Obtaining_energy_flux

August 28, 2020

[1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

[2]:
```
import os
import numpy as np
import astropy.units as u
from astropy.coordinates import SkyCoord, Angle
from regions import CircleSkyRegion
from gammapy.spectrum import (
    SpectrumDatasetOnOff,
    SpectrumDataset,
    SpectrumDatasetMaker,
    FluxPointsEstimator,
    FluxPointsDataset,
    ReflectedRegionsBackgroundMaker,
    plot_spectrum_datasets_off_regions,
)
from gammapy.modeling import Fit, Parameter
from gammapy.modeling.models import (
    PowerLawSpectralModel,
    SpectralModel,
    SkyModel,
    ExpCutoffPowerLawSpectralModel,
)
from gammapy.irf import load_cta_irfs
from gammapy.data import Observation
from gammapy.maps import MapAxis
from itertools import combinations
```

[3]:
```
import scipy.stats as stats
import math
import statistics
```

[4]:
```
os.environ['CALDB'] = '/home/rishank/anaconda2/envs/cta/share/caldb/'
!echo $CALDB
!ls $CALDB
```

```
/home/rishank/anaconda2/envs/cta/share/caldb/
```

```
data
```

[5]:
```python
irfs = load_cta_irfs(
    "$CALDB/data/cta/prod3b-v2/bcf/South_z20_50h/irf_file.fits"
)
```

[6]:
```python
livetime = 8 * u.h
n_obs = 125
pointing = SkyCoord(0, 0, unit="deg", frame="galactic")
offset = 0.5 * u.deg
# Reconstructed and true energy axis
energy_axis = MapAxis.from_edges(
    np.logspace(-1.5, 2.0, 10), unit="TeV", name="energy", interp="log"
)
energy_axis_true = MapAxis.from_edges(
    np.logspace(-1.5, 2.0, 31), unit="TeV", name="energy", interp="log"
)

on_region_radius = Angle("0.11 deg")
on_region = CircleSkyRegion(center=pointing, radius=on_region_radius)
```

[7]:
```python
# Define spectral model - a simple Power Law in this case
model_simu = PowerLawSpectralModel(
    index=2.22,
    amplitude=1.289e-12 * u.Unit("cm-2 s-1 TeV-1"),
    reference=1 * u.TeV,
)
print(model_simu)
# we set the sky model used in the dataset
model = SkyModel(spectral_model=model_simu)
```

```
PowerLawSpectralModel

   name     value    error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.220e+00   nan                nan nan  False
amplitude 1.289e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
```

[8]:
```python
obs = Observation.create(pointing=pointing, livetime=livetime, irfs=irfs)
print(obs)
```

```
Info for OBS_ID = 1
- Pointing pos: RA 266.40 deg / Dec -28.94 deg
- Livetime duration: 28800.0 s


WARNING: AstropyDeprecationWarning: The truth value of a Quantity is ambiguous.
```

```
In the future this will raise a ValueError. [astropy.units.quantity]
```

[9]:
```python
# Make the SpectrumDataset
dataset_empty = SpectrumDataset.create(
    e_reco=energy_axis.edges, e_true=energy_axis_true.edges, region=on_region
)
maker = SpectrumDatasetMaker(selection=["aeff", "edisp", "background"])
dataset = maker.run(dataset_empty, obs)
```

[10]:
```python
# Set the model on the dataset, and fake
dataset.model = model
dataset.fake(random_state=42)
print(dataset)
```

```
SpectrumDataset

    Name                            : 1

    Total counts                    : 3375
    Total predicted counts          : nan
    Total background counts         : 3384.90

    Effective area min              : 3.44e+04 m2
    Effective area max              : 5.41e+06 m2

    Livetime                        : 2.88e+04 s

    Number of total bins            : 9
    Number of fit bins              : 9

    Fit statistic type              : cash
    Fit statistic value (-2 log(L)) : nan

    Number of parameters            : 0
    Number of free parameters       : 0
```

[11]:
```python
dataset_onoff = SpectrumDatasetOnOff(
    aeff=dataset.aeff,
    edisp=dataset.edisp,
    models=model,
    livetime=livetime,
    acceptance=1,
    acceptance_off=5,
)
dataset_onoff.fake(background_model=dataset.background)
```

```
print(dataset_onoff)
```

```
SpectrumDatasetOnOff

    Name                            :

    Total counts                    : 6092
    Total predicted counts          : 6221.39
    Total off counts                : 17002.00

    Total background counts         : 3400.40

    Effective area min              : 3.44e+04 m2
    Effective area max              : 5.41e+06 m2

    Livetime                        : 8.00e+00 h

    Number of total bins            : 9
    Number of fit bins              : 9

    Fit statistic type              : wstat
    Fit statistic value (-2 log(L)) : 9.53

    Number of parameters            : 3
    Number of free parameters       : 2

    Model type                      : SkyModels
    Acceptance mean:                : 1.0
```

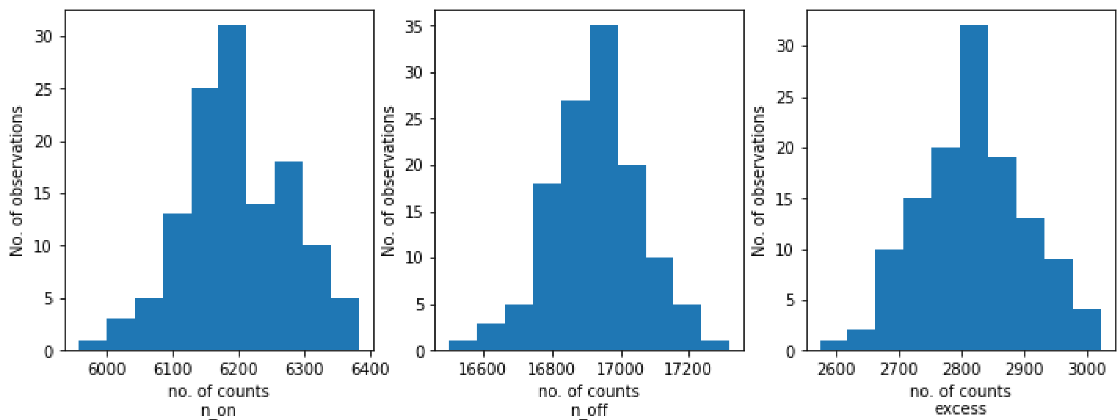
[12]:
```
%%time

datasets = []

for idx in range(n_obs):
    dataset_onoff.fake(random_state=idx, background_model=dataset.background)
    dataset_onoff.name = f"obs_{idx}"
    datasets.append(dataset_onoff.copy())
```

```
CPU times: user 1.13 s, sys: 3.12 ms, total: 1.13 s
Wall time: 1.18 s
```

