SimWeights Tutorial

Kevin Meagher
University of Wisconsin
14 June 2024
IceProd/SimProd Workshop

IceCube Neutrino Observatory

Wisconsin IceCube Particle Astrophysics Center

Wisconsin University of Wisconsin–Madison

NSF

What are Simulation Weights?

We want to know how many events we observe, theoretically this is an integral:

$$N_{obs} = \int \mathrm{d}E \int \mathrm{d}A \int \mathrm{d}\Omega \int \mathrm{d}t \cdot \mathscr{D}(E, \theta, \phi, \vec{x}) \cdot \Phi(E, \theta, \phi, t)$$

In IceCube the integration is a flux $\Phi(E, \theta, \phi, t)$ incident on the surface of the Earth integrated over energy, time, area and solid angle. With some function $\mathscr{D}(E, \theta, \phi)$ which represents the probability of an event passing our final level quality cuts. We can use Monte Carlo integration to convert this into a sum

$$R = \sum_{i=1}^{N_{gen}} \mathscr{D}_i \cdot \Phi(E_i, \theta_i, \phi_i)$$

Where R is the rate of events and $\mathscr{D}_i$ is a binary 1 for passes quality cuts and 0 otherwise

Use importance sampling

Over sample a certain region of the generation surface with probability $p(E)$ so that our sum becomes

$$R = \sum_{i=1}^{N_{gen}} g_i \cdot \mathcal{D}_i \cdot \Phi(E_i, \theta_i, \phi_i)$$

Where the weight is defined as

$$g_i = \frac{1}{N_{gen} \cdot p(E_i)}$$

Calculating the weight

In order to get the math to work out right $p(E_i)$ must be the probability of generating an event on the generation surface.

The generation surface contains Area, Solid Angle, as well as energy and for neutrinos the probability of interaction.

For Neutrino Generator with an E^-1 generation spectrum this becomes:

$$g_i = \frac{p_{int}}{N_{gen} \cdot \pi R^2 \cdot 4\pi \cdot \ln(E_2/E_1) E_i^{-1}}$$

What if you use more than one dataset?

If you try to combine datasets with different energy spectra you need to combine the weights like so:

$$g_i = \frac{p_{int}}{N_1 \cdot p_1(E_i) + N_2 \cdot p_2(E_i)}$$

Most of the quantities you need to weight data are available in the I3Frame but correctly calculating weights becomes a book keeping problem as it is often unclear how many files from the dataset were used.

Past Attempts to Weight IceCube Simulation

- Copypasta — Just copy the correct formulas from somebody else.
  - Pros: Easy to see the formula you are using
  - Cons: Very Error prone, not clear if formula from one dataset applies to another dataset
- OneWeight - Save g_i to the I3Frame
  - Pros: Works great if you are only using one dataset or datasets with identical generation surface
  - Cons: Fails horribly when combining datasets, also it is stored as the reciprocal of how you should be thinking about weights
- IceTray's `icecube.weighting` module
  - Pros: Can combine datasets easily
  - Cons: requires database infrastructure, database is missing lots of datasets, requires complete IceTray

Sim Weights was developed with the following requirements

- Replacement for icetray's weighting project
- Installable with pip
- Doesn't depend on IceTray
- Calculate weights based solely on files generated by hdfwriter
  - Doesn't query database
  - Doesn't require sidecar files
- Easily combine weights from different datasets

- Calculating weights is Straightforward
- Create a Weight object from an hdf5 file
- To get the weights pass a flux model to Weight.get_weights()

```
import simweights, pandas
simfile = pandas.HDFStore('Level2_IC86.2020_corsika.021111.hdf5','r')
flux_model = simweights.GaisserH4a()
weight_obj = simweights.CorsikaWeighter(simfile)
weights = weight_obj.get_weights(flux_model)
print('Rate',weights.sum(),'Hz')
```

- Cosmic Ray flux models are available in simweights
- Hdfwriter will correctly add data from S-Frames to its output if the key is included

