```
amplitude 1.179e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.668e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.181e+00   nan                nan nan  False
amplitude 1.169e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.047e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.317e+00   nan                nan nan  False
amplitude 1.005e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 3.616e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.160e+00   nan                nan nan  False
amplitude 1.280e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.329e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.301e+00   nan                nan nan  False
amplitude 1.151e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.140e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.171e+00   nan                nan nan  False
amplitude 1.243e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.495e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
```

[19]:
```python
index = [[0 for i in range(cols)] for j in range(rows)]
amplitude = [[0 for i in range(cols)] for j in range(rows)]
reference = [[0 for i in range(cols)] for j in range(rows)]
lambda_ = [[0 for i in range(5)] for j in range(rows)]
alpha = [[0 for i in range(5)] for j in range(rows)]
covar = [[0 for i in range(cols)] for j in range(rows)]
```

```
for i in range(8):
    a = np.array([_["index"] for _ in res[i][0]])
    b = np.array([_["amplitude"] for _ in res[i][0]])
    c = np.array([_["reference"] for _ in res[i][0]])
    index[i][0]=a
    amplitude[i][0]=b
    reference[i][0]=c
for j in range(5):
    for i in range(8):
        a = np.array([_["index"] for _ in res[i][j+1]])
        b = np.array([_["amplitude"] for _ in res[i][j+1]])
        c = np.array([_["reference"] for _ in res[i][j+1]])
        d = np.array([_["lambda_"] for _ in res[i][j+1]])
        e = np.array([_["alpha"] for _ in res[i][j+1]])
        index[i][j+1]=a
        amplitude[i][j+1]=b
        reference[i][j+1]=c
        lambda_[i][j]=d
        alpha[i][j]=e

mu = [[0 for i in range(cols)] for j in range(rows)]
sigma = [[0 for i in range(cols)] for j in range(rows)]

for i in range(8):
    x = np.array([index[i][0], amplitude[i][0], reference[i][0]])
    covar[i][0]=np.cov(x)
for j in range(5):
    for i in range(8):
        x = np.array([index[i][j+1], amplitude[i][j+1], reference[i][j+1],␣
 ↪lambda_[i][j], alpha[i][j]])
        covar[i][j+1]=np.cov(x)
```

[20]:
```
def evaluate_err(self, covar, energy, epsilon=1e-4):


        p_cov = covar
        eps = np.sqrt(np.diag(covar)) * epsilon

        df_dp = self._evaluate_gradient(energy, eps)
        f_cov = df_dp.T @ p_cov @ df_dp
        f_err = np.sqrt(np.diagonal(f_cov))

        q = self(energy)
        return u.Quantity([q.value, f_err], unit=q.unit)
```

[21]:
```
def plot_error(
        self,
        covar,
```

```
        energy_range,
        ax=None,
        energy_unit="TeV",
        flux_unit="cm-2 s-1 TeV-1",
        energy_power=0,
        n_points=100,
        **kwargs,
    ):

        ax = plt.gca() if ax is None else ax

        kwargs.setdefault("facecolor", "black")
        kwargs.setdefault("alpha", 0.2)
        kwargs.setdefault("linewidth", 0)

        emin, emax = energy_range
        energy = MapAxis.from_energy_bounds(emin, emax, n_points, energy_unit).
↪edges

        flux, flux_err = evaluate_err(self,covar,energy).to(flux_unit)

        y_lo = self._plot_scale_flux(energy, flux - flux_err, energy_power)
        y_hi = self._plot_scale_flux(energy, flux + flux_err, energy_power)

        where = (energy >= energy_range[0]) & (energy <= energy_range[1])
        ax.fill_between(energy.value, y_lo.value, y_hi.value, where=where,␣
↪**kwargs)

        self._plot_format_ax(ax, energy, y_lo, energy_power)
        return ax
```

[22]:
```
for j in range(6):
    print(f"model: {j}")
    for i in range(8):
        mu[i][j]=index[i][j].mean()
        sigma[i][j]=index[i][j].std()
        print(f"index: {index[i][j].mean()} += {index[i][j].std()}")
```

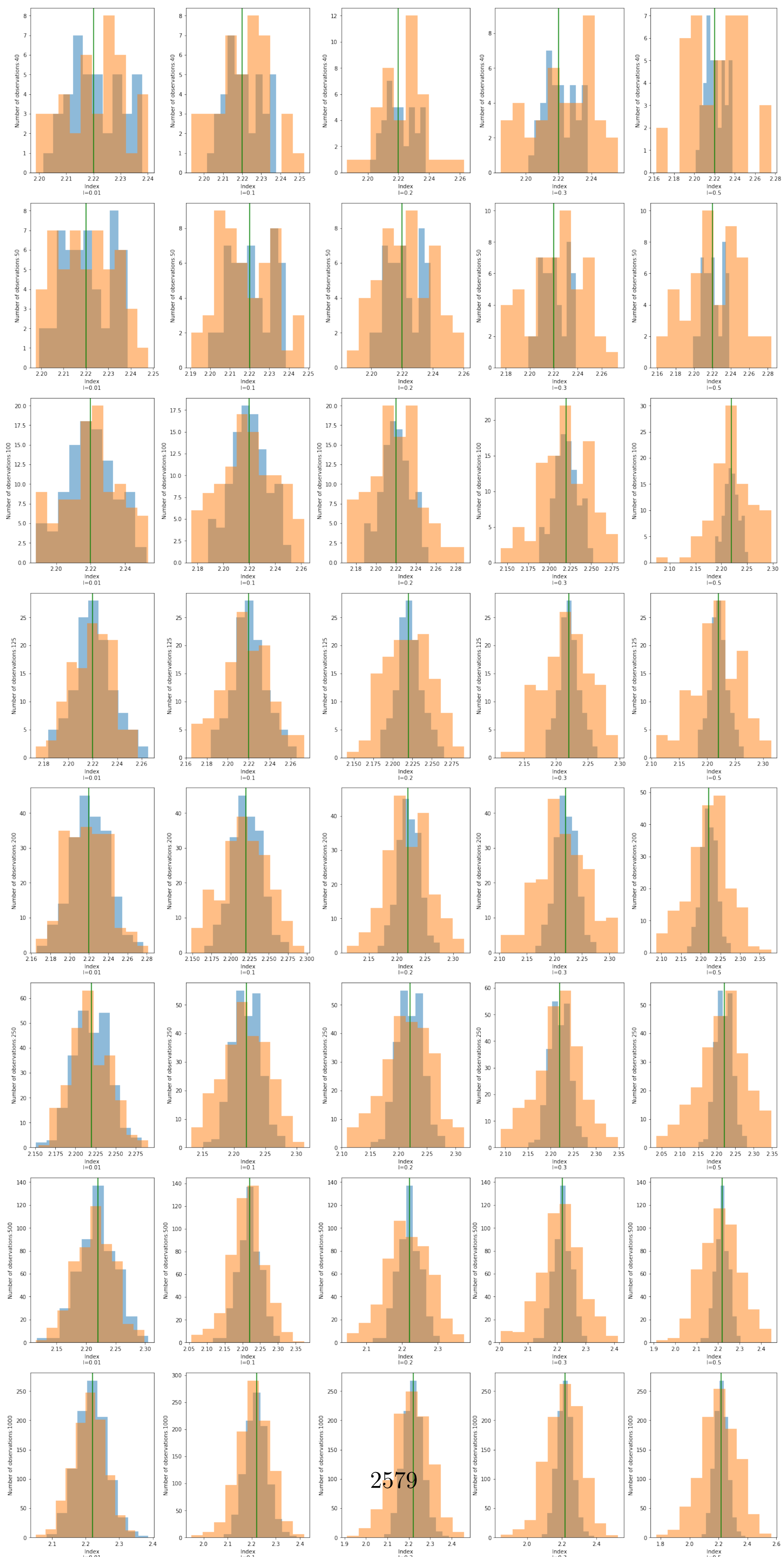
```
model: 0
index: 2.2208519415872168 += 0.009464742485055524
index: 2.2204071142702646 += 0.010643317914159624
index: 2.2203004860671913 += 0.014097739094078581
index: 2.220852816179499 += 0.016172109015838997
index: 2.2212194274890815 += 0.019215872375112058
index: 2.2199563069339674 += 0.022096616282109238
index: 2.2190755629793766 += 0.030988506389346 52
index: 2.219433716740164 += 0.045081260977561495
```

```
model: 1
index: 2.2204719698984876 += 0.011057911452127133
index: 2.2201525277145957 += 0.012376042811368266
index: 2.220766004292375 += 0.016254280452423428
index: 2.2192855660840825 += 0.016979350127733894
index: 2.2181942859260526 += 0.021566497637438602
index: 2.218340501343136 += 0.0246029745440613
index: 2.2141768111037137 += 0.03231075552018245
index: 2.2107577986756977 += 0.04844514153588179
model: 2
index: 2.2211847400271347 += 0.013650423195759882
index: 2.2186708810222173 += 0.013877953235843704
index: 2.2183847765756646 += 0.021281357410544443
index: 2.217352129681799 += 0.02251754427503892
index: 2.217878098387325 += 0.030017541358336335
index: 2.2180707820822607 += 0.03578842176931339
index: 2.2134213141794454 += 0.048266151351770606
index: 2.2096672550463956 += 0.06709402486086152
model: 3
index: 2.223099032157907 += 0.014582982639192716
index: 2.222378160459559 += 0.017317813920792062
index: 2.2193213565228627 += 0.02471830690583128
index: 2.2196622329825892 += 0.029492982027657766
index: 2.219270342305731 += 0.03810505221442646
index: 2.2164086393715294 += 0.042035907033355205
index: 2.214966998284206 += 0.061051815397505094
index: 2.210479984202571 += 0.08501738899244768
model: 4
index: 2.222540582907901 += 0.019103690191683814
index: 2.2226157841278593 += 0.022856886060554298
index: 2.2192448538730125 += 0.029345636261254127
index: 2.216770671204509 += 0.03677833344036589
index: 2.214961563969888 += 0.045036732115477464
index: 2.2177066375784715 += 0.04962759511556764
index: 2.2157036821177947 += 0.07098309361241216
index: 2.211728268618703 += 0.0990796020052331
model: 5
index: 2.2214952793651683 += 0.026250377207844712
index: 2.221494526385288 += 0.02940775753647641
index: 2.219397005708672 += 0.03882027909001693
index: 2.217771068100871 += 0.04167907227969314
index: 2.2204774474082605 += 0.05161966300111515
index: 2.2154705626624485 += 0.06378648415321286
index: 2.212090541476322 += 0.08989263357284584
index: 2.203583012383354 += 0.12668573135299435
```

[23]:

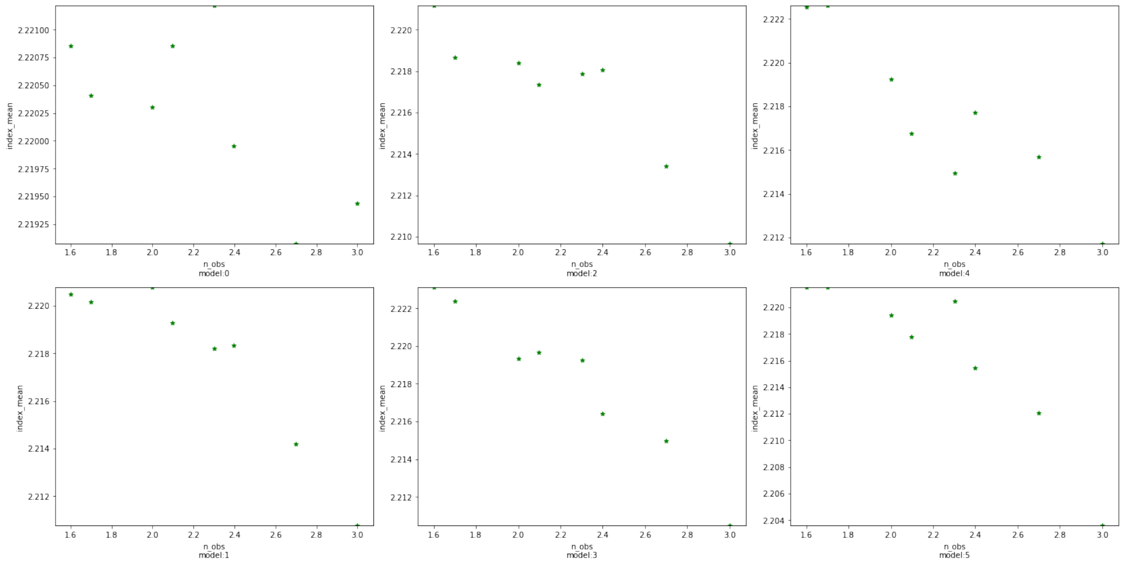
```
fig = plt.figure(figsize=[20,40],constrained_layout=True)

import matplotlib.gridspec as gridspec

gs0 = gridspec.GridSpec(1, 5, figure=fig)

gs1 = gridspec.GridSpecFromSubplotSpec(8, 1, subplot_spec=gs0[0])
for n in range(8):
    ax = fig.add_subplot(gs1[n])
    plt.hist(index[n][0], bins=10, alpha=0.5)
    plt.hist(index[n][1], bins=10, alpha=0.5)
    plt.axvline(x=model_simu.parameters["index"].value, color="green")
    plt.xlabel('Index\nl=0.01')
    plt.ylabel(f'Number of observations:{n_obs[n]}')

gs2 = gridspec.GridSpecFromSubplotSpec(8, 1, subplot_spec=gs0[1])
for n in range(8):
    ax = fig.add_subplot(gs2[n])
    plt.hist(index[n][0], bins=10, alpha=0.5)
    plt.hist(index[n][2], bins=10, alpha=0.5)
    plt.axvline(x=model_simu.parameters["index"].value, color="green")
    plt.xlabel('Index\nl=0.1')
    plt.ylabel(f'Number of observations:{n_obs[n]}')

gs3 = gridspec.GridSpecFromSubplotSpec(8, 1, subplot_spec=gs0[2])
for n in range(8):
    ax = fig.add_subplot(gs3[n])
    plt.hist(index[n][0], bins=10, alpha=0.5)
    plt.hist(index[n][3], bins=10, alpha=0.5)
    plt.axvline(x=model_simu.parameters["index"].value, color="green")
    plt.xlabel('Index\nl=0.2')
    plt.ylabel(f'Number of observations:{n_obs[n]}')

gs4 = gridspec.GridSpecFromSubplotSpec(8, 1, subplot_spec=gs0[3])
for n in range(8):
    ax = fig.add_subplot(gs4[n])
    plt.hist(index[n][0], bins=10, alpha=0.5)
    plt.hist(index[n][4], bins=10, alpha=0.5)
    plt.axvline(x=model_simu.parameters["index"].value, color="green")
    plt.xlabel('Index\nl=0.3')
    plt.ylabel(f'Number of observations:{n_obs[n]}')

gs5 = gridspec.GridSpecFromSubplotSpec(8, 1, subplot_spec=gs0[4])
for n in range(8):
    ax = fig.add_subplot(gs5[n])
    plt.hist(index[n][0], bins=10, alpha=0.5)
    plt.hist(index[n][5], bins=10, alpha=0.5)
```

```
    plt.axvline(x=model_simu.parameters["index"].value, color="green")
    plt.xlabel('Index\nl=0.5')
    plt.ylabel(f'Number of observations:{n_obs[n]}')

plt.show()
```


