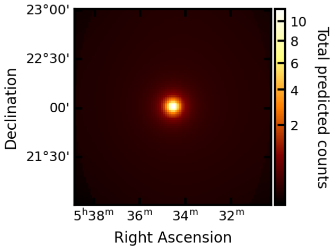


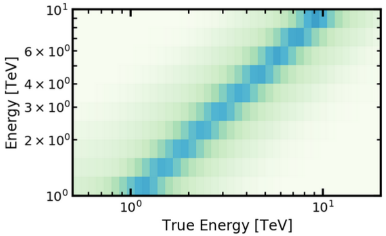


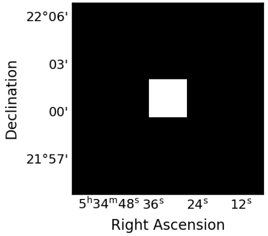


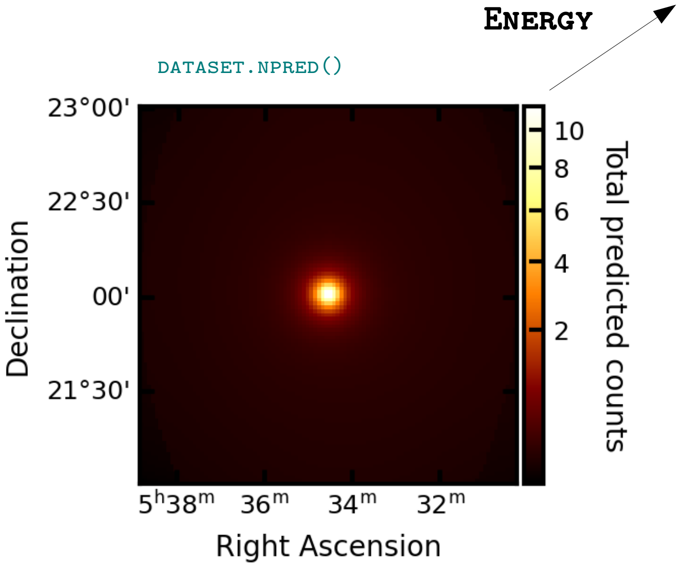
(b) Gamma/hadron cuts

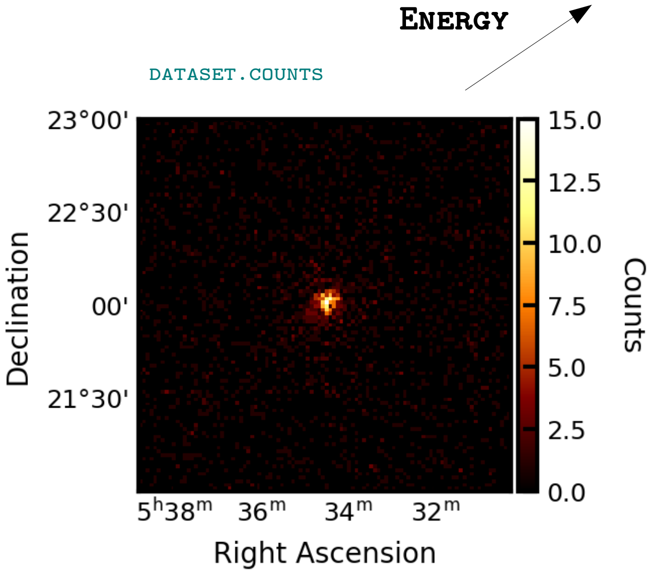
$$N_C(E_{reco},x_{reco}) = N_{bkg}(E_{reco},x_{reco}) + \int dx_{true} \int dE_{true}\; EDISP(E_{true},E_{reco}) \times PSF(E_{true},x_{reco},x_{true}) \times Aeff(E_{true}) \times t_{live} \times \left(\frac{dN_\gamma}{dE_{true}}(x_{true})\right)$$











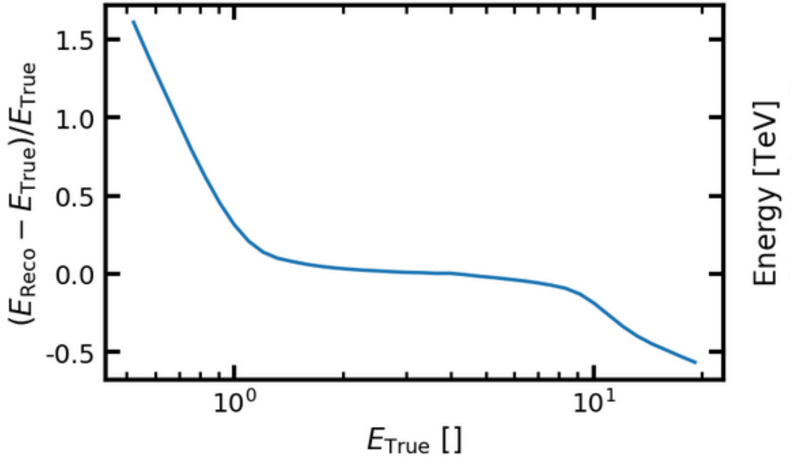
Now we can fit a model to the data by comparing prediction to observation!

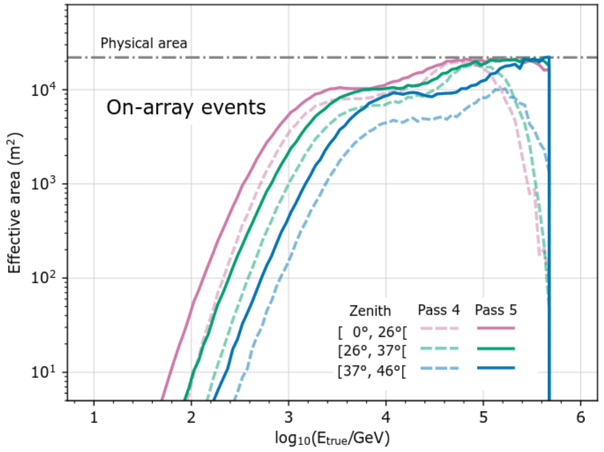
Data Analysis

- What does the data of gamma-ray instruments look like?
- What do I need to go from data to physical quantities?
- **How to determine the validity of data?**
- What is "data reduction"?

In which range is my data "valid"?

The IRFs also provide a very important tool in selecting our analysis range of validity





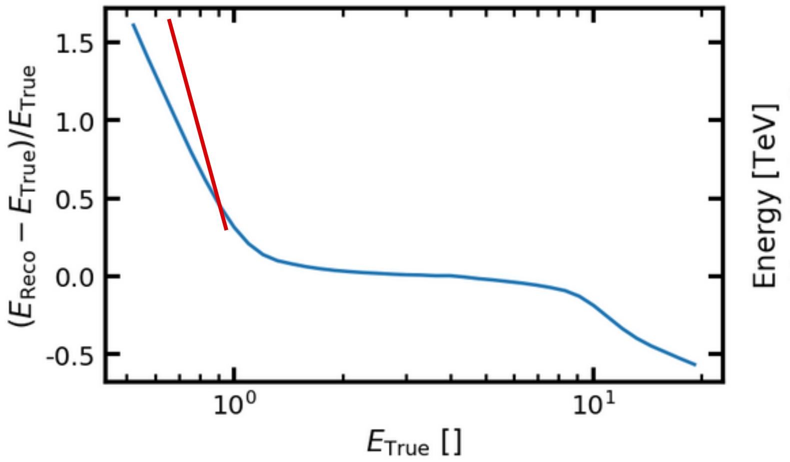
(b) Gamma/hadron cuts

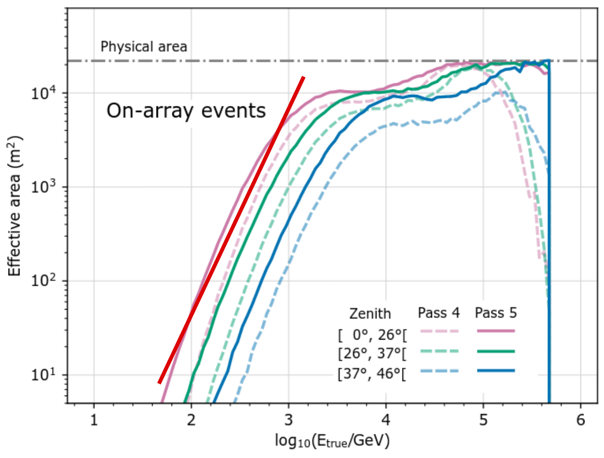
In which range is my data "valid"?

The IRFs also provide a very important tool in selecting our analysis range of validity

Remember: they are made with simulations. So they rely heavily on **MC/data consistency**

→ ranges where IRFs change rapidly can be dangerous!





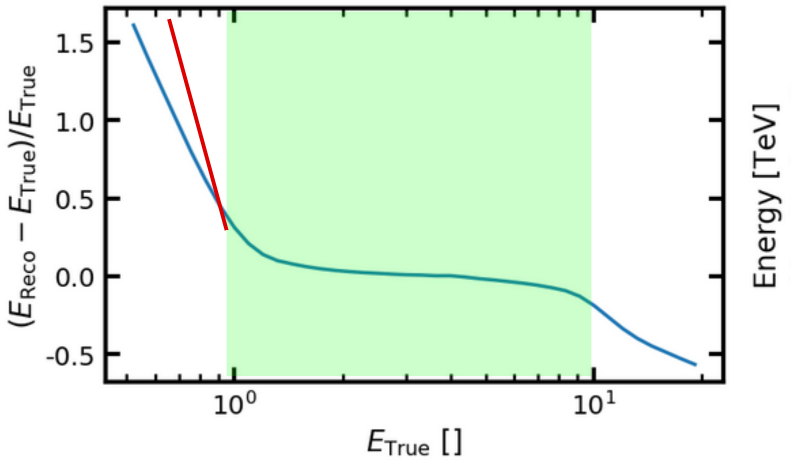
(b) Gamma/hadron cuts

In which range is my data "valid"?

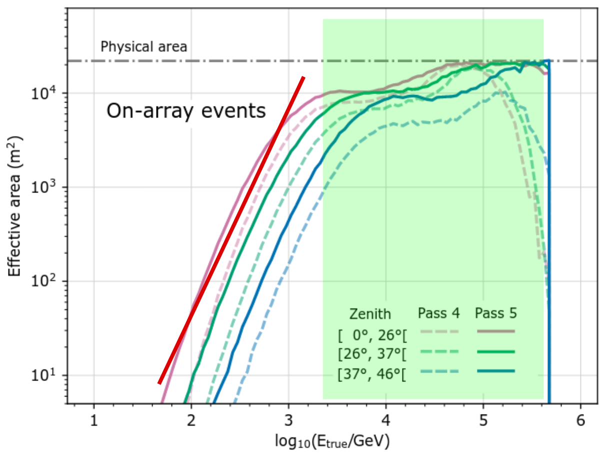
The IRFs also provide a very important tool in selecting our analysis range of validity

Remember: they are made with simulations. So they rely heavily on **MC/data consistency**

→ ranges where IRFs change rapidly can be dangerous!



`dataset.mask_safe.plot_grid()`



(b) Gamma/hadron cuts

Data Analysis

- What does the data of gamma-ray instruments look like?
- What do I need to go from data to physical quantities?
- How to determine the validity of data?
- **What is "data reduction"?**

Select and read data

```
from gammapy.data import DataStore

# Read bundle of event list, GTI and their corresponding IRF
data_store = DataStore.from_dir("$GAMMAPY_DATA/hess-dl3-dr1")

# Select runs around the Crab (only needed for pointing observations!)
selection = dict(
    type="sky_circle",
    frame="icrs",
    lon="83.633 deg",
    lat="22.014 deg",
    radius="5 deg",
)
selected_obs_table = data_store.obs_table.select_observations(selection)
observations = data_store.get_observations(selected_obs_table["OBS_ID"])

# Look at data
observations[0].events.table # event table
observations[0].GTI.table # event table

observations[0].aeff.peek() #aeff
observations[0].edisp.peek() #edisp
observations[0].psf.peek() #psf
observations[0].bkg.peek() #bkg
```

IN “DETECTOR” COORDINATES
(OFFSET, ZENITH…)

Reduce data

GO FROM DETECTOR TO SKY COORDINATES AND BUNDLE

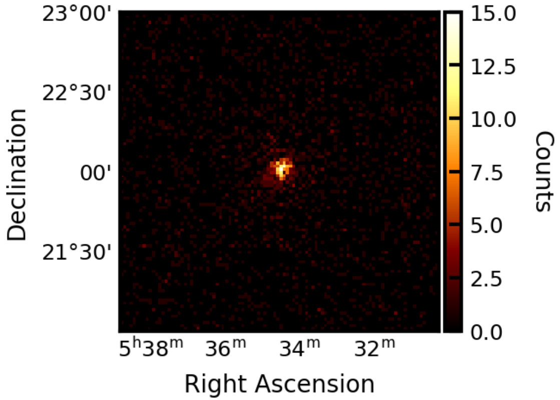
```
from gammapy.maps import WcsGeom, MapAxis
from gammapy.datasets import MapDataset
from regions import CircleSkyRegion

# Define energy axes
energy_axis = MapAxis.from_energy_bounds(1.0, 10.0, 10, unit="TeV")
energy_axis_true = MapAxis.from_energy_bounds(0.5, 20, 40, unit="TeV", name="energy_true") # always more range and more bins than reco!

# Define the sky geometry
geom = WcsGeom.create(
    skydir=(83.633, 22.014),
    binsz=0.02,
    width=(2, 2),
    frame="icrs",
    proj="CAR",
    axes=[energy_axis],
)

# create an empty dataset
stacked = MapDataset.create(geom=geom, energy_axis_true=energy_axis_true, name="crab-stacked")
```

Reduce data

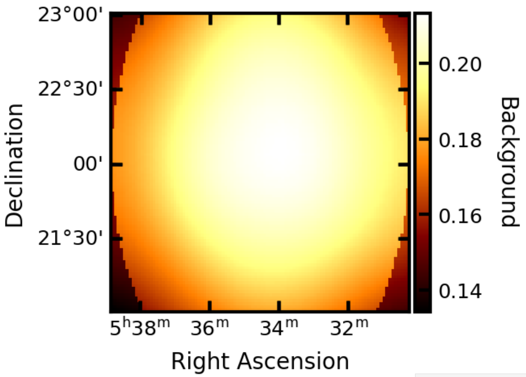
GO FROM DETECTOR TO SKY COORDINATES AND BUNDLE

```
# Define the makers
maker = MapDatasetMaker()
maker_safe_mask = SafeMaskMaker(methods=["offset-max", "aeff-max"], offset_max="2.5 deg")

circle = CircleSkyRegion(center=SkyCoord("83.63 deg", "22.14 deg"), radius=0.2 * u.deg)
exclusion_mask = geom.region_mask(regions=[circle], inside=False)
maker_fov_bkg = FoVBackgroundMaker(method="fit", exclusion_mask=exclusion_mask)

for obs in observations:
    # First a cutout of the target map is produced
    cutout = stacked.cutout(
        obs.get_pointing_icrs(obs.tmid), width=2 * offset_max, name=f"obs-{obs.obs_id}"
    )
    # A MapDataset is filled in this cutout geometry
    dataset = maker.run(cutout, obs)
    # The data quality cut is applied
    dataset = maker_safe_mask.run(dataset, obs)
    # fit background model
    dataset = maker_fov.run(dataset)
    print(
        f"Background norm obs {obs.obs_id}: {dataset.background_model.spectral_model.norm.value:.2f}"
    )
    # The resulting dataset cutout is stacked onto the final one
    stacked.stack(dataset)
```

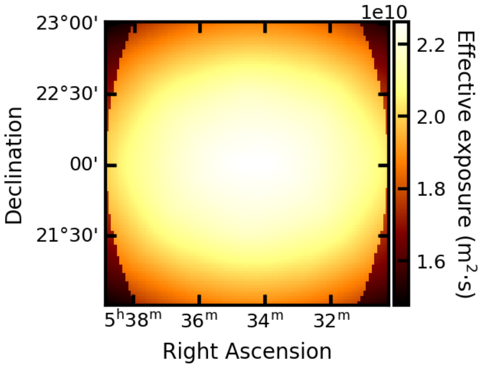
```python
ax = stacked.counts.reduce_over_axes().plot(add_cbar=True)
cbar = ax.images[-1].colorbar
cbar.set_label('Counts', rotation=270, labelpad=30)
```



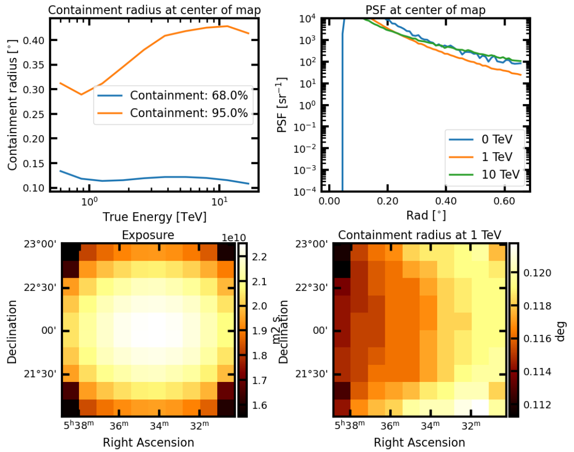

```python
ax = stacked.background.reduce_over_axes().plot(add_cbar=True)
cbar = ax.images[-1].colorbar
cbar.set_label('Background', rotation=270, labelpad=30)
```



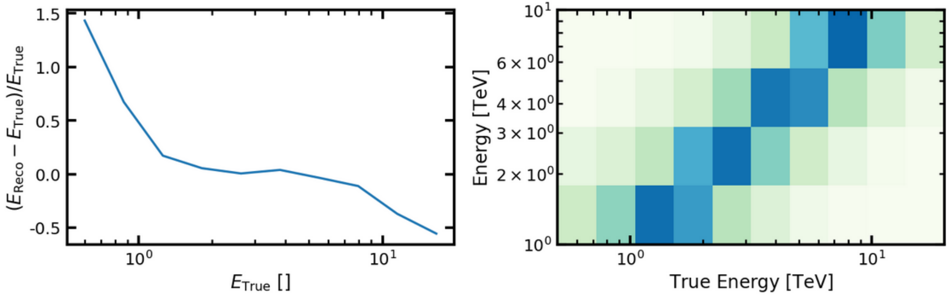

```python
ax = stacked.exposure.reduce_over_axes().plot(add_cbar=True)
cbar = ax.images[-1].colorbar
cbar.set_label('Effective exposure (m$^2$·s)', rotation=270, labelpad=30)
```




```python
stacked.psf.peek()
```




```python
stacked.edisp.peek()
```



DL3
γ-like events

DL4
Binned data

DL5
Science products

Data reduction

CHOOSE A SKY GEOMETRY AND BIN/PROJECT THE EVENTS AND IRFs

.data
- DataStore
- Observations
- Observation
- GTI

.makers
- MapDatasetMaker
- SafeMaskMaker
- FoVBackgroundMaker
- RingBackgroundMaker
- etc.

.datasets
- Datasets
- MapDataset
- MapDatasetOnOff
- etc.

Likelihood fitting

.modeling
- Fit, Models, SkyModel
- FoVBackgroundModel
- etc.

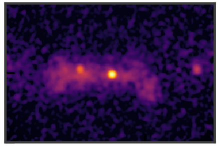
.estimators
- FluxPointsEstimator
- FluxMapEstimator
- etc.