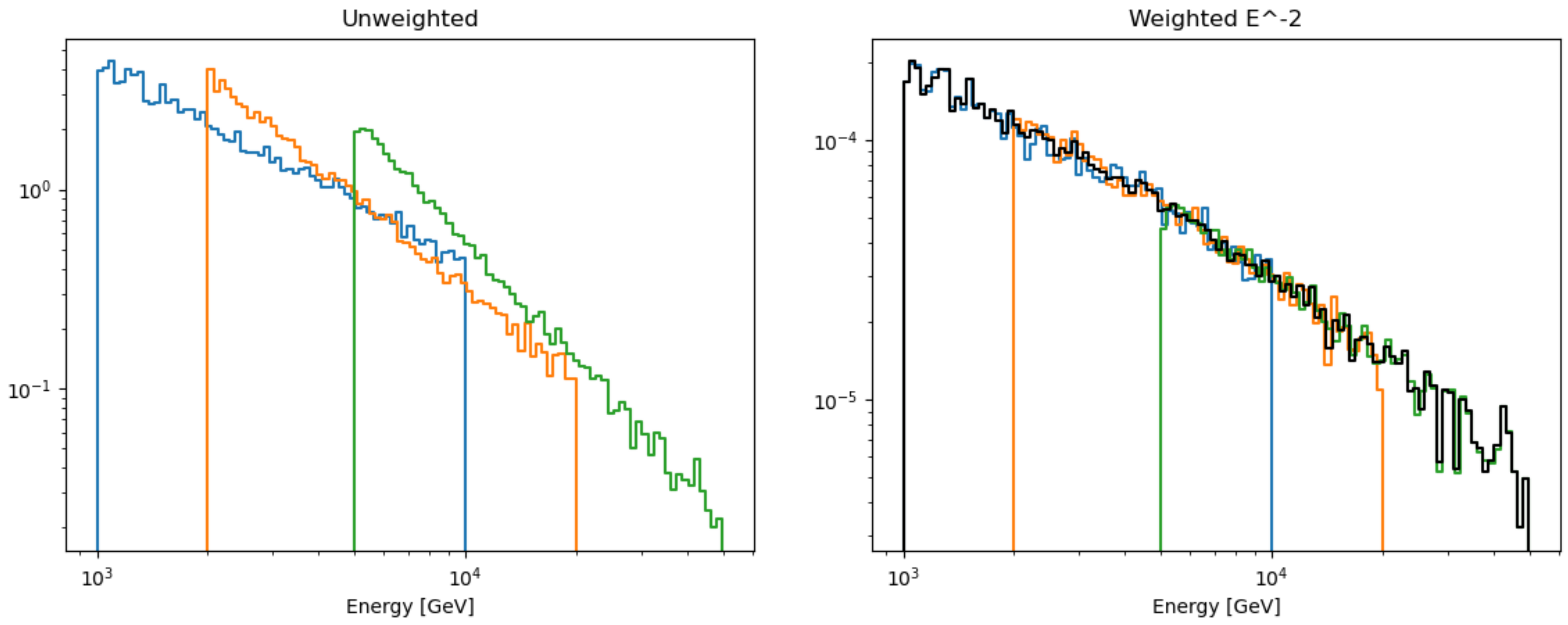
For older CORSIKA you still need to pass nfiles parameter

```
simfile = pandas.HDFStore('Level2_IC86.2016_corsika.020789.hdf5','r')
weight_obj = simweights.CorsikaWeighter(simfile,nfiles=10)
```

simweights will automatically convert the units for NuFlux models

```
simfile = pandas.HDFStore('Level2_IC86.2016_NuMu.020878.000000.hdf5')
flux_model = nuflux.makeFlux('CORSIKA_GaisserH3a_QGSJET-II')
weight_obj=simweights.NuGenWeighter(simfile,nfiles=10)
weights = weight_obj.get_weights(flux_model)
```

Combining datasets with different generation spectra can be done by adding weight objects




```
w1 = NuGenWeighter(pd.HDFStore(‘f1.hdf5’,’r’),nfiles=1)
w2 = NuGenWeighter(pd.HDFStore(‘f2.hdf5’,’r’),nfiles=1)
w3 = NuGenWeighter(pd.HDFStore(‘f3.hdf5’,’r’),nfiles=1)
wtotal = w1 + w2 + w3
```

Simulation types supported by simweights

Simulation Type	Supported	Notes
Dynamic Stack CORSIKA	✅	Uses S-Frames
CORSIKA-in-ice	✅	Needs nfiles
neutrino-generator	✅	Needs nfiles
CORSIKA-ice-top	✅	Uses S-Frames
Old GENIE	❌	Similar to nugen
New GENIE	✅	Uses S-Frames
LeptonInjector	❌	Needs Side Car Files
MuonGun	❌	Needs Side Car Files
WimpSim	❌	Uses Discrete Energy
MonopoleGenerator	❌	Uses Discrete Energy

Tutorial

Creating an IceTray venv on cobalt

```
ssh cobalt
/cvmfs/icecube.opensciencegrid.org/py3-v4.3.0/icetray-env icetray/v1.9.2
python -m venv ~/.venv311 --system-site-packages
source ~/.venv311/bin/activate
pip install simweights nuflux
```

