Ingredients

full statistical model

model parameters (physics)

stochasticity inherent to physics model
(generation of neutrino energies)

true particle properties (latent random variables)
marginalized out of final model

stochasticity inherent to detection model
(e.g. particle interactions, electronics)

observables (random variables)
to go into likelihood

(could explain relationship to hierarchical models, marginalization on blackboard) our **model parameters** (3 in total) are related to the **neutrino fluxes**

one relative **normalization factor** for atmospheric neutrino flux (background)

(intensity of the background)

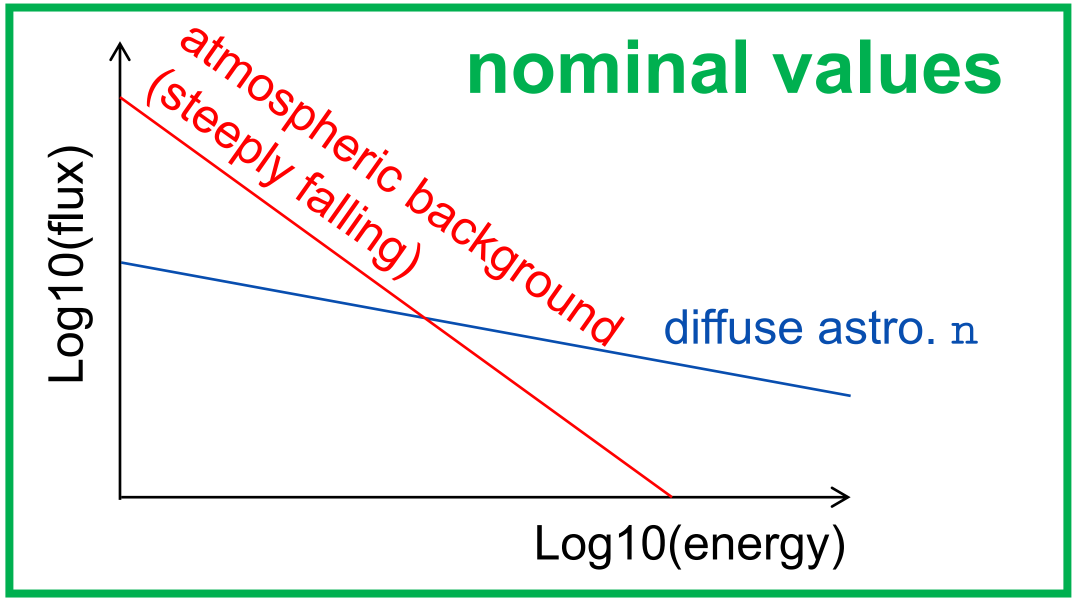
one **normalization factor** for astrophysical neutrino flux (signal)

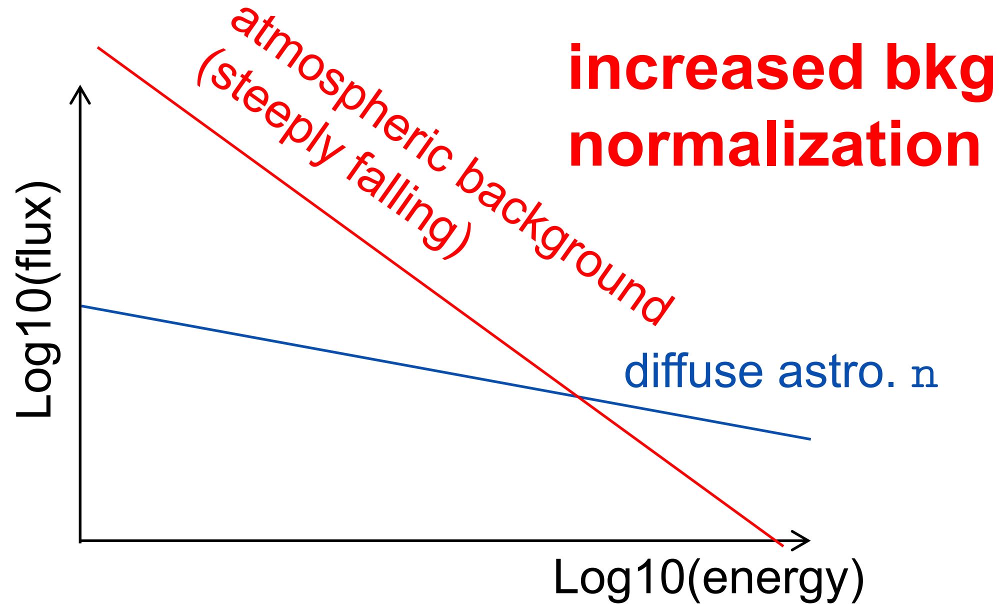
(intensity of signal)

