Preliminary results on solar neutrino measurement feasibility with CYGNO

S.Torelli - E.Baracchini - S.Piacentini

Expected rate

- Total cross section of neutrino on electron

$$\sigma_{\nu_e}(E_\nu) = \frac{G_F^2 m_e}{2\pi}\Bigg\{(g_V+g_A+2)^2\left[\frac{2E_\nu^2}{(m_e+2E_\nu)} - T'_{e,Thr}\right] + \\ -(g_V-g_A)^2\frac{E_\nu}{3}\left[\left(1-\frac{2E_\nu}{m_e+2E_\nu}\right)^3 - \left(1-\frac{T'_{e,Thr}}{E_\nu}\right)^3\right] + \\ -(g_V-g_A)(g_V+g_A+2)\frac{m_e}{2}\left[\frac{4E_\nu^2}{(m_e+2E_\nu)^2} - \frac{T'^{\,2}_{e,Thr}}{E_\nu^2}\right]\Bigg\}$$

- Expected rate calculated on 60:40 He/CF_4 gas mixture @ 1atm 25°C

- Oscillation taken into account

$$P(\nu_e \rightarrow \nu_\mu) = P_{e\mu} = \frac{1}{2}\sin^2(2\theta_{12})$$

- pp flux tabulated taken from Bahcall

q [MeV]	P(q)	q [MeV]	P(q)	q [MeV]
0.00504	0.0035	0.11089	1.2477	0.21675
0.01008	0.0138	0.11593	1.3417	0.22179
0.01512	0.0307	0.12097	1.4370	0.22683
0.02016	0.0538	0.12601	1.5335	0.23187
0.02520	0.0830	0.13106	1.6310	0.23691
0.03024	0.1179	0.13610	1.7291	0.24195
0.03528	0.1582	0.14114	1.8278	0.24699
0.04032	0.2038	0.14618	1.9267	0.25203
0.04537	0.2543	0.15122	2.0258	0.25707

- Resulted rate:

$$R = N_e \sum_i \varphi(E_i)(P_{ee}\sigma_{\nu_e}(E_{\nu,i}) + P_{e\mu}\sigma_{\nu_\mu}(E_{\nu,i}))\Delta E$$

i-th flux component

$$R = 2.9 \cdot 10^{-8}\,\frac{events}{s \cdot m^3} = 0.9\,\frac{events}{y \cdot m^3}$$

Threshold on e^- E at $20\ keV$

Bayesian framework

- Sensitivity studies performed with the Bayesian framework:

Posterior probability — Likelihood — Prior probabilities — Normalization factor

$$p(\mu_s, \mu_b | D) = \frac{p(D | \mu_s, \mu_b) \cdot \pi(\mu_s) \cdot \pi(\mu_b)}{p(D)}$$

- Prior probability: a priori knowledge of the signal and background probability distribution, it will be assumed possoanian for the background and flat for the signal

- Likelihood: probability of observing the data given μ_s and μ_b, calculated as the product of the probability of $n_{i,j}$ events in the i, j bin with expected value $\lambda_{i,j}$