Generate neutrinos with no cuts

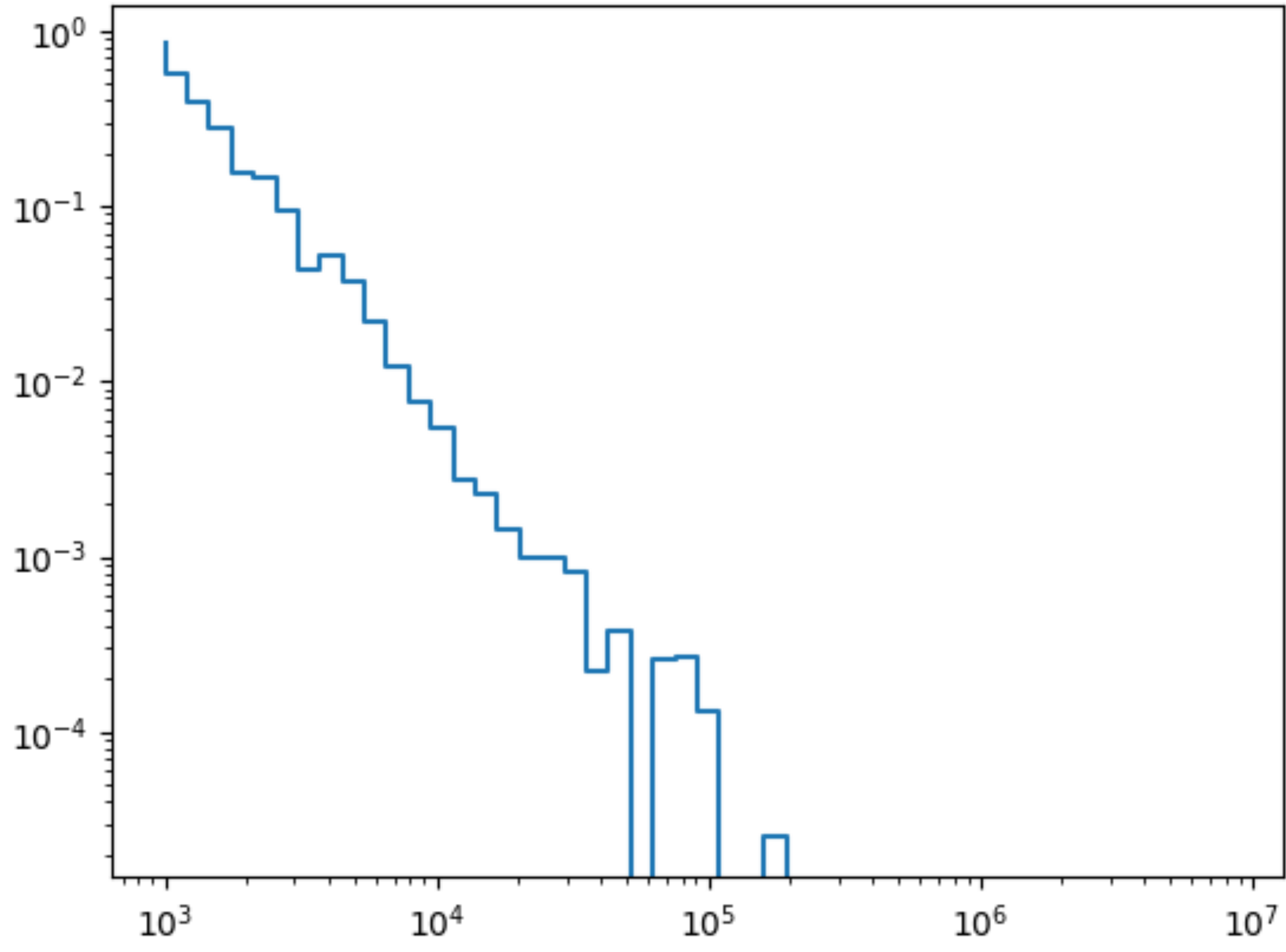
```
from icecube import icetray, phys_services,hdfwriter
from icecube.simprod import segments
from icecube.simprod.util import DAQCounter

print(dir(segments))

N = 1000
tray = icetray.I3Tray()
tray.Add("I3GSLRandomServiceFactory")
tray.AddModule("I3InfiniteSource", "TheSource",
                   Stream=icetray.I3Frame.DAQ)
tray.AddModule(DAQCounter, "counter3", nevents=N)
tray.Add(segments.GenerateNeutrinos,NumEvents=N)
tray.Add("Dump")
tray.Add(hdfwriter.I3SimHDFWriter,
    keys=["I3MCWeightDict","NuGPrimary"],
    OutPut="nugen.hdf5",
)
tray.Execute()
```