YAML → .analysis

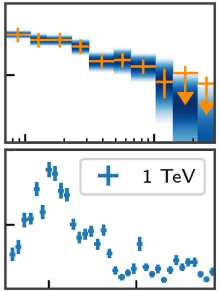
Source Catalogs

Name	Flux	Size
SNR	1e-12	1 deg
PWN	1e-11	0.2 deg
GRB	1e-10	0 deg

Flux & TS Maps



SEDs & Lightcurves



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

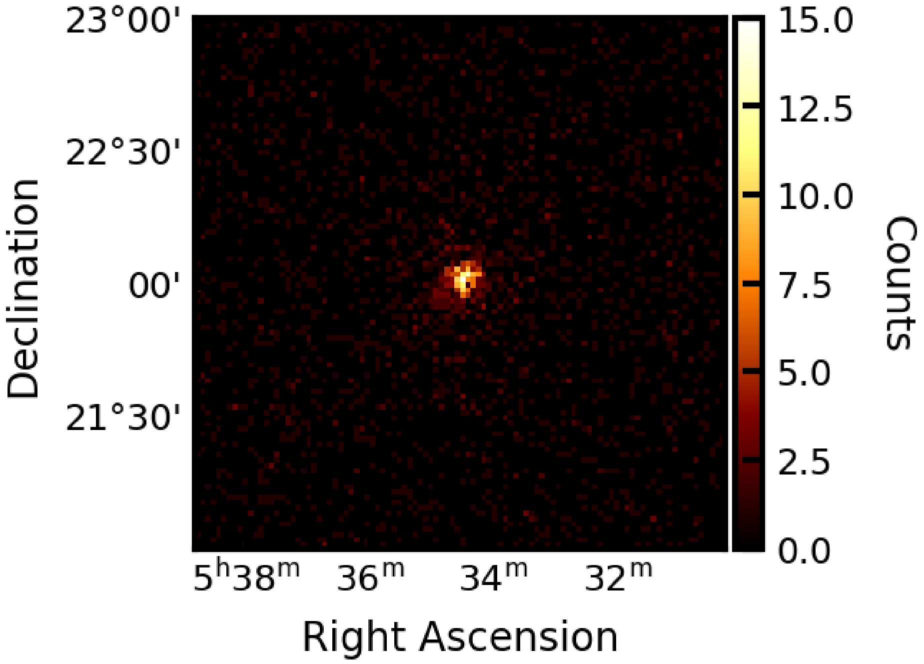
*if we have time

Data Analysis

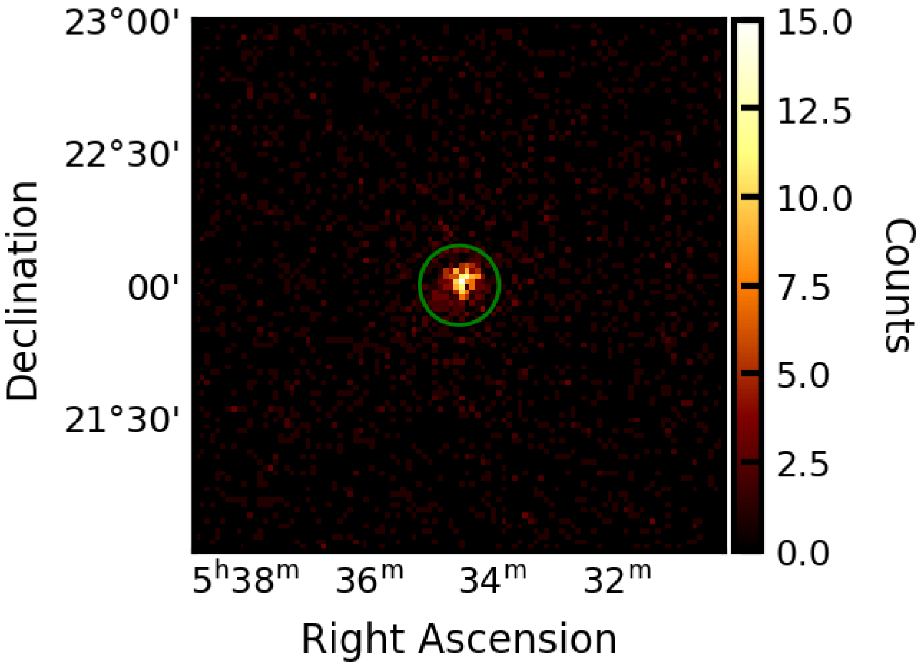
- Is there a source there?
- What are its properties? (spectral, spatial, temporal*)
- How to present results
- Combining data from different instruments
- Sources of uncertainty

*if we have time

Simple approach – let's take a cutout



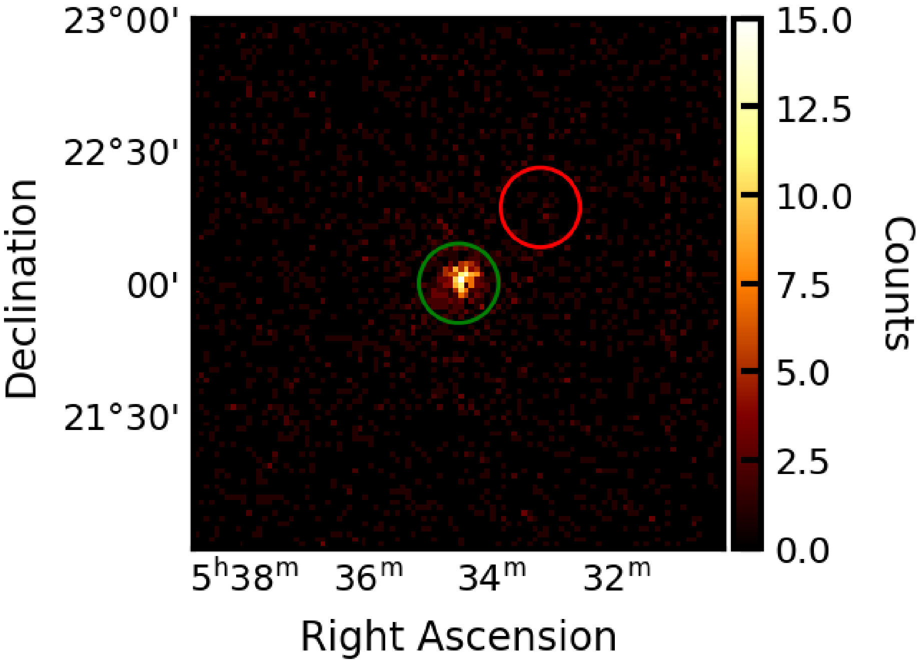
Simple approach – let's take a cutout




```
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.

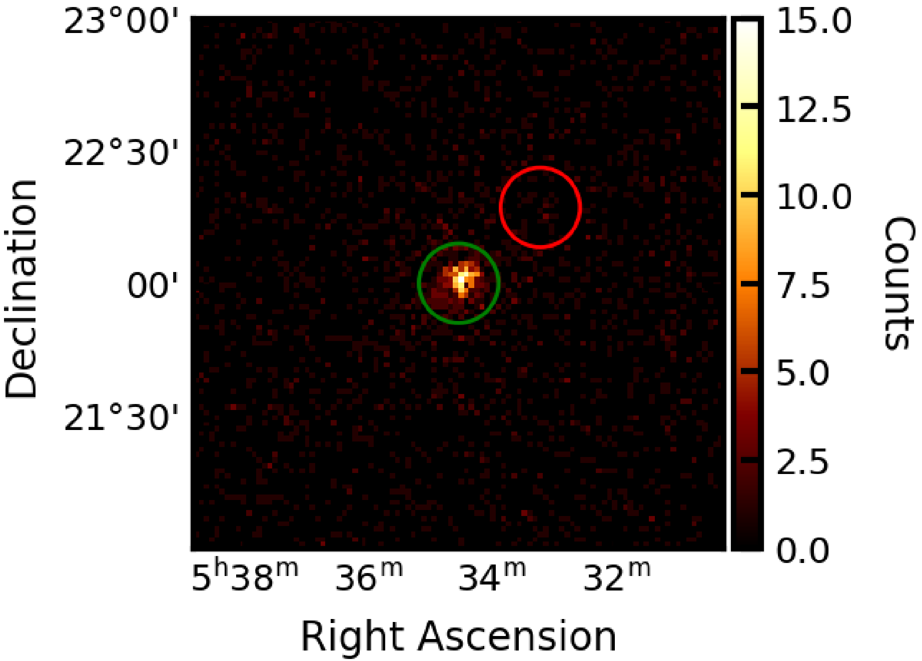
We use the difference in test statistic (TS), **the (Poisson) likelihood ratio**

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

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!)

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

- Significance: ~35σ

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

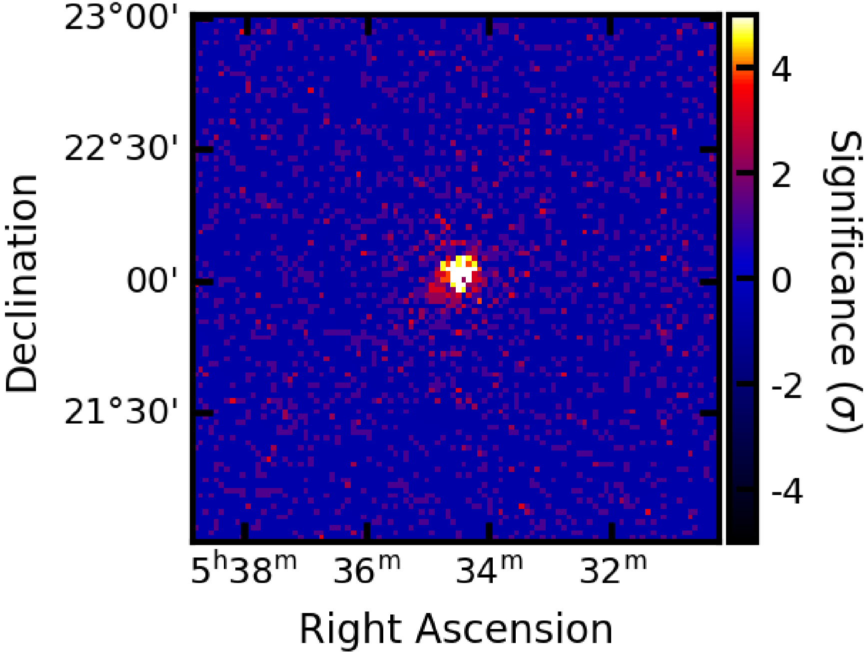
That is kind of unsatisfying – what about a map?

Here we have two options:

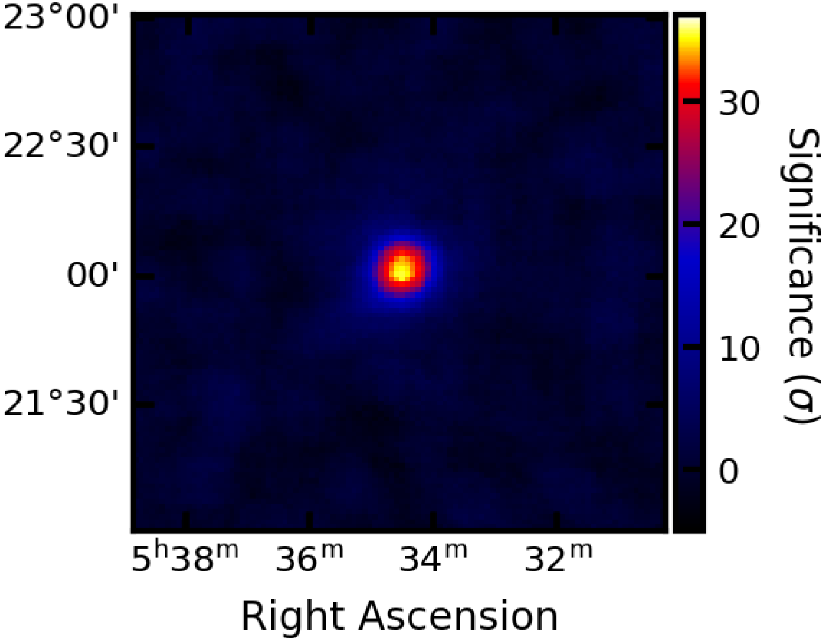
- **Based on excess:** for each pixel of the map, compare the measured counts with the expected background using Li&Ma. Not model dependent but usually done with some smoothing, which impacts scale of visible structures → **common in IACTs**
- **Based on model:** For each pixel compare the likelihood 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

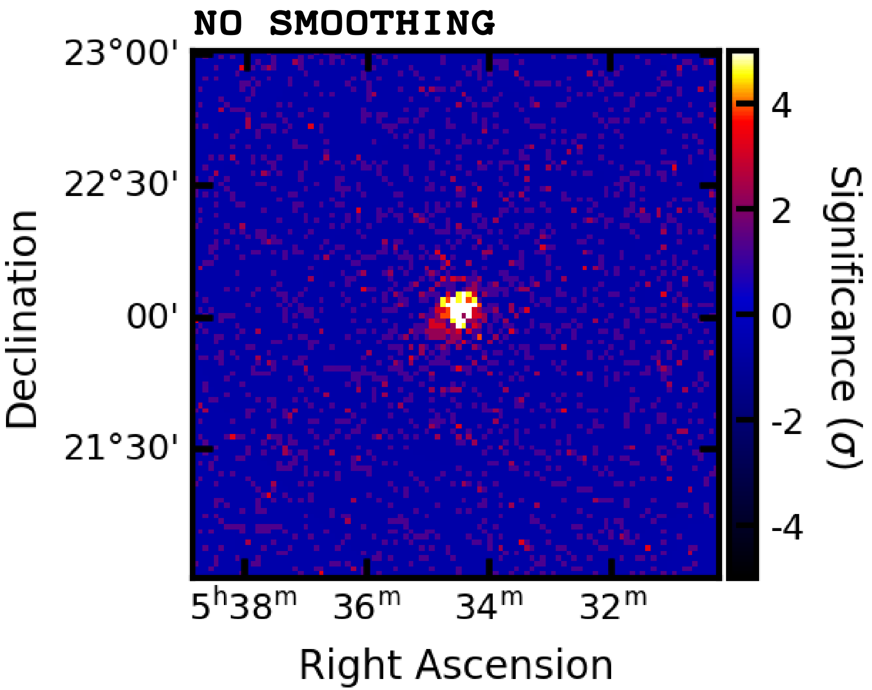


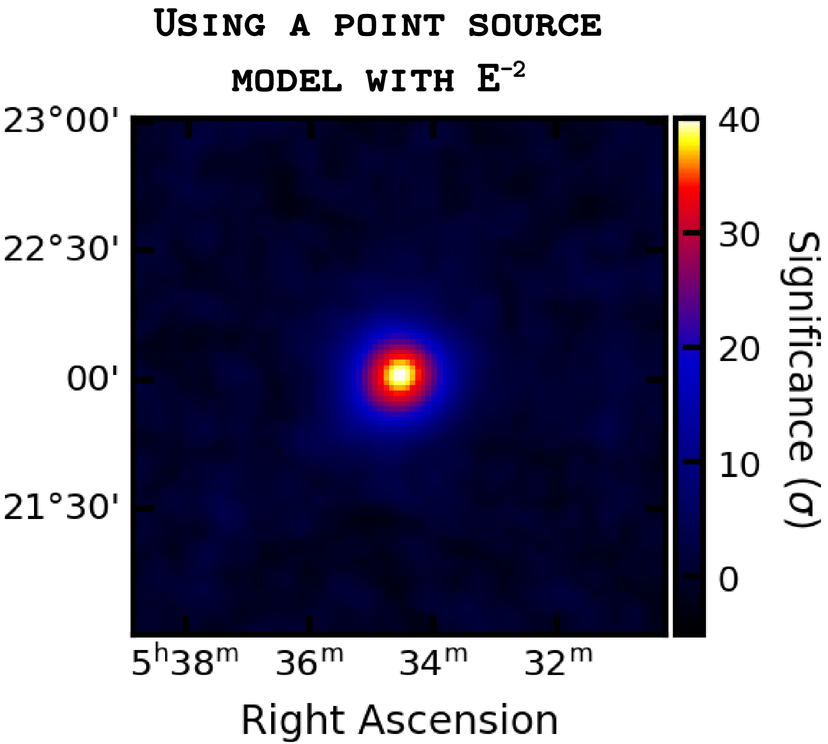
Reasonable smoothing: similar to PSF




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

Significance maps

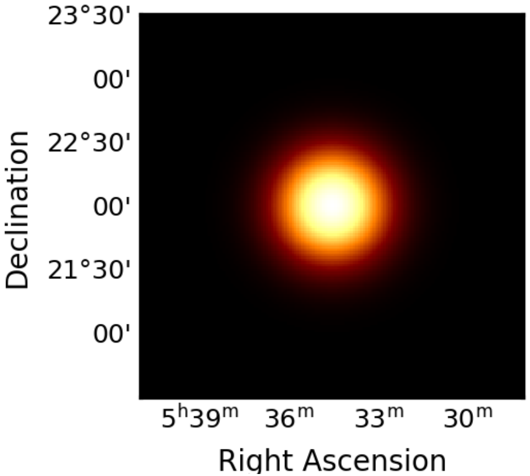