$$L = \prod_{i,j} \frac{\lambda_{i,j}^{n_{i,j}} e^{-\lambda_{i,j}}}{n_{i,j}!}$$

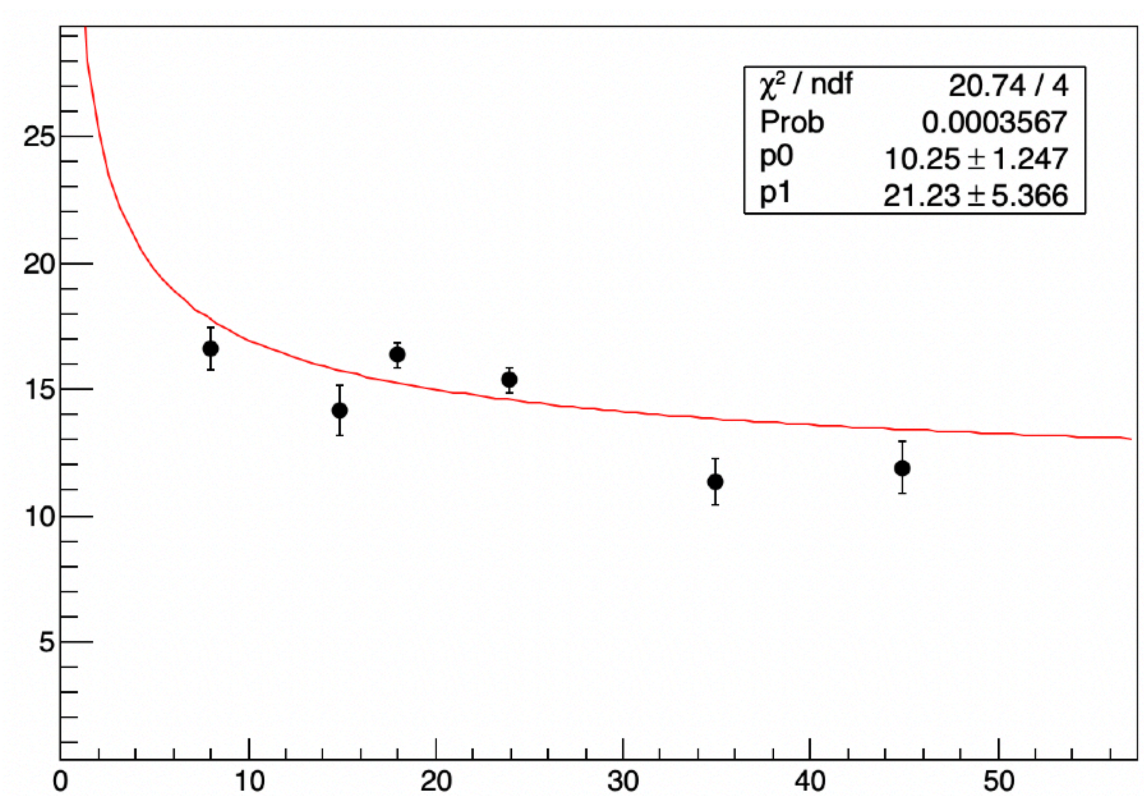
- Normalization factor: difficult to estimate a priori, it will be calculated by integrating the numerator distribution with a Markov-Chain Montecarlo based algorithm (posterior must be nomalized to 1)