Number of observations 40
Number of observations 50
Number of observations 100
Number of observations 125
Number of observations 200
Number of observations 250
Number of observations 500
Number of observations 1000
Index
l=0.01
l=0.1
l=0.2
l=0.3
l=0.5

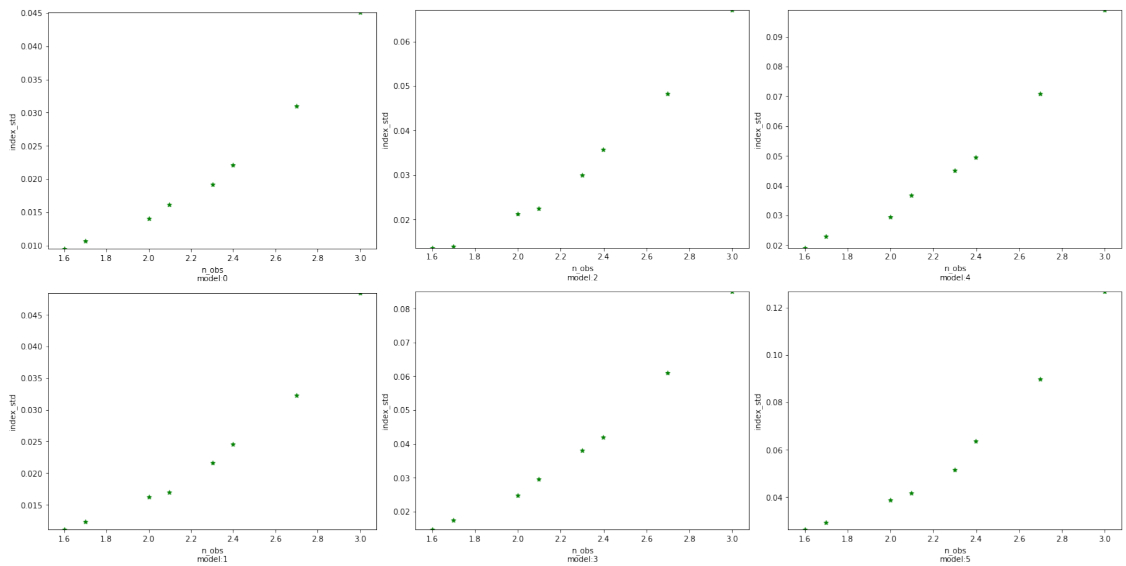
[24]:

```
fig = plt.figure(figsize=[20,10],constrained_layout=True)

import matplotlib.gridspec as gridspec

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

gs1 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[0])
for n in range(2):
    ax = fig.add_subplot(gs1[n])
    plt.scatter(np.log10(n_obs), [row[n] for row in mu], label= "stars", color=␣
↪"green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n}')
    plt.ylabel('index_mean')
    plt.ylim(min([row[n] for row in mu]),max([row[n] for row in mu]))

gs2 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[1])
for n in range(2):
    ax = fig.add_subplot(gs2[n])
    plt.scatter(np.log10(n_obs), [row[n+2] for row in mu], label= "stars",␣
↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+2}')
    plt.ylabel('index_mean')
    plt.ylim(min([row[n+2] for row in mu]),max([row[n+2] for row in mu]))

gs3 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[2])
for n in range(2):
    ax = fig.add_subplot(gs3[n])
    plt.scatter(np.log10(n_obs), [row[n+4] for row in mu], label= "stars",␣
↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+4}')
    plt.ylabel('index_mean')
    plt.ylim(min([row[n+4] for row in mu]),max([row[n+4] for row in mu]))

plt.show()
```



[25]:
```
fig = plt.figure(figsize=[20,10],constrained_layout=True)

import matplotlib.gridspec as gridspec

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

gs1 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[0])
for n in range(2):
    ax = fig.add_subplot(gs1[n])
    plt.scatter(np.log10(n_obs), [row[n] for row in sigma], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n}')
    plt.ylabel('index_std')
    plt.ylim(min([row[n] for row in sigma]),max([row[n] for row in sigma]))

gs2 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[1])
for n in range(2):
    ax = fig.add_subplot(gs2[n])
    plt.scatter(np.log10(n_obs), [row[n+2] for row in sigma], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+2}')
    plt.ylabel('index_std')
    plt.ylim(min([row[n+2] for row in sigma]),max([row[n+2] for row in sigma]))

gs3 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[2])
for n in range(2):
```

```
    ax = fig.add_subplot(gs3[n])
    plt.scatter(np.log10(n_obs), [row[n+4] for row in sigma], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+4}')
    plt.ylabel('index_std')
    plt.ylim(min([row[n+4] for row in sigma]),max([row[n+4] for row in sigma]))

plt.show()
```



[26]:
```
for j in range(6):
    print(f"model: {j}")
    for i in range(8):
        mu[i][j]=amplitude[i][j].mean()
        sigma[i][j]=amplitude[i][j].std()
        print(f"amplitude: {amplitude[i][j].mean()} += {amplitude[i][j].std()}")
```

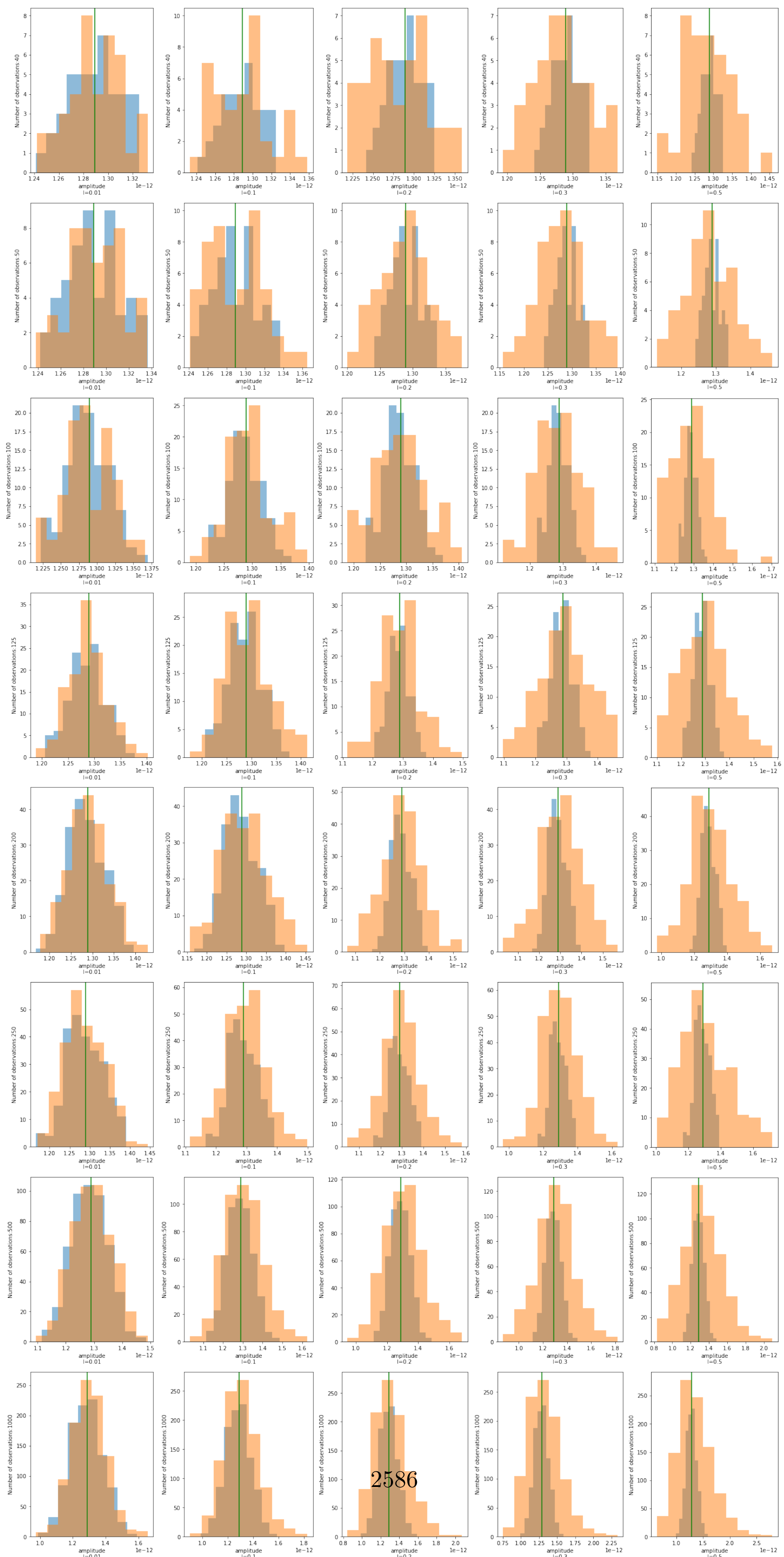
```
model: 0
amplitude: 1.2882109734645357e-12 += 2.020774416400257e-14
amplitude: 1.2887930617364896e-12 += 2.321243734503526e-14
amplitude: 1.2875057637201671e-12 += 2.943738660653206e-14
amplitude: 1.2855551328222816e-12 += 3.4620322478409695e-14
amplitude: 1.2852616877373924e-12 += 4.1590131875188173e-14
amplitude: 1.2870704254498414e-12 += 4.604036230233484e-14
amplitude: 1.2848003538014906e-12 += 6.382360983727948e-14
amplitude: 1.2834022687629476e-12 += 9.812239252655629e-14
model: 1
amplitude: 1.2887740733863474e-12 += 2.1624008113134554e-14
amplitude: 1.290197249320048e-12 += 2.3665712755630992e-14
```

```
amplitude: 1.2873240819719735e-12 += 3.2647047697068356e-14
amplitude: 1.287907206311591e-12 += 3.64426974840847e-14
amplitude: 1.288016568088648e-12 += 4.6423404433813026e-14
amplitude: 1.2878012385332741e-12 += 4.866857799313919e-14
amplitude: 1.2941386042810284e-12 += 6.980329125688868e-14
amplitude: 1.299253568211679e-12 += 1.032452149663184e-13
model: 2
amplitude: 1.288833703100092e-12 += 2.961453439686997e-14
amplitude: 1.2893006402619744e-12 += 2.8680818735483707e-14
amplitude: 1.2927832874446586e-12 += 3.988864742203177e-14
amplitude: 1.2927113396963073e-12 += 4.504148198217889e-14
amplitude: 1.2939662295353098e-12 += 5.825767301477067e-14
amplitude: 1.2947862319018303e-12 += 6.618781500750589e-14
amplitude: 1.3031686219183516e-12 += 9.697874818989171e-14
amplitude: 1.3082828025427663e-12 += 1.3892983026905873e-13
model: 3
amplitude: 1.2805526881050745e-12 += 3.574018932168858e-14
amplitude: 1.2872566281400055e-12 += 3.844094690616137e-14
amplitude: 1.286795800901381e-12 += 4.978203409314228e-14
amplitude: 1.2882630704524287e-12 += 6.442621190119476e-14
amplitude: 1.2890386543647814e-12 += 8.12412098445936e-14
amplitude: 1.296306142469232e-12 += 8.772249466134022e-14
amplitude: 1.3041335523312662e-12 += 1.240115995567233e-13
amplitude: 1.3152171172761559e-12 += 1.8316141365511399e-13
model: 4
amplitude: 1.281598735194535e-12 += 4.1968762789464806e-14
amplitude: 1.2775118349799447e-12 += 4.9074789194400446e-14
amplitude: 1.2846474844088973e-12 += 6.617702199344377e-14
amplitude: 1.2934970969874811e-12 += 8.226292423276059e-14
amplitude: 1.3006942825709336e-12 += 1.0004240556633877e-13
amplitude: 1.2956693454424653e-12 += 1.0648717184660167e-13
amplitude: 1.3060211186389223e-12 += 1.5765578083564723e-13
amplitude: 1.3217526457383794e-12 += 2.287718530959276e-13
model: 5
amplitude: 1.2852430965817646e-12 += 6.24353802813177e-14
amplitude: 1.2825788705042499e-12 += 7.073938775851406e-14
amplitude: 1.2884404569577469e-12 += 9.577571033132101e-14
amplitude: 1.2980170584135136e-12 += 9.830052853675669e-14
amplitude: 1.2971893243690907e-12 += 1.2750395315534658e-13
amplitude: 1.308078615540899e-12 += 1.4960139056854337e-13
amplitude: 1.3287393295332176e-12 += 2.1348786203556218e-13
amplitude: 1.3542334224473133e-12 += 3.2352153412172364e-13
```

[27]:
```python
fig = plt.figure(figsize=[20,40],constrained_layout=True)

import matplotlib.gridspec as gridspec
```