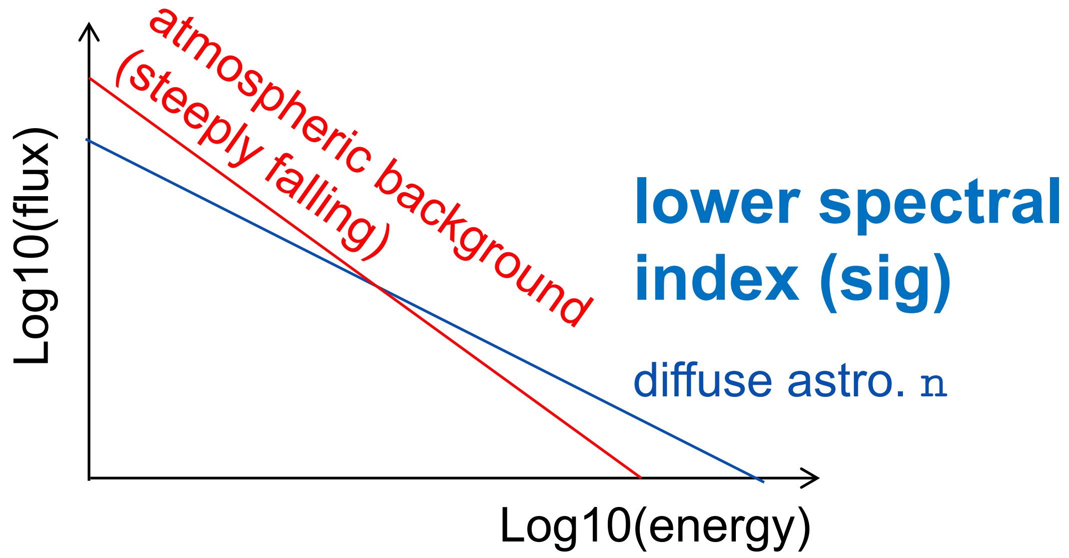
one **spectral index** for the astrophysical neutrino flux (signal)

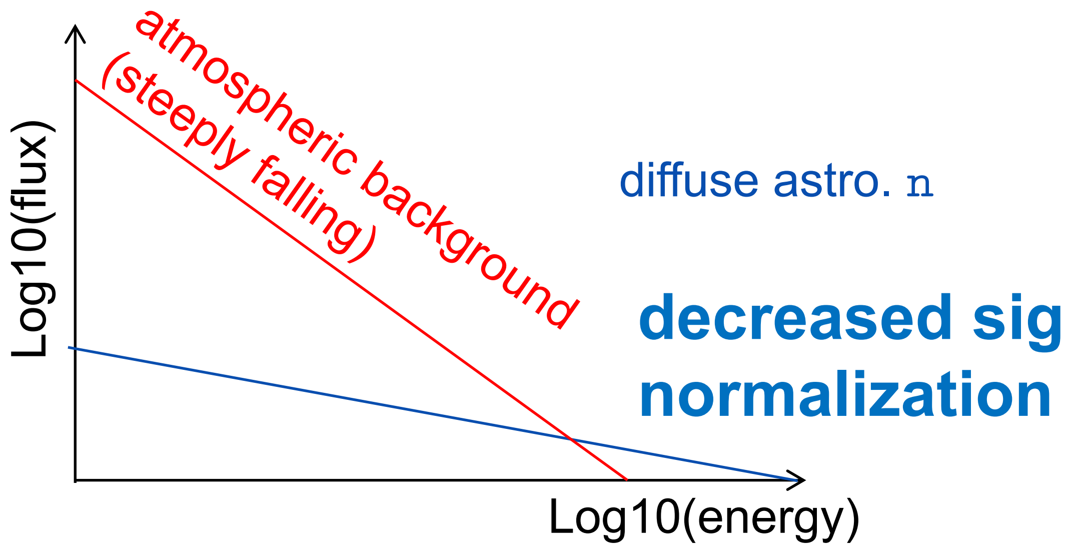
(shape of signal)