[13]:
```
n_on = [dataset.counts.data.sum() for dataset in datasets]
n_off = [dataset.counts_off.data.sum() for dataset in datasets]
excess = [dataset.excess.data.sum() for dataset in datasets]

fix, axes = plt.subplots(1, 3, figsize=(12, 4))
```

```
axes[0].hist(n_on)
axes[0].set_xlabel("no. of counts\nn_on")
axes[0].set_ylabel("No. of observations")
axes[1].hist(n_off)
axes[1].set_xlabel("no. of counts\nn_off")
axes[1].set_ylabel("No. of observations")
axes[2].hist(excess)
axes[2].set_xlabel("no. of counts\nexcess");
axes[2].set_ylabel("No. of observations")
```

[13]: Text(0, 0.5, 'No. of observations')



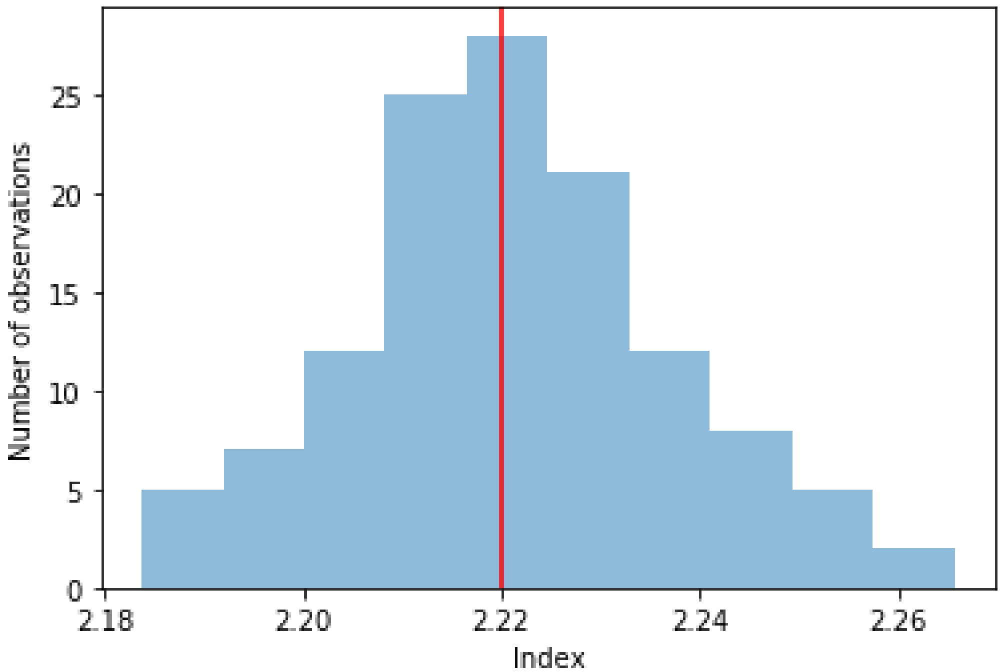
[14]:
```
%%time
e_edges = np.logspace(-1.5, 2.0, 10) * u.TeV
results = []
fpes = []
model_best_joints = []
for dataset in datasets:
    dataset.models = model.copy()
    fit = Fit([dataset])
    result = fit.run()
    results.append(
        {
            "index": result.parameters["index"].value,
            "amplitude": result.parameters["amplitude"].value,
            "reference":result.parameters["reference"].value,
        }
    )
    print(result.parameters.to_table())
    fpe = FluxPointsEstimator(datasets=[dataset], e_edges=e_edges)
    flux_points = fpe.run()
    print(flux_points.table_formatted)
```

```
10.661  6.813  16.681 …       7.933e-16       8.252e-16       7.620e-16
26.102 16.681  40.842 …       1.649e-16       1.772e-16       1.530e-16
63.908 40.842 100.000 …       3.672e-17       4.021e-17       3.340e-17
   name      value     error         unit      min max frozen
--------- --------- --------- -------------- --- --- ------
    index 2.210e+00 1.651e-02                nan nan  False
amplitude 1.254e-12 3.420e-14 cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00 0.000e+00            TeV nan nan   True
e_ref  e_min   e_max  …     dnde_err       dnde_errp       dnde_errn
 TeV    TeV     TeV   … 1 / (cm2 s TeV) 1 / (cm2 s TeV) 1 / (cm2 s TeV)
------ ------ ------- … --------------- --------------- ---------------
 0.049  0.032   0.077 …       8.904e-11       8.945e-11       8.861e-11
 0.121  0.077   0.190 …       7.891e-12       7.946e-12       7.836e-12
 0.297  0.190   0.464 …       1.053e-12       1.063e-12       1.043e-12
 0.726  0.464   1.136 …       1.586e-13       1.610e-13       1.562e-13
 1.778  1.136   2.783 …       2.221e-14       2.272e-14       2.172e-14
 4.354  2.783   6.813 …       3.816e-15       3.922e-15       3.712e-15
10.661  6.813  16.681 …       7.888e-16       8.205e-16       7.578e-16
26.102 16.681  40.842 …       1.810e-16       1.941e-16       1.685e-16
63.908 40.842 100.000 …       3.827e-17       4.177e-17       3.493e-17
CPU times: user 2min 57s, sys: 560 ms, total: 2min 57s
Wall time: 2min 59s
```

[15]:

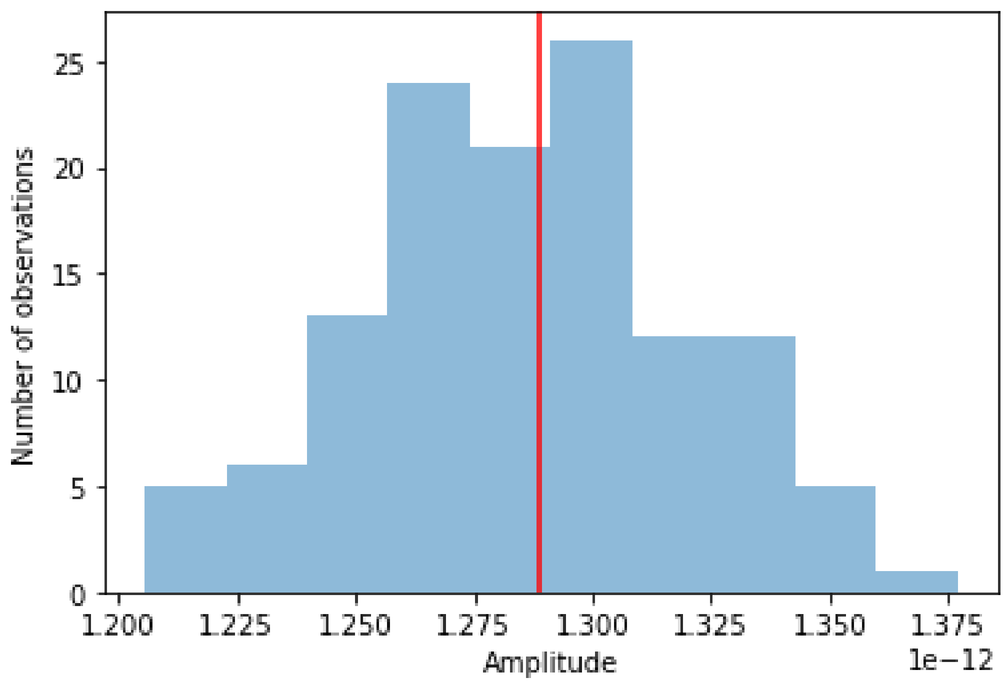
```python
index = np.array([_["index"] for _ in results])
plt.hist(index, bins=10, alpha=0.5)
plt.axvline(x=model_simu.parameters["index"].value, color="red")
plt.xlabel('Index')
plt.ylabel('Number of observations')
print(f"index: {index.mean()} += {index.std()}")
```

```
index: 2.220852816179499 += 0.016172109015838997
```



[16]:
```
amplitude = np.array([_["amplitude"] for _ in results])
plt.hist(amplitude, bins=10, alpha=0.5)
plt.axvline(x=model_simu.parameters["amplitude"].value, color="red")
plt.xlabel('Amplitude')
plt.ylabel('Number of observations')
print(f"amplitude: {amplitude.mean()} += {amplitude.std()}")
```

```
amplitude: 1.2855551328222816e-12 += 3.4620322478409695e-14
```



[17]:
```
reference = np.array([_["reference"] for _ in results])
x = np.array([index, amplitude, reference])
covar=np.cov(x)
print(covar)
```

```
[[ 2.63646280e-04 -1.45497159e-16  0.00000000e+00]
 [-1.45497159e-16  1.20823259e-27  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]]
```

[18]:
```
simu = PowerLawSpectralModel(
    index=index.mean(),
    amplitude=amplitude.mean() * u.Unit("cm-2 s-1 TeV-1"),
    reference=1 * u.TeV,
)
print(simu)
```

```
PowerLawSpectralModel

   name     value    error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.221e+00   nan                nan nan  False
amplitude 1.286e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
```

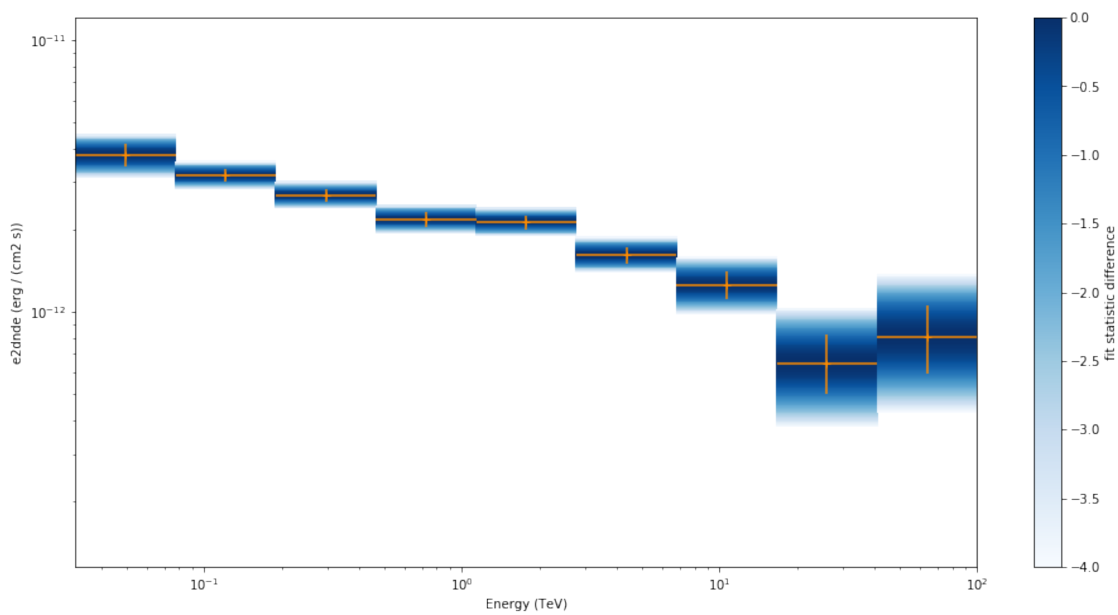
[19]:
```
i = 0
fpes[i].table_formatted
```