```
estimator = TSMapEstimator()
rresult = estimator.run(stacked)
```

Beware!

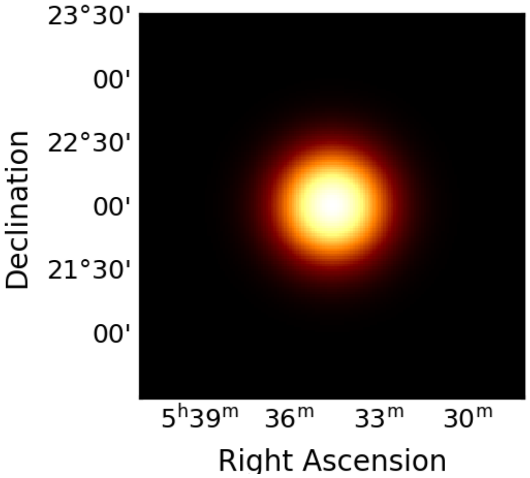
IF SPATIALLY EXTENDED SOURCE



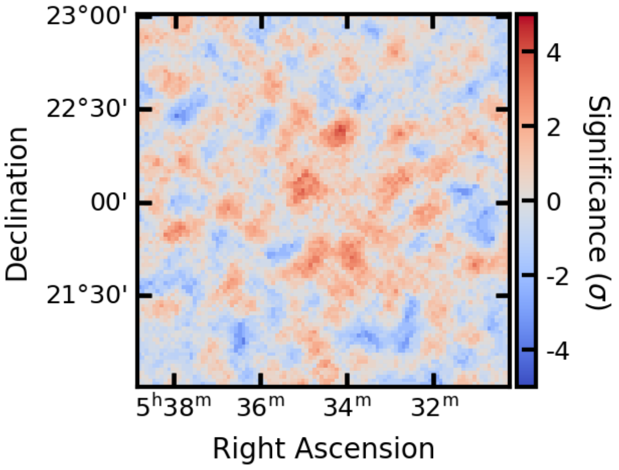
Simulate a source by giving a dataset a model and doing dataset.fake()

Beware!

IF SPATIALLY EXTENDED SOURCE



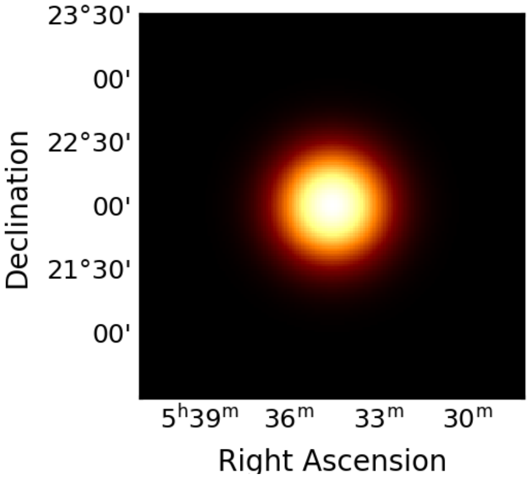
Simulate a source by giving a dataset a model and doing dataset.fake()



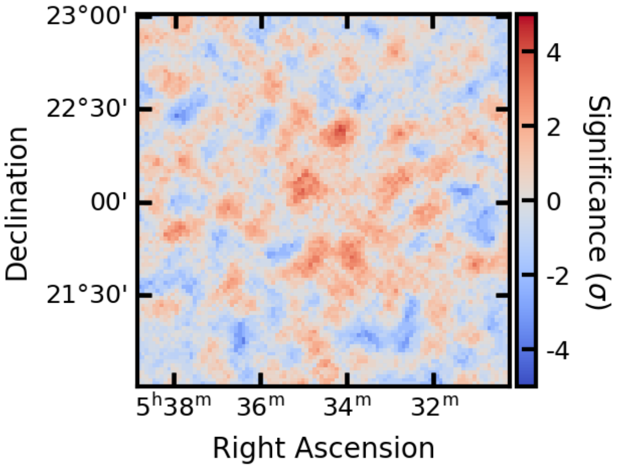
USING CORRELATION RADIUS LIKE THE PSF: WE SEE NOTHING!

Beware!

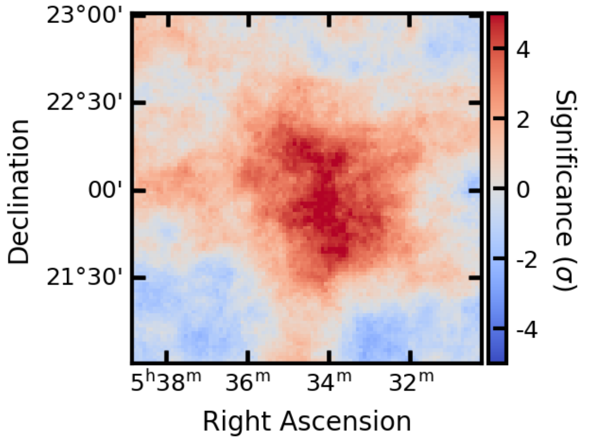
IF SPATIALLY EXTENDED SOURCE



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



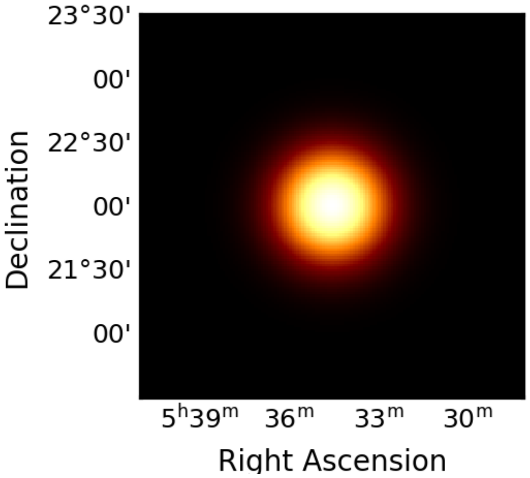
USING CORRELATION RADIUS LIKE THE PSF: WE SEE NOTHING!



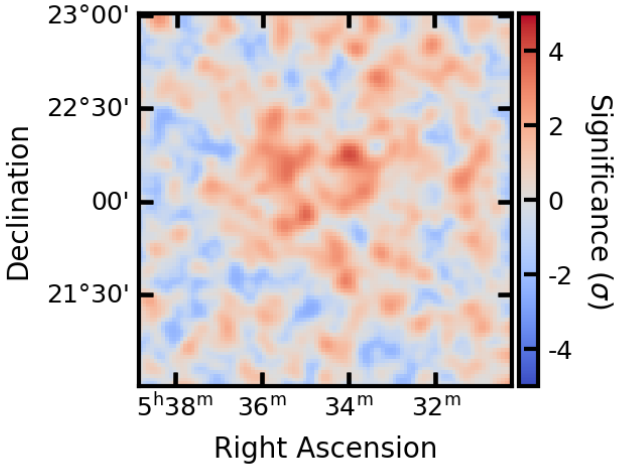
WITH LARGER RADIUS IT SHOWS UP – BUT BACKGROUND SYSTEMATICS ARE DANGEROUS!

Beware!

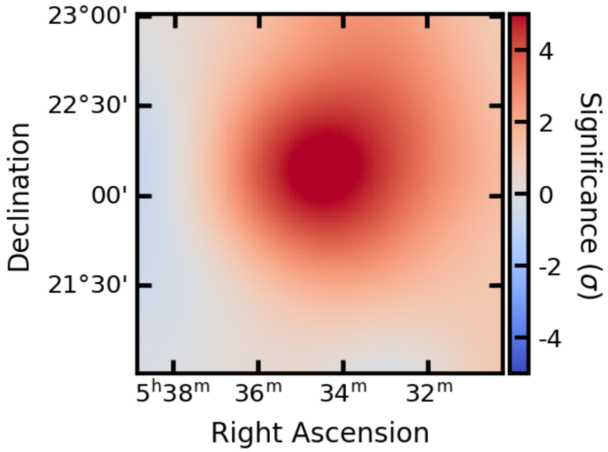
IF SPATIALLY EXTENDED SOURCE



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?

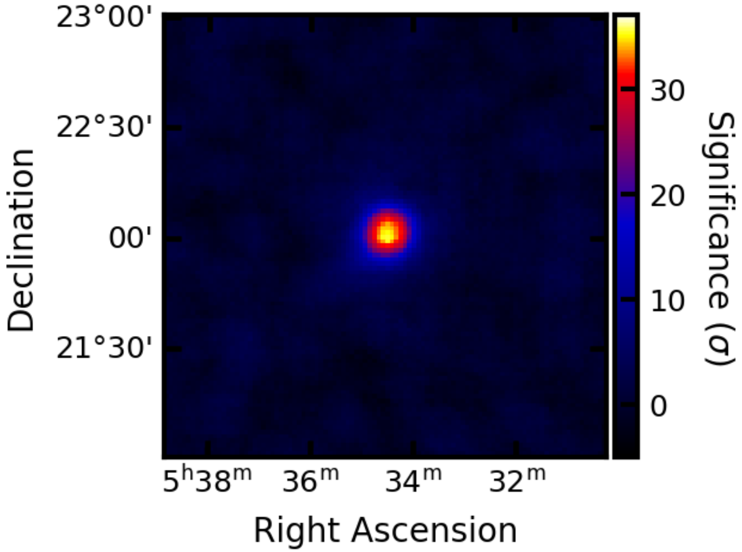
In the absence of signal, the significance distribution should be well described by a **Gaussian distribution with mean=0 and width=1**. This = as many +ve fluctuations as -ve!

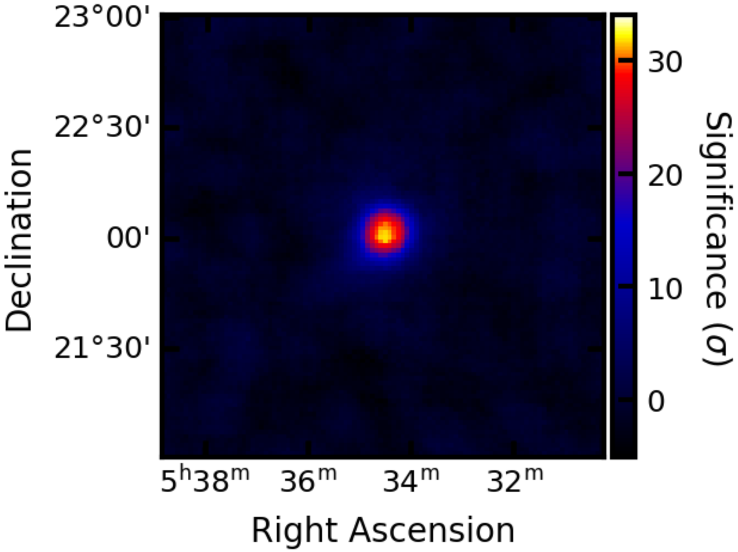
Fluctuations should be randomly distributed so that map has no structures such as gradients or other

Background normalization

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

Can you tell me which one?

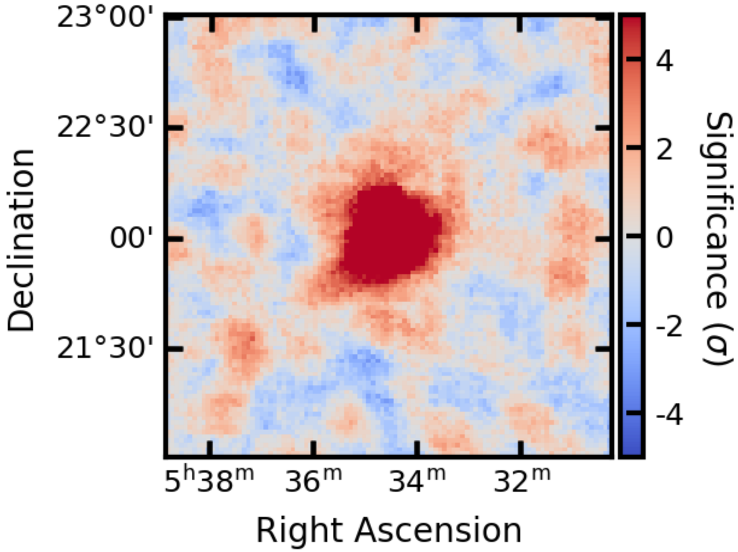


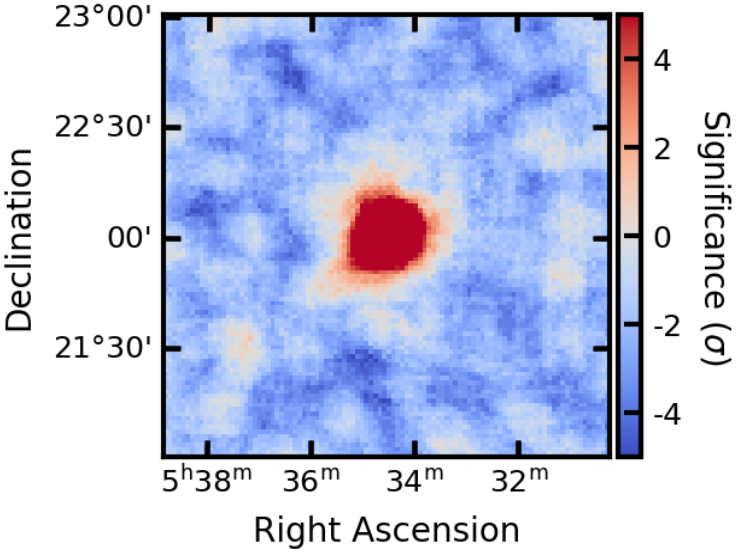


Background normalization

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

Can you tell me which one?

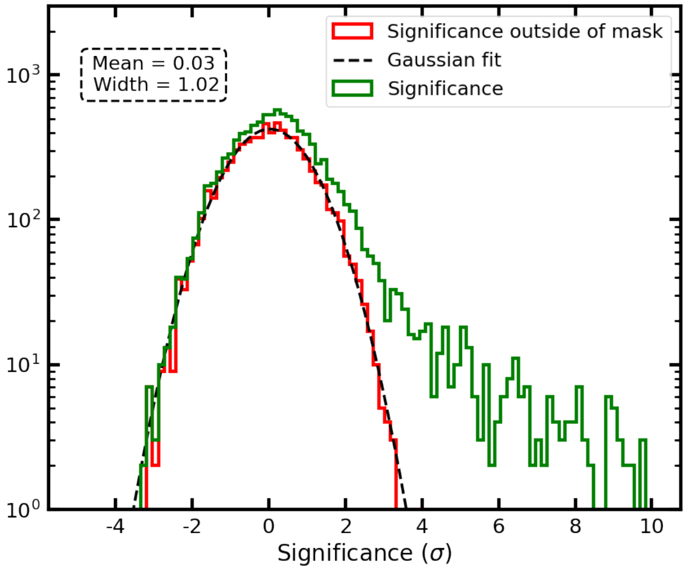


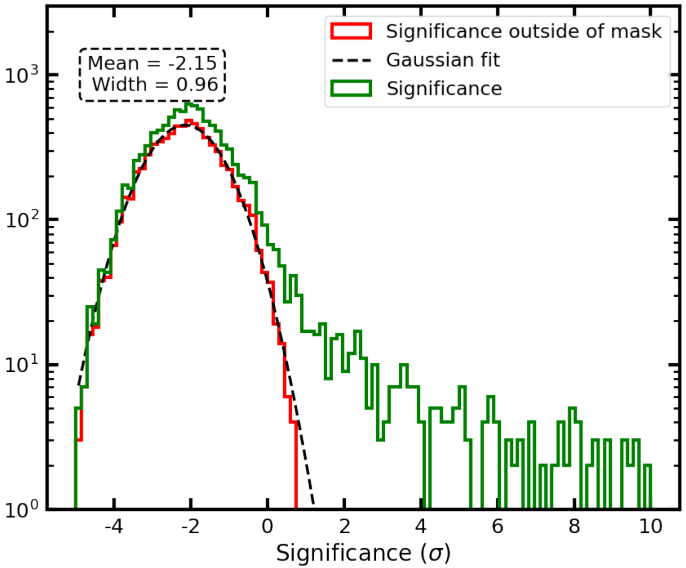


Background normalization

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

Can you tell me which one?



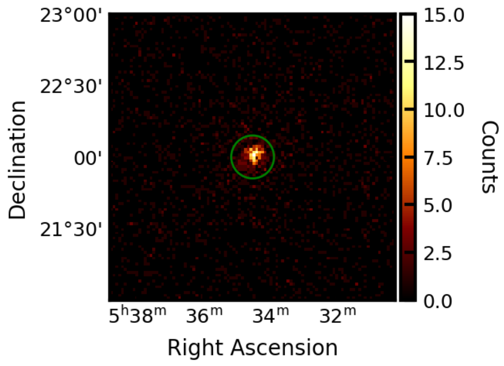


Data Analysis

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

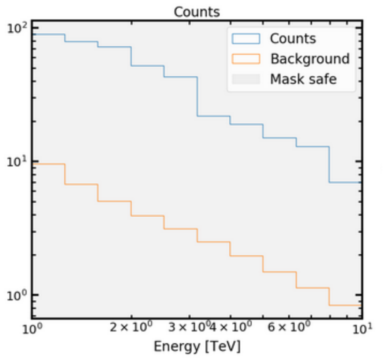
*if we have time

Modeling our source – 1D analysis



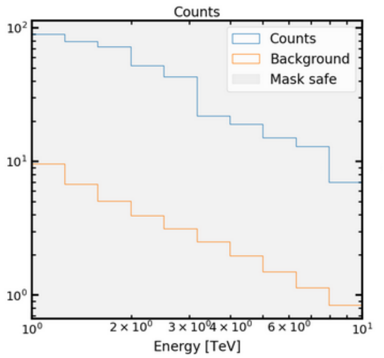

```
SPEC =STACKED.TO_SPECTRUM_DATASET(ON_REGION)
```

Modeling our source – 1D analysis

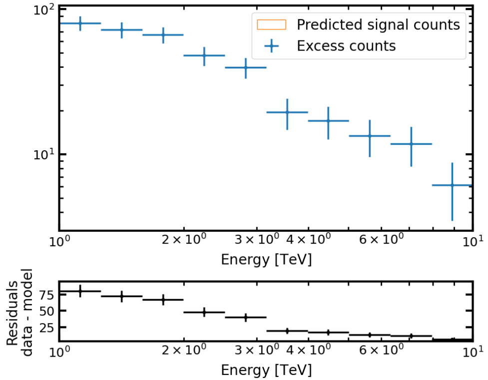


SPEC =STACKED.TO_SPECTRUM_DATASET(ON_REGION)

Modeling our source – 1D analysis

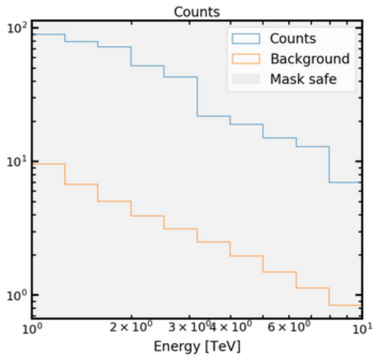


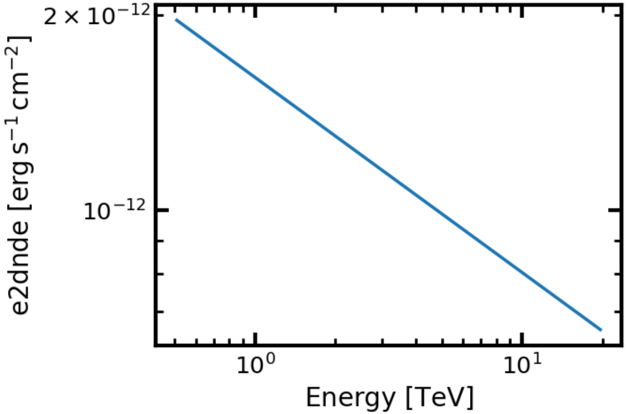
EXCESS= COUNTS-BKG



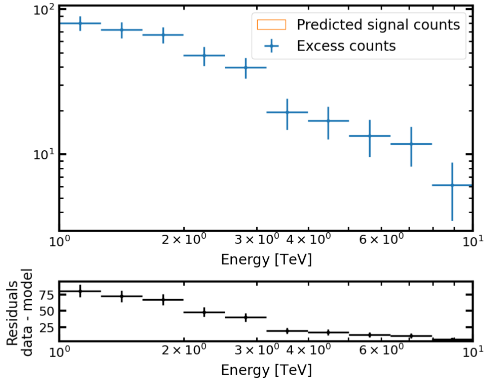
Modeling our source – 1D analysis

```
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]
```



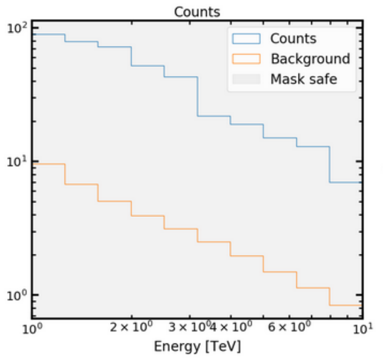


EXCESS= COUNTS-BKG

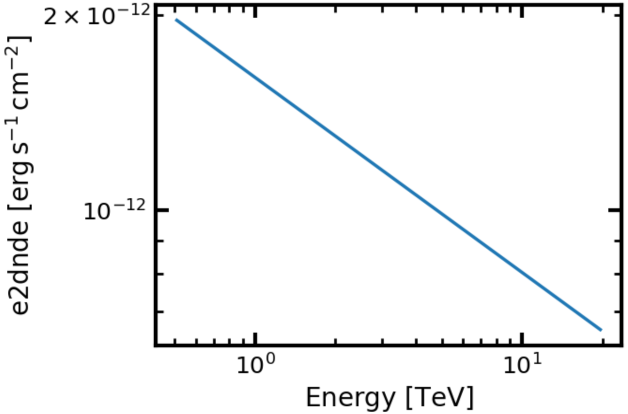


Modeling our source – 1D analysis

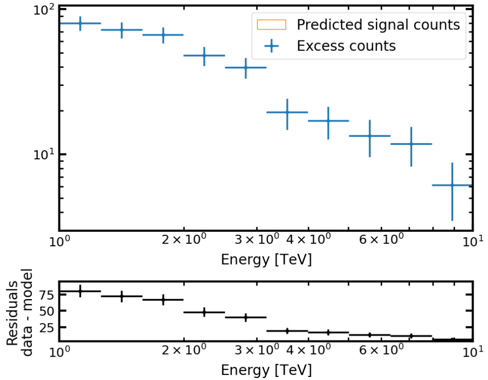
```
spectral_model = PowerLawSpectralModel(
    amplitude=1e-12 * u.Unit("cm-2 s-1 TeV-1"),
    index=2,
    reference=1 * u.TeV,
)
model = SkyModel(spectral_model=spectral_model, name="crab")
spec.models = [model]
```



EXCESS= COUNTS-BKG

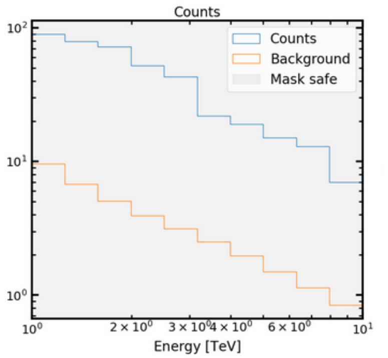


USING IRFs, CALCULATE PREDICTED EXCESS FOR EACH SET OF MODEL PARAMETERS

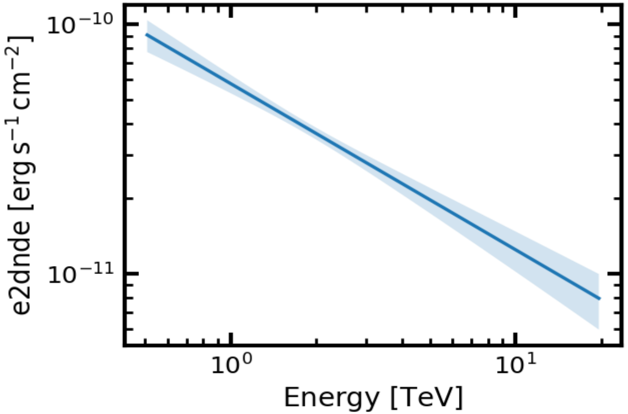


Modeling our source – 1D analysis