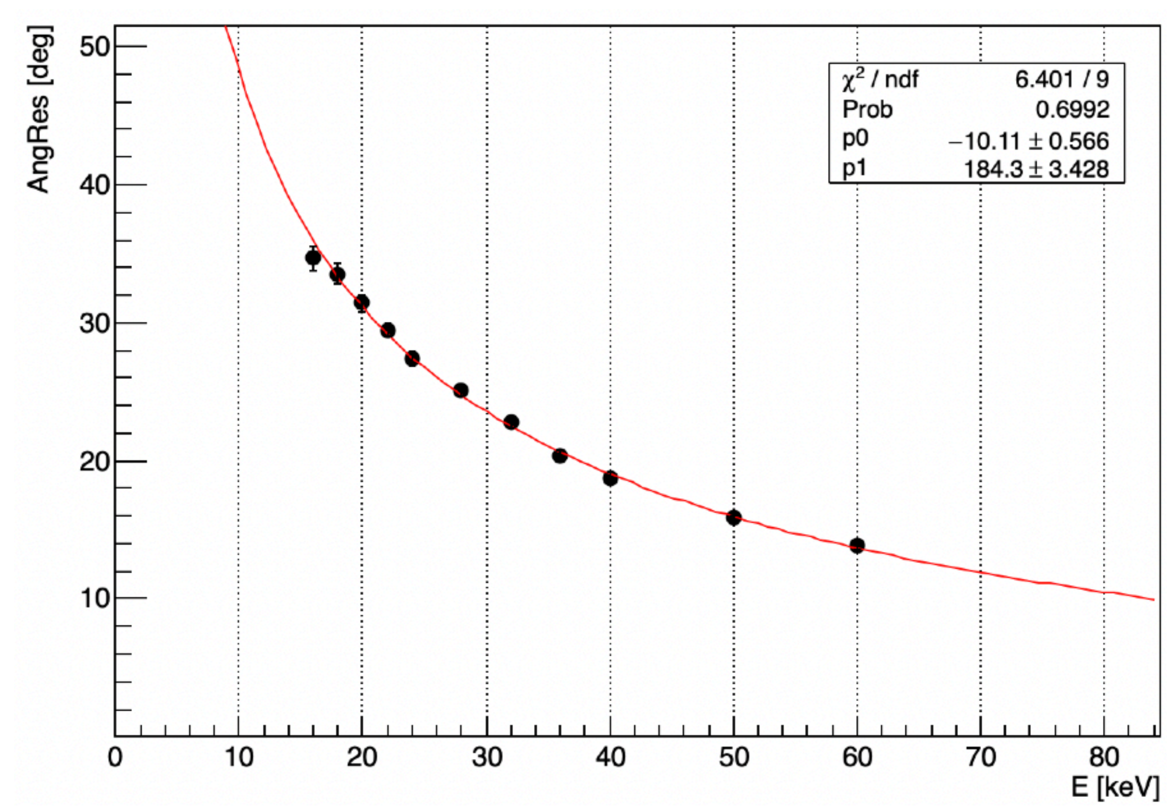
Analysis tool

- Work done with Stefano Piacentini in Roma Sapienza
- Framework moved from python to C++ for fastness and future scalability of the code
- Tool used is the BAT: Bayesian Analysis Toolkit
- Tool developed to solve statistical problems encountered in Bayesian inference
- It contains inside the tools for defining models, marginalize distributions, compare different hypothesis models…
- For this case of study BAT requires the models (S,B), the likelihood function, the priors and the expected values of each bins

Resolutions used and assumptions

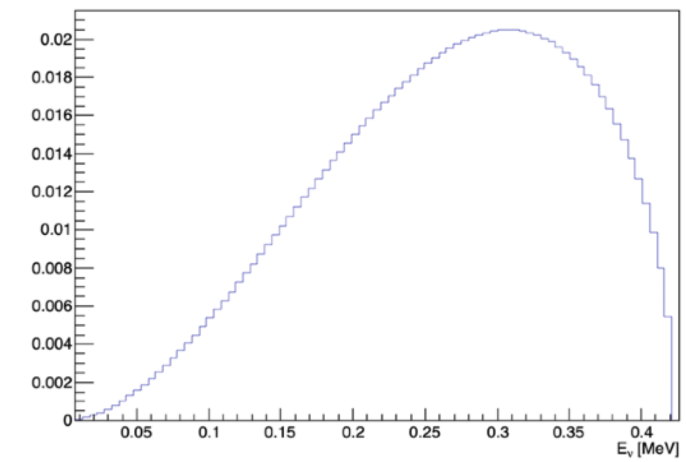
- For the templates generation ($\lambda_{i,j}$ information) the energy resolutions from the data and the angular resolution from the MC have been used



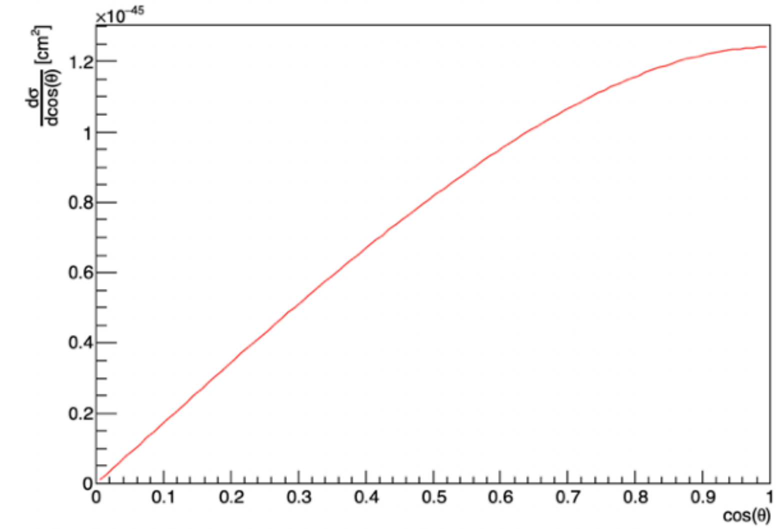


- Assumptions:
  - Same resolution in both theta (on the GEM plane) and phi (respect to the perpendicular to the GEM plane)
  - Isotropic gamma background