The keys needed for each type of simulation can be found in https://docs.icecube.aq/simweights/main/reading_files.html

type	S-Frame	Q-Frame
Triggered CORSIKA	I3PrimaryInjectorInfo	I3CorsikaWeight
S-Frame CORSIKA	I3CorsikaInfo	PolyplopiaPrimary
CORSIKA without S-Frames	none	CorsikaWeightMap, PolyplopiaPrimary
neutrino-generator	none	I3MCWeightDict
genie-reader	I3GenieInfo	I3GenieResult

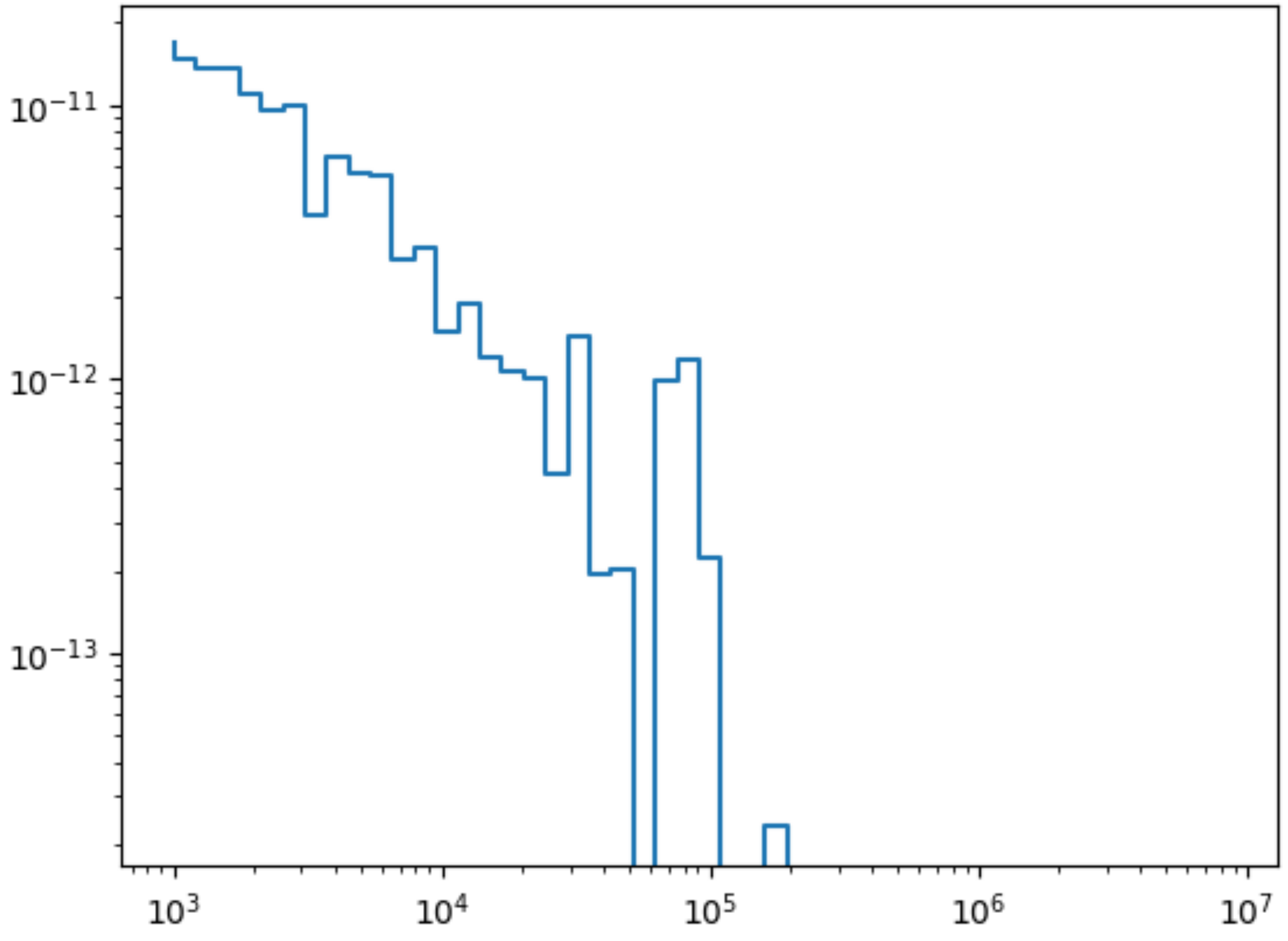
Plotting NuGen Unweighted

```
from pprint import pprint
import tables
import numpy as np
import pylab as plt
import simweights

def log_plot(A,B,data,weights,bins=10):
    hy,hx = np.histogram(data, bins = np.geomspace(A,B,bins),weights=weights)
    hd = hx[1:] - hx[:-1]
    plt.step(hx[:-1],hy/hd)
    plt.loglog()

f1 = tables.open_file('nugen.hdf5')
weighter = simweights.NuGenWeighter(f1,nfiles=1)

E1 = 10**f1.root.I3MCWeightDict.cols.MinEnergyLog[0]
E2 = 10**f1.root.I3MCWeightDict.cols.MaxEnergyLog[0]
N = f1.root.I3MCWeightDict.shape
print(N)

log_plot(E1,E2,weighter.get_weight_column("energy"),np.ones(N),bins=50)
plt.savefig('nugen_unweighted.png')
```