```
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)
```

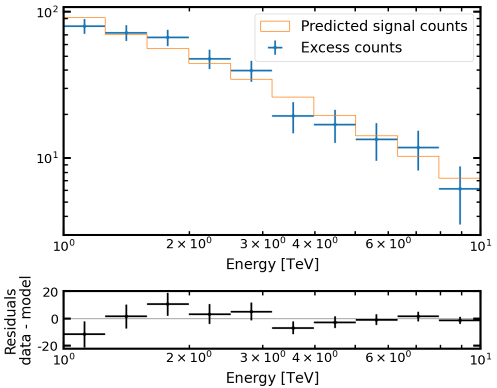


EXCESS= COUNTS-BKG



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

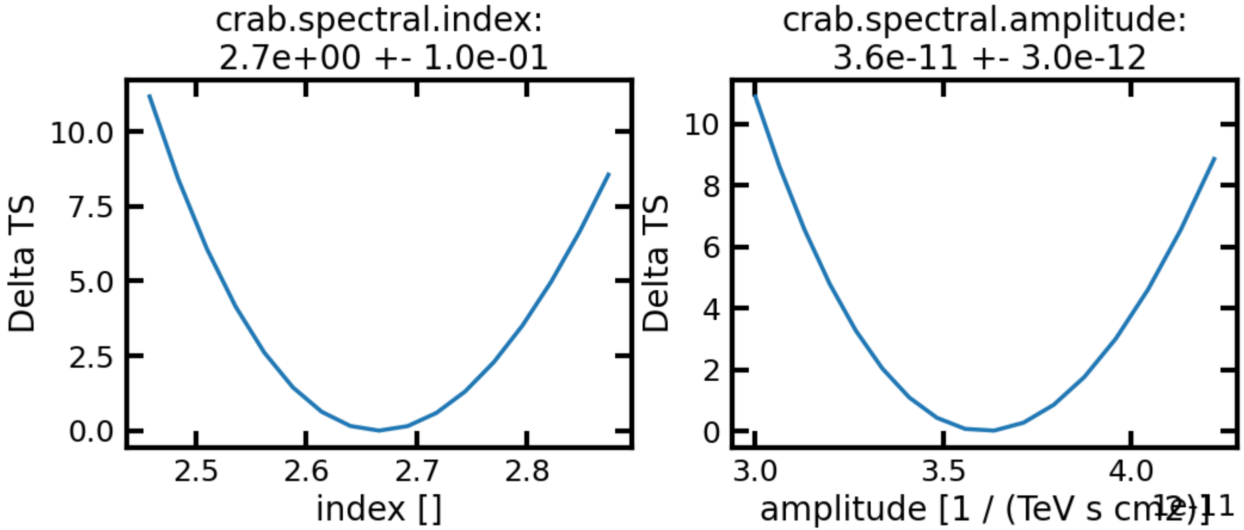
FIT: FIND SET OF PARAMETERS WHICH MAXIMIZES THE LIKELIHOOD (WHICH MAKES THE PREDICTION CLOSEST TO THE OBSERVED EXCESS)



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?



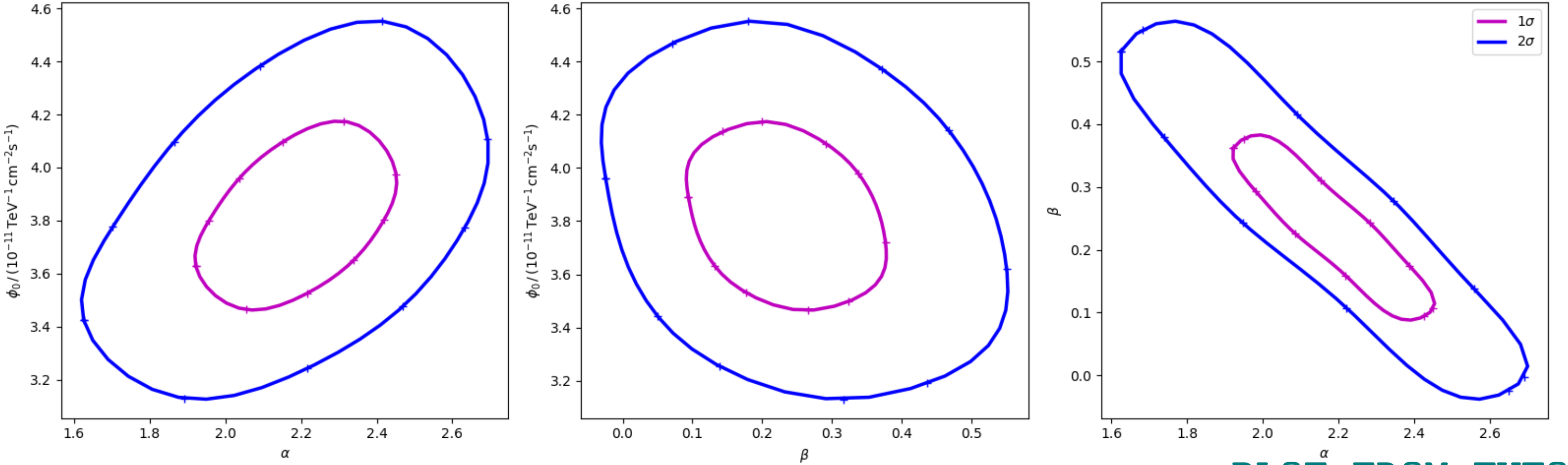
[Link to tutorial]

How to determine fit quality?

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

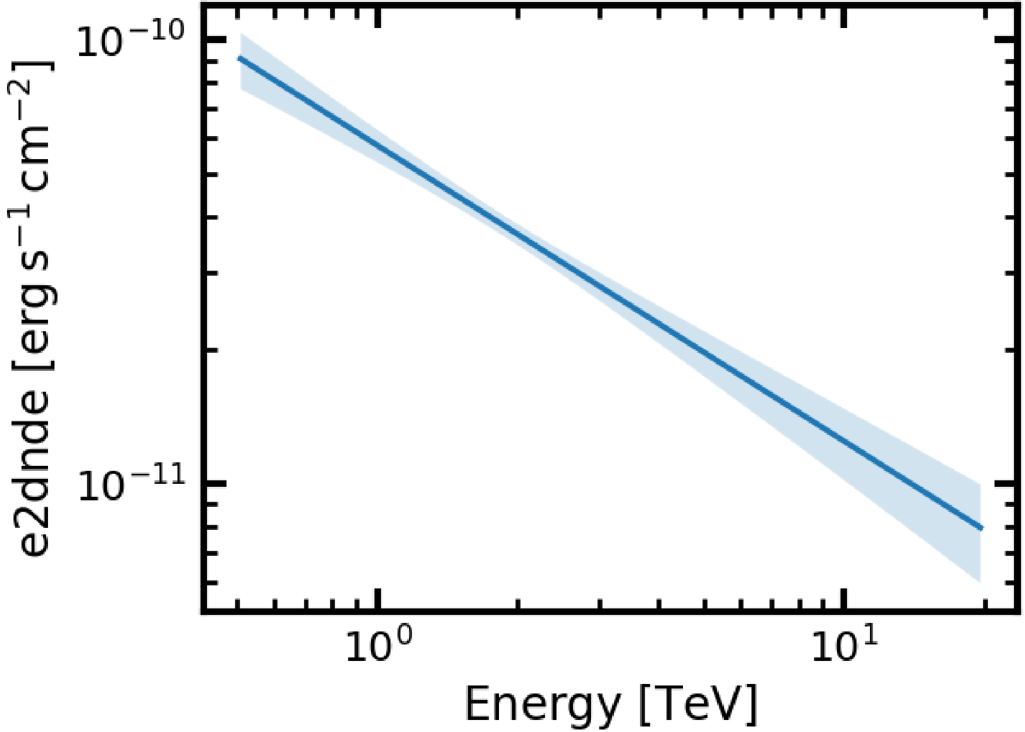
- Maximum likelihood = minimum test statistic
- Is the minimum well defined?

HOW TO



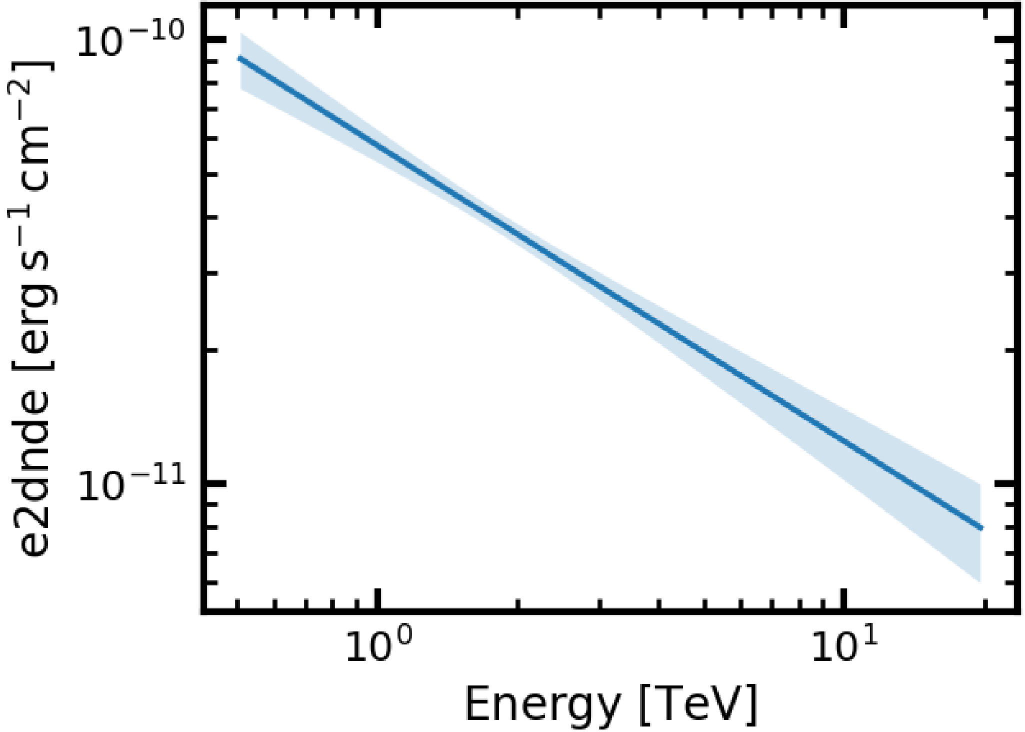
PLOT FROM TUTORIAL

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.
```

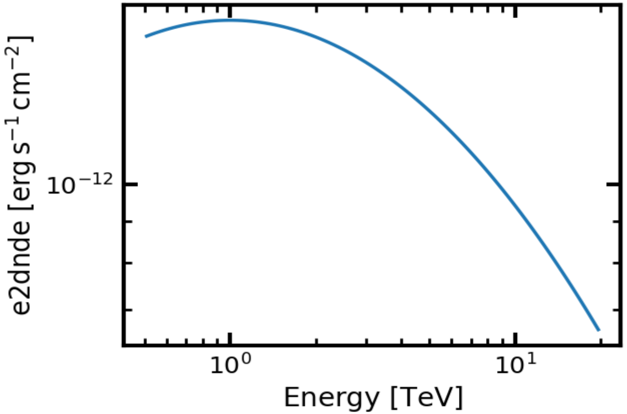
Modeling our source – 1D analysis

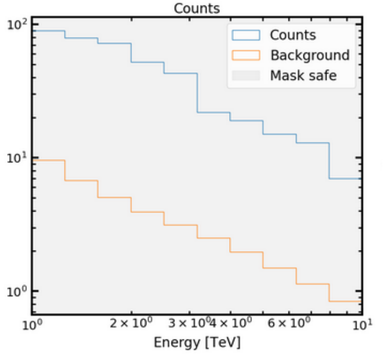



```
OptimizeResult

	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.
```

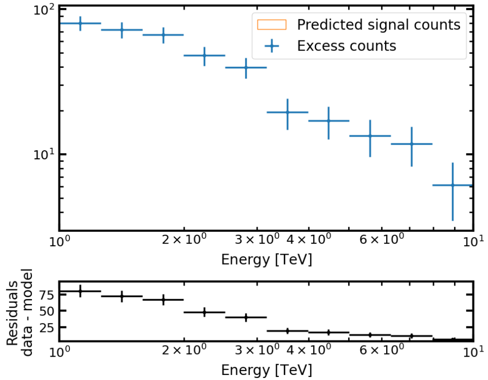
Was that the best model?

```
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)
```



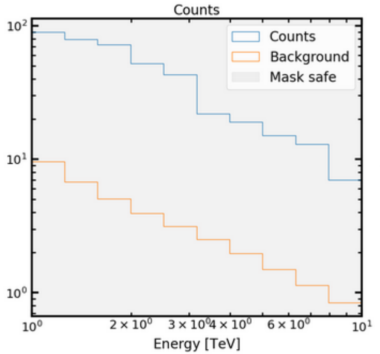


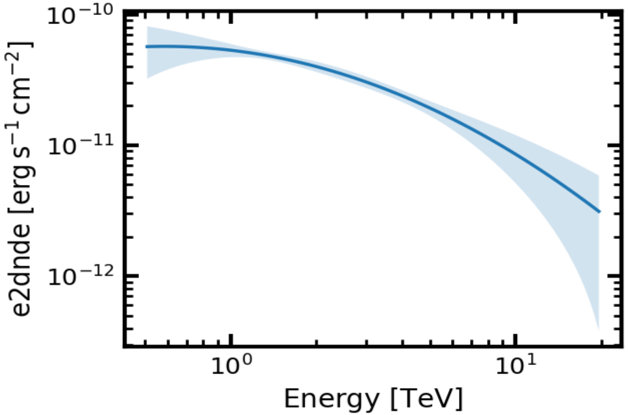
EXCESS= COUNTS-BKG



WAS THAT THE BEST MODEL?

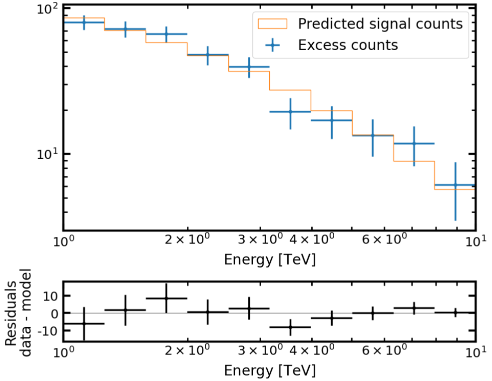
```
spectral_model = LogParabolaSpectralModel(
    amplitude=1e-12 * u.Unit("cm-2 s-1 TeV-1"),
    index=2,
    beta=0.01,
    reference=1 * u.TeV,
)
model = SkyModel(spectral_model=spectral_model, name="crab-lp")
spec.models = [model]
fit = Fit()
result = fit.run(datasets=spec)
```






```
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 χ^2 distribution with N_{DOF} degrees of freedom, where N_{DOF} is the difference of free parameters between the two hypothesis as long as they are nested

- 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 <u>with one extra parameter</u>
- 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:

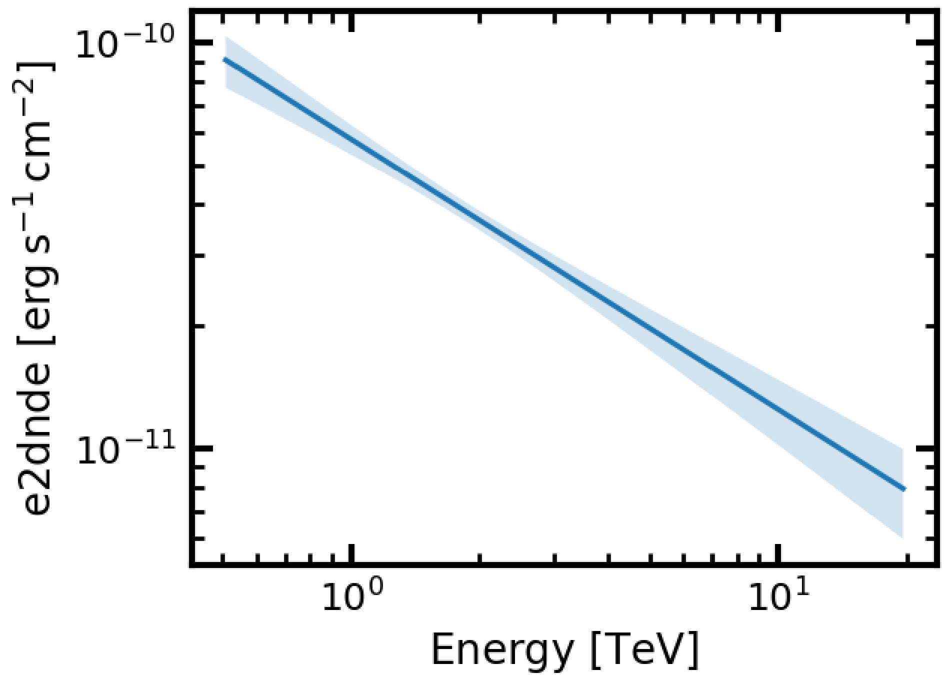
- "ts" : fit statistic difference with null hypothesis
- "fit_results" : results for the best fit
- "fit_results_null" : fit results for the null hypothesis

```
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'])
```

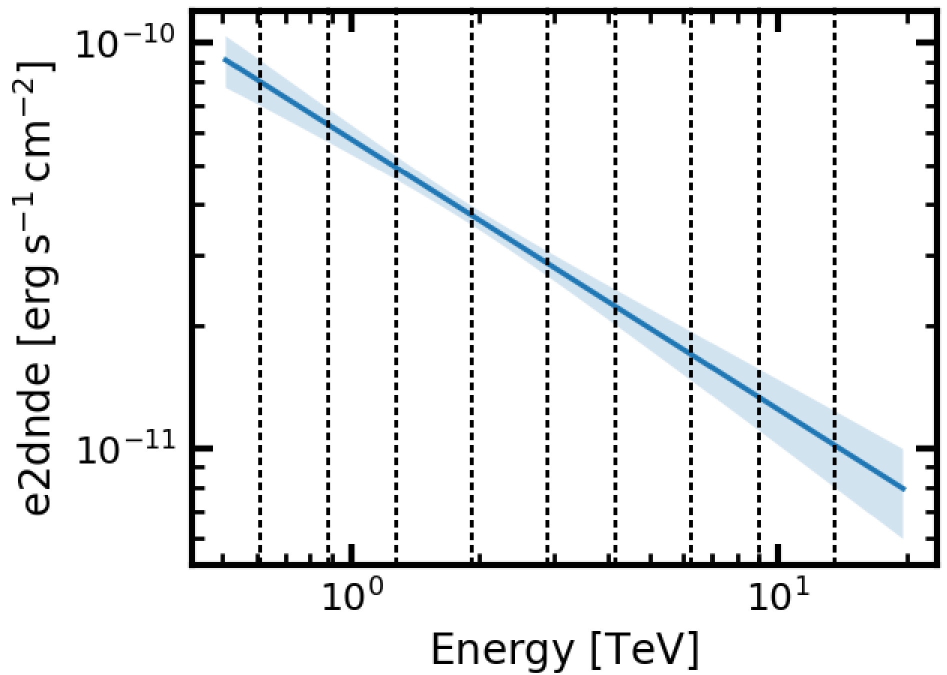
```
from gammapy.modeling.selection import select_nested_models