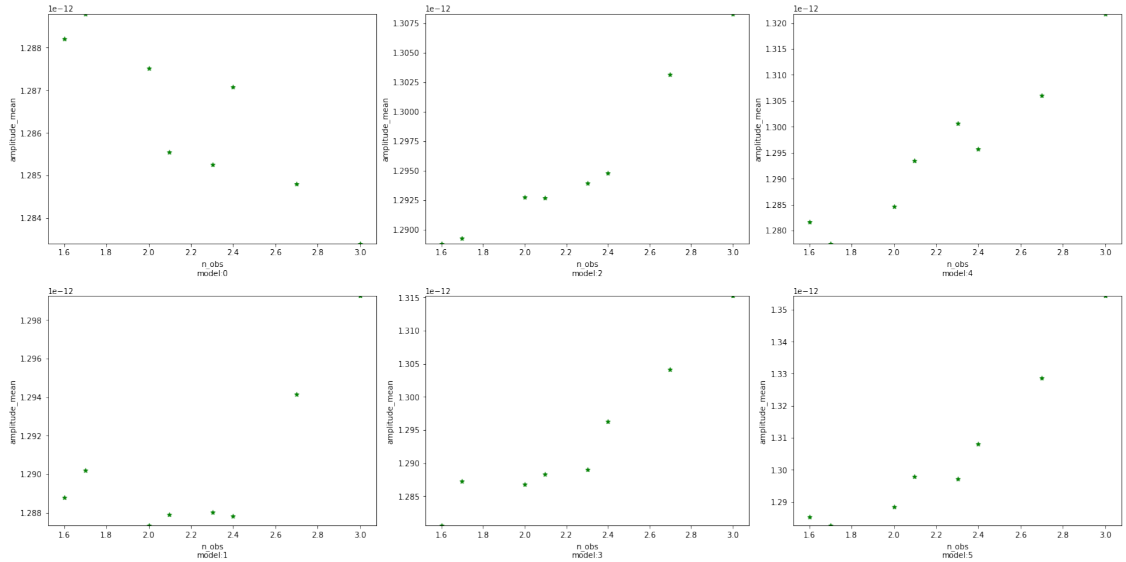
```python
gs0 = gridspec.GridSpec(1, 5, figure=fig)

gs1 = gridspec.GridSpecFromSubplotSpec(8, 1, subplot_spec=gs0[0])
for n in range(8):
    ax = fig.add_subplot(gs1[n])
    plt.hist(amplitude[n][0], bins=10, alpha=0.5)
    plt.hist(amplitude[n][1], bins=10, alpha=0.5)
    plt.axvline(x=model_simu.parameters["amplitude"].value, color="green")
    plt.xlabel('amplitude\nl=0.01')
    plt.ylabel(f'Number of observations:{n_obs[n]}')

gs2 = gridspec.GridSpecFromSubplotSpec(8, 1, subplot_spec=gs0[1])
for n in range(8):
    ax = fig.add_subplot(gs2[n])
    plt.hist(amplitude[n][0], bins=10, alpha=0.5)
    plt.hist(amplitude[n][2], bins=10, alpha=0.5)
    plt.axvline(x=model_simu.parameters["amplitude"].value, color="green")
    plt.xlabel('amplitude\nl=0.1')
    plt.ylabel(f'Number of observations:{n_obs[n]}')

gs3 = gridspec.GridSpecFromSubplotSpec(8, 1, subplot_spec=gs0[2])
for n in range(8):
    ax = fig.add_subplot(gs3[n])
    plt.hist(amplitude[n][0], bins=10, alpha=0.5)
    plt.hist(amplitude[n][3], bins=10, alpha=0.5)
    plt.axvline(x=model_simu.parameters["amplitude"].value, color="green")
    plt.xlabel('amplitude\nl=0.2')
    plt.ylabel(f'Number of observations:{n_obs[n]}')

gs4 = gridspec.GridSpecFromSubplotSpec(8, 1, subplot_spec=gs0[3])
for n in range(8):
    ax = fig.add_subplot(gs4[n])
    plt.hist(amplitude[n][0], bins=10, alpha=0.5)
    plt.hist(amplitude[n][4], bins=10, alpha=0.5)
    plt.axvline(x=model_simu.parameters["amplitude"].value, color="green")
    plt.xlabel('amplitude\nl=0.3')
    plt.ylabel(f'Number of observations:{n_obs[n]}')

gs5 = gridspec.GridSpecFromSubplotSpec(8, 1, subplot_spec=gs0[4])
for n in range(8):
    ax = fig.add_subplot(gs5[n])
    plt.hist(amplitude[n][0], bins=10, alpha=0.5)
    plt.hist(amplitude[n][5], bins=10, alpha=0.5)
    plt.axvline(x=model_simu.parameters["amplitude"].value, color="green")
    plt.xlabel('amplitude\nl=0.5')
    plt.ylabel(f'Number of observations:{n_obs[n]}')
```

```
plt.show()
```



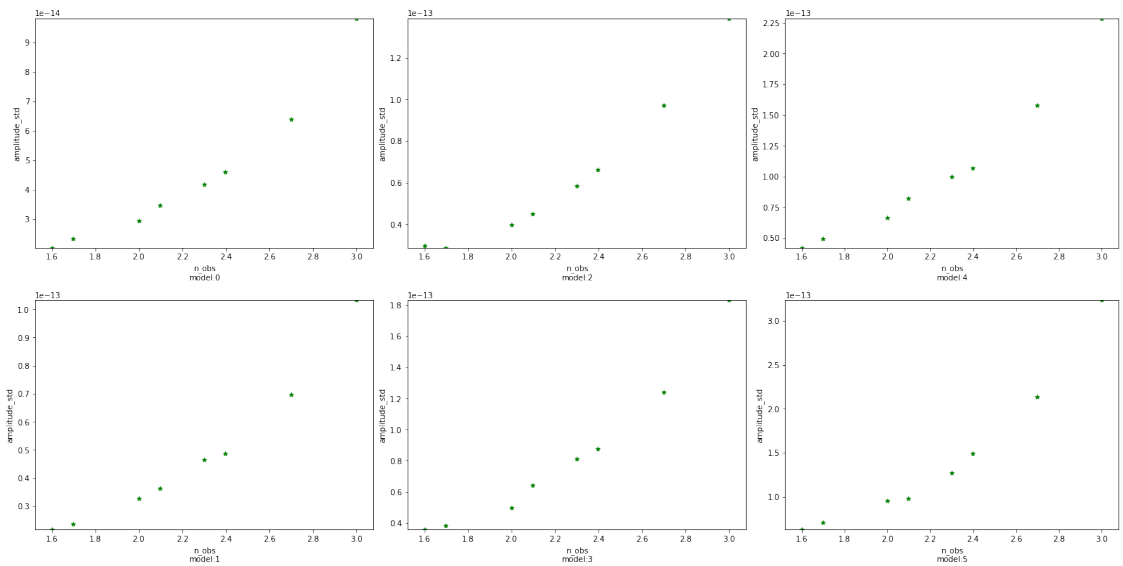
[28]:

```
fig = plt.figure(figsize=[20,10],constrained_layout=True)

import matplotlib.gridspec as gridspec

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

gs1 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[0])
for n in range(2):
    ax = fig.add_subplot(gs1[n])
    plt.scatter(np.log10(n_obs), [row[n] for row in mu], label= "stars", color=␣
 ↪"green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n}')
    plt.ylabel('amplitude_mean')
    plt.ylim(min([row[n] for row in mu]),max([row[n] for row in mu]))

gs2 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[1])
for n in range(2):
    ax = fig.add_subplot(gs2[n])
    plt.scatter(np.log10(n_obs), [row[n+2] for row in mu], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+2}')
    plt.ylabel('amplitude_mean')
    plt.ylim(min([row[n+2] for row in mu]),max([row[n+2] for row in mu]))

gs3 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[2])
for n in range(2):
    ax = fig.add_subplot(gs3[n])
    plt.scatter(np.log10(n_obs), [row[n+4] for row in mu], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+4}')
    plt.ylabel('amplitude_mean')
    plt.ylim(min([row[n+4] for row in mu]),max([row[n+4] for row in mu]))

plt.show()
```



[29]:
```
fig = plt.figure(figsize=[20,10],constrained_layout=True)

import matplotlib.gridspec as gridspec

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

gs1 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[0])
for n in range(2):
    ax = fig.add_subplot(gs1[n])
    plt.scatter(np.log10(n_obs), [row[n] for row in sigma], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n}')
    plt.ylabel('amplitude_std')
    plt.ylim(min([row[n] for row in sigma]),max([row[n] for row in sigma]))

gs2 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[1])
for n in range(2):
    ax = fig.add_subplot(gs2[n])
    plt.scatter(np.log10(n_obs), [row[n+2] for row in sigma], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+2}')
    plt.ylabel('amplitude_std')
    plt.ylim(min([row[n+2] for row in sigma]),max([row[n+2] for row in sigma]))

gs3 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[2])
for n in range(2):
```

```
    ax = fig.add_subplot(gs3[n])
    plt.scatter(np.log10(n_obs), [row[n+4] for row in sigma], label= "stars",␣
↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+4}')
    plt.ylabel('amplitude_std')
    plt.ylim(min([row[n+4] for row in sigma]),max([row[n+4] for row in sigma]))

plt.show()
```



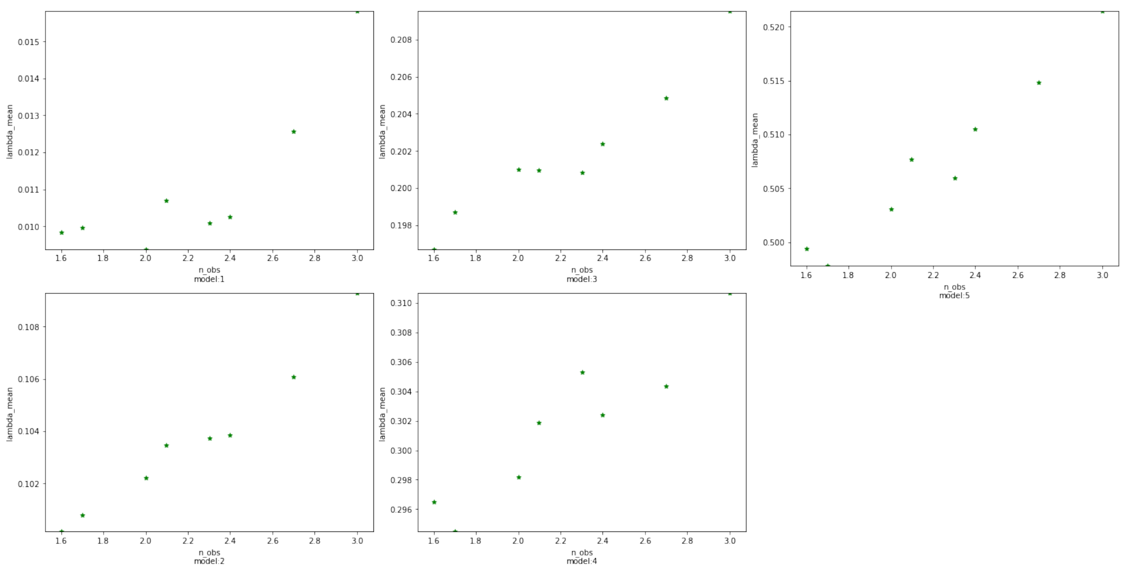
[30]:
```
for j in range(5):
    print(f"model: {j}")
    for i in range(8):
        mu[i][j]=lambda_[i][j].mean()
        sigma[i][j]=lambda_[i][j].std()
        print(f"lambda_: {lambda_[i][j].mean()} += {lambda_[i][j].std()}")
```

```
model: 0
lambda_: 0.009833288727017455 += 0.0035274128156517 05
lambda_: 0.00997096588039571 += 0.003855226271571548
lambda_: 0.009380252964157535 += 0.00519607994559184
lambda_: 0.01069061050567699 += 0.005867950169742963
lambda_: 0.01008256122878477 += 0.007212265102101397
lambda_: 0.010256444061833998 += 0.007588274308204233
lambda_: 0.012567652159184253 += 0.010904987005412904
lambda_: 0.015828571114078694 += 0.016919554721768242
model: 1
lambda_: 0.10015432945628366 += 0.006648005418110846 5
lambda_: 0.10078827921563097 += 0.006449629883359707
```

```
lambda_: 0.1022222867459217 += 0.011100524725229686
lambda_: 0.10344886668194597 += 0.013432590106282118
lambda_: 0.10373798639547463 += 0.01640438049913423
lambda_: 0.10383728053928191 += 0.01839816538948341
lambda_: 0.10607361404282202 += 0.027229784201596493
lambda_: 0.109270706562282 7 += 0.03675385733319002
model: 2
lambda_: 0.1967069864590812 += 0.010744366602316709
lambda_: 0.19869165590213667 += 0.013229014698656682
lambda_: 0.20099872976971228 += 0.017449749215610166
lambda_: 0.2009619078702176 += 0.02269908573695963
lambda_: 0.20083837141583402 += 0.025916384082007707
lambda_: 0.2023882407275771 += 0.030565647809145686
lambda_: 0.2048623674512616 += 0.04117116322471812
lambda_: 0.20954778762943943 += 0.06034534310983556
model: 3
lambda_: 0.2964729205851758 += 0.014721944589363148
lambda_: 0.29447525989345485 += 0.01685203921803965
lambda_: 0.29817104986664195 += 0.027015407948679744
lambda_: 0.30185892306290885 += 0.030489761434818297
lambda_: 0.3052778294324254 += 0.03695667157246595
lambda_: 0.3024098305014937 += 0.03962871920754218
lambda_: 0.30436087505589615 += 0.05804036947085071
lambda_: 0.31065997881282587 += 0.08196051750382335
model: 4
lambda_: 0.4993696065408228 += 0.026190993490909777
lambda_: 0.49778683814020547 += 0.029522341153212735
lambda_: 0.5030743790002709 += 0.038467654459497
lambda_: 0.5076719641434785 += 0.04428667967358409
lambda_: 0.5059721243539873 += 0.05600715517347292
lambda_: 0.5105196288691412 += 0.070808907235574
lambda_: 0.5148255650991596 += 0.0966096298646213
lambda_: 0.5214624796108971 += 0.13687383699270922
```