effect of model parameters









the list of observables

1) **reconstructed deposited energy** of the neutrino induced particle shower

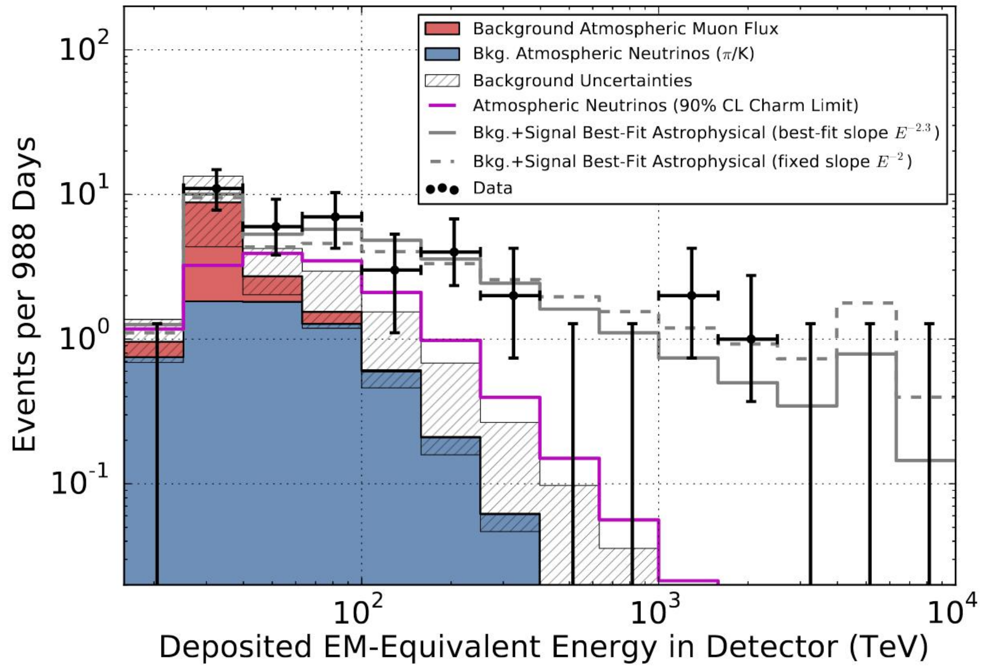
2) **reconstructed zenith angle direction** of the particle shower

(this one is not used in our example. advanced students are welcome to expand the example to study by how much the fit and significance improve once the additional observable is included!)

Question
How do the observable distributions depend on the assumptions about the neutrino fluxes?

Answer
dictated by detector response (no analytic expression)
BUT we understand the individual steps involved!

(hierarchical model!)



idea 1) use a binned analysis

idea 2) for a given set of input parameters perform a large **Monte Carlo simulation** of the detector

for each MC event tabulate the generated observables
-> predict what fraction of events survives analysis cuts
-> predict what fraction of surviving events lands in each analysis bin (this is all we need in a binned analysis)

idea 3) instead of generating one simulation for each set of parameter values we do it once for some arbitrary assumptions

correct predictions (for each parameter combination) can be obtained from **importance weights**!

(supplementary explanations on blackboard)

Basic Importance Weights

We often face the problem of evaluating integrals through Monte-Carlo sampling. In some problems naive implementations can turn out to be rather inefficient, i.e. have high variance. Importance sampling is a method to reduce variance at minimial extra computational cost.
Consider the example of calculating the average value $\mu = E_f\left(g(x)\right)$ of function $g(x)$ w.r.t the pdf $f(x)$. A suitable MC estimate can be obtained by sampling f(x) directly ($X_i \sim f(x)$)

$$\mu = \int g(x)f(x)dx \tag{1}$$

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} g(x_i) \tag{2}$$

However if g(x) is large in a region where density f(x) is small, we will rarely obtain samples in the region of interest and thus the estimate will be noisy.

Basic Importance Weights

In order to reduce noise, one can instead choose to base the estimate on samples generated from a pdf $h(x)$ with higher density in the region of interest ($x_i \sim h(x)$).

$$\mu = \int g(x)f(x)dx = \int \frac{g(x)f(x)}{h(x)} h(x)dx = E_h(\frac{g(x)f(x)}{h(x)}) \tag{3}$$

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} \frac{g(x_i)f(x_i)}{h(x_i)} \tag{4}$$

The fact that we used $h(x)$ instead of $f(x)$ to estimate μ is corrected by using the importance weights $w_i = \frac{f(x_i)}{h(x_i)}$ in what has now become a weighted average.
A demonstration of this technique can be found in two ipython notebooks (check course repo):

- example 1: weighted_average_ex1.ipynb
- example 2: weighted_average_ex2.ipynb

This technique is used extensively in IceCube (and other HEP experiments) to estimate quantities of interest from Monte Carlo datasets.