# 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

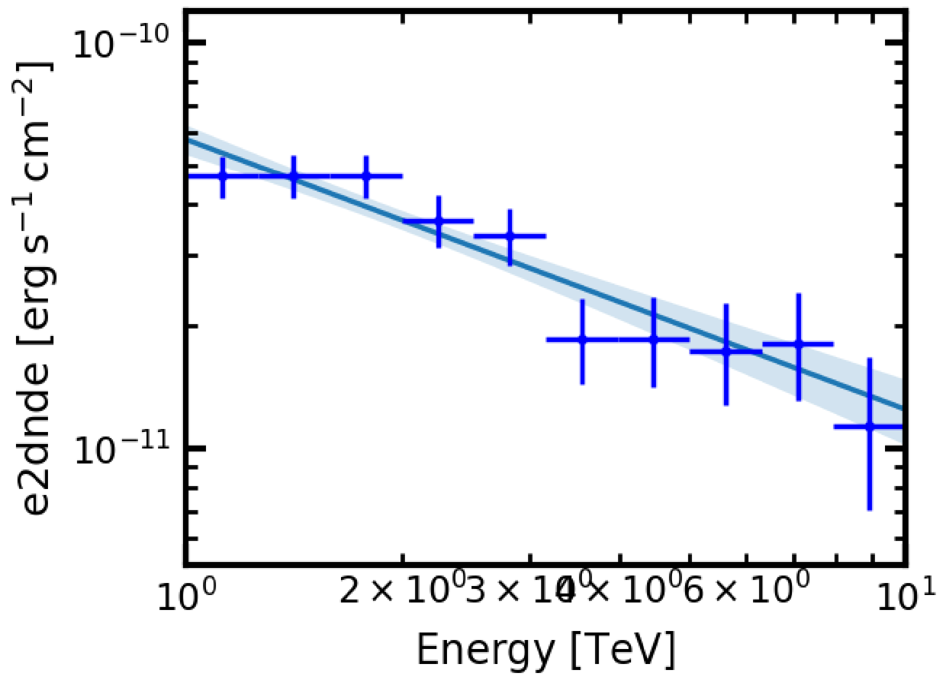


Modeling our source – 1D analysis



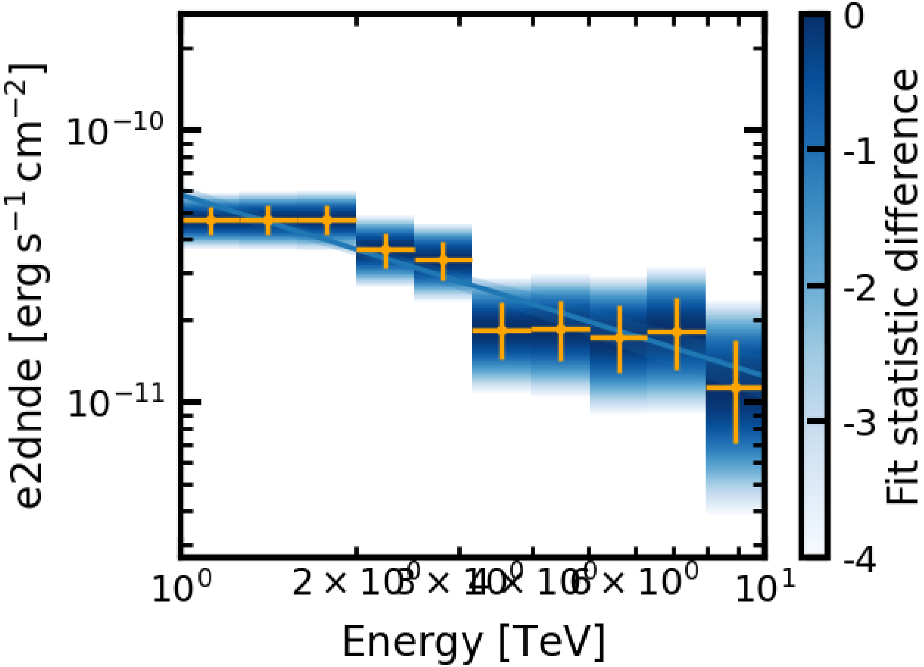
Fit again in each small bin assuming the overall shape but letting the normalization free

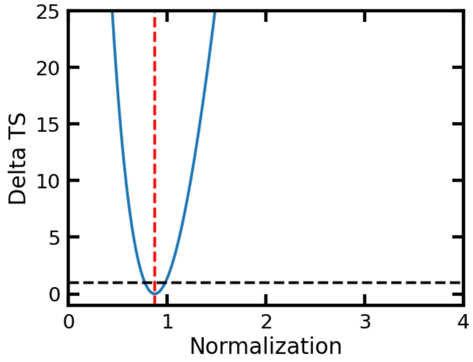
Modeling our source – 1D analysis



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

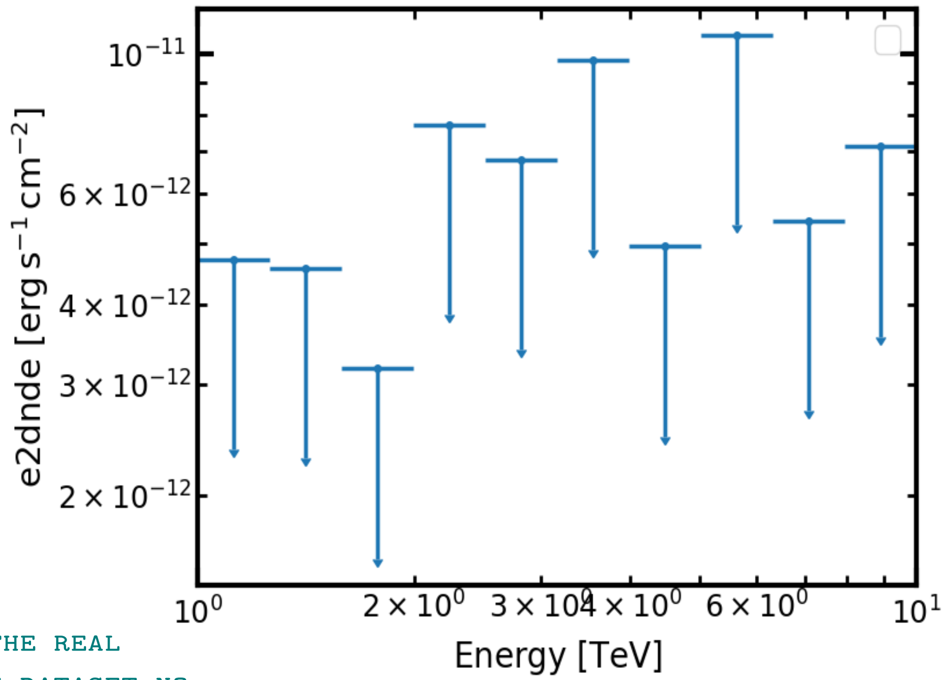
Modeling our source – 1D analysis





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

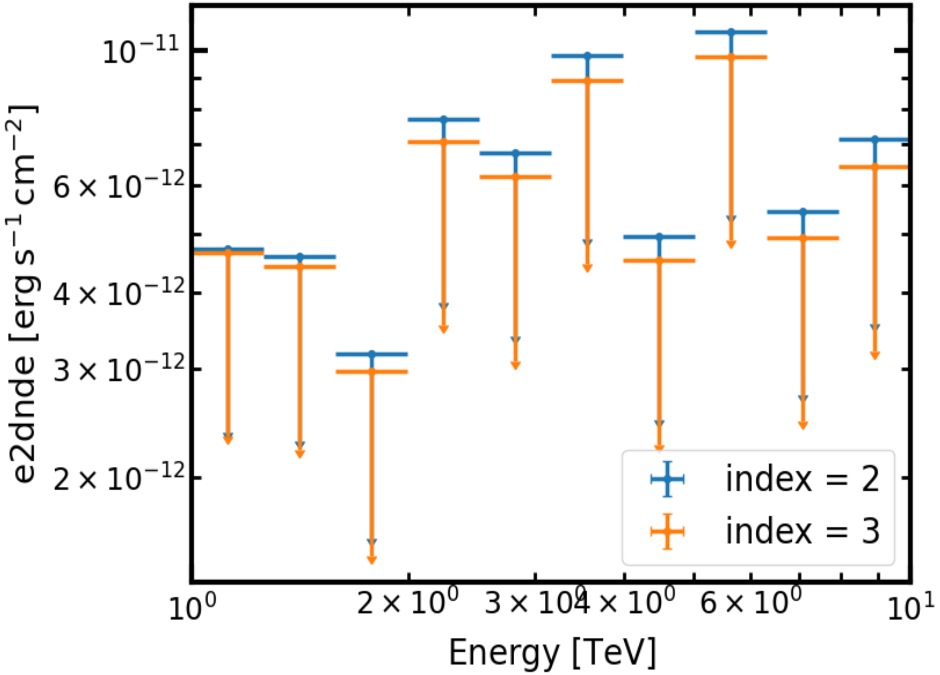
What if nothing is detected? upper limits



You need to assume a spectral shape!

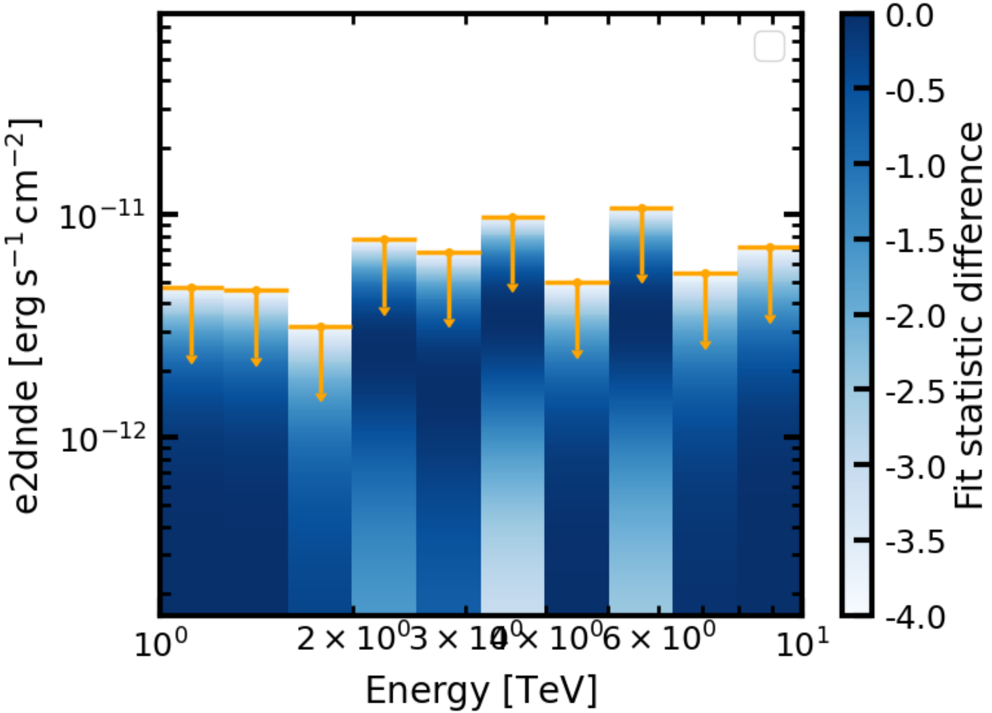
Here I removed the real counts, gave the dataset no model and did dataset.fake()

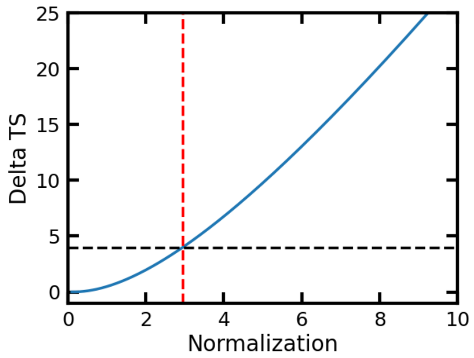
What if nothing is detected? upper limits



You need to assume a spectral shape!

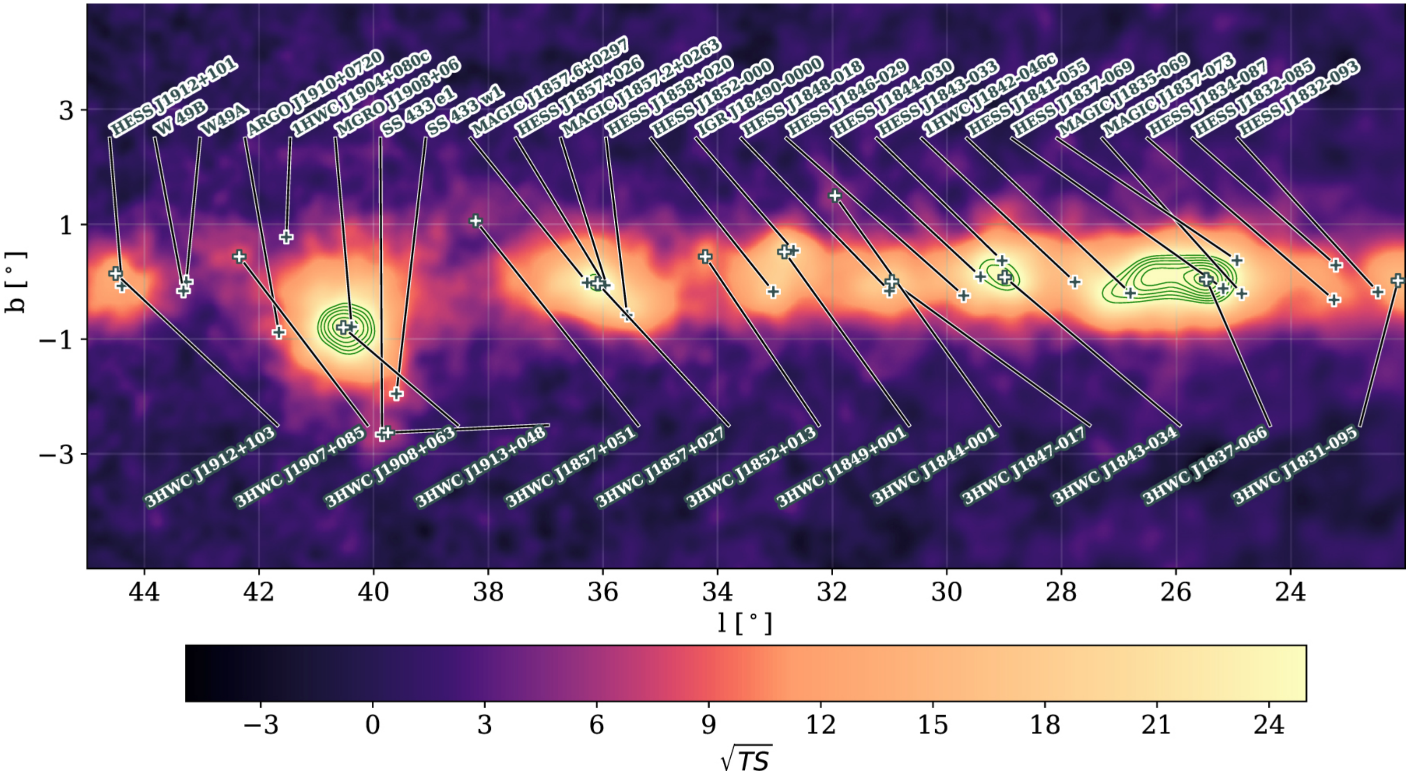
What if nothing is detected? upper limits



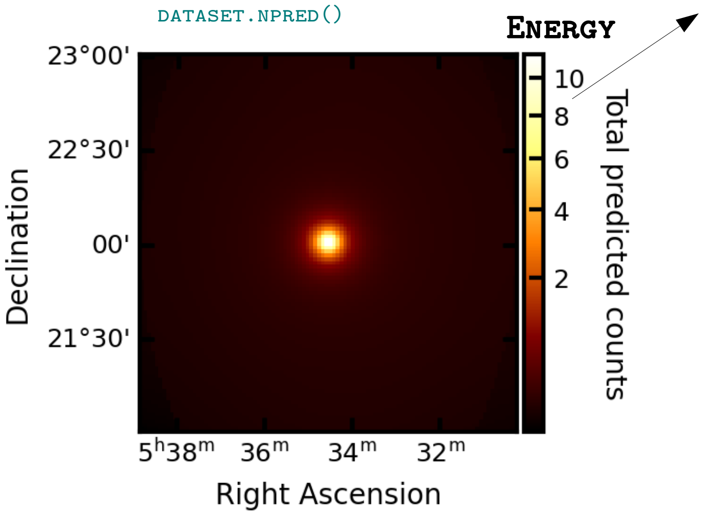


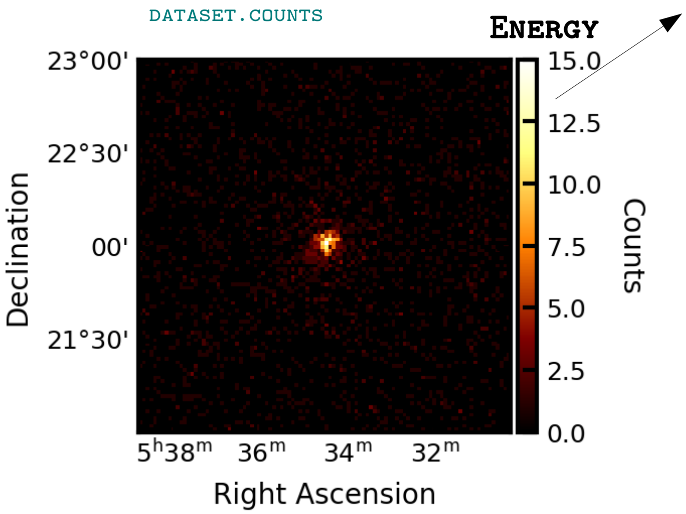
What if the region is complicated?

3HAWC Catalog



Fitting spectra and morphology at once – 3D analysis

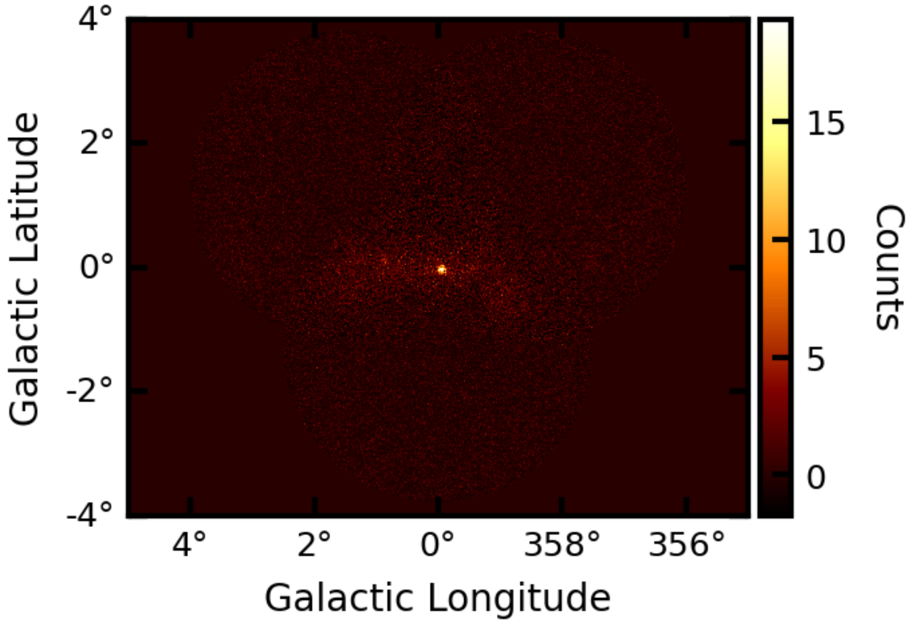


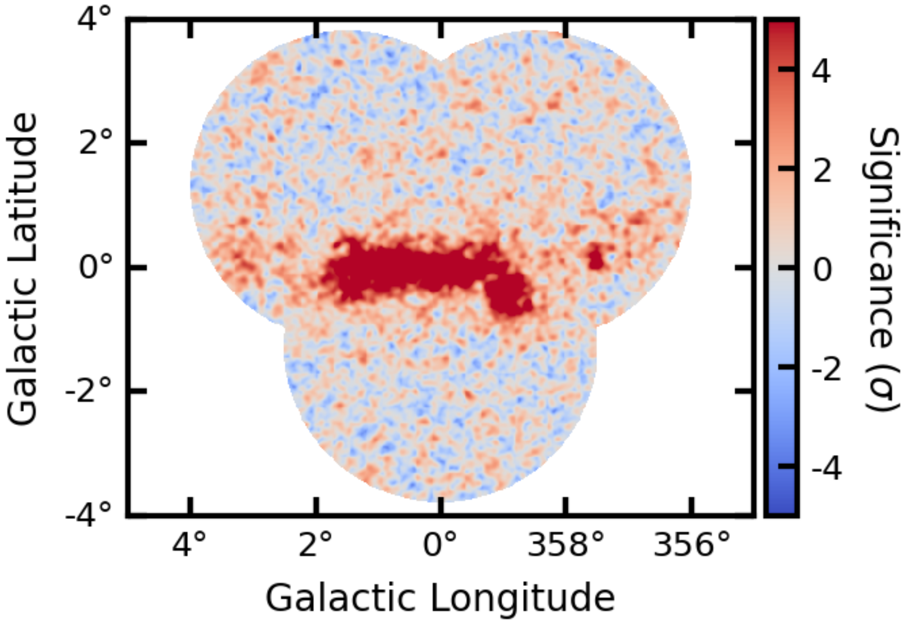


Now we can fit a model to the data by comparing prediction to observation!