Plotting Correctly weighted NuGen

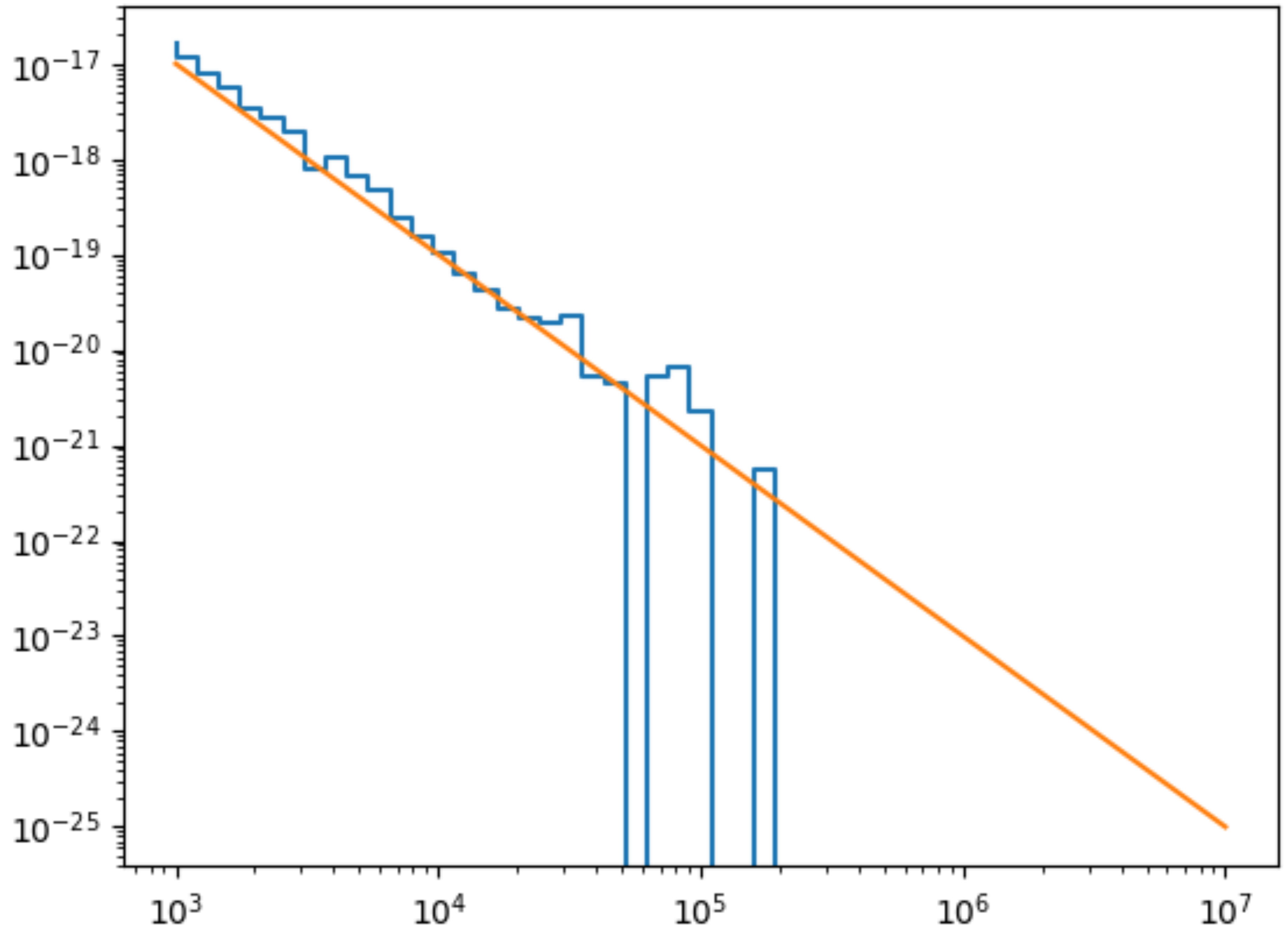
```
plt.figure()
def flux_model(energy):
    return 1e-11 * energy ** -2

plt.figure()
log_plot(E1,E2,weighter.get_weight_column("energy"),weighter.get_weights(flux_model),bins=50)
plt.savefig('nugen_weighted.png')
```