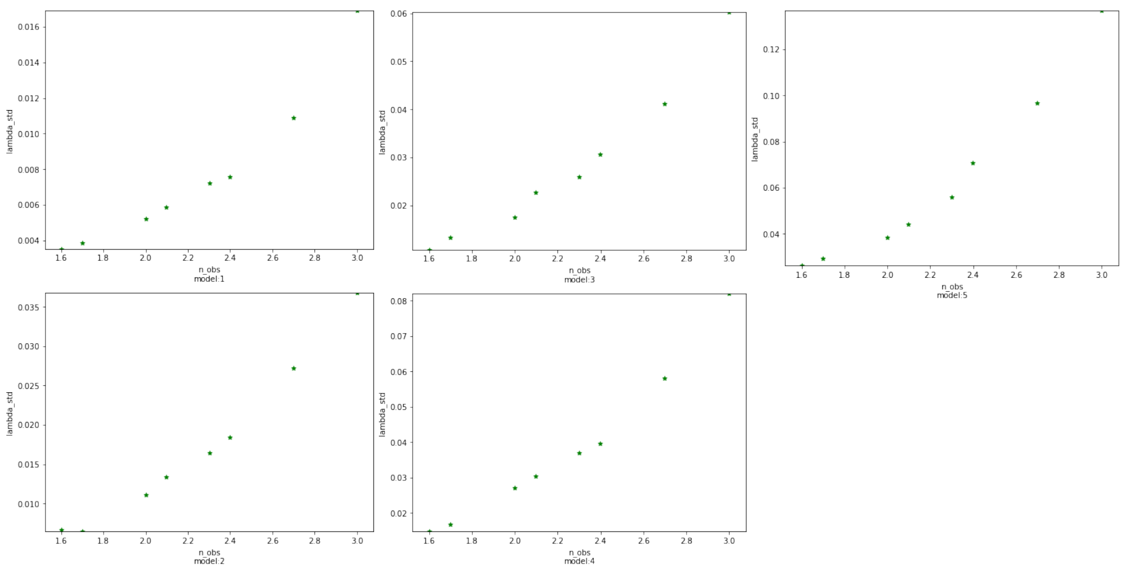
[31]:
```
fig = plt.figure(figsize=[20,10],constrained_layout=True)

import matplotlib.gridspec as gridspec

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

gs1 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[0])
for n in range(2):
    ax = fig.add_subplot(gs1[n])
    plt.scatter(np.log10(n_obs), [row[n] for row in mu], label= "stars", color=␣
 ↪"green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+1}')
```

```
    plt.ylabel('lambda_mean')
    plt.ylim(min([row[n] for row in mu]),max([row[n] for row in mu]))

gs2 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[1])
for n in range(2):
    ax = fig.add_subplot(gs2[n])
    plt.scatter(np.log10(n_obs), [row[n+2] for row in mu], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+3}')
    plt.ylabel('lambda_mean')
    plt.ylim(min([row[n+2] for row in mu]),max([row[n+2] for row in mu]))

gs3 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[2])
for n in range(1):
    ax = fig.add_subplot(gs3[n])
    plt.scatter(np.log10(n_obs), [row[n+4] for row in mu], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+5}')
    plt.ylabel('lambda_mean')
    plt.ylim(min([row[n+4] for row in mu]),max([row[n+4] for row in mu]))

plt.show()
```



[32]:
```
fig = plt.figure(figsize=[20,10],constrained_layout=True)

import matplotlib.gridspec as gridspec
```

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

gs1 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[0])
for n in range(2):
    ax = fig.add_subplot(gs1[n])
    plt.scatter(np.log10(n_obs), [row[n] for row in sigma], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+1}')
    plt.ylabel('lambda_std')
    plt.ylim(min([row[n] for row in sigma]),max([row[n] for row in sigma]))

gs2 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[1])
for n in range(2):
    ax = fig.add_subplot(gs2[n])
    plt.scatter(np.log10(n_obs), [row[n+2] for row in sigma], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+3}')
    plt.ylabel('lambda_std')
    plt.ylim(min([row[n+2] for row in sigma]),max([row[n+2] for row in sigma]))

gs3 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[2])
for n in range(1):
    ax = fig.add_subplot(gs3[n])
    plt.scatter(np.log10(n_obs), [row[n+4] for row in sigma], label= "stars",␣
 ↪color= "green",
                marker= "*", s=30)
    plt.xlabel(f'n_obs\nmodel:{n+5}')
    plt.ylabel('lambda_std')
    plt.ylim(min([row[n+4] for row in sigma]),max([row[n+4] for row in sigma]))

plt.show()
```



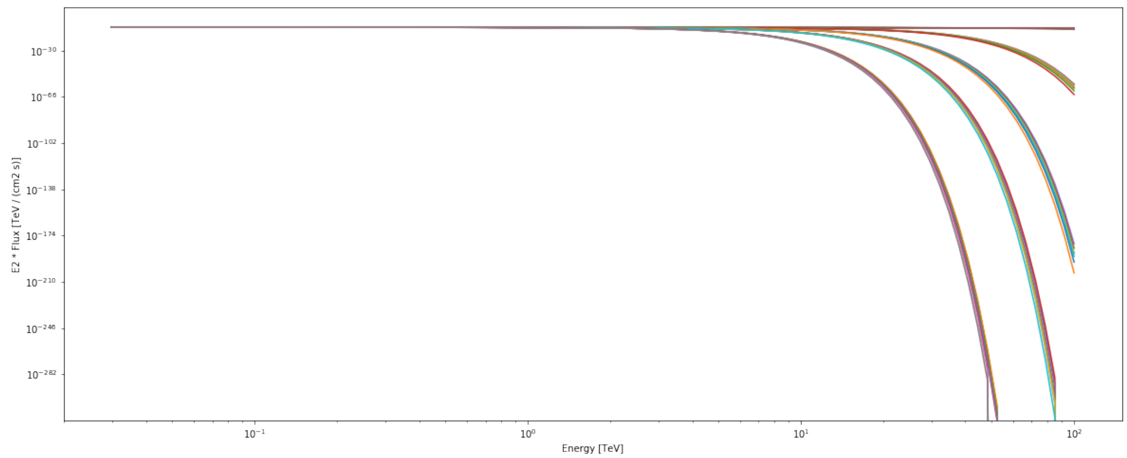

```
[33]: sim = [[0 for i in range(cols)] for j in range(rows)]
for i in range(8):
    s = PowerLawSpectralModel(
        index=index[i][0].mean(),
        amplitude=amplitude[i][0].mean() * u.Unit("cm-2 s-1 TeV-1"),
        reference=1 * u.TeV,
    )
    print(s)
    sim[i][0]=s

for j in range(5):
    for i in range(8):
        s = ExpCutoffPowerLawSpectralModel(
            index=index[i][j+1].mean(),
            amplitude=amplitude[i][j+1].mean() * u.Unit("cm-2 s-1 TeV-1"),
            reference=1 * u.TeV,
            lambda_=lambda_[i][j].mean() * u.Unit("TeV-1"),
            alpha = 2,
        )
        print(s)
        sim[i][j+1]=s
```

```
PowerLawSpectralModel

   name      value    error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------

    index 2.221e+00   nan                nan nan  False
amplitude 1.288e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
```

```
   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.215e+00   nan                nan nan  False
amplitude 1.308e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.105e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
ExpCutoffPowerLawSpectralModel

   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.212e+00   nan                nan nan  False
amplitude 1.329e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.148e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
ExpCutoffPowerLawSpectralModel

   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.204e+00   nan                nan nan  False
amplitude 1.354e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.215e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
```

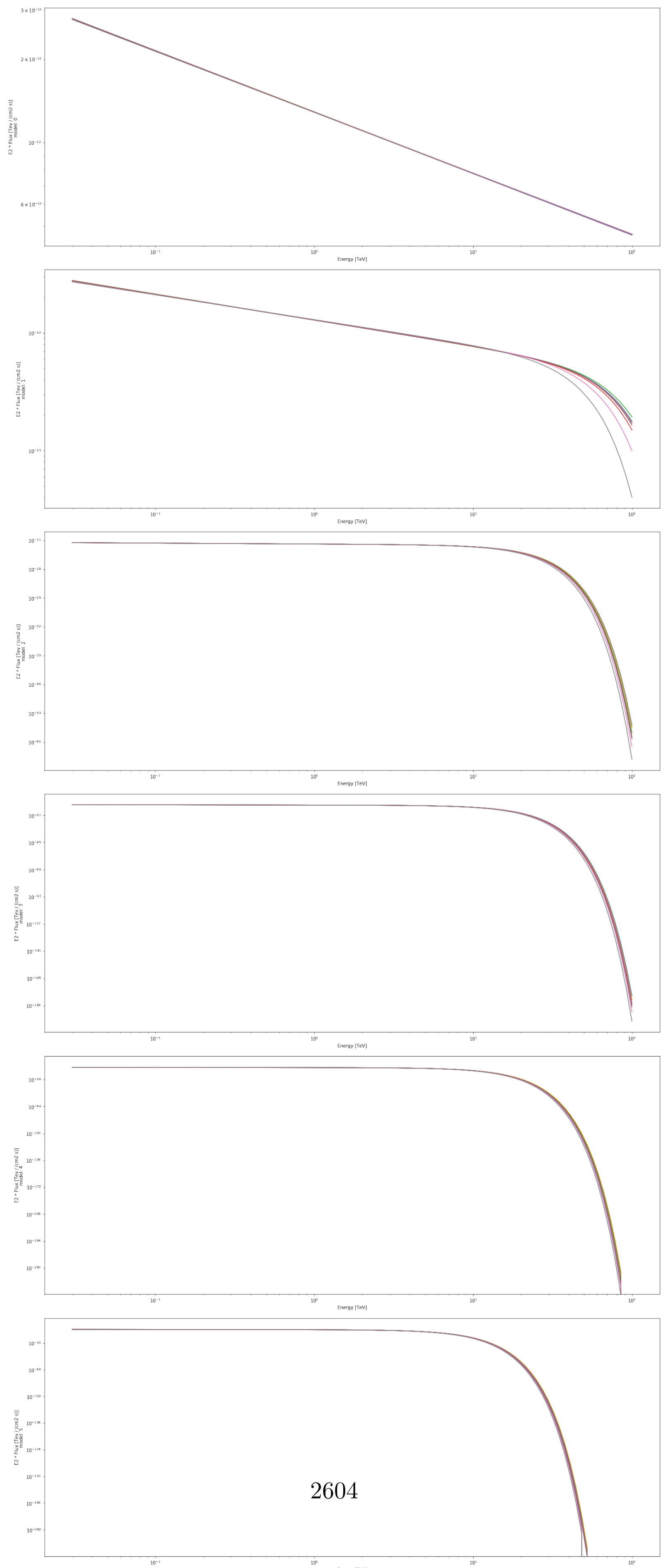
[34]:

```
plt.figure(figsize=[20,8])
energy_range = [0.03, 100] * u.TeV
for j in range(6):
    for i in range(8):
        sim[i][j].plot(energy_range=energy_range, energy_power=2)
plt.show
plt.savefig('energy_lightcurve')
```



[35]:

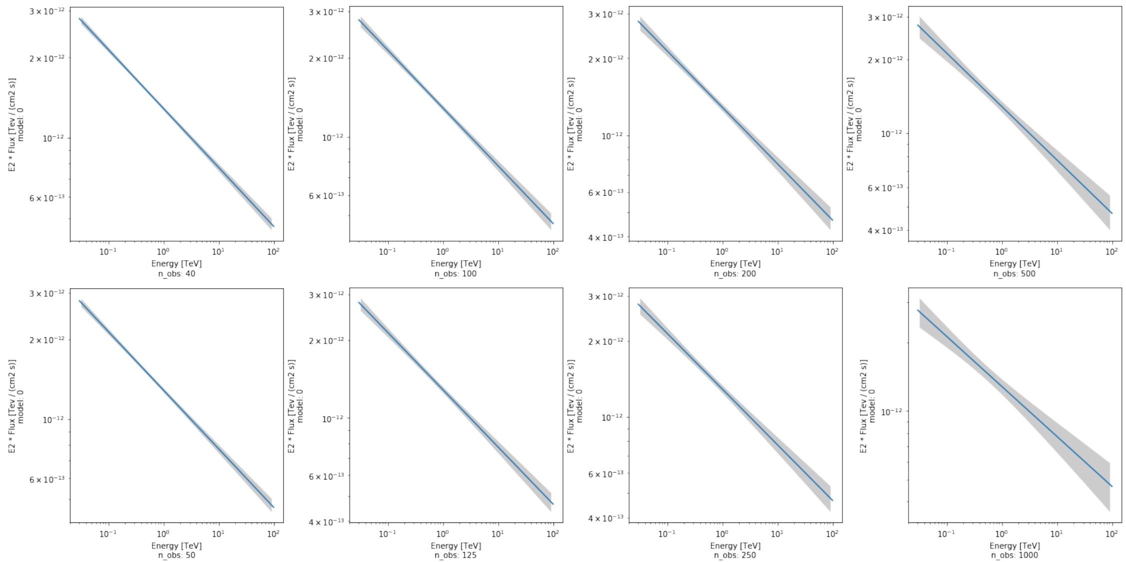
```
fig = plt.figure(figsize=[20,48],constrained_layout=True)

import matplotlib.gridspec as gridspec

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

gs1 = gridspec.GridSpecFromSubplotSpec(6, 1, subplot_spec=gs0[0])
for n in range(6):
    ax = fig.add_subplot(gs1[n])
    for i in range(8):
        sim[i][n].plot(energy_range=energy_range, energy_power=2)
    plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {n}')