Signal templates



- Extraction of a random neutrino energy according to the pp solar flux



- Extraction of a random $\cos\theta$ value according to the differential cross section for extracted neutrino energy

- Calculation of the electron energy

$$T'_e(\theta) = \frac{2E_\nu^2 m_e cos^2(\theta)}{(E_\nu + m_e)^2 - E_\nu^2 cos^2(\theta)}$$

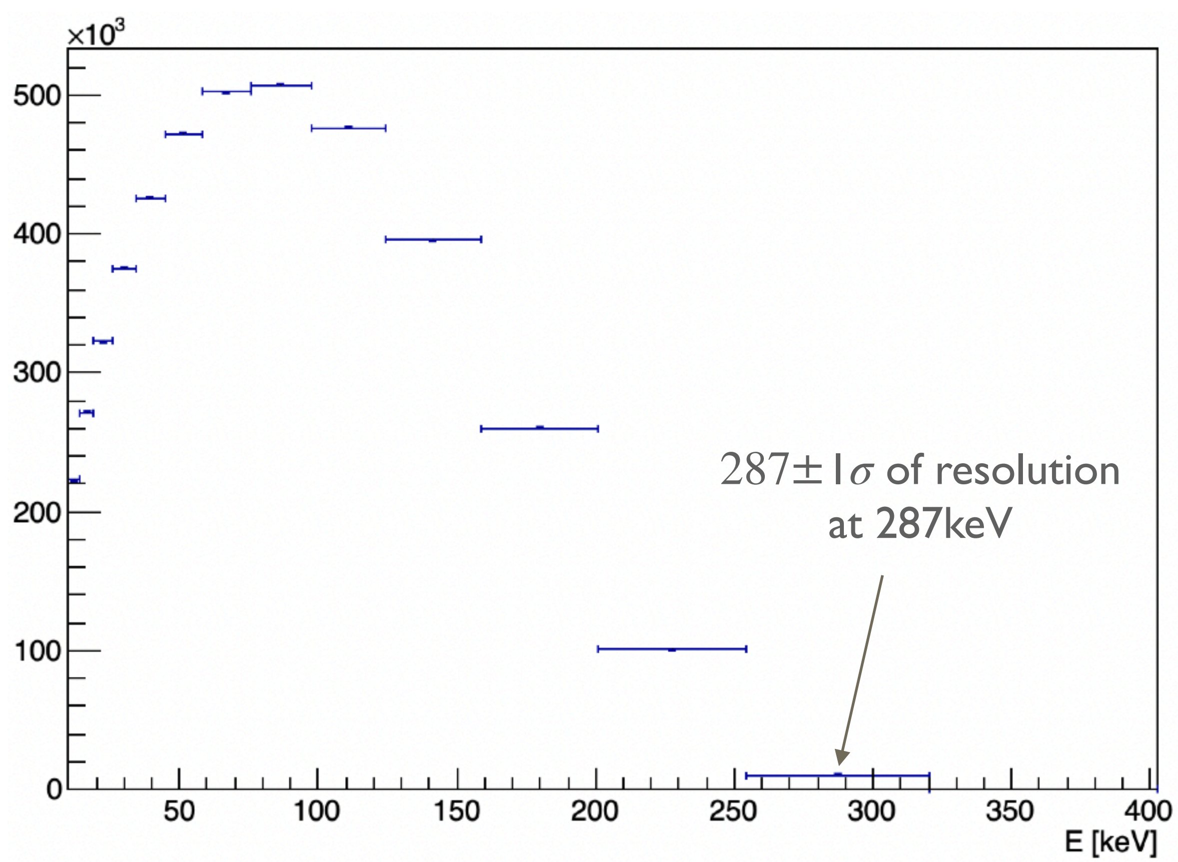
- Smearing of the energy and angle according to the resolutions