Get the flux back from the simulation

You need to divide the weight but the interaction probability Solid angle and injected area to get back to the flux you started with

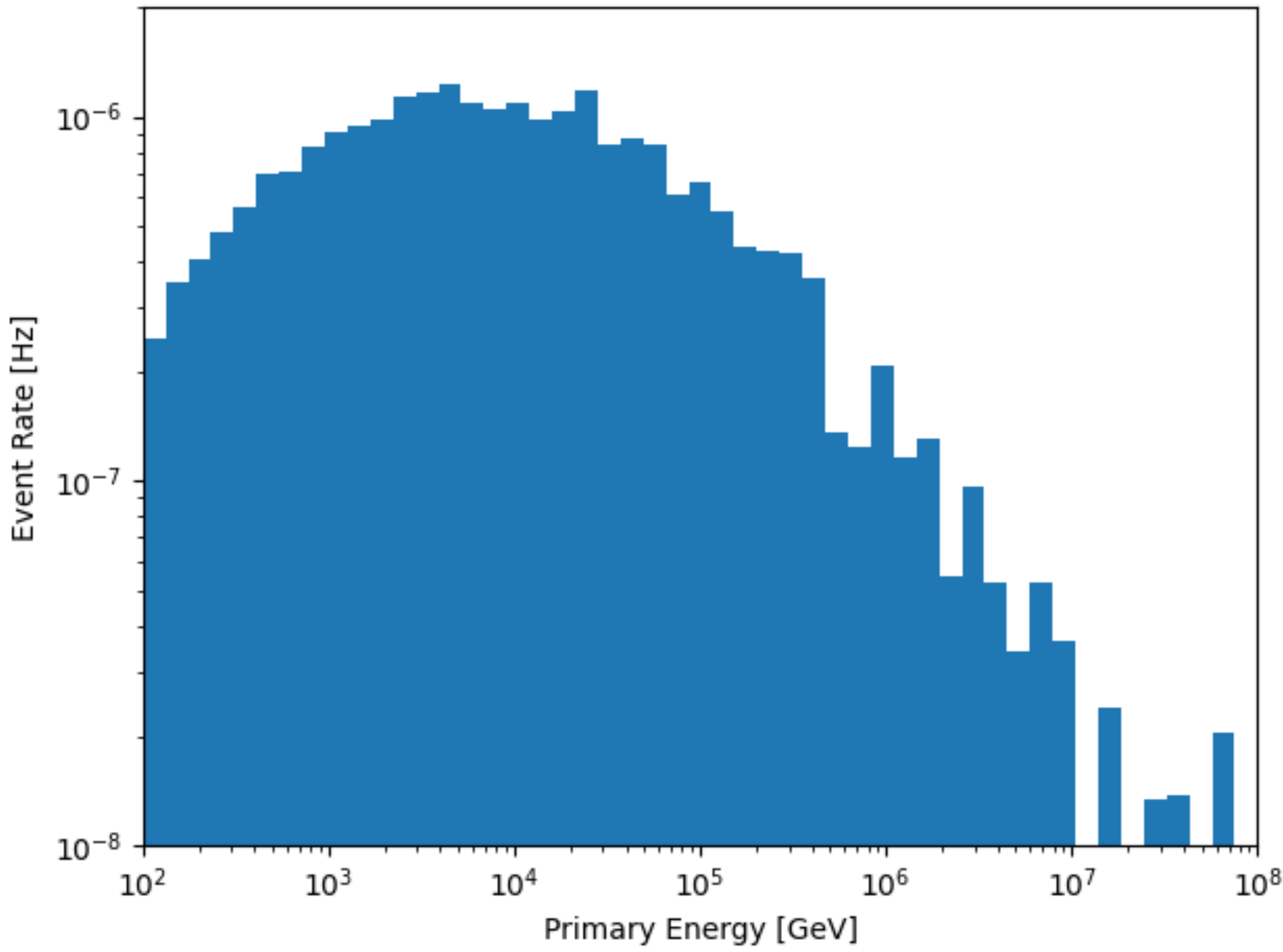
```
plt.figure()
weight = weighter.get_weights(flux_model)
weight /= f1.root.I3MCWeightDict.cols.InteractionWeight[:]
weight /= f1.root.I3MCWeightDict.cols.InjectionAreaCGS[:]
weight /= f1.root.I3MCWeightDict.cols.SolidAngle[:]
log_plot(E1, E2, weighter.get_weight_column("energy"), weight, bins=50)
EE = np.geomspace(E1, E2, 1000)
plt.plot(EE, flux_model(EE))
plt.savefig("nugen_flux.png")
```