Fitting a complicated region

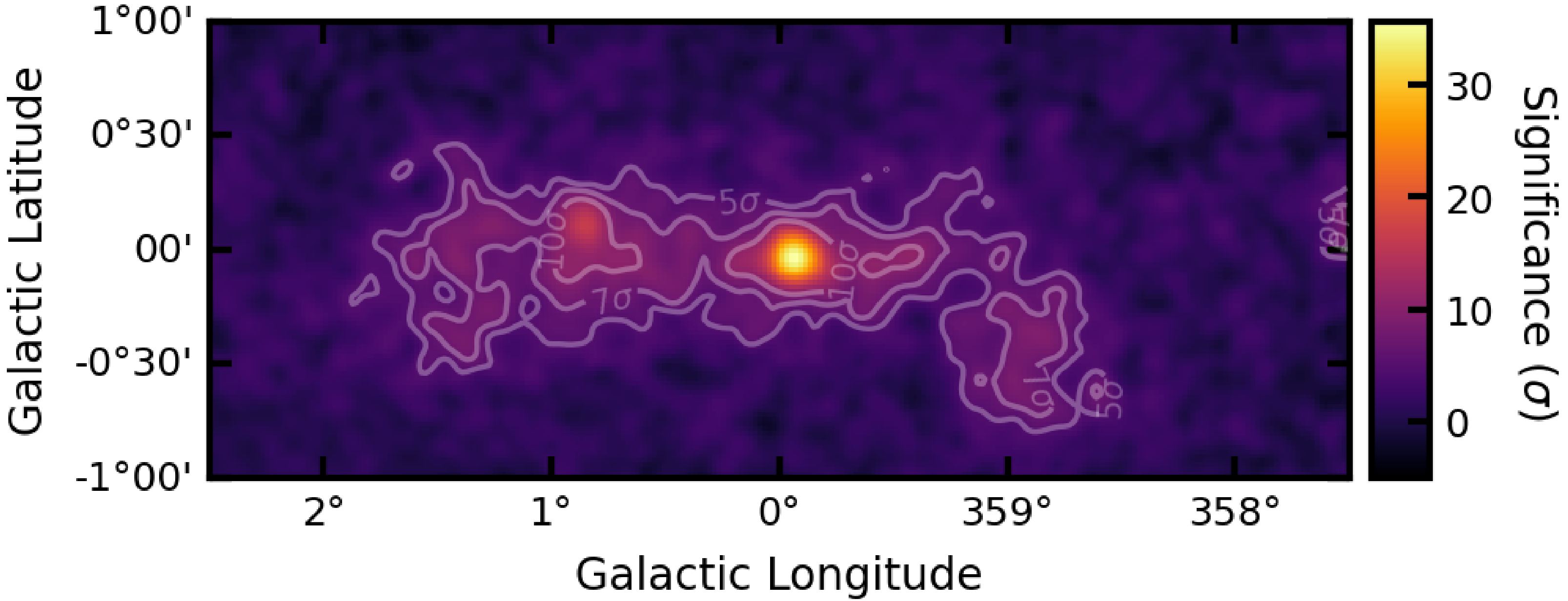
Using simulated CTA observations of the Galactic center





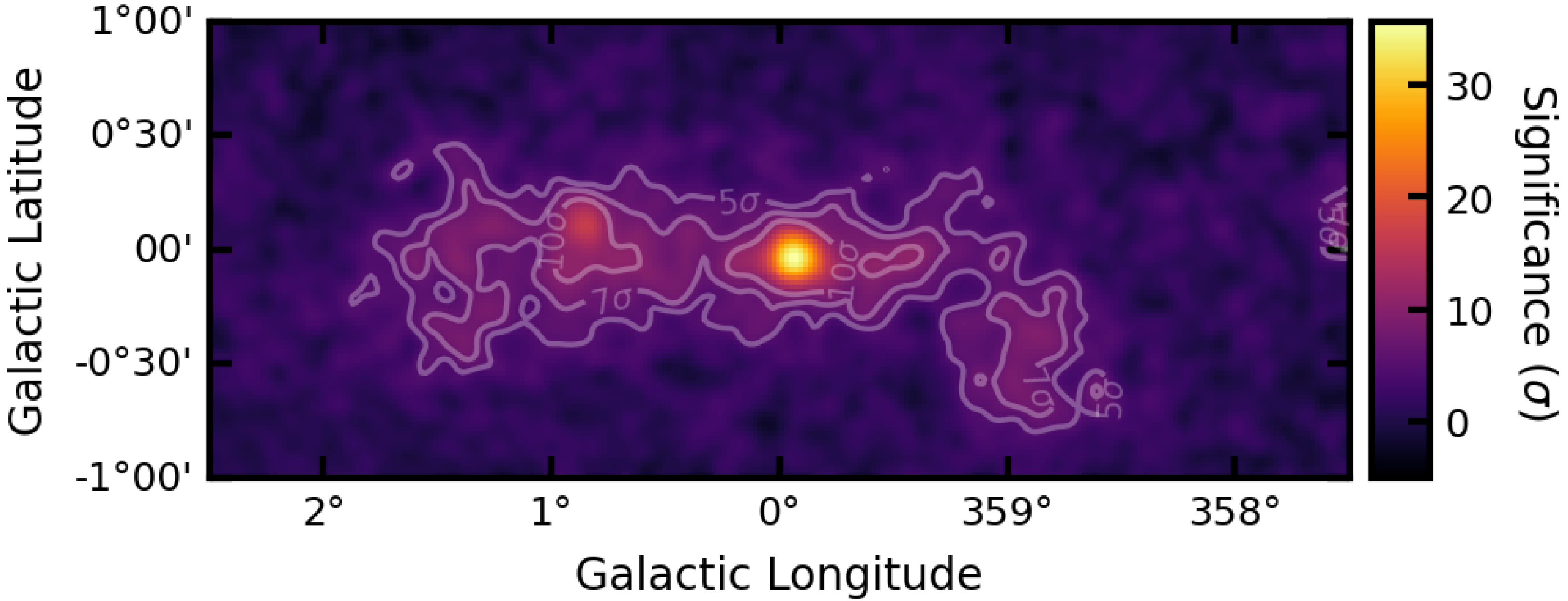
Fitting a complicated region

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

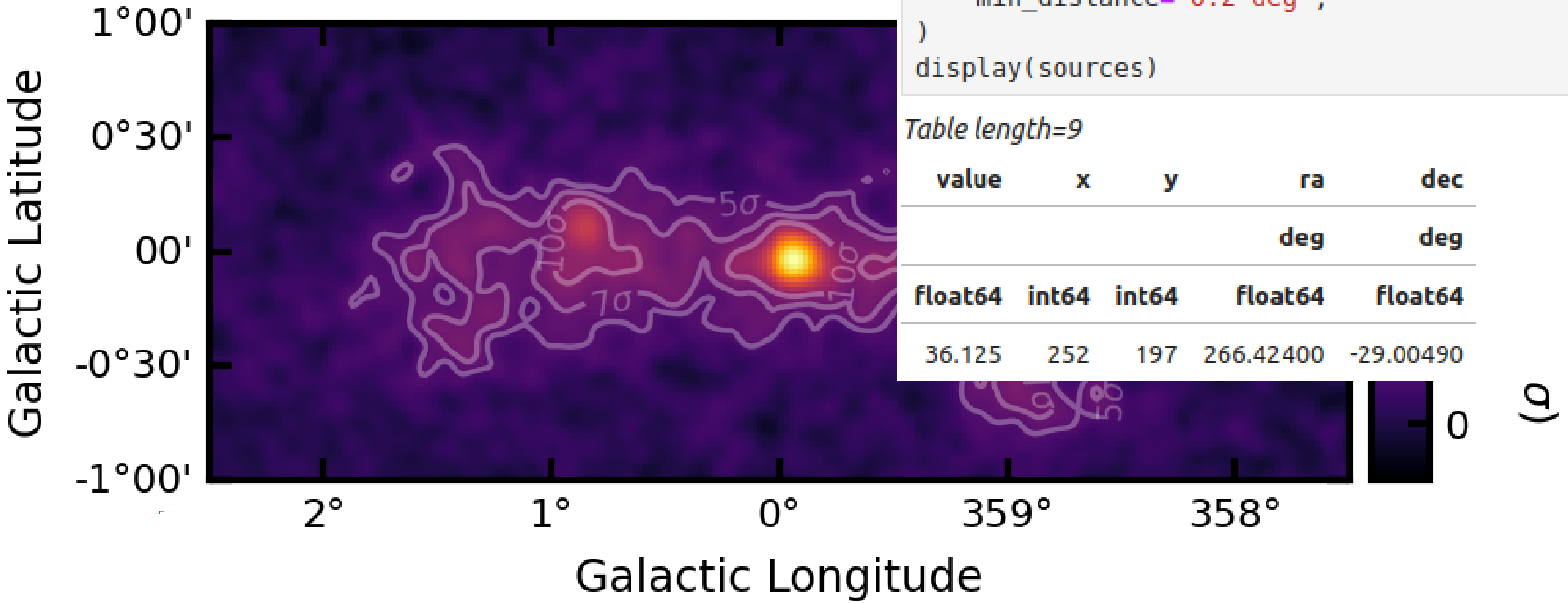
Alternative: Find peaks in TS map and start with already a handful of sources, and then iteratively make the model more complex