[19]: <Table length=9>

e_ref	e_min	e_max	…	dnde_err	dnde_errp	dnde_errn
TeV	TeV	TeV	…	1 / (cm2 s TeV)	1 / (cm2 s TeV)	1 / (cm2 s TeV)
float64	float64	float64	…	float64	float64	float64
0.049	0.032	0.077	…	8.840e-11	8.852e-11	8.829e-11
0.121	0.077	0.190	…	7.892e-12	7.931e-12	7.854e-12
0.297	0.190	0.464	…	1.055e-12	1.065e-12	1.044e-12
0.726	0.464	1.136	…	1.560e-13	1.586e-13	1.535e-13
1.778	1.136	2.783	…	2.441e-14	2.487e-14	2.396e-14
4.354	2.783	6.813	…	3.904e-15	4.001e-15	3.809e-15
10.661	6.813	16.681	…	8.213e-16	8.577e-16	7.861e-16
26.102	16.681	40.842	…	1.479e-16	1.605e-16	1.359e-16
63.908	40.842	100.000	…	3.549e-17	3.896e-17	3.218e-17

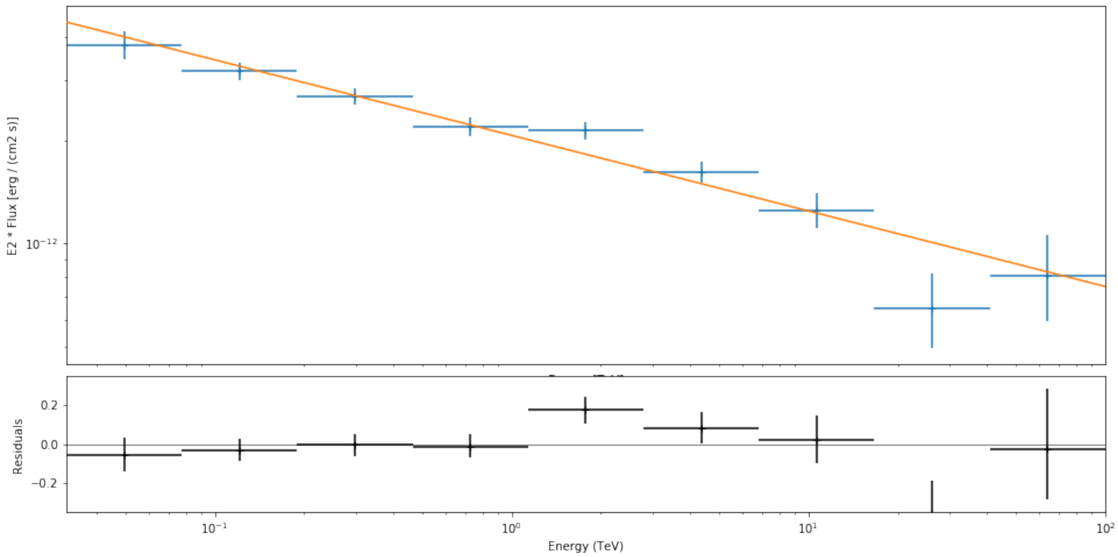
[20]:
```
plt.figure(figsize=(16, 8))
fpes[i].table["is_ul"] = fpes[i].table["ts"] < 4
ax = fpes[i].plot(
    energy_power=2, flux_unit="erg-1 cm-2 s-1", color="darkorange"
)
fpes[i].to_sed_type("e2dnde").plot_ts_profiles(ax=ax)
```

[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7f42a962c4a8>



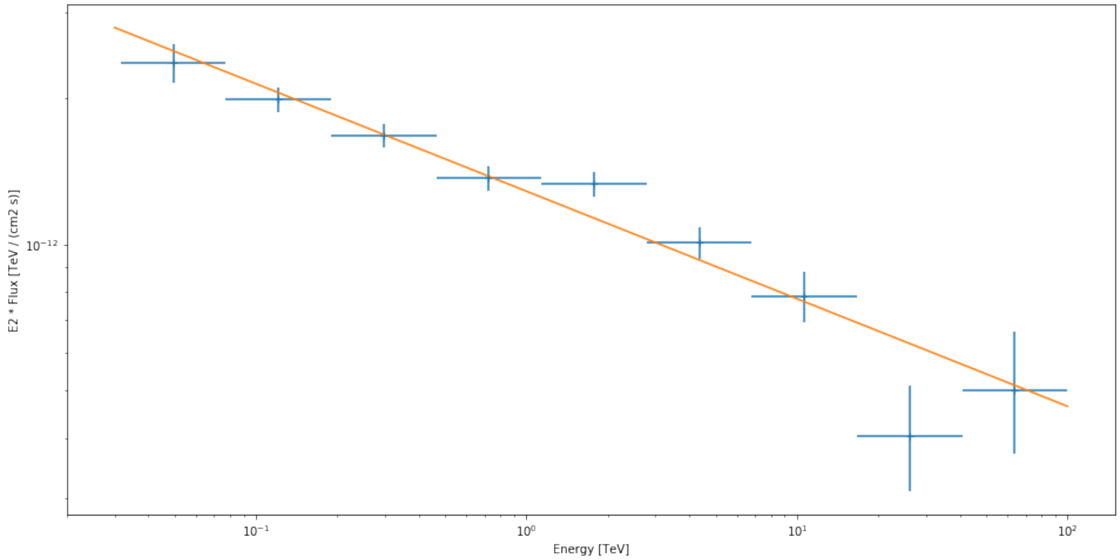
[21]:
```
flux_points_dataset = FluxPointsDataset(
    data=fpes[i], models=model_best_joints[i]
)
```

[22]:
```
plt.figure(figsize=(16, 8))
flux_points_dataset.peek();
```



[23]:
```
energy_range = [0.03, 100] * u.TeV
plt.figure(figsize=[16,8])
fpes[i].plot(energy_power=2)
simu.plot(energy_range=energy_range, energy_power=2)
plt.show
```