Booking an existing nugen file to hdf5 file

```
from pathlib import Path

from icecube import icetray, hdfwriter, simclasses

FILE_DIR = Path("/data/sim/IceCube/2016/filtered/level2/neutrino-generator/21217/0000000-0000999/")
files = sorted(str(f) for f in FILE_DIR.glob("Level2_IC86.2016_NuMu.021217.00000*.i3.zst"))

tray = icetray.I3Tray()
tray.Add("I3Reader", FileNameList=files)
tray.Add(
    hdfwriter.I3HDFWriter,
    SubEventStreams=["InIceSplit"],
    keys=["PolyplopiaPrimary", "I3MCWeightDict"],
    output="Level2_IC86.2016_NuMu.021217.hdf5",
)

tray.Execute()
```

```python
import pandas as pd
import pylab as plt
import simweights

# load the hdf5 file that we just created using pandas
hdffile = pd.HDFStore("Level2_IC86.2016_NuMu.021217.hdf5", "r")

# instantiate the weighter object by passing the pandas file to it
weighter = simweights.NuGenWeighter(hdffile, nfiles=10)

def northern_track(energy: ArrayLike) -> ArrayLike:
    return 1.44e-18 / 2 * (energy / 1e5) ** -2.2


# get the weights by passing the flux to the weighter
weights = weighter.get_weights(northern_track)

# print some info about the weighting object
print(weighter.tostring(northern_track))

# create equal spaced bins in log space
bins = plt.geomspace(1e2, 1e8, 50)

# get energy of the primary cosmic-ray from `PolyplopiaPrimary`
primary_energy = weighter.get_column("PolyplopiaPrimary", "energy")

# histogram the primary energy with the weights
plt.hist(primary_energy, weights=weights, bins=bins)

# make the plot look good
plt.loglog()
plt.xlabel("Primary Energy [GeV]")
plt.ylabel("Event Rate [Hz]")
plt.xlim(bins[0], bins[-1])
plt.ylim(1e-8, 2e-6)
plt.tight_layout()
plt.savefig('Level2_IC86.2016_NuMu.021217.png')
```