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)
```

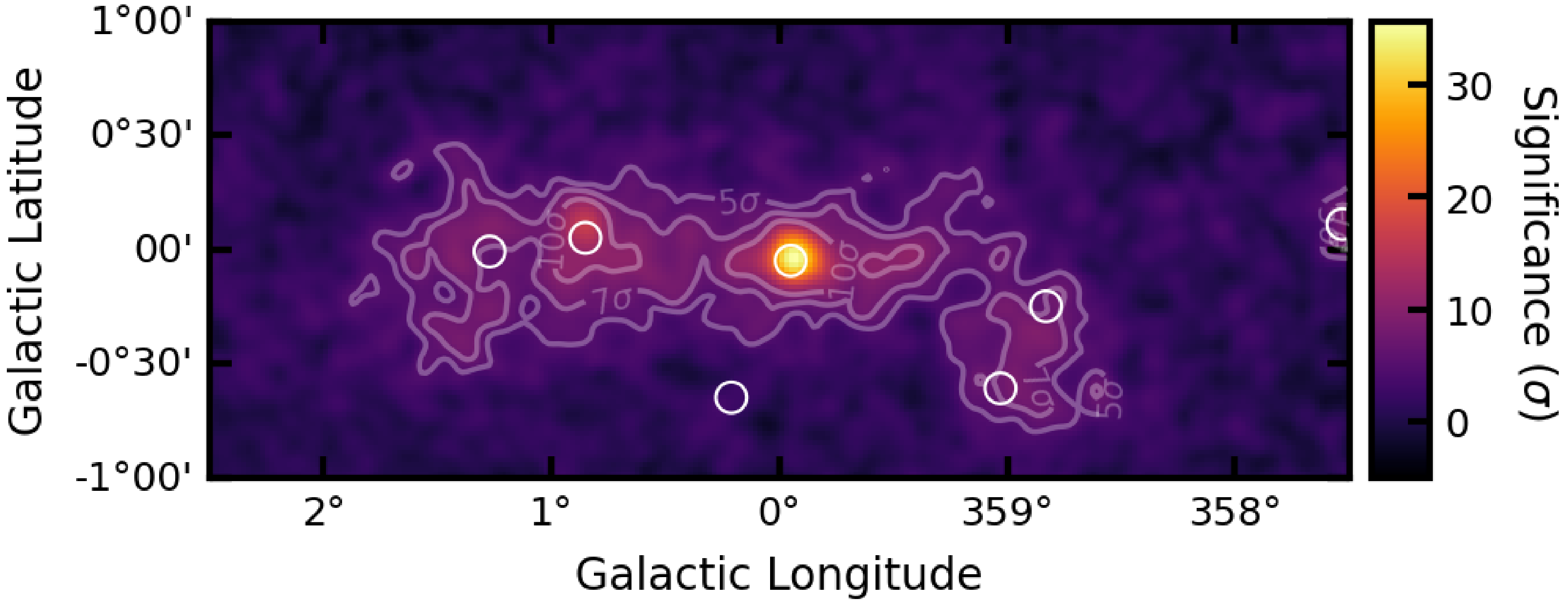
Table length=9

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]

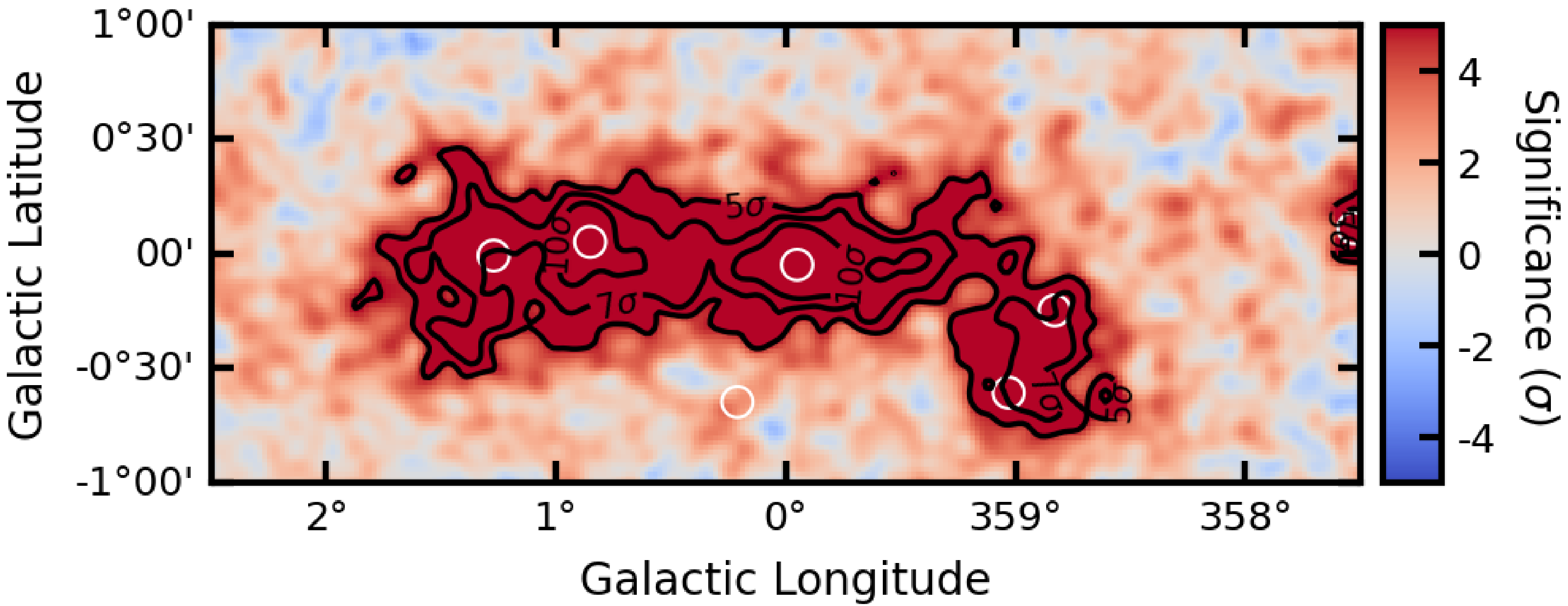
Alternative: Find peaks in TS map and start with already a handful of sources, and then iteratively make the model more complex



Fitting a complicated region

[LINK TO TUTORIAL]

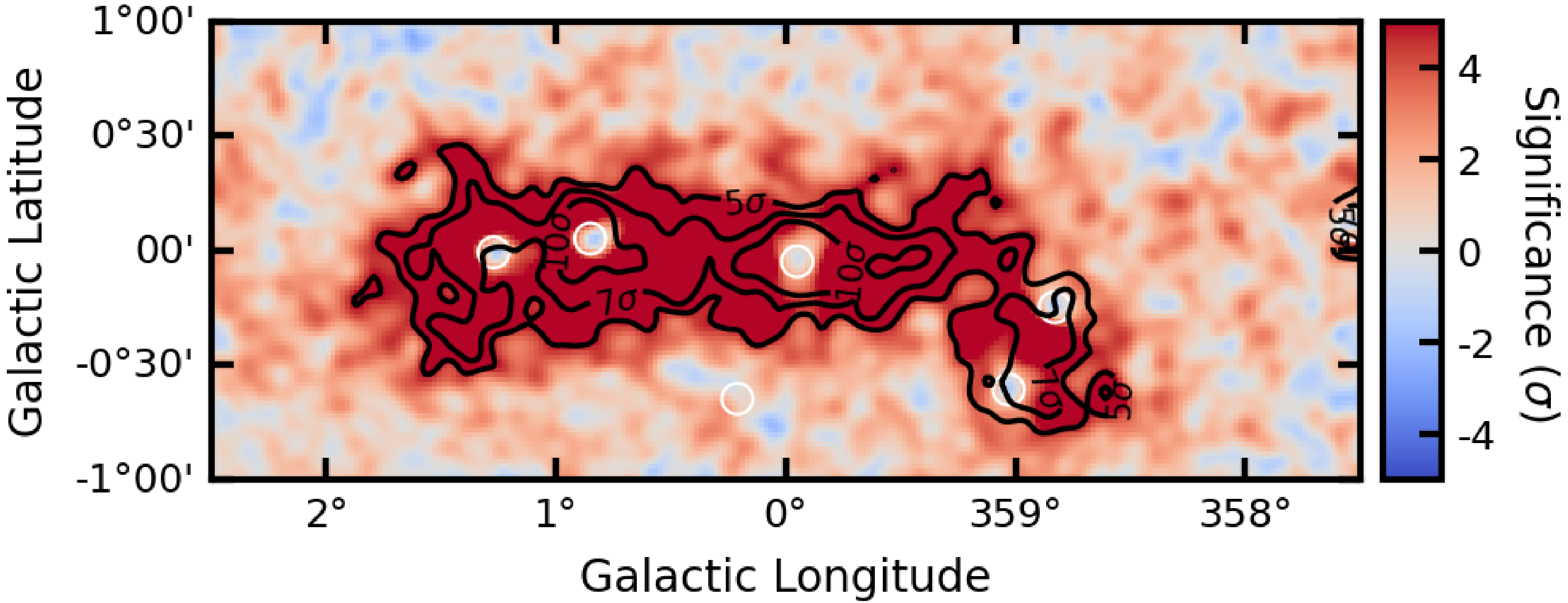
Alternative: Find peaks in TS map and start with already a handful of sources, and then iteratively make the model more complex



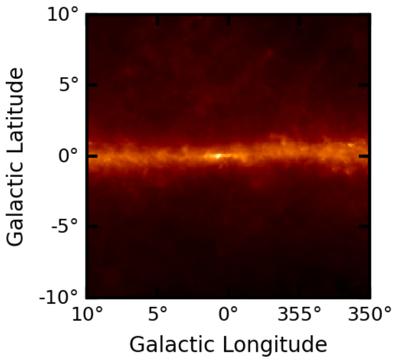
Fitting a complicated region

[LINK TO TUTORIAL]

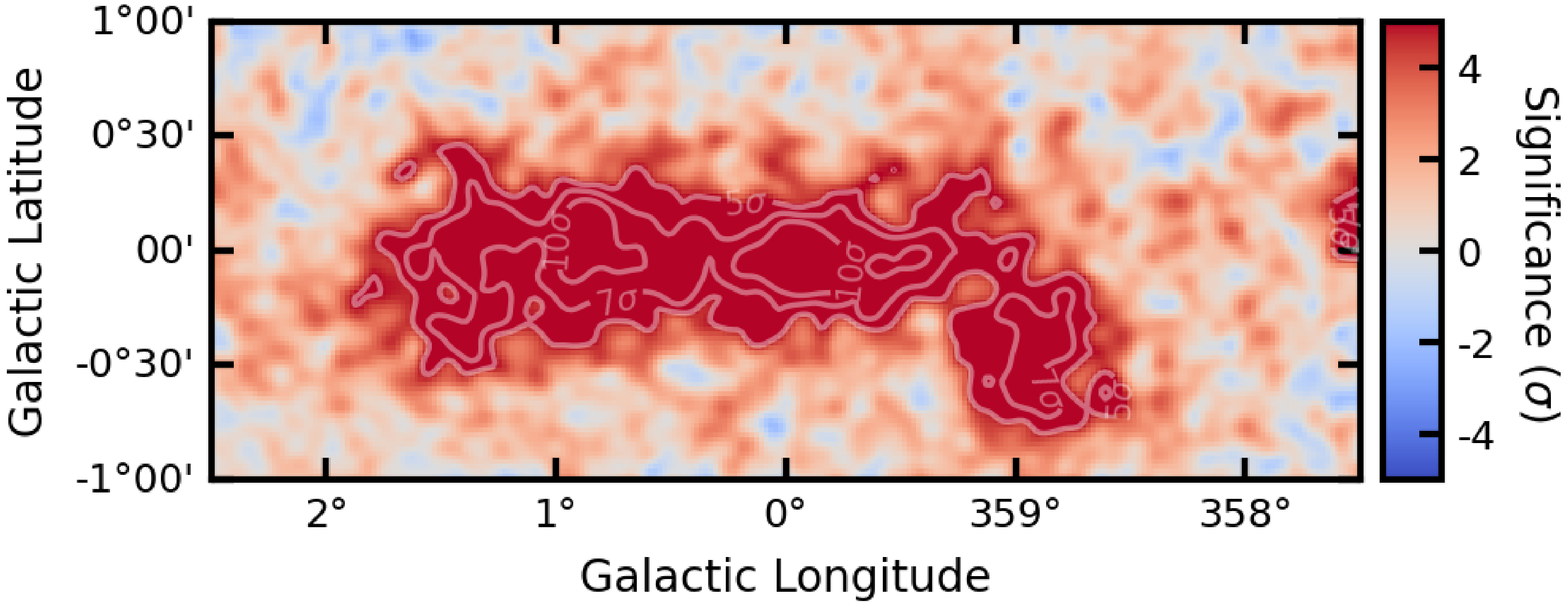
Alternative: Find peaks in TS map and start with already a handful of sources, and then iteratively make the model more complex



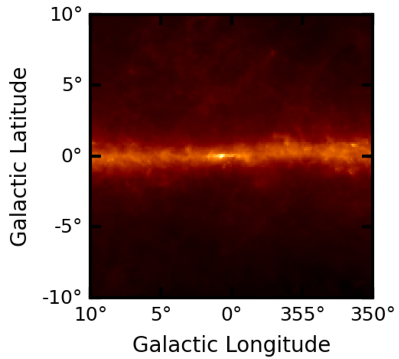
Fitting a complicated region



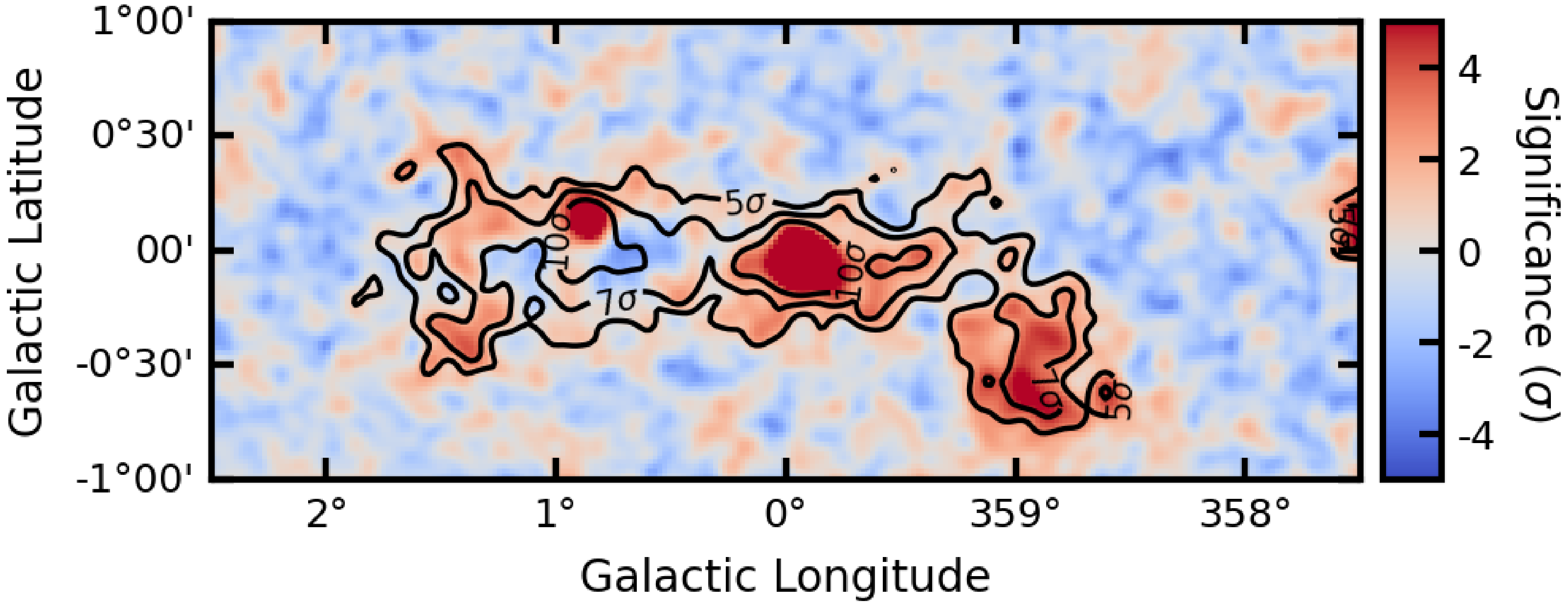
Models don't need to be analytical!



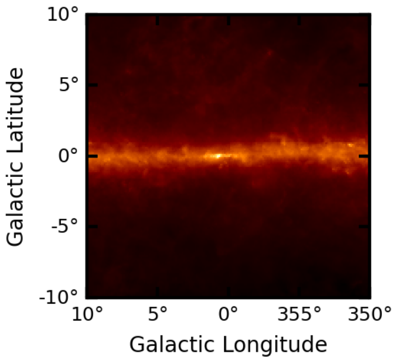
Fitting a complicated region



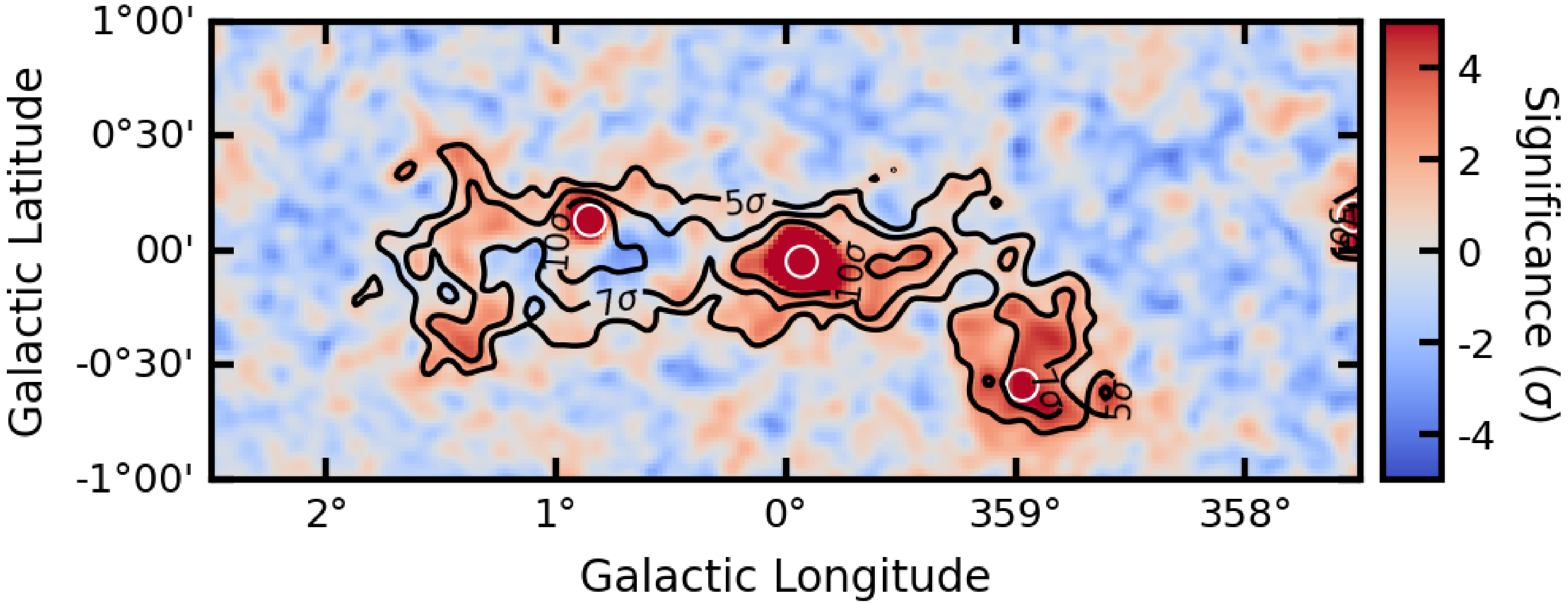
Models don't need to be analytical!



Fitting a complicated region

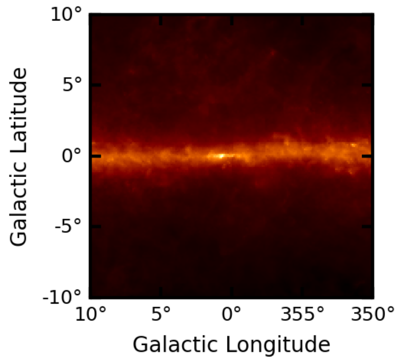


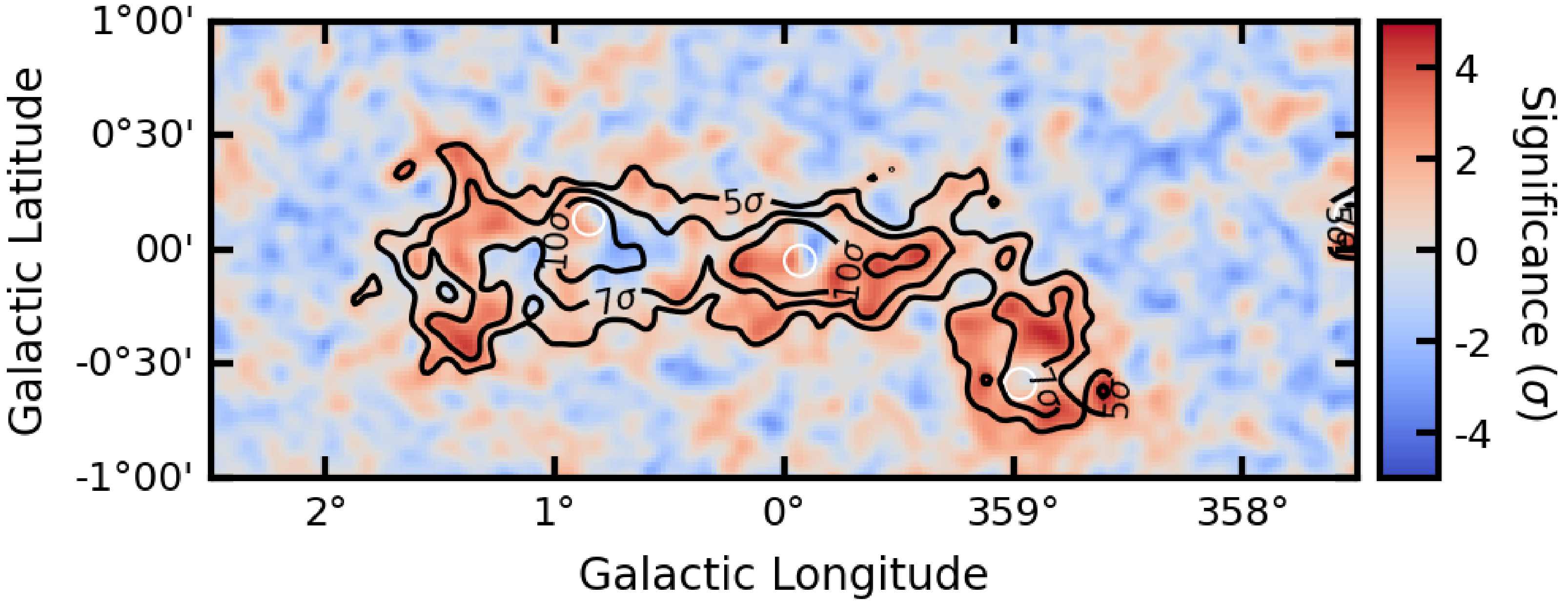
Models don't need to be analytical!



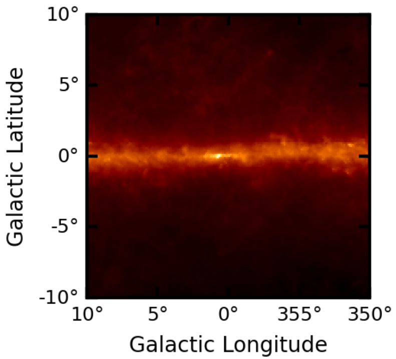
Fitting a complicated region

Models don't need to be analytical!

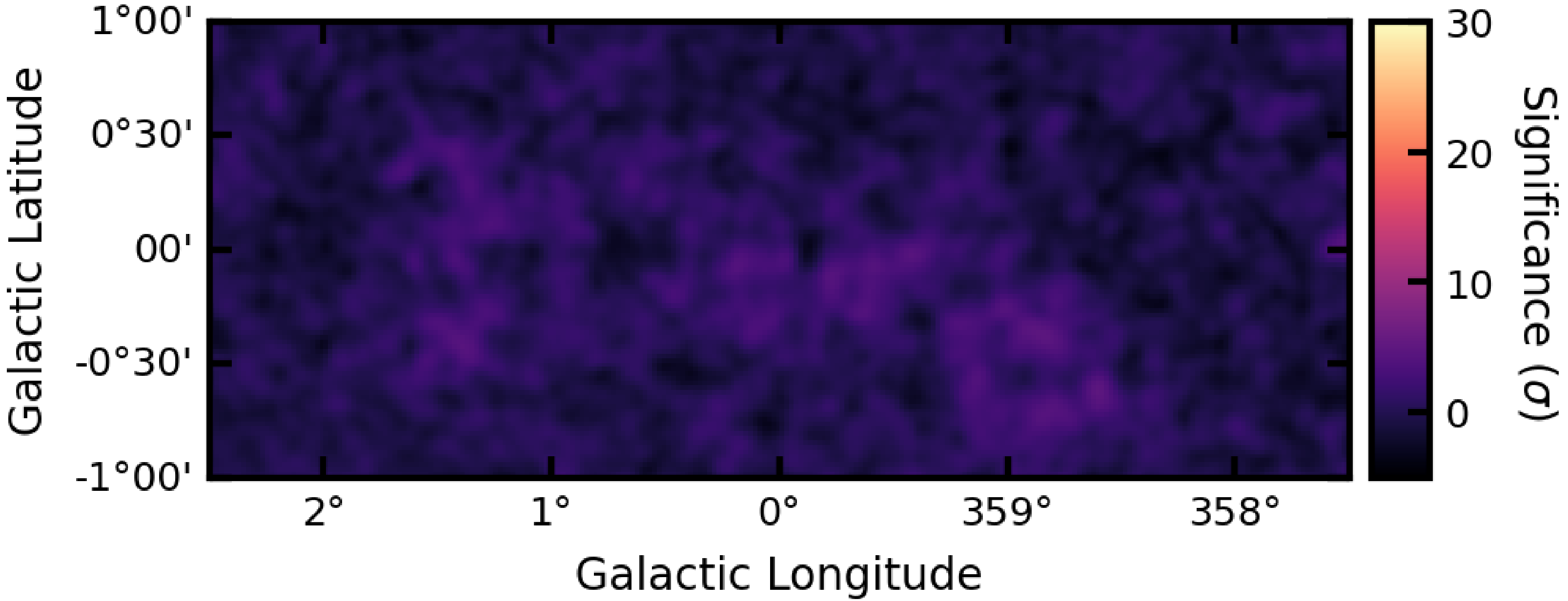




Fitting a complicated region



BE CAREFUL WITH YOUR COLORMAPS!



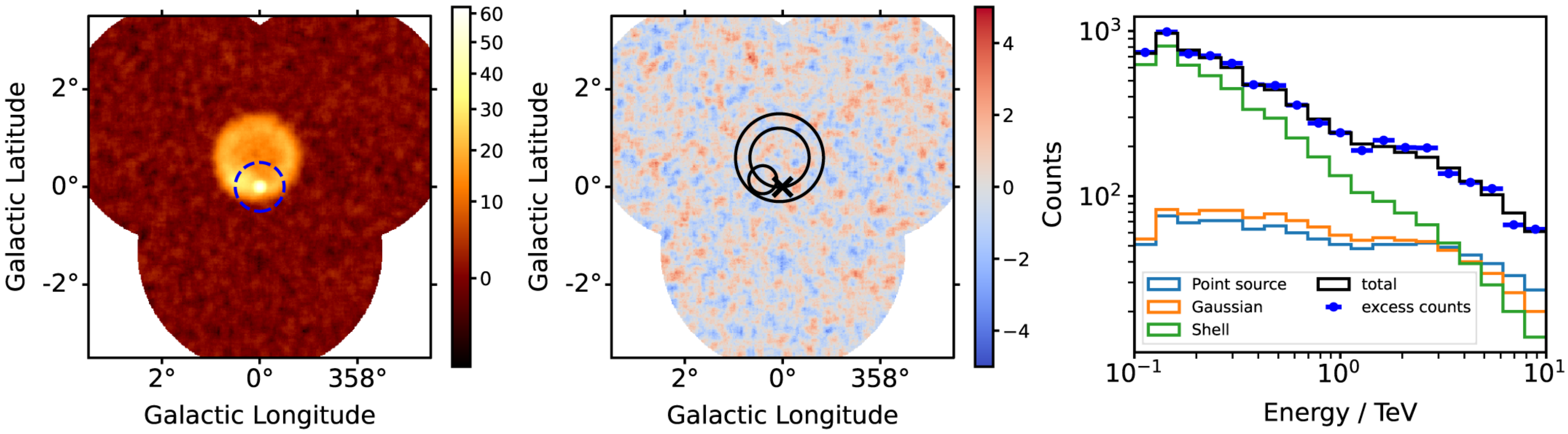
Power of 3D analysis

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

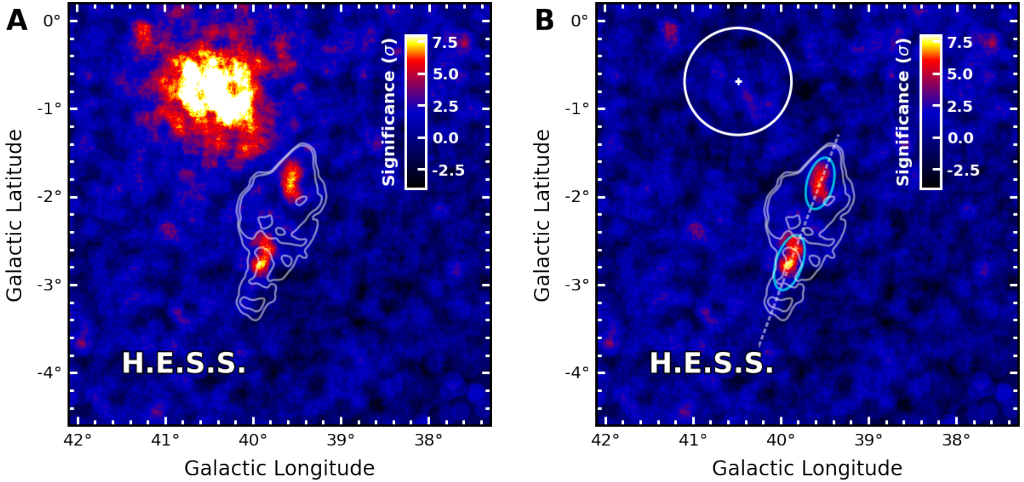
A shell with power law spectrumm

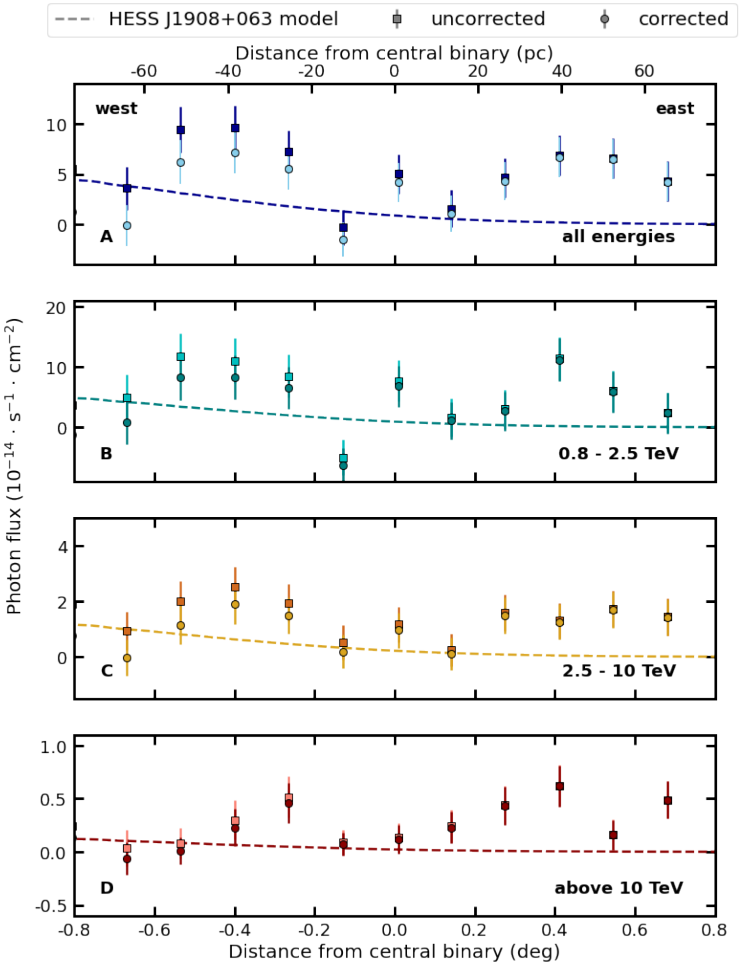


A. Donath et al

Also spatially!

H.E.S.S. Coll. 2024





[LINK TO TUTORIAL]

Data Analysis

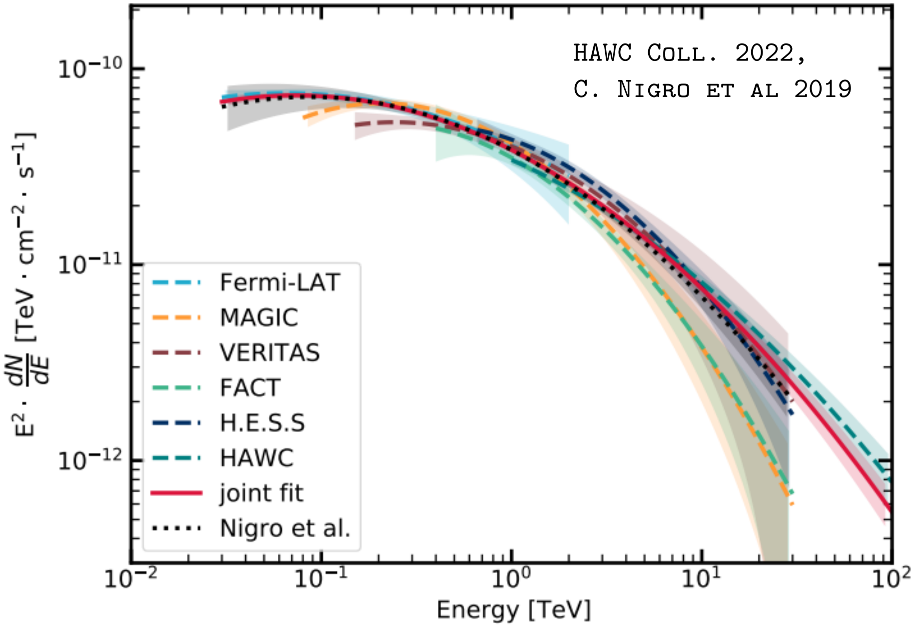
- Is there a source there?