Generating observables through Monte Carlo

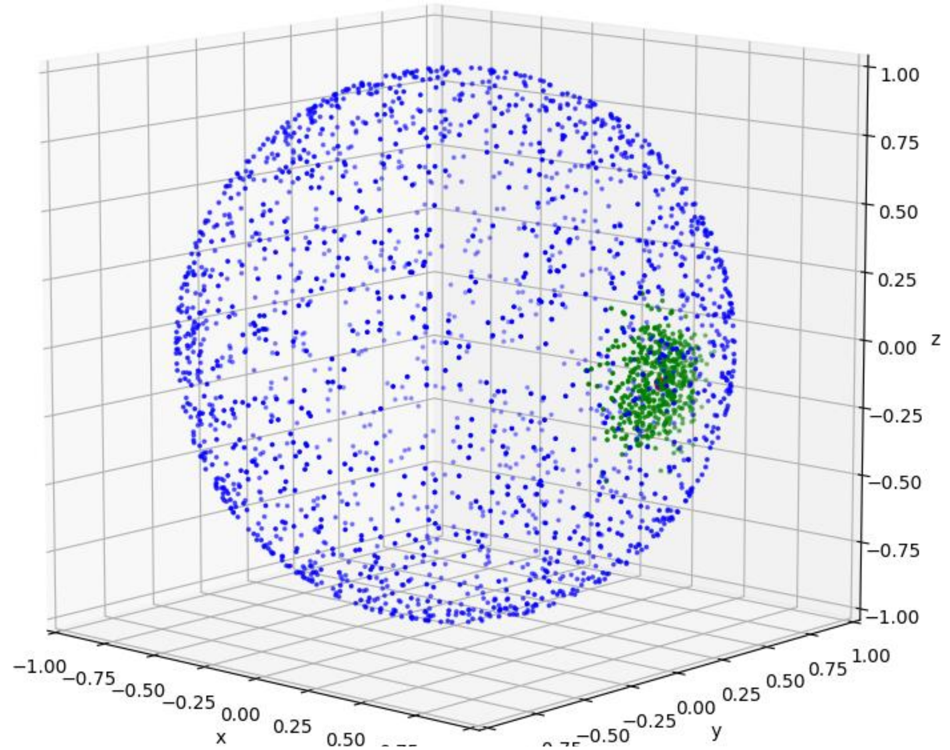
For i in range(nevents)

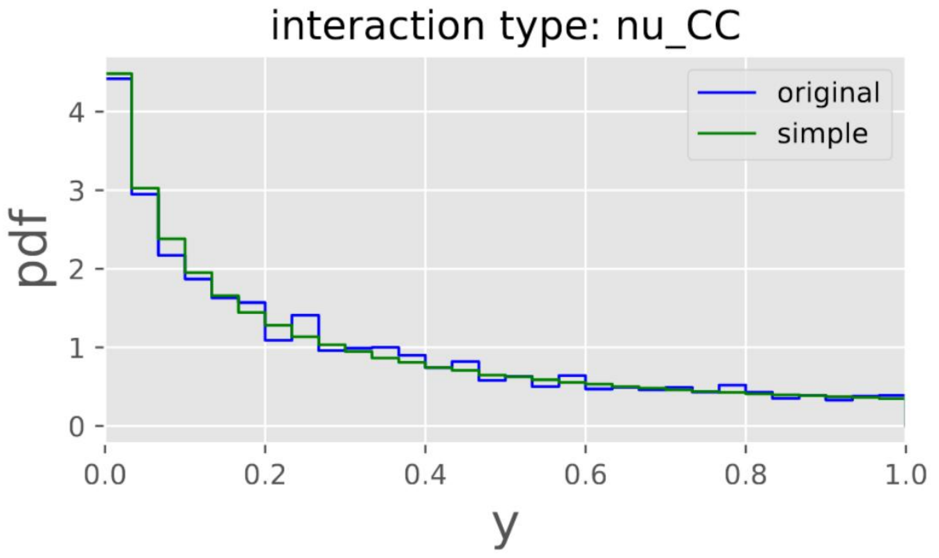
step 1
generate **neutrino energy from powerlaw** (some index, e.g. E^{-1})
(inverse CDF sampling)

generate **neutrino direction uniformly across sky**
(accept reject sampling)

step 2
calculate **generation weight** from **generation pdf** and **effective area** for the chosen channel

step 3
generate **inelasticity y** from an energy dependent function
(inverse CDF sampling)





The model

step 4
calculate hadronic and electromagnetic part of the shower / cascade

$E_{em} = (1-y)E_{nu}$ (only for CC) and **$E_{had} = yE_{nu}$ (for CC and NC)**

calculate deposited energy as **$E_{dep} = E_{em} + E_{had}$** ($E_{em} = 0$ for NC events)

step 5
generate **reconstructed quantities** from normal distribution around latent true random variables generated in the previous steps (with resolutions parameterized as function of E_{dep})