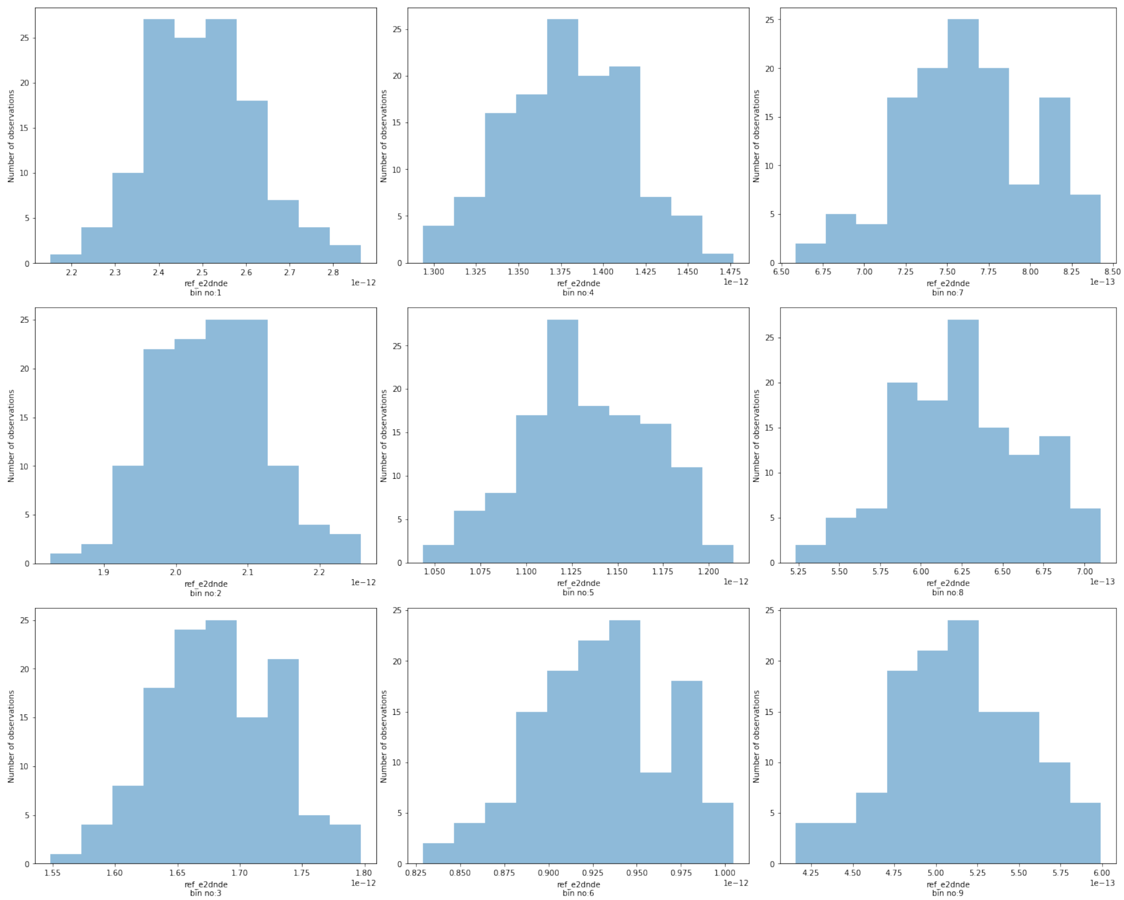
[23]: `<function matplotlib.pyplot.show(*args, **kw)>`



[24]:

```python
fig = plt.figure(figsize=[20,16],constrained_layout=True)

import matplotlib.gridspec as gridspec

gs0 = gridspec.GridSpec(1, 3, figure=fig)

gs1 = gridspec.GridSpecFromSubplotSpec(3, 1, subplot_spec=gs0[0])
for n in range(3):
    ax = fig.add_subplot(gs1[n])
    e_ref_first = np.array([_.table['ref_e2dnde'][n] for _ in fpes])
    plt.hist(e_ref_first, bins=10, alpha=0.5)
    plt.xlabel(f'ref_e2dnde\nbin no:{n+1}')
    plt.ylabel('Number of observations')

gs2 = gridspec.GridSpecFromSubplotSpec(3, 1, subplot_spec=gs0[1])
for n in range(3):
    ax = fig.add_subplot(gs2[n])
    e_ref_first = np.array([_.table['ref_e2dnde'][n+3] for _ in fpes])
    plt.hist(e_ref_first, bins=10, alpha=0.5)
    plt.xlabel(f'ref_e2dnde\nbin no:{n+4}')
    plt.ylabel('Number of observations')

gs3 = gridspec.GridSpecFromSubplotSpec(3, 1, subplot_spec=gs0[2])
for n in range(3):
    ax = fig.add_subplot(gs3[n])
    e_ref_first = np.array([_.table['ref_e2dnde'][n+6] for _ in fpes])
    plt.hist(e_ref_first, bins=10, alpha=0.5)
    plt.xlabel(f'ref_e2dnde\nbin no:{n+7}')
```

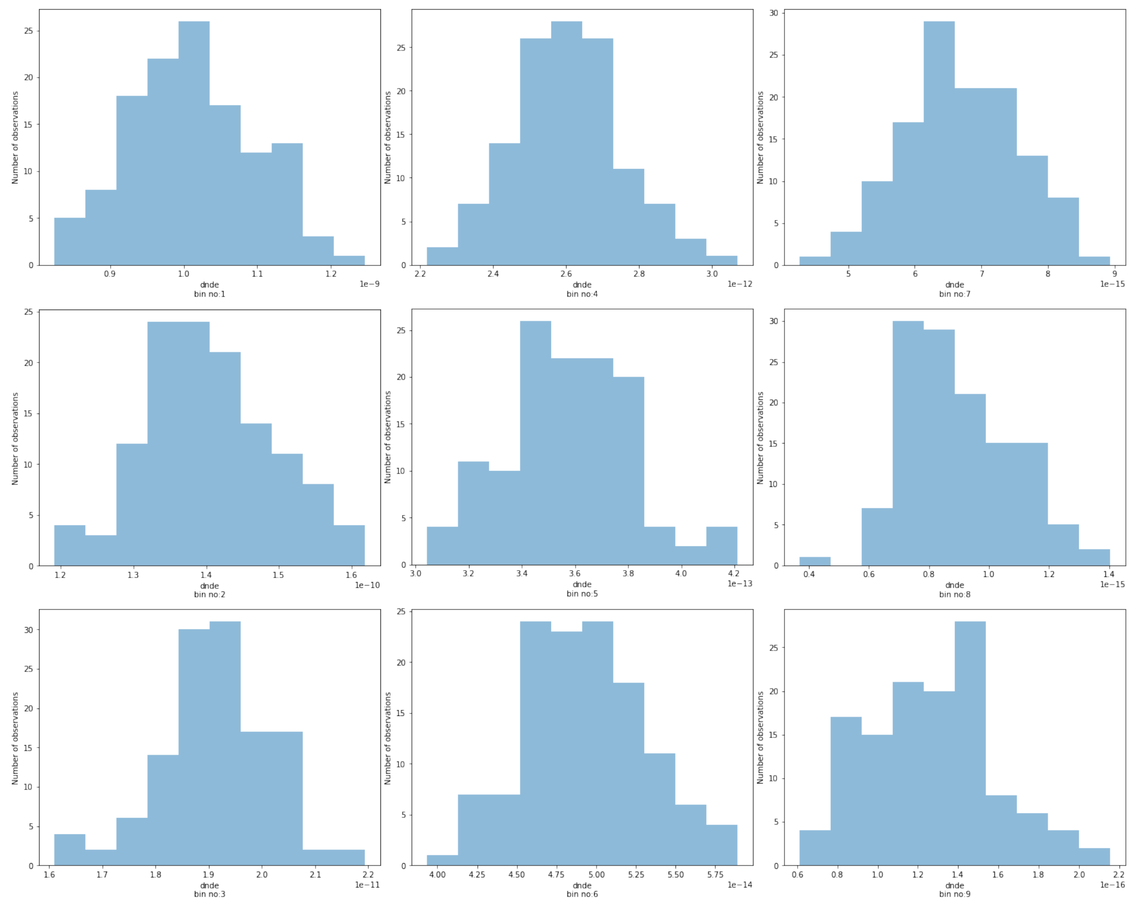
```
    plt.ylabel('Number of observations')

plt.show()
```



[25]:

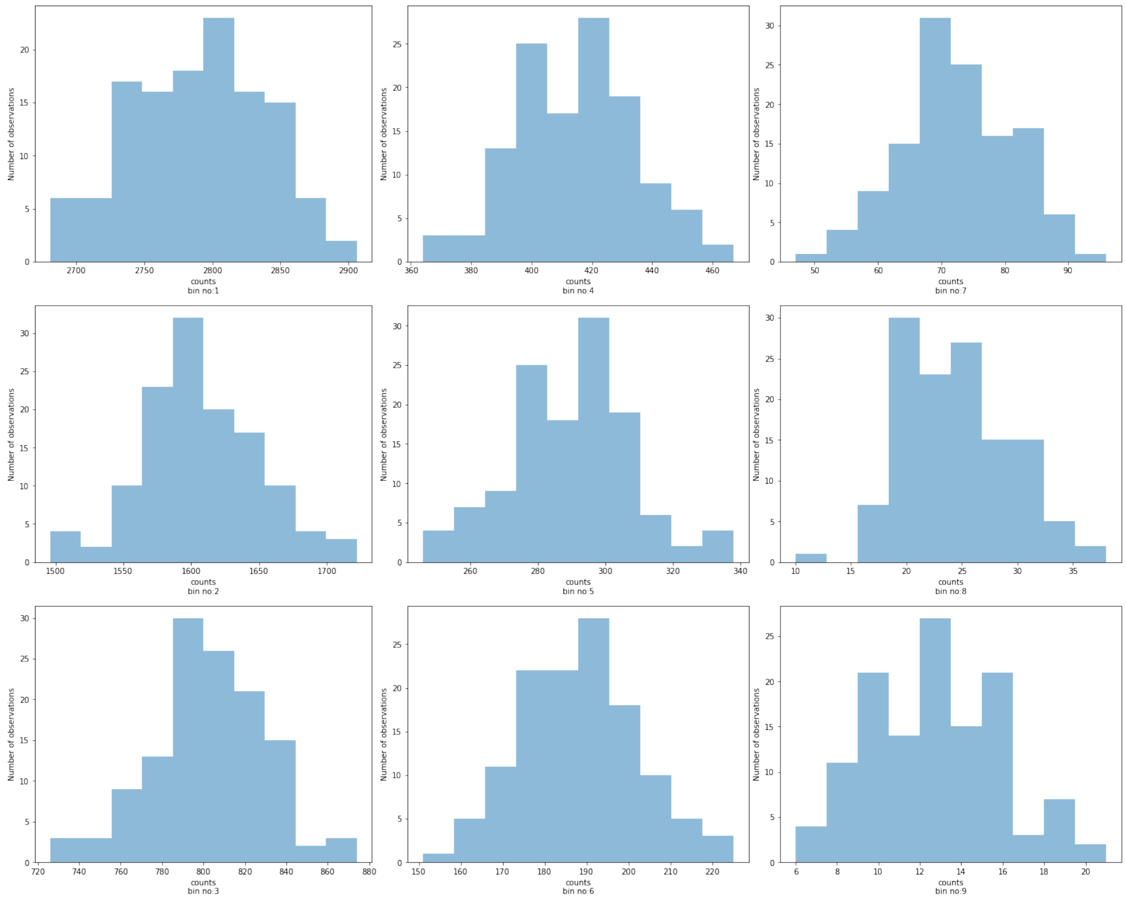
```
fig = plt.figure(figsize=[20,16],constrained_layout=True)

import matplotlib.gridspec as gridspec

gs0 = gridspec.GridSpec(1, 3, figure=fig)

gs1 = gridspec.GridSpecFromSubplotSpec(3, 1, subplot_spec=gs0[0])
for n in range(3):
    ax = fig.add_subplot(gs1[n])
    e_ref_first = np.array([_.table['dnde'][n] for _ in fpes])
    plt.hist(e_ref_first, bins=10, alpha=0.5)
    plt.xlabel(f'dnde\nbin no:{n+1}')
    plt.ylabel('Number of observations')

gs2 = gridspec.GridSpecFromSubplotSpec(3, 1, subplot_spec=gs0[1])
```

```
for n in range(3):
    ax = fig.add_subplot(gs2[n])
    e_ref_first = np.array([_.table['dnde'][n+3] for _ in fpes])
    plt.hist(e_ref_first, bins=10, alpha=0.5)
    plt.xlabel(f'dnde\nbin no:{n+4}')
    plt.ylabel('Number of observations')

gs3 = gridspec.GridSpecFromSubplotSpec(3, 1, subplot_spec=gs0[2])
for n in range(3):
    ax = fig.add_subplot(gs3[n])
    e_ref_first = np.array([_.table['dnde'][n+6] for _ in fpes])
    plt.hist(e_ref_first, bins=10, alpha=0.5)
    plt.xlabel(f'dnde\nbin no:{n+7}')
    plt.ylabel('Number of observations')

plt.show()
```



[26]:
```
fig = plt.figure(figsize=[20,16],constrained_layout=True)
```

```python
import matplotlib.gridspec as gridspec

gs0 = gridspec.GridSpec(1, 3, figure=fig)

gs1 = gridspec.GridSpecFromSubplotSpec(3, 1, subplot_spec=gs0[0])
for n in range(3):
    ax = fig.add_subplot(gs1[n])
    e_ref_first = np.array([_.table['counts'][n] for _ in fpes])
    plt.hist(e_ref_first, bins=10, alpha=0.5)
    plt.xlabel(f'counts\nbin no:{n+1}')
    plt.ylabel('Number of observations')

gs2 = gridspec.GridSpecFromSubplotSpec(3, 1, subplot_spec=gs0[1])
for n in range(3):
    ax = fig.add_subplot(gs2[n])
    e_ref_first = np.array([_.table['counts'][n+3] for _ in fpes])
    plt.hist(e_ref_first, bins=10, alpha=0.5)
    plt.xlabel(f'counts\nbin no:{n+4}')
    plt.ylabel('Number of observations')

gs3 = gridspec.GridSpecFromSubplotSpec(3, 1, subplot_spec=gs0[2])
for n in range(3):
    ax = fig.add_subplot(gs3[n])
    e_ref_first = np.array([_.table['counts'][n+6] for _ in fpes])
    plt.hist(e_ref_first, bins=10, alpha=0.5)
    plt.xlabel(f'counts\nbin no:{n+7}')
    plt.ylabel('Number of observations')

plt.show()
```



[27]:
```
x =␣
↪['ref_dnde','ref_flux','ref_eflux','ref_e2dnde','norm','stat','norm_err','counts','norm_err
```

[28]:
```
flux_points_mean = fpe.run()
flux_points_mean.table_formatted
```

[28]:
```
<Table length=9>
 e_ref   e_min   e_max  ...     dnde_err        dnde_errp       dnde_errn
  TeV     TeV     TeV   ... 1 / (cm2 s TeV) 1 / (cm2 s TeV) 1 / (cm2 s TeV)
float64 float64 float64 ...     float64         float64         float64
------- ------- ------- ... --------------- --------------- ---------------
  0.049   0.032   0.077 ...       8.903e-11       8.945e-11       8.861e-11
  0.121   0.077   0.190 ...       7.891e-12       7.946e-12       7.836e-12
  0.297   0.190   0.464 ...       1.053e-12       1.063e-12       1.043e-12
  0.726   0.464   1.136 ...       1.586e-13       1.610e-13       1.562e-13
  1.778   1.136   2.783 ...       2.221e-14       2.272e-14       2.172e-14
  4.354   2.783   6.813 ...       3.816e-15       3.922e-15       3.712e-15
 10.661   6.813  16.681 ...       7.888e-16       8.205e-16       7.578e-16
 26.102  16.681  40.842 ...       1.810e-16       1.941e-16       1.685e-16
```

```
 63.908  40.842 100.000 ...       3.827e-17       4.177e-17       3.493e-17
```

[29]:

```
for _ in x:
    y = 0
    for bin in fpes:
        y = y + bin.table[_]
    y = y/len(fpes)
    flux_points_mean.table[_] = y
```

[30]:

```
flux_points_mean.table_formatted
```

[30]:

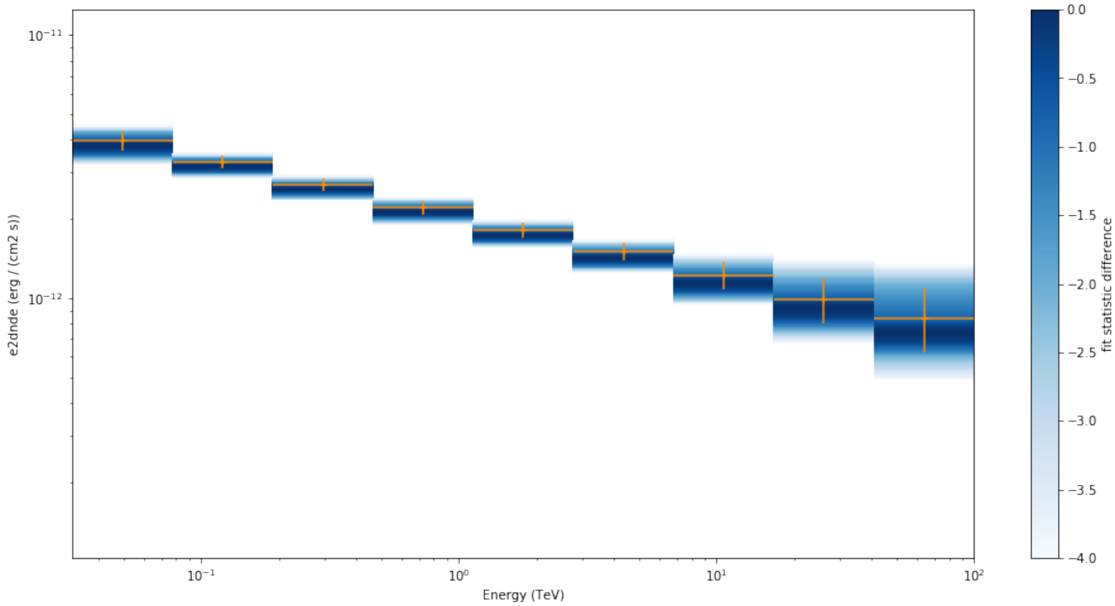
```
<Table length=9>
 e_ref   e_min   e_max  ...     dnde_err        dnde_errp       dnde_errn
  TeV     TeV     TeV   ... 1 / (cm2 s TeV) 1 / (cm2 s TeV) 1 / (cm2 s TeV)
float64 float64 float64 ...     float64         float64         float64
------- ------- ------- ... --------------- --------------- ---------------
  0.049   0.032   0.077 ...       8.936e-11       8.969e-11       8.904e-11
  0.121   0.077   0.190 ...       7.969e-12       8.011e-12       7.927e-12
  0.297   0.190   0.464 ...       1.055e-12       1.065e-12       1.046e-12
  0.726   0.464   1.136 ...       1.576e-13       1.601e-13       1.550e-13
  1.778   1.136   2.783 ...       2.271e-14       2.318e-14       2.224e-14
  4.354   2.783   6.813 ...       3.771e-15       3.870e-15       3.675e-15
 10.661   6.813  16.681 ...       8.130e-16       8.457e-16       7.810e-16
 26.102  16.681  40.842 ...       1.815e-16       1.941e-16       1.693e-16
 63.908  40.842 100.000 ...       3.584e-17       3.935e-17       3.250e-17
```

[31]:

```
plt.figure(figsize=(16, 8))
flux_points_mean.table["is_ul"] = flux_points_mean.table["ts"] < 4
ax = flux_points_mean.plot(
    energy_power=2, flux_unit="erg-1 cm-2 s-1", color="darkorange"
)
flux_points_mean.to_sed_type("e2dnde").plot_ts_profiles(ax=ax)
```

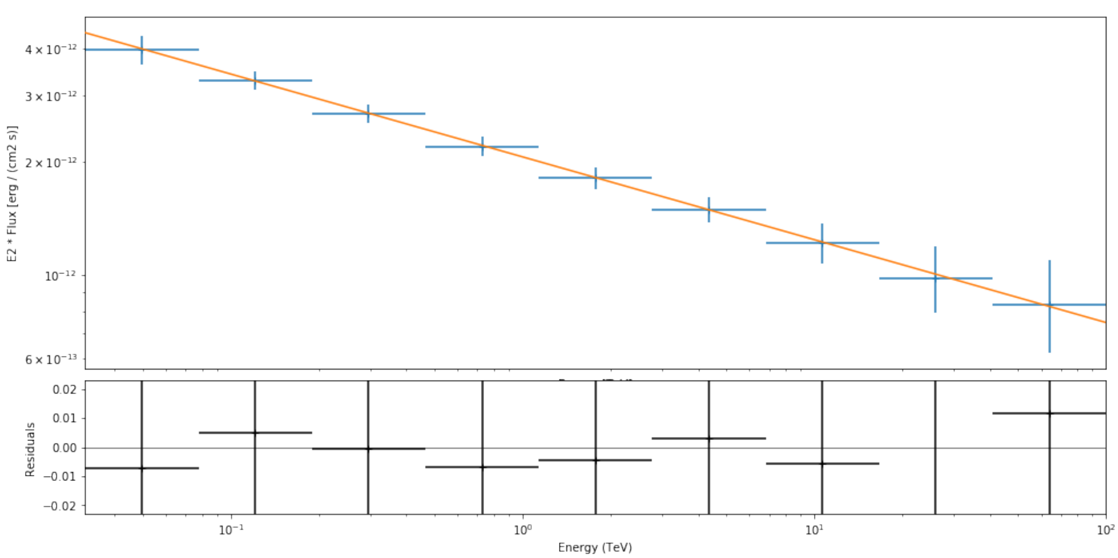
[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f42a89f2e48>
```



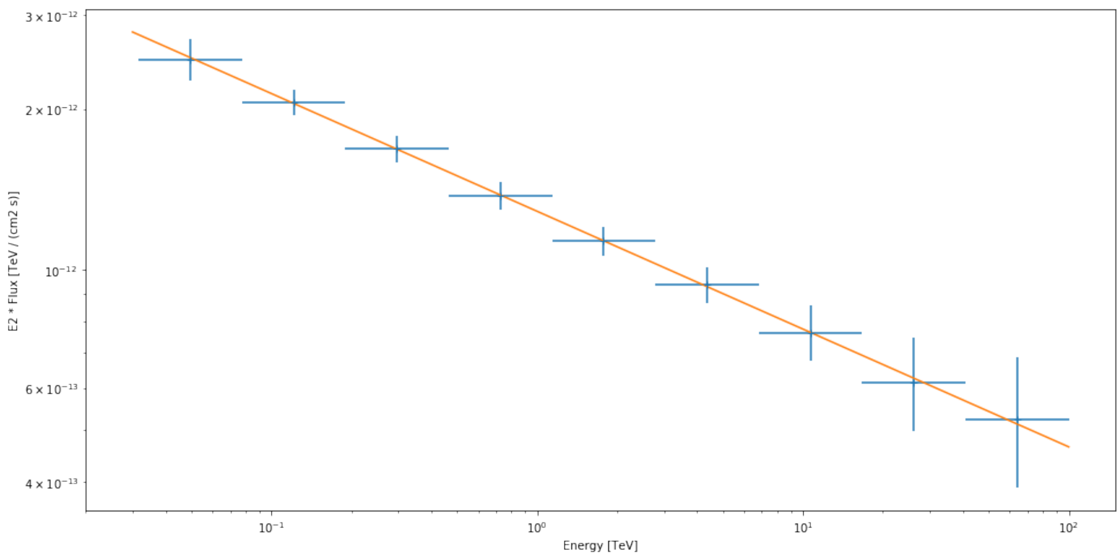

```
[32]: flux_points_dataset = FluxPointsDataset(
          data=flux_points_mean, models=model_best_joints[124]
      )
```

```
[33]: plt.figure(figsize=(16, 8))
      flux_points_dataset.peek();
```



[34]:
```
plt.figure(figsize=[16,8])
flux_points_mean.plot(energy_power=2)
simu.plot(energy_range=energy_range, energy_power=2)
plt.show
```

[34]: <function matplotlib.pyplot.show(*args, **kw)>



[ ]:
```
```