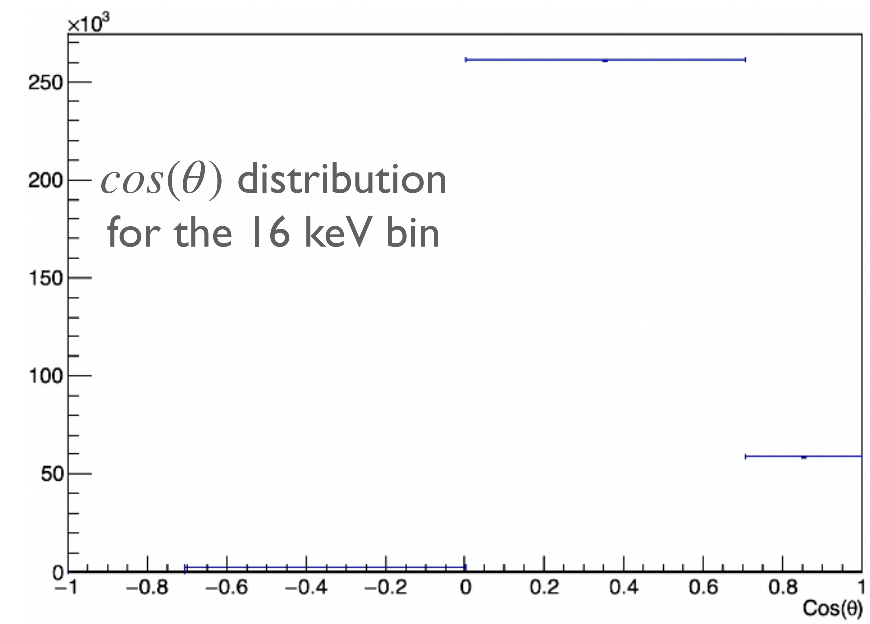
$$\frac{\sigma_E}{E} = 10.25 + \frac{21.23}{\sqrt{E}}$$

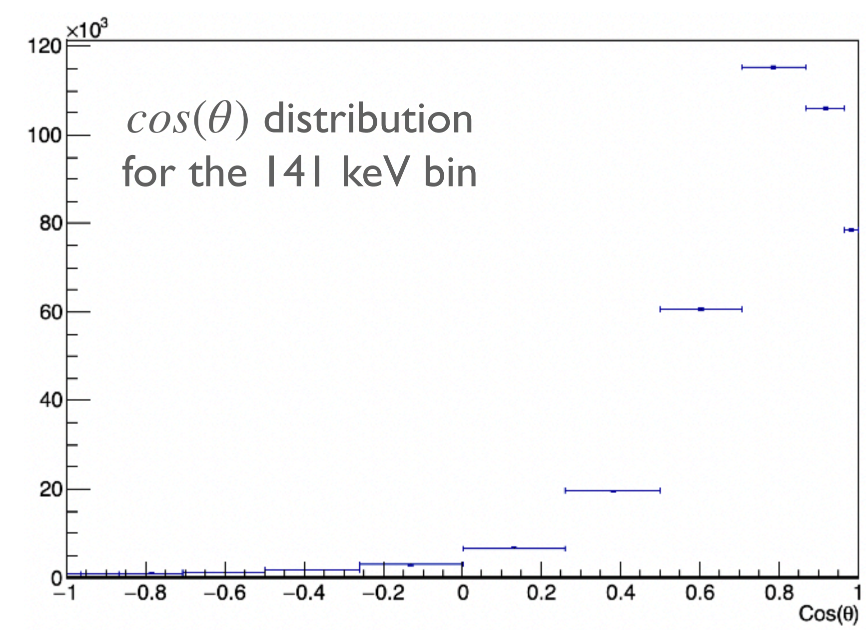
$$\sigma_\theta = -10.11 + \frac{184.3}{\sqrt{E}}$$