- What are its properties? (spectral, spatial, temporal*)
- How to present results
- **Combining data from different instruments**
- Sources of uncertainty

*if we have time

Joint Analysis

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

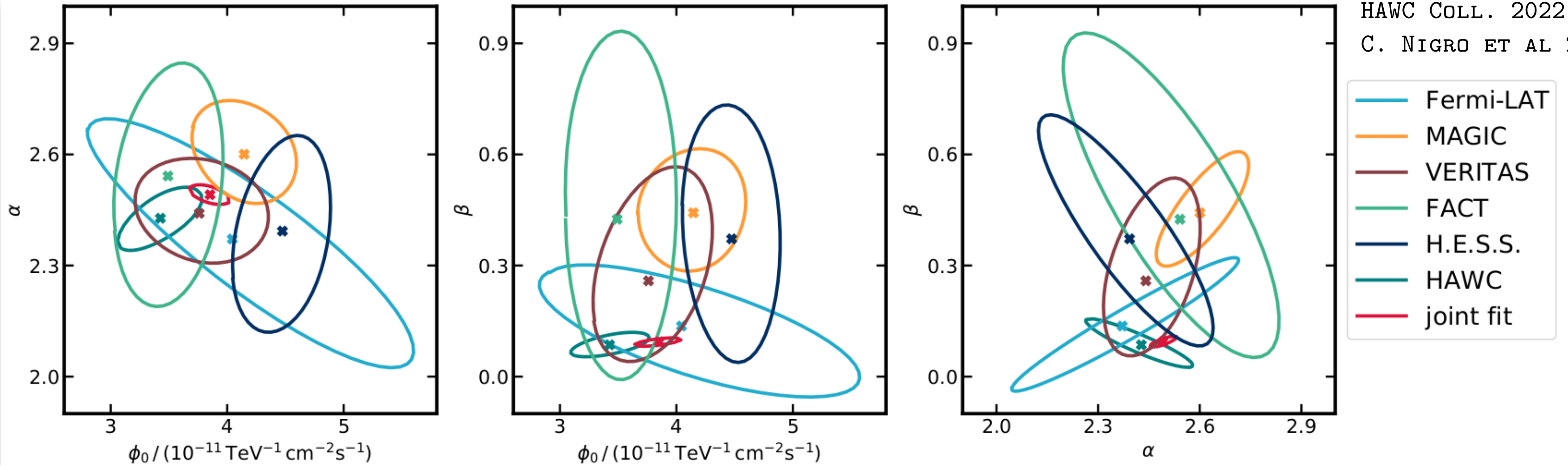
Gammapy doesn't care which instrument took the data in your dataset



Joint Analysis

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



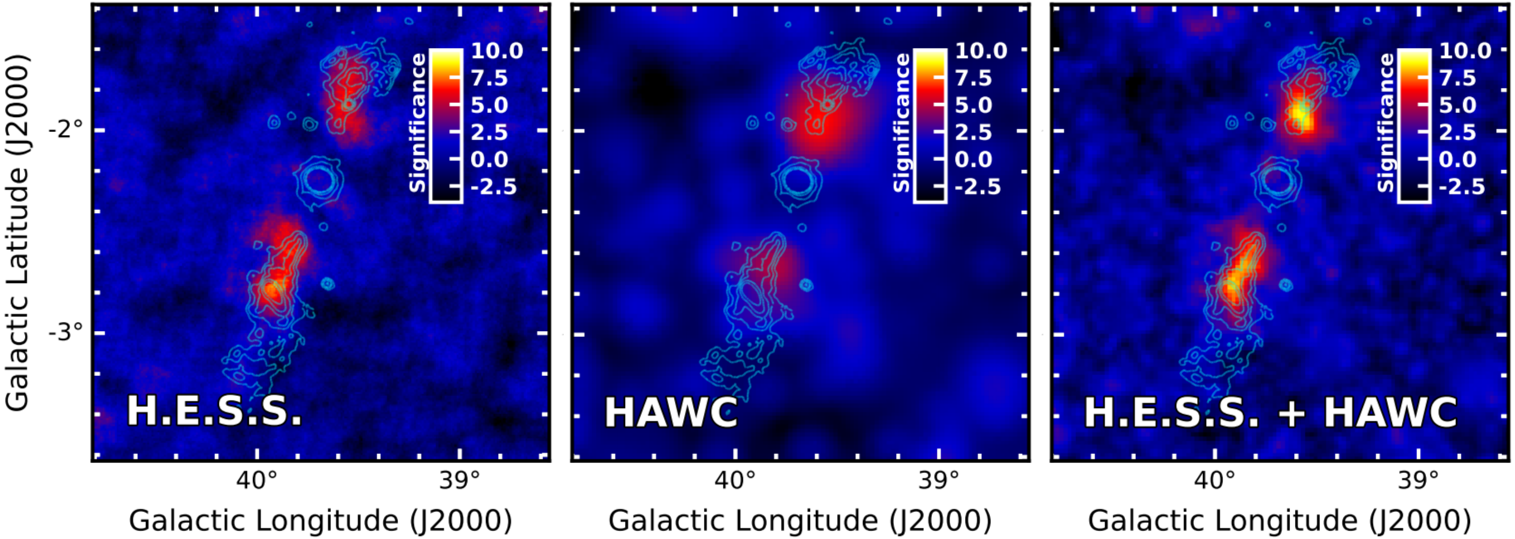
HAWC Coll. 2022,
C. Nigro et al 2019

Joint Analysis

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

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**

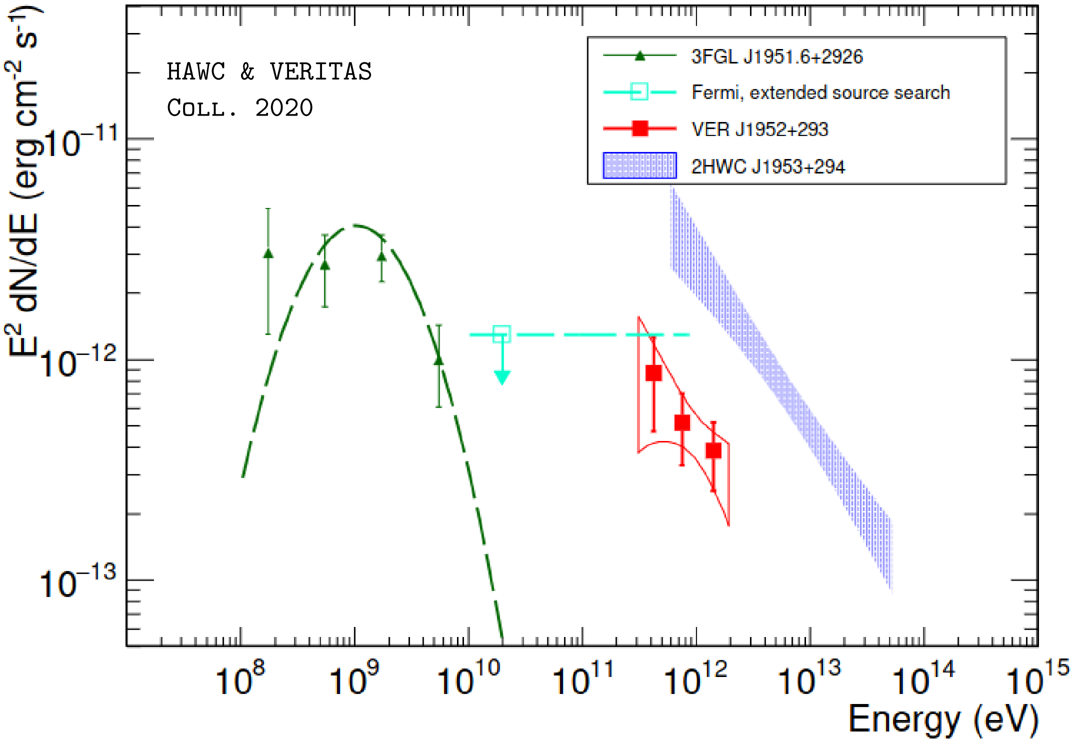
*if we have time

Systematic sources of uncertainty

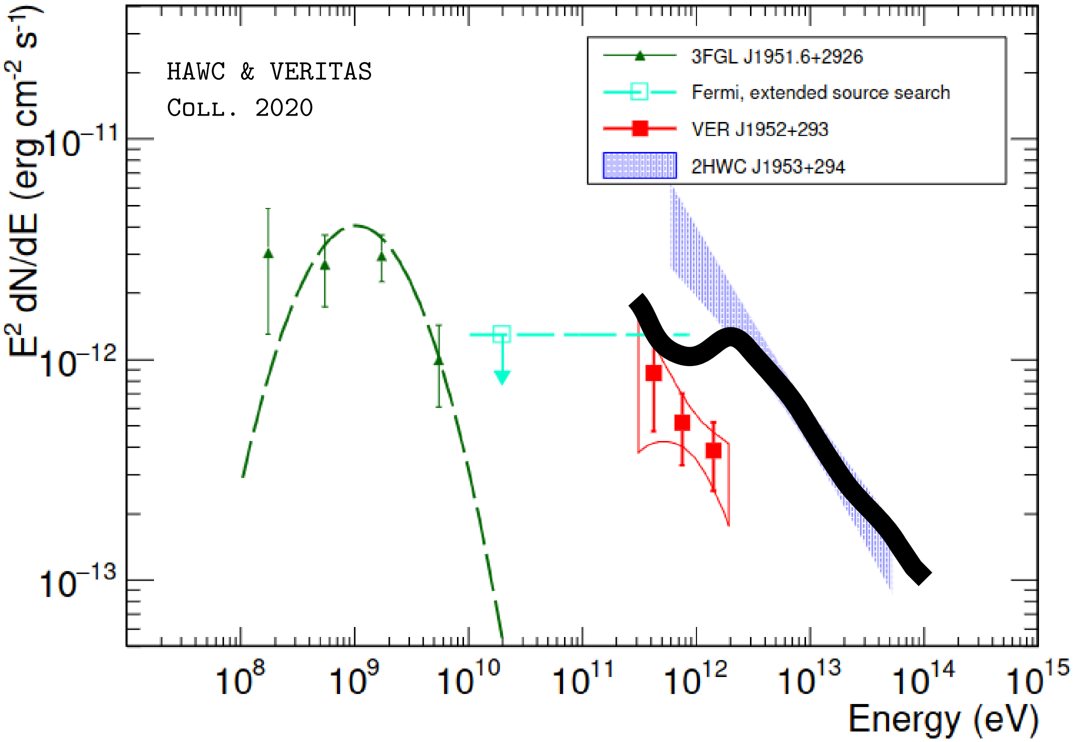
- Statistical uncertainty is reduced by taking more data
- Systematic uncertainty is **not**!
- One needs to estimate it and account for it → "reasonable guess"
- Different approaches, more instrument-dependent
- Think about things you might be getting wrong. What's their impact?
- E.g. "What if our 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

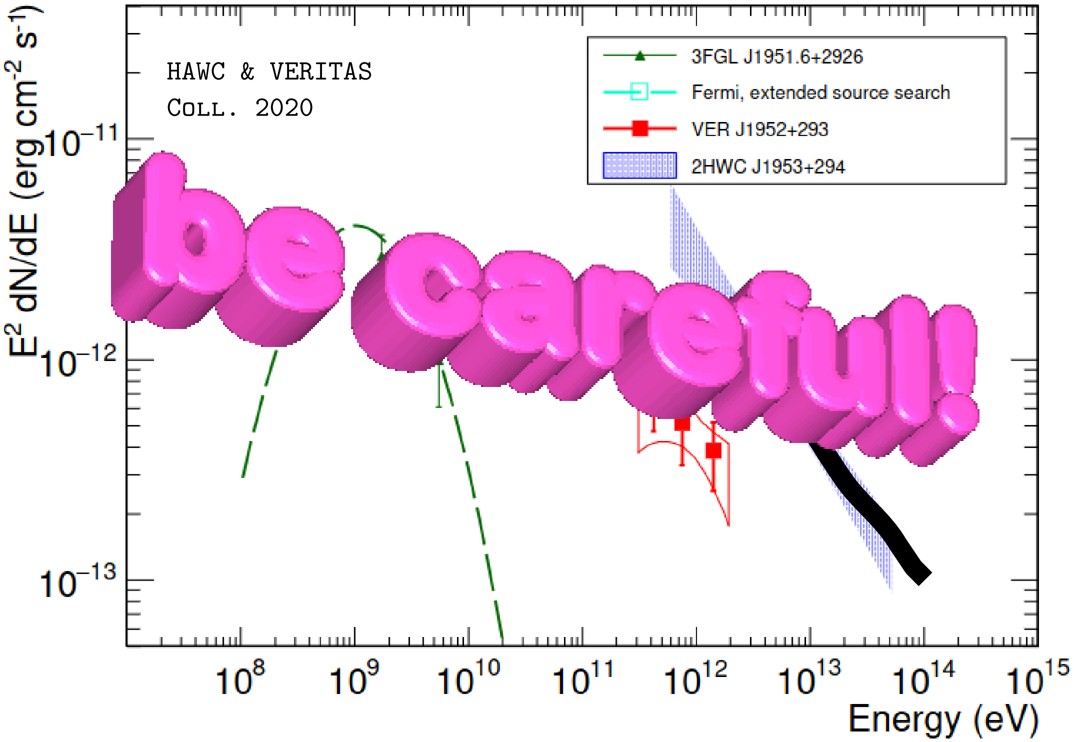
Systematic sources of uncertainty & Joint analyses



Systematic sources of uncertainty & Joint analyses



Systematic sources of uncertainty & Joint analyses



Conclusion (=random thoughts)

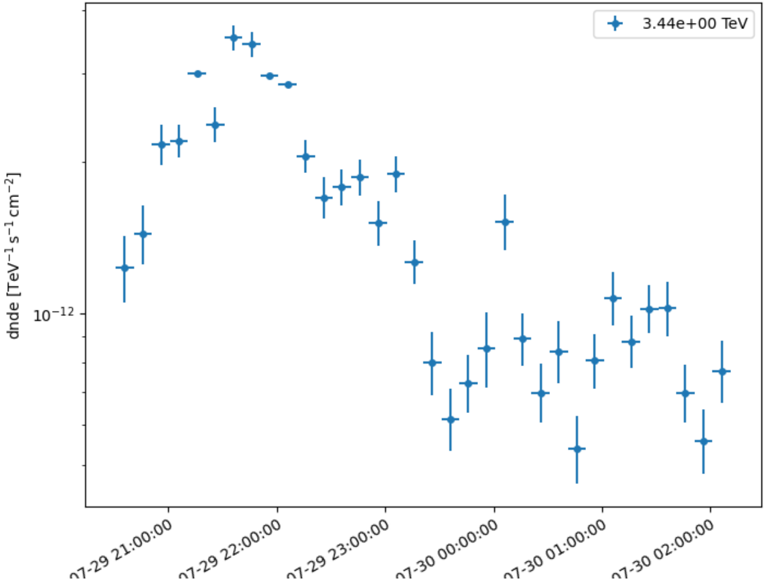
- A common data format allows for efficient data sharing, common tools and joint analysis
- Data/simulation consistency is the basis on which all of our analyses rest on
- Almost every high-level data product is produced with assumptions.
- Do not ignore systematics!!!!
- make as many sanity checks and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

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!

PLEASE PLEASE INSTALL THE SOFTWARE AHEAD OF TOMORROW!!!!
(SEE EMAIL FROM BRIAN LAST WEEK)

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