plt.show()
plt.savefig('energy_lightcurve')
```

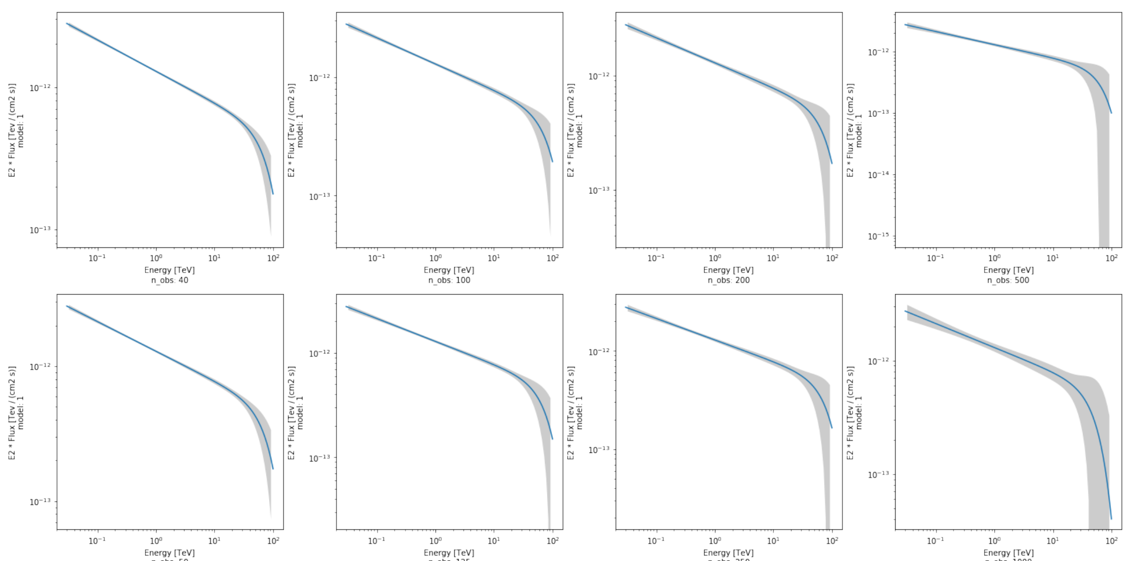

Energy [TeV]

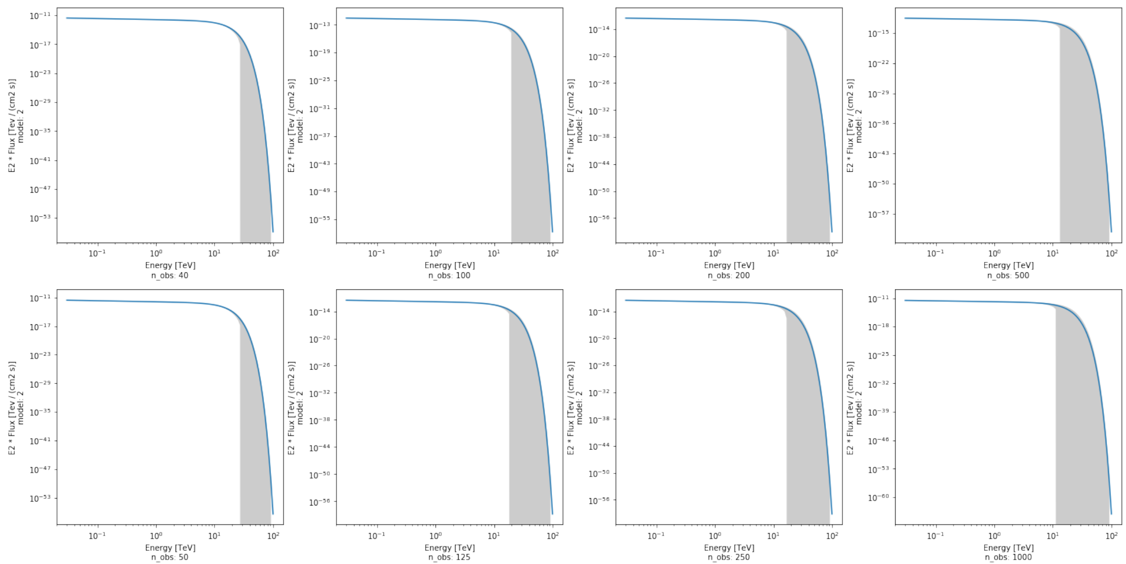
```
<Figure size 432x288 with 0 Axes>
```

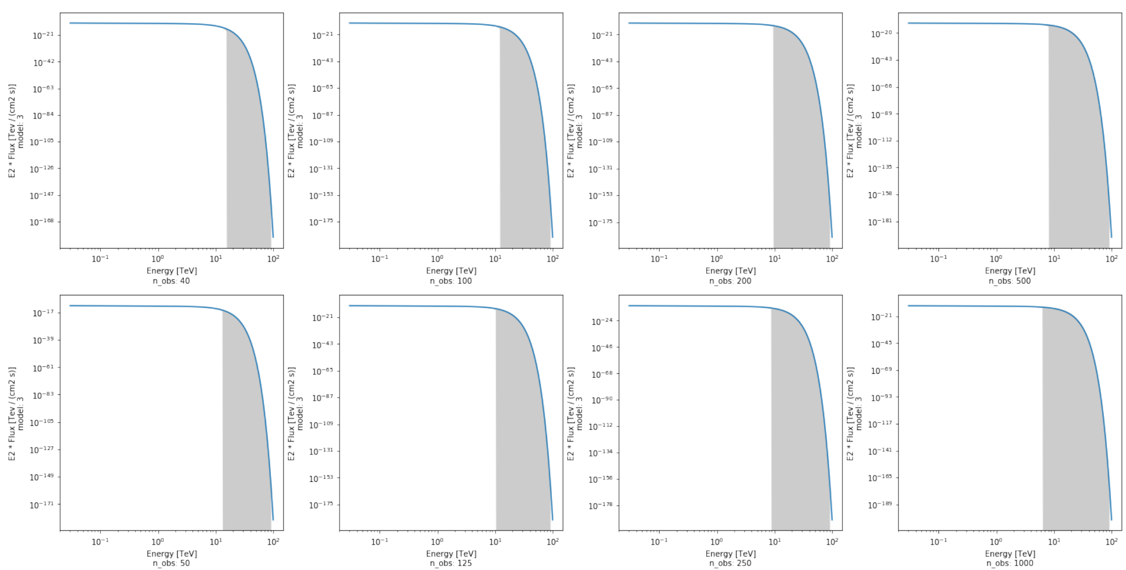
```
[36]: for i in range(6):
    fig = plt.figure(figsize=[20,10],constrained_layout=True)

    import matplotlib.gridspec as gridspec

    gs0 = gridspec.GridSpec(1, 4, figure=fig)

    gs1 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[0])
    for n in range(2):
        ax = fig.add_subplot(gs1[n])
        sim[n][i].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n][i], covar=covar[n][i],energy_range=energy_range,␣
↪energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i}')

    gs2 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[1])
    for n in range(2):
        ax = fig.add_subplot(gs2[n])
        sim[n+2][i].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n+2][i],␣
↪covar=covar[n+2][i],energy_range=energy_range, energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n+2]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i}')

    gs3 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[2])
    for n in range(2):
        ax = fig.add_subplot(gs3[n])
        sim[n+4][i].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n+4][i],␣
↪covar=covar[n+4][i],energy_range=energy_range, energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n+4]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i}')

    gs4 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[3])
    for n in range(2):
        ax = fig.add_subplot(gs4[n])
        sim[n+6][i].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n+6][i],␣
↪covar=covar[n+6][i],energy_range=energy_range, energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n+6]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i}')
```

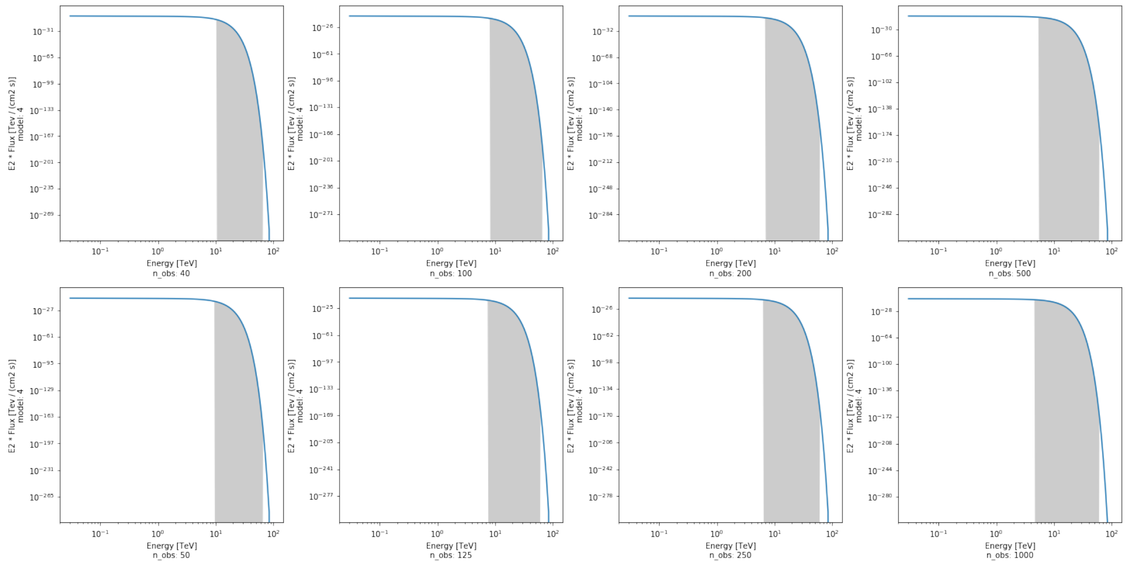
```
plt.show()
```


E2 * Flux [Tev / (cm2 s)]
model: 0
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000

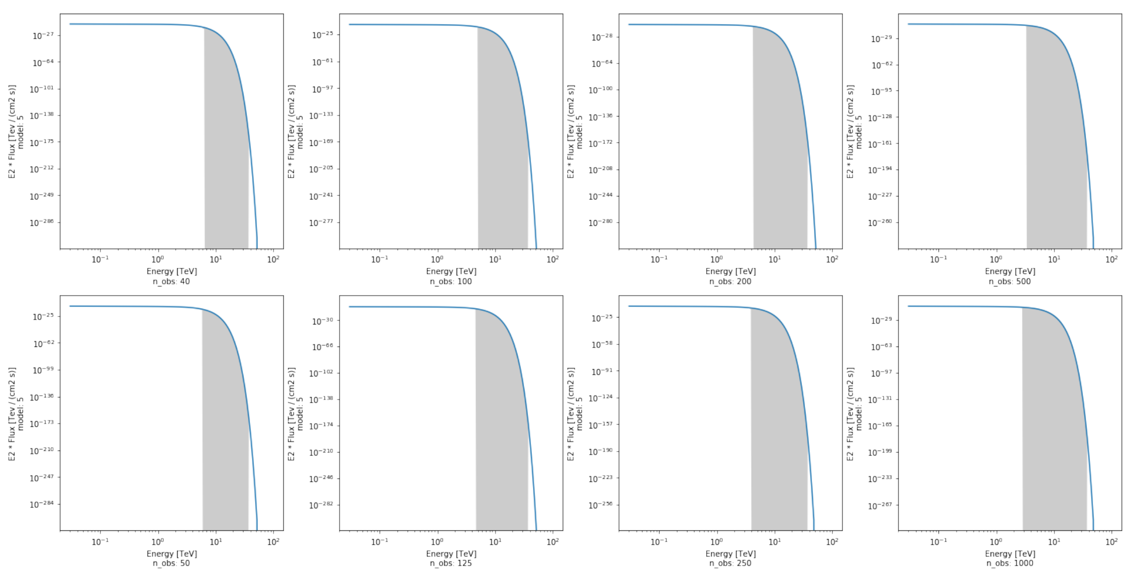

E2 * Flux [Tev / (cm2 s)]
model: 1
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000


E2 * Flux [Tev / (cm2 s)]
model: 2
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000


E2 * Flux [Tev / (cm2 s)]
model: 3
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000

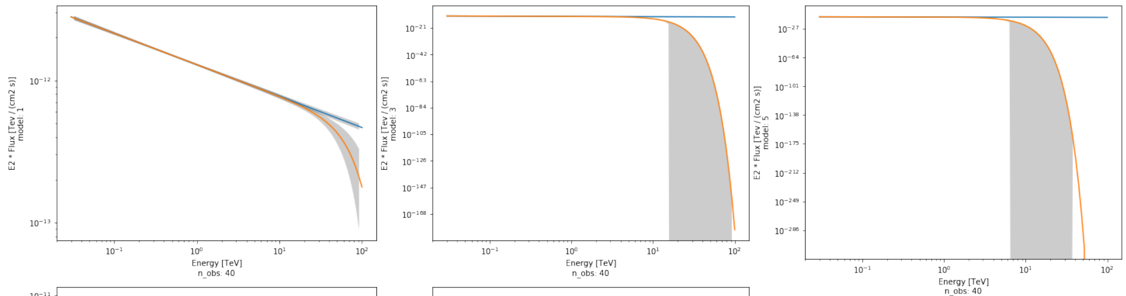


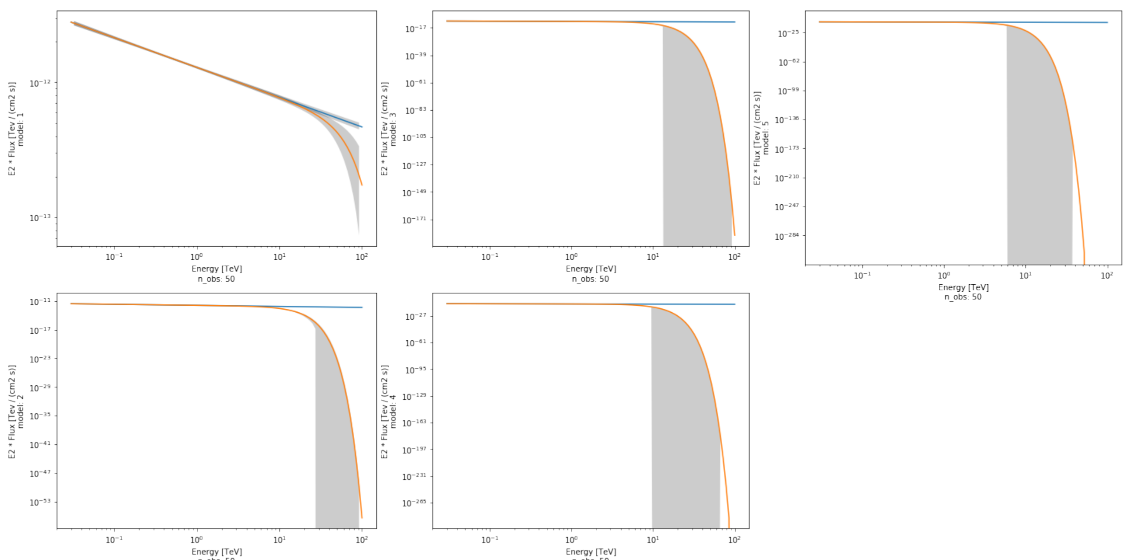