Signal templates

- Histograms construction:
  - Energy histogram with bin size $\pm 1\sigma$ of resolution
  - For each energy bin a $\cos(\theta)$ histogram with bin size $\pm 1\sigma$ of resolution

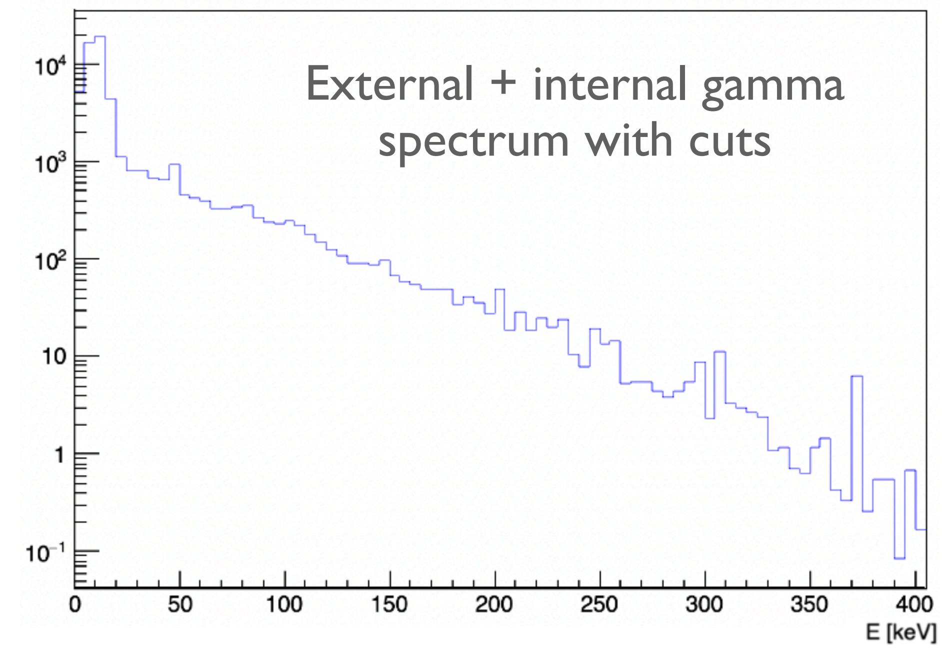


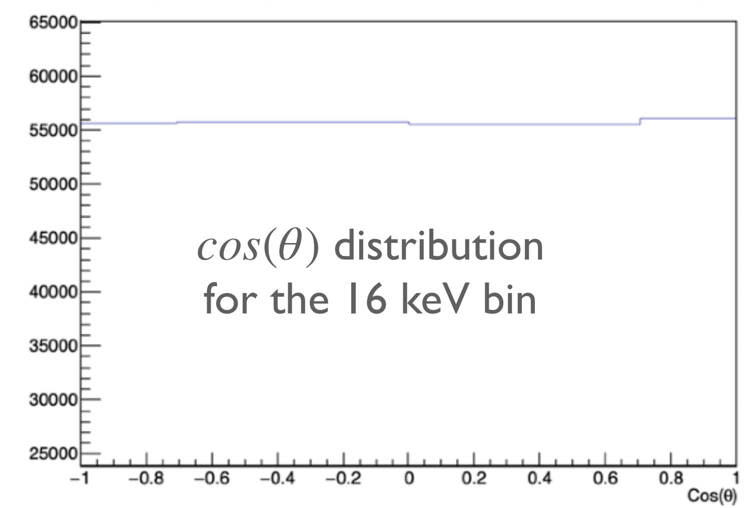


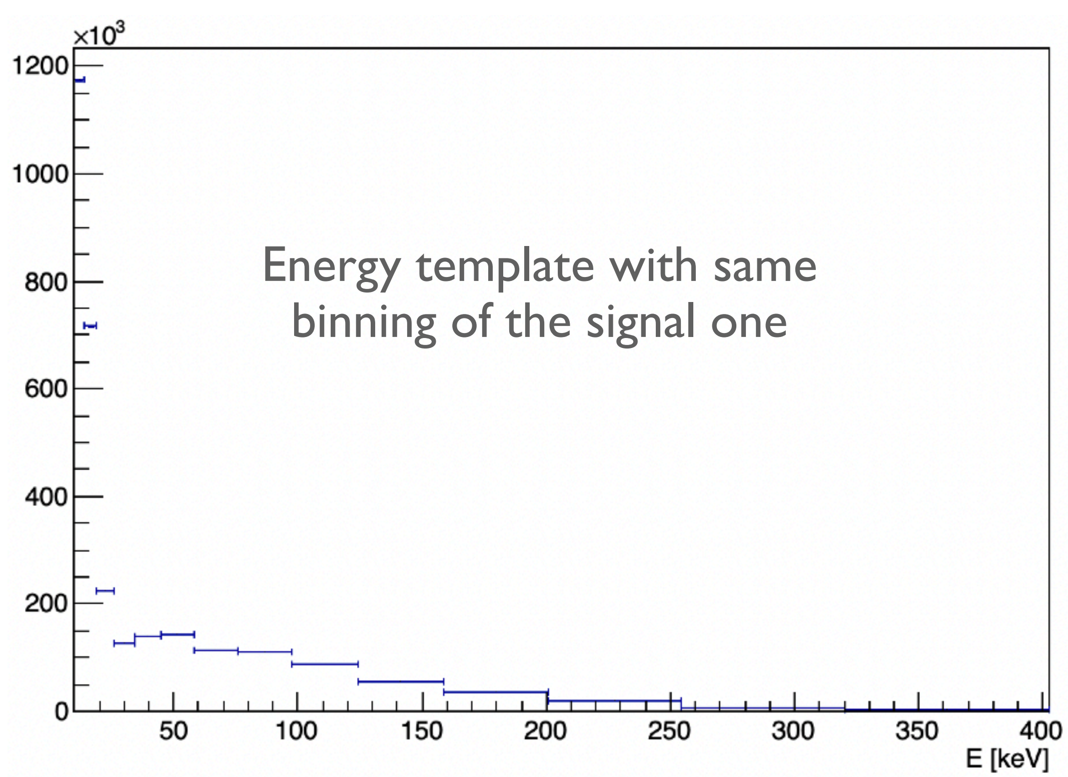


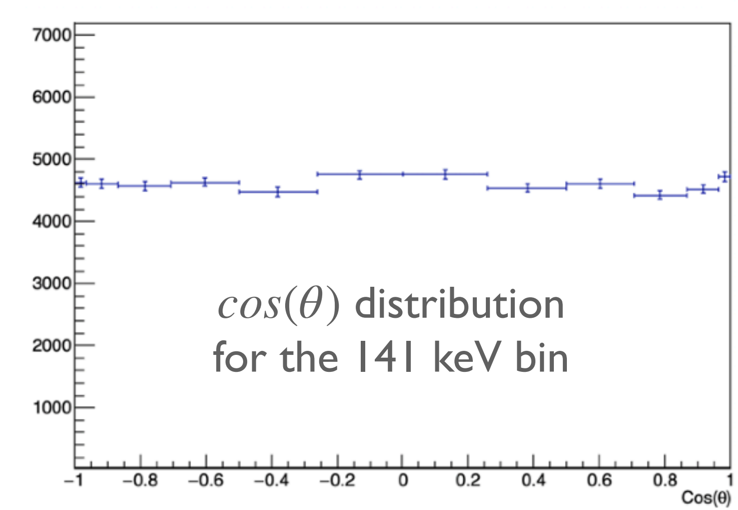
Background templates

- For the background, the templates were generated in the same way
- I preliminary used the LIME bkg spectra calculated by Flaminia, summing the external and internal flux of gamma with cuts
- Extraction of a random energy according to the spectrum, and theta from flat distribution → smearing both