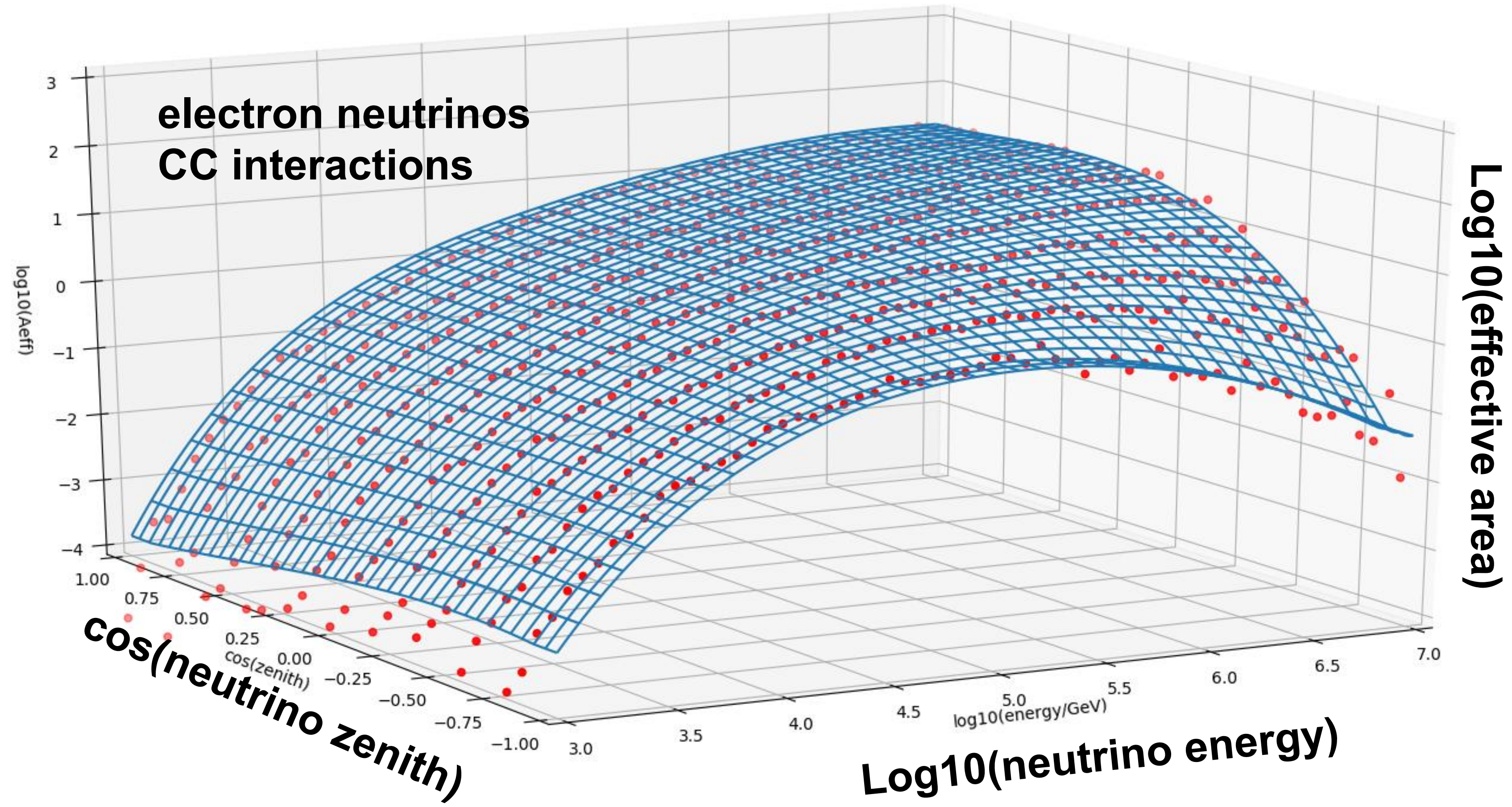
for energy: gaussian centered at E_{dep}

for direction: von Mises-Fisher
(symmetric gaussian on the sphere)

(additional explanations on blackboard and ipython notebook
understanding_weighted_simulation.ipynb)

The model uses a realistic IceCube effective area

we perform **one simulation for each component** i,j
with i in interaction types (NC, CC) and j in neutrino flavors (nue, numu) - but skip numu NC
re-combination of all channels via importance weights


electron neutrinos
CC interactions
log10(Aeff)
Log10(effective area)
cos(zenith)
cos(neutrino zenith)
log10(energy/GeV)
Log10(neutrino energy)