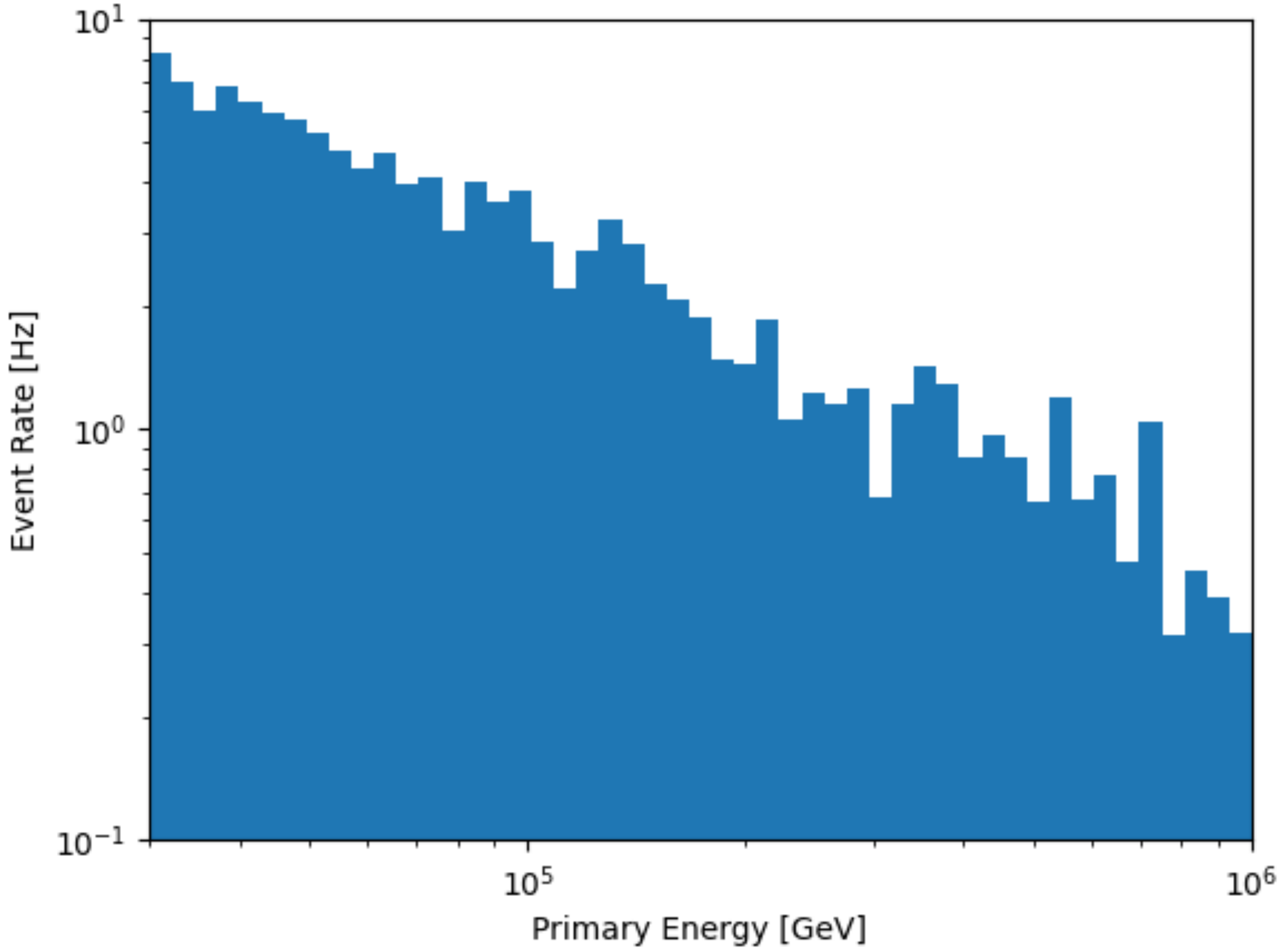
Booking Triggered CORSIKA

```
from pathlib import Path
from icecube import icetray, hdfwriter, simclasses

FILE_DIR = Path("/data/sim/IceCube/2016/filtered/level2/CORSIKA-in-ice/21889/0000000-0000999")
files = sorted(str(f) for f in FILE_DIR.glob("Level2_IC86.2016_corsika.021889.00000*.i3.zst"))

tray = icetray.I3Tray()
tray.Add("I3Reader", FileNameList=files)
tray.Add(
    hdfwriter.I3HDFWriter,
    SubEventStreams=["InIceSplit"],
    keys=["PolyplopiaPrimary", "I3PrimaryInjectorInfo", "I3CorsikaWeight"],
    output="Level2_IC86.2016_corsika.021889.hdf5",
)

tray.Execute()
```

I3PrimaryInjectorInfo Is an S-Frame and
can be booked just like objects in Q-Frames

Weighting Triggered CORSIKA

```
import pandas as pd
import pylab as plt
import simweights
# load the hdf5 file that we just created using pandas
hdffile = pd.HDFStore("Level2_IC86.2016_corsika.021889.hdf5", "r")
# instantiate the weighter object by passing the pandas file to it
weighter = simweights.CorsikaWeighter(hdffile)
# create an object to represent our cosmic-ray primary flux model
flux = simweights.GaisserH4a()
# get the weights by passing the flux to the weighter
weights = weighter.get_weights(flux)
# print some info about the weighting object
print(weighter.tostring(flux))
# create equal spaced bins in log space
bins = plt.geomspace(3e4, 1e6, 50)
# get energy of the primary cosmic-ray from `PolyplopiaPrimary`
primary_energy = weighter.get_column("PolyplopiaPrimary", “energy")
# histogram the primary energy with the weights
plt.hist(primary_energy, weights=weights, bins=bins)
# make the plot look good
plt.loglog()
plt.xlabel("Primary Energy [GeV]")
plt.ylabel("Event Rate [Hz]")
plt.xlim(bins[0], bins[-1])
plt.ylim(0.1, 10)
plt.savefig("Level2_IC86.2016_corsika.021889.png")
```



Triggered CORSIKA Has S-Frames so there is no need to keep track of the Number of files

Getting Help

- Consult the documentation at https://docs.icecube.aq/simweights/main/index.html
- Ask for help on #software
- File an issue on GitHub: https://github.com/icecube/simweights/issues