```
/home/rishank/anaconda2/envs/gammapy-0.15/lib/python3.6/site-
packages/matplotlib/ticker.py:1123: RuntimeWarning: divide by zero encountered
in double_scalars
  coeff = np.round(x / b ** exponent)
```

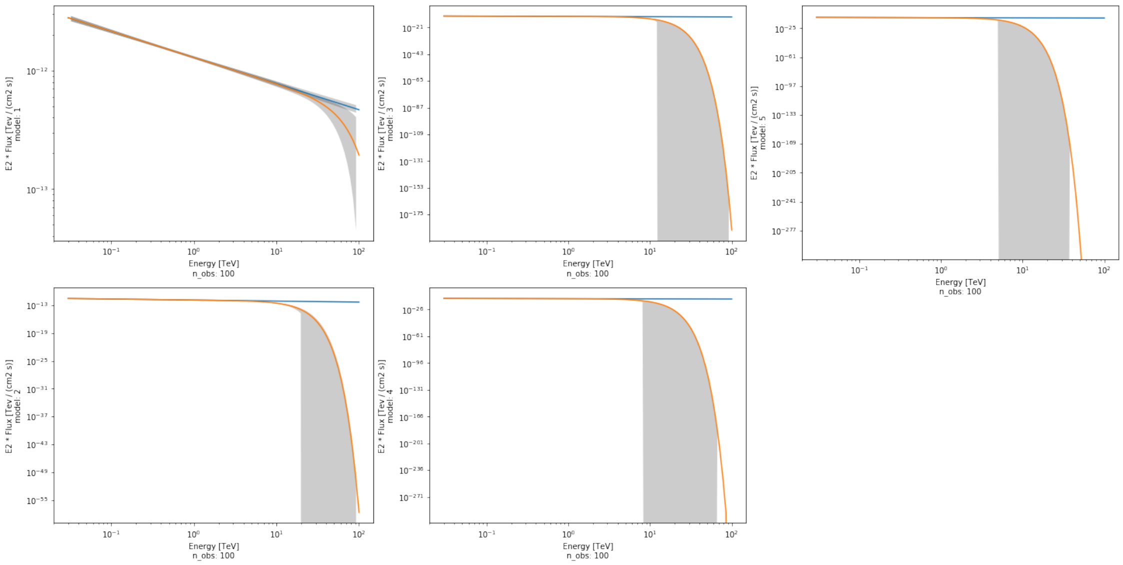


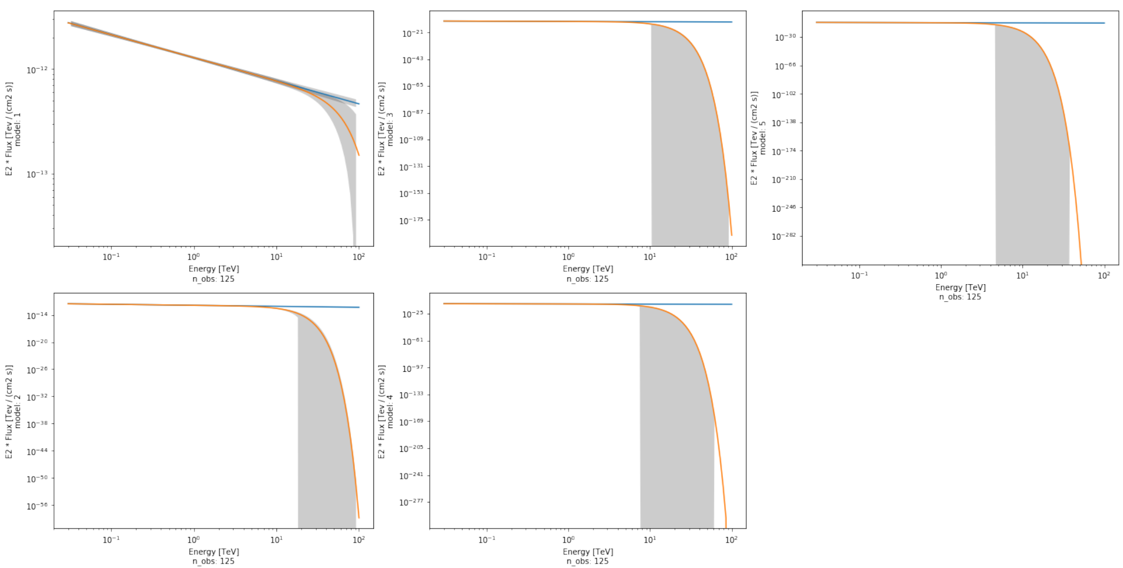
[37]:
```python
for i in range(8):
    fig = plt.figure(figsize=[20,10],constrained_layout=True)

    import matplotlib.gridspec as gridspec
```

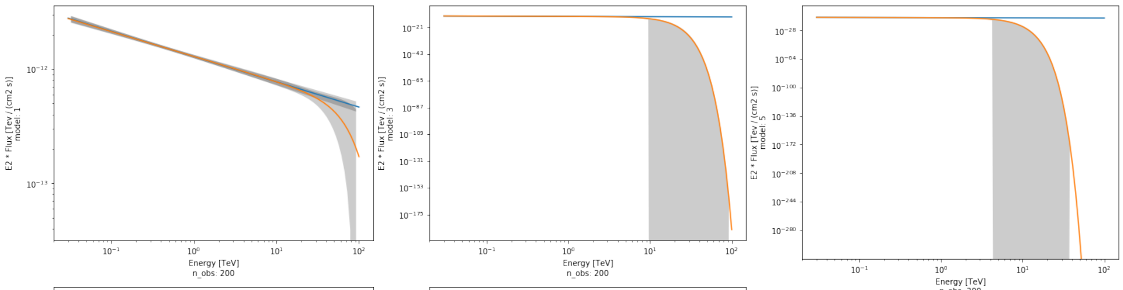
```python
gs0 = gridspec.GridSpec(1, 3, figure=fig)

gs1 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[0])
for n in range(2):
    ax = fig.add_subplot(gs1[n])
    sim[i][0].plot(energy_range=energy_range, energy_power=2)
    plot_error(self=sim[i][0], covar=covar[i][0],energy_range=energy_range,␣
↪energy_power=2)
    sim[i][n+1].plot(energy_range=energy_range, energy_power=2)
    plot_error(self=sim[i][n+1],␣
↪covar=covar[i][n+1],energy_range=energy_range, energy_power=2)
    plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[i]}')
    plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {n+1}')

gs2 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[1])
for n in range(2):
    ax = fig.add_subplot(gs2[n])
    sim[i][0].plot(energy_range=energy_range, energy_power=2)
    plot_error(self=sim[i][0], covar=covar[i][0],energy_range=energy_range,␣
↪energy_power=2)
    sim[i][n+3].plot(energy_range=energy_range, energy_power=2)
    plot_error(self=sim[i][n+3],␣
↪covar=covar[i][n+3],energy_range=energy_range, energy_power=2)
    plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[i]}')
    plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {n+3}')

gs3 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[2])
for n in range(1):
    ax = fig.add_subplot(gs3[n])
    sim[i][0].plot(energy_range=energy_range, energy_power=2)
    plot_error(self=sim[i][0], covar=covar[i][0],energy_range=energy_range,␣
↪energy_power=2)
    sim[i][n+5].plot(energy_range=energy_range, energy_power=2)
    plot_error(self=sim[i][n+5],␣
↪covar=covar[i][n+5],energy_range=energy_range, energy_power=2)
    plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[i]}')
    plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {n+5}')

plt.show()
```

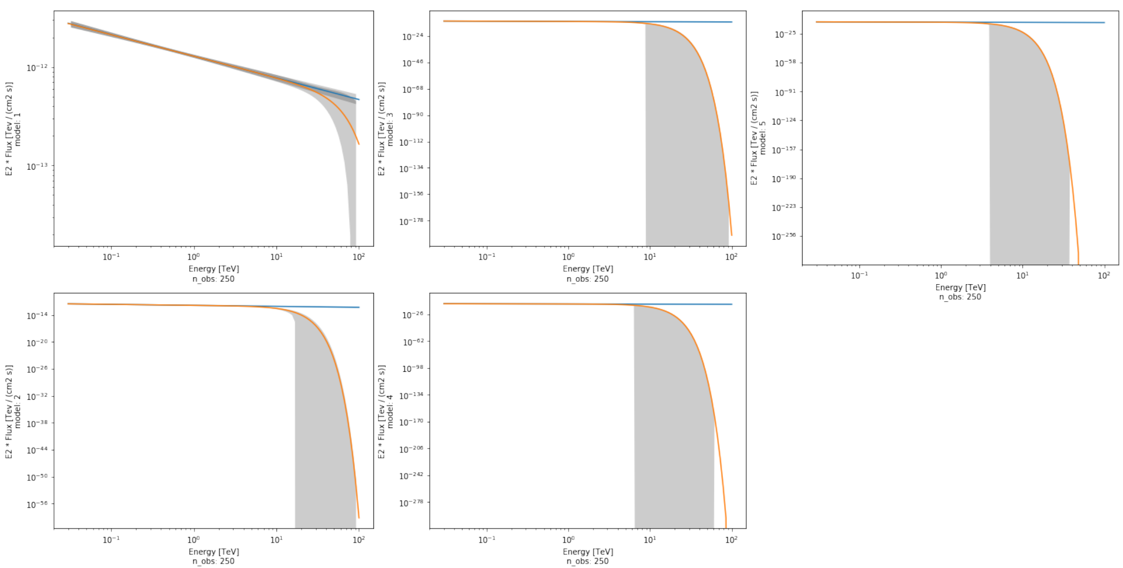

E2 * Flux [Tev / (cm2 s)]
model: 1
model: 2
model: 3
model: 4
model: 5
Energy [TeV]
n_obs: 40

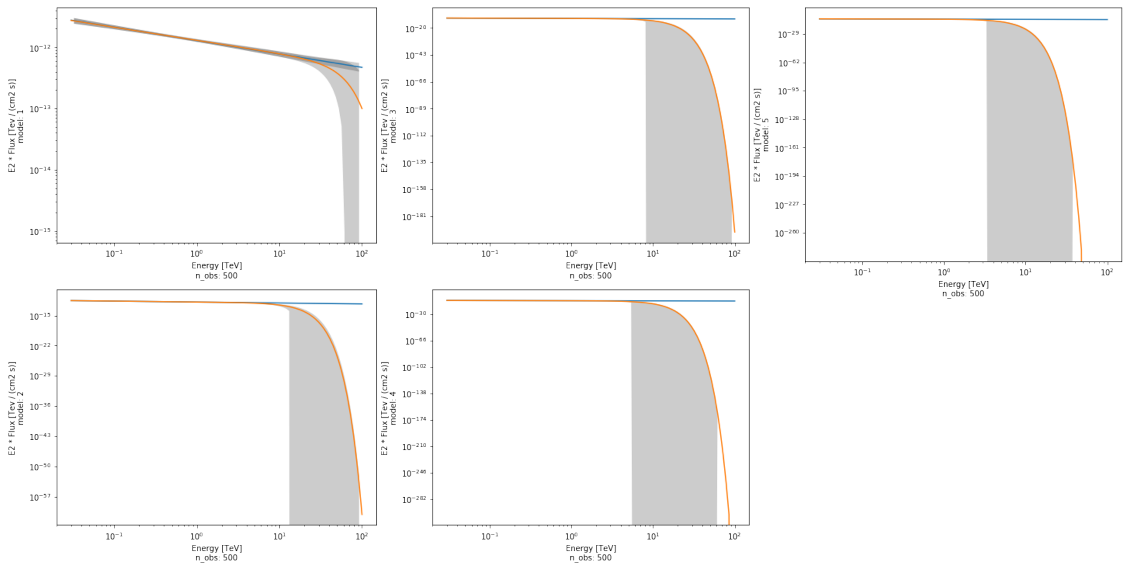

E2 * Flux [Tev / (cm2 s)]
model: 1
model: 2
model: 3
model: 4
model: 5
Energy [TeV]
n_obs: 50

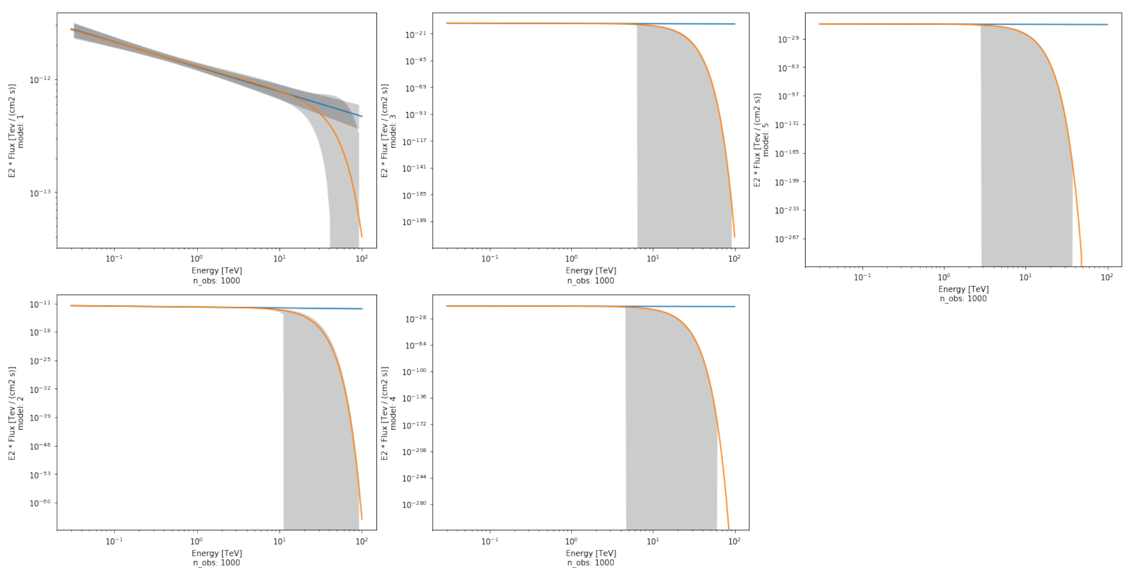

E2 * Flux [Tev / (cm2 s)]
model: 1
model: 2
model: 3
model: 4
model: 5
Energy [TeV]
n_obs: 100


E2 * Flux [Tev / (cm2 s)]
model: 1
model: 2
model: 3
model: 4
model: 5
Energy [TeV]
n_obs: 125


E2 * Flux [Tev / (cm2 s)]
model: 1
model: 2
model: 3
model: 4
model: 5
Energy [TeV]
n_obs: 200

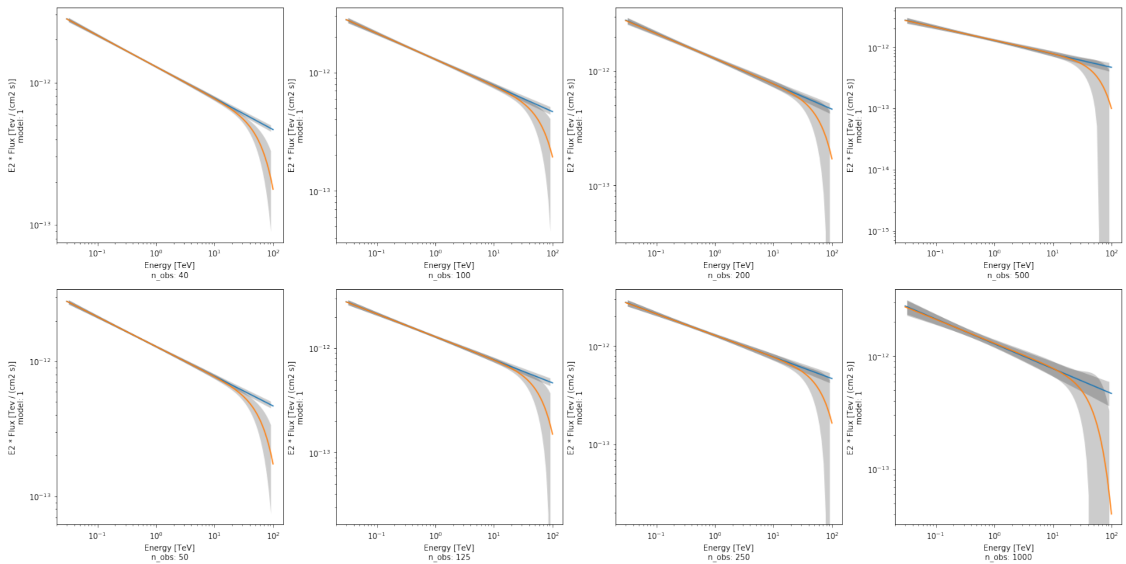

E2 * Flux [Tev / (cm2 s)]
model: 1
model: 2
model: 3
model: 4
model: 5
Energy [TeV]
n_obs: 250

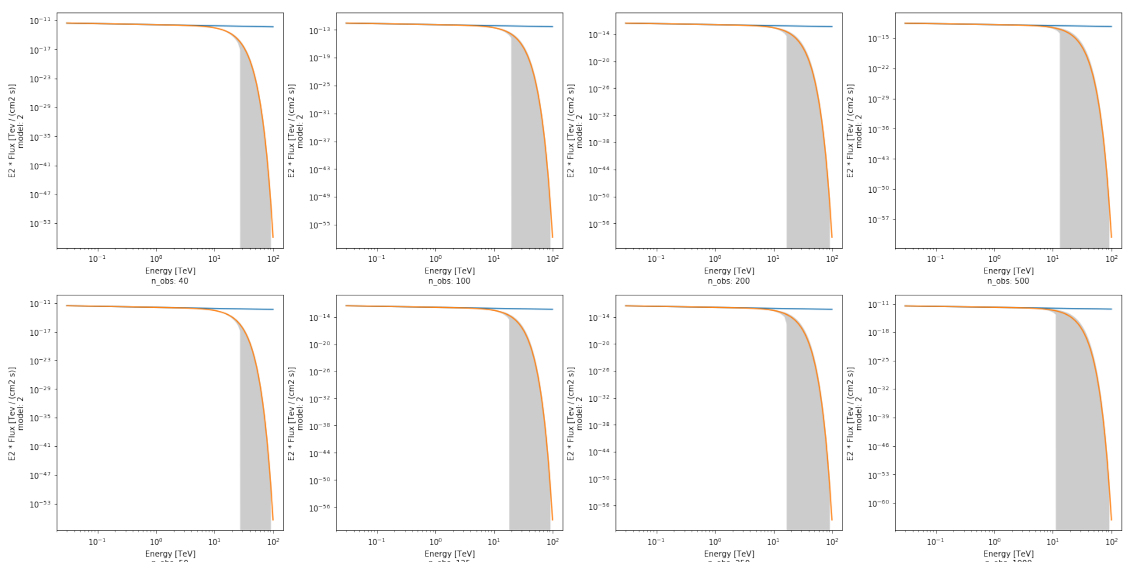


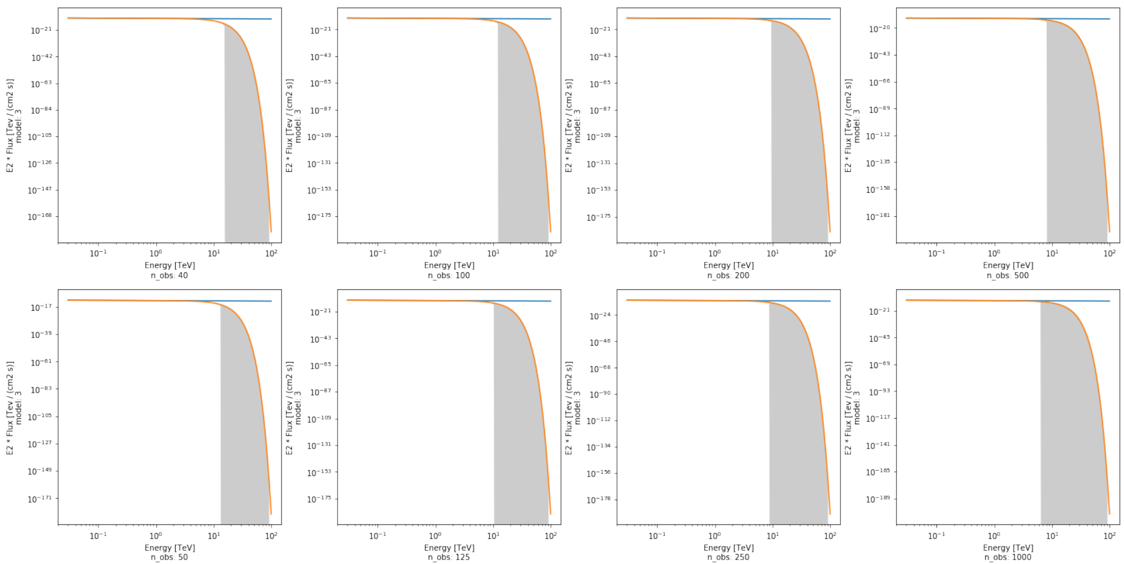


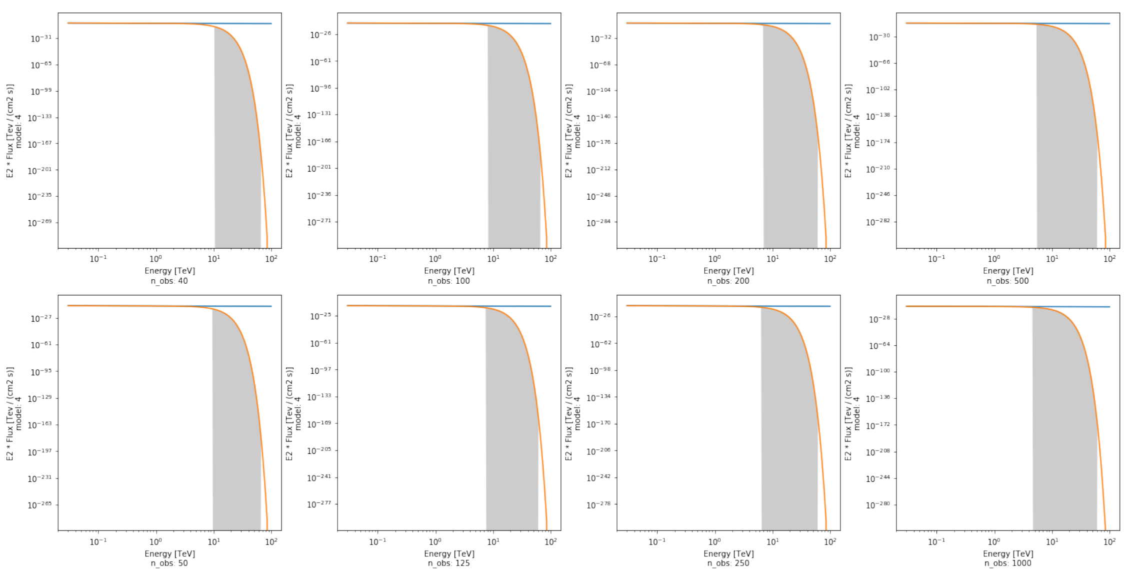
[38]:
```python
for i in range(5):
    fig = plt.figure(figsize=[20,10],constrained_layout=True)

    import matplotlib.gridspec as gridspec

    gs0 = gridspec.GridSpec(1, 4, figure=fig)

    gs1 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[0])
    for n in range(2):
        ax = fig.add_subplot(gs1[n])
```

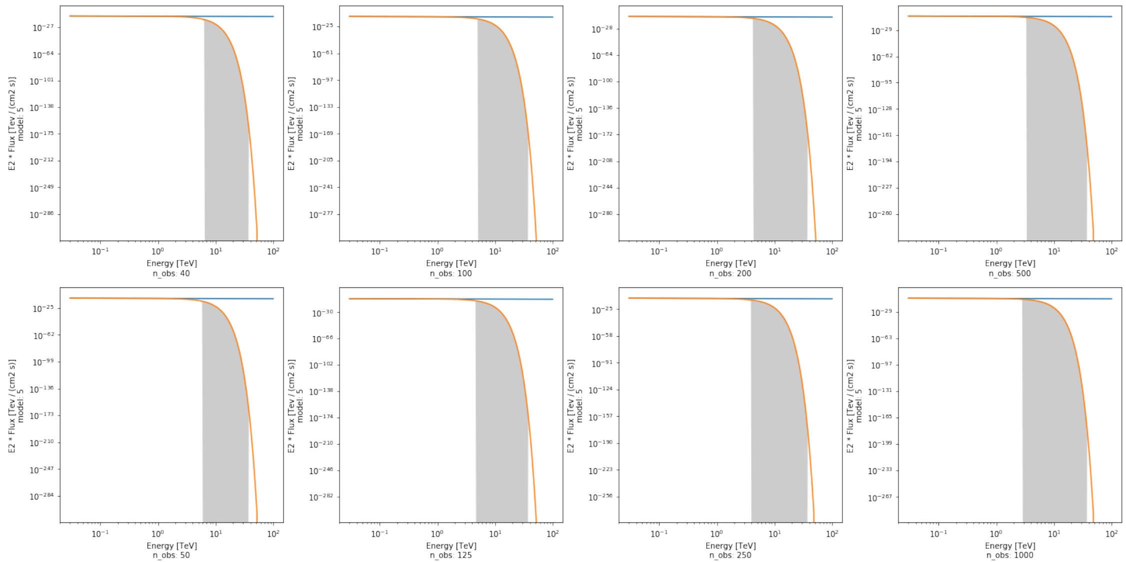
```
        sim[n][0].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n][0], covar=covar[n][0],energy_range=energy_range, energy_power=2)
        sim[n][i+1].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n][i+1], covar=covar[n][i+1],energy_range=energy_range, energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i+1}')

    gs2 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[1])
    for n in range(2):
        ax = fig.add_subplot(gs2[n])
        sim[n+2][0].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n+2][0], covar=covar[n+2][0],energy_range=energy_range, energy_power=2)
        sim[n+2][i+1].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n+2][i+1], covar=covar[n+2][i+1],energy_range=energy_range, energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n+2]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i+1}')

    gs3 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[2])
    for n in range(2):
        ax = fig.add_subplot(gs3[n])
        sim[n+4][0].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n+4][0], covar=covar[n+4][0],energy_range=energy_range, energy_power=2)
        sim[n+4][i+1].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n+4][i+1], covar=covar[n+4][i+1],energy_range=energy_range, energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n+4]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i+1}')

    gs4 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[3])
    for n in range(2):
        ax = fig.add_subplot(gs4[n])
        sim[n+6][0].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n+6][0], covar=covar[n+6][0],energy_range=energy_range, energy_power=2)
        sim[n+6][i+1].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim[n+6][i+1], covar=covar[n+6][i+1],energy_range=energy_range, energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n+6]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i+1}')

    plt.show()
```


E2 * Flux [Tev / (cm2 s)]
model: 1
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000


E2 * Flux [Tev / (cm2 s)]
model: 2
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000


E2 * Flux [Tev / (cm2 s)]
model: 3
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000


E2 * Flux [Tev / (cm2 s)]
model: 4
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000



[39]:
```
%%time
res_1 = [[0 for i in range(cols)] for j in range(rows)]
minuit_opts = {"tol": 0.001, "strategy": 1}
for i in range(8):
    results_1 = []
    for dataset in datas[i][0]:
        dataset.models = model[i][0].copy()
        fit_1 = Fit([dataset])
        result_1 = fit_1.run(optimize_opts=minuit_opts)
        results_1.append(
            {
                "index": result_1.parameters["index"].value,
                "amplitude": result_1.parameters["amplitude"].value,
                "reference":result_1.parameters["reference"].value,
                "covariance": result_1.parameters.covariance,
            }
        )
        print(result_1.parameters.to_table())
    res_1[i][0]=results_1
```

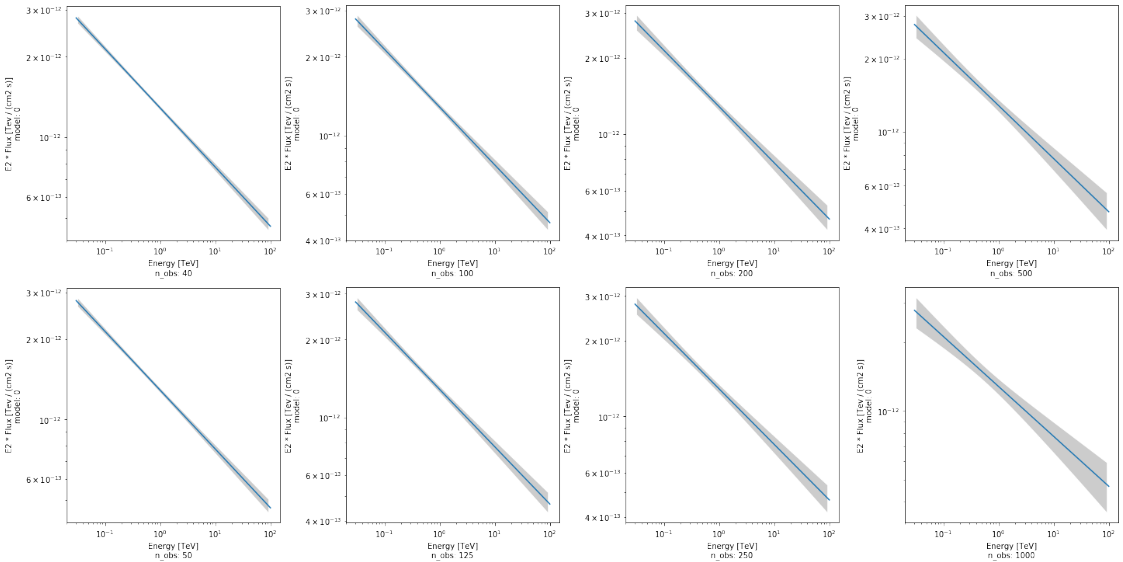
```
   name     value     error         unit      min max frozen
--------- --------- --------- -------------- --- --- ------
    index 2.210e+00 9.028e-03                nan nan  False
amplitude 1.318e-12 1.964e-14 cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00 0.000e+00            TeV nan nan   True
   name     value     error         unit      min max frozen
--------- --------- --------- -------------- --- --- ------
    index 2.222e+00 9.318e-03                nan nan  False
amplitude 1.258e-12 1.930e-14 cm-2 s-1 TeV-1 nan nan  False
```

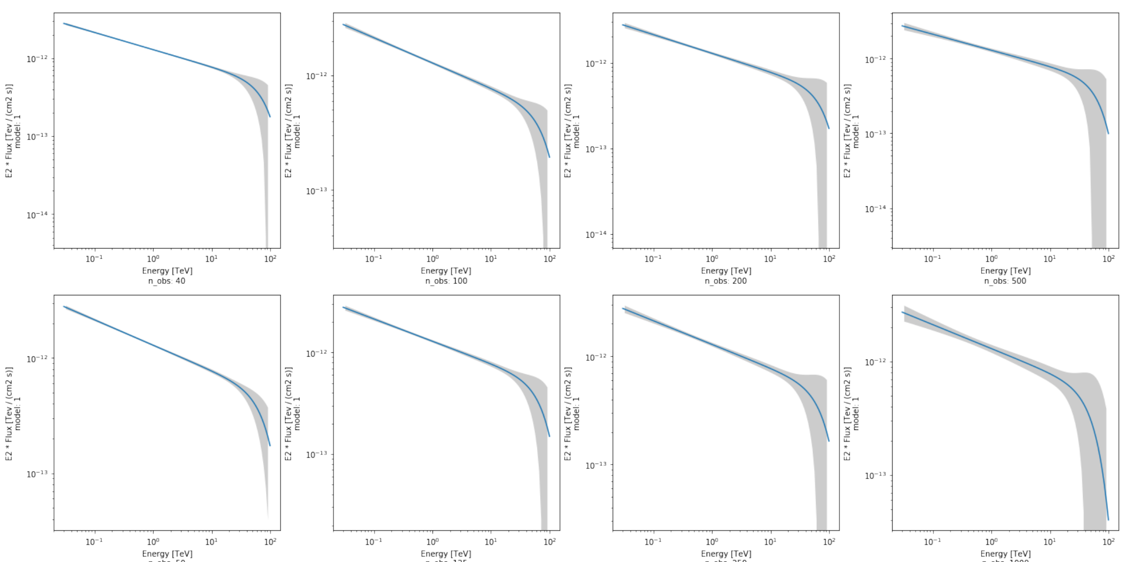
```
    alpha 2.000e+00   nan                nan nan   True
ExpCutoffPowerLawSpectralModel

   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.220e+00   nan                nan nan  False
amplitude 1.297e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.060e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
ExpCutoffPowerLawSpectralModel

   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.215e+00   nan                nan nan  False
amplitude 1.308e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.105e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
ExpCutoffPowerLawSpectralModel

   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.212e+00   nan                nan nan  False
amplitude 1.329e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.148e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
ExpCutoffPowerLawSpectralModel

   name     value   error      unit      min max frozen
--------- --------- ----- -------------- --- --- ------
    index 2.204e+00   nan                nan nan  False
amplitude 1.354e-12   nan cm-2 s-1 TeV-1 nan nan  False
reference 1.000e+00   nan            TeV nan nan   True
  lambda_ 5.215e-01   nan          TeV-1 nan nan  False
    alpha 2.000e+00   nan                nan nan   True
```

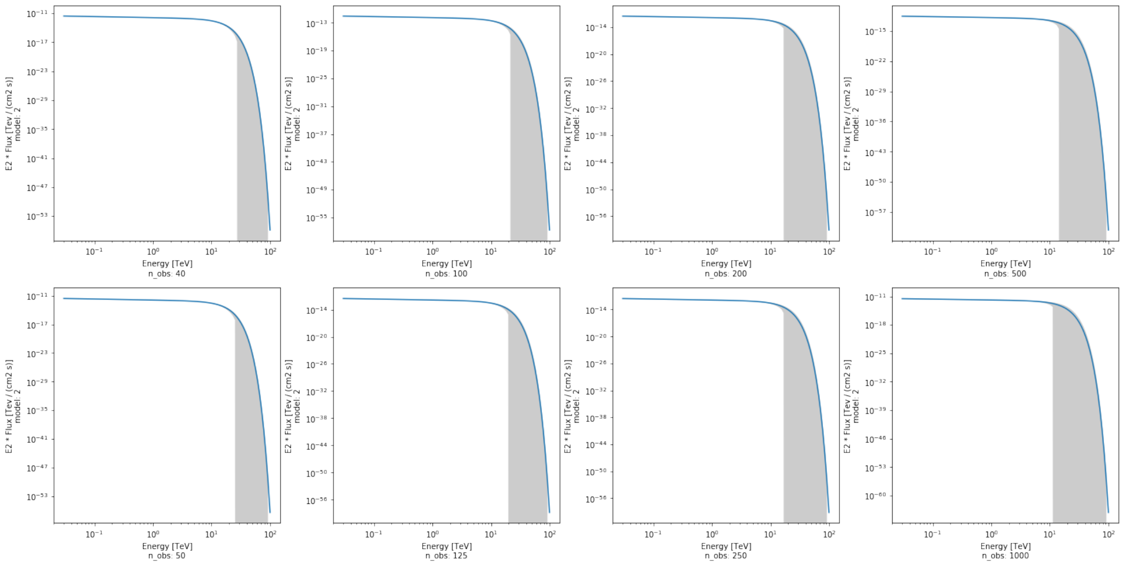
[43]:

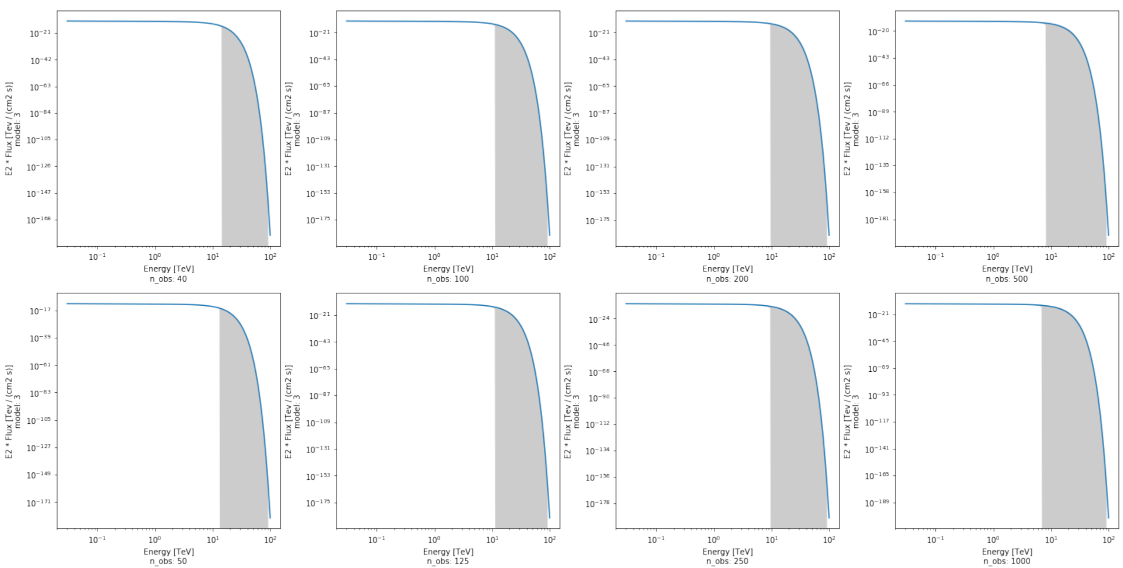
```
for i in range(6):
    fig = plt.figure(figsize=[20,10],constrained_layout=True)

    import matplotlib.gridspec as gridspec

    gs0 = gridspec.GridSpec(1, 4, figure=fig)

    gs1 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[0])
    for n in range(2):
```

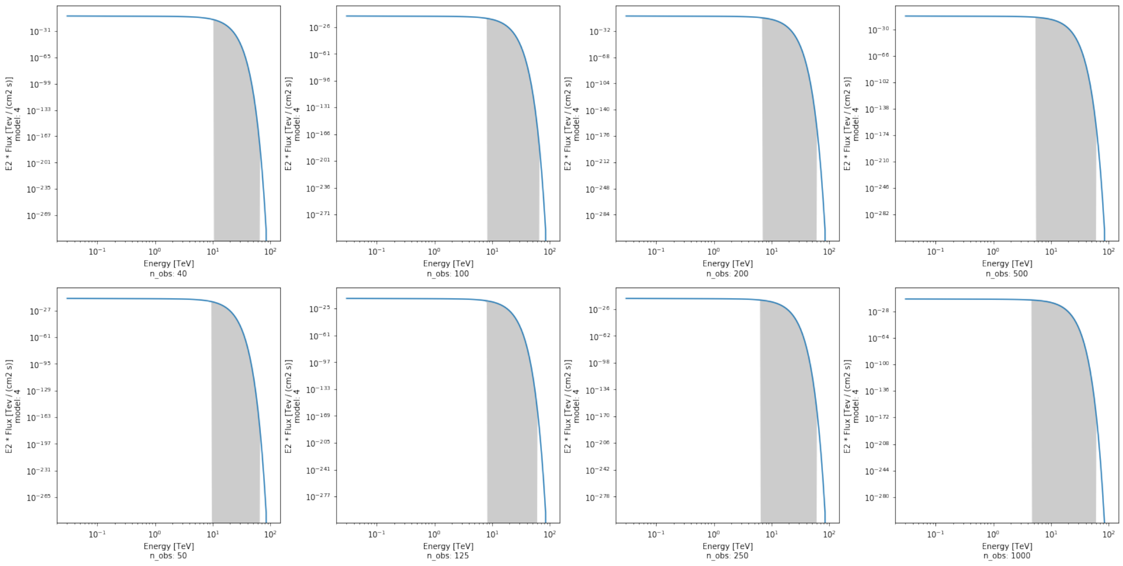
```
        ax = fig.add_subplot(gs1[n])
        sim_1[n][i].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim_1[n][i], covar=np.mean(covar_1[n][i], axis =␣
↪0),energy_range=energy_range, energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i}')

    gs2 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[1])
    for n in range(2):
        ax = fig.add_subplot(gs2[n])
        sim_1[n+2][i].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim_1[n+2][i], covar=np.mean(covar_1[n+2][i], axis =␣
↪0),energy_range=energy_range, energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n+2]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i}')

    gs3 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[2])
    for n in range(2):
        ax = fig.add_subplot(gs3[n])
        sim_1[n+4][i].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim_1[n+4][i], covar=np.mean(covar_1[n+4][i], axis =␣
↪0),energy_range=energy_range, energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n+4]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i}')

    gs4 = gridspec.GridSpecFromSubplotSpec(2, 1, subplot_spec=gs0[3])
    for n in range(2):
        ax = fig.add_subplot(gs4[n])
        sim_1[n+6][i].plot(energy_range=energy_range, energy_power=2)
        plot_error(self=sim_1[n+6][i], covar=np.mean(covar_1[n+6][i], axis =␣
↪0),energy_range=energy_range, energy_power=2)
        plt.xlabel(f'Energy [TeV]\nn_obs: {n_obs[n+6]}')
        plt.ylabel(f'E2 * Flux [Tev / (cm2 s)]\nmodel: {i}')

    plt.show()
```

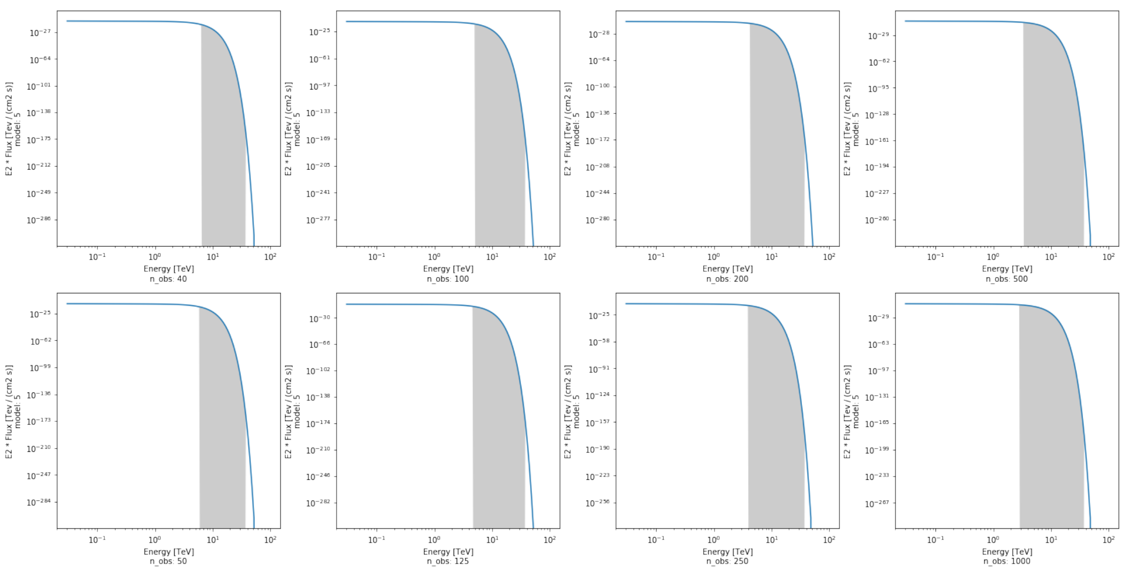

E2 * Flux [Tev / (cm2 s)]
model: 0
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000


E2 * Flux [Tev / (cm2 s)]
model: 1
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000


E2 * Flux [Tev / (cm2 s)]
model: 2
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000


E2 * Flux [Tev / (cm2 s)]
model: 3
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000


E2 * Flux [Tev / (cm2 s)]
model: 4
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000


E2 * Flux [Tev / (cm2 s)]
model: 5
Energy [TeV]
n_obs: 40
n_obs: 100
n_obs: 200
n_obs: 500
n_obs: 50
n_obs: 125
n_obs: 250
n_obs: 1000

[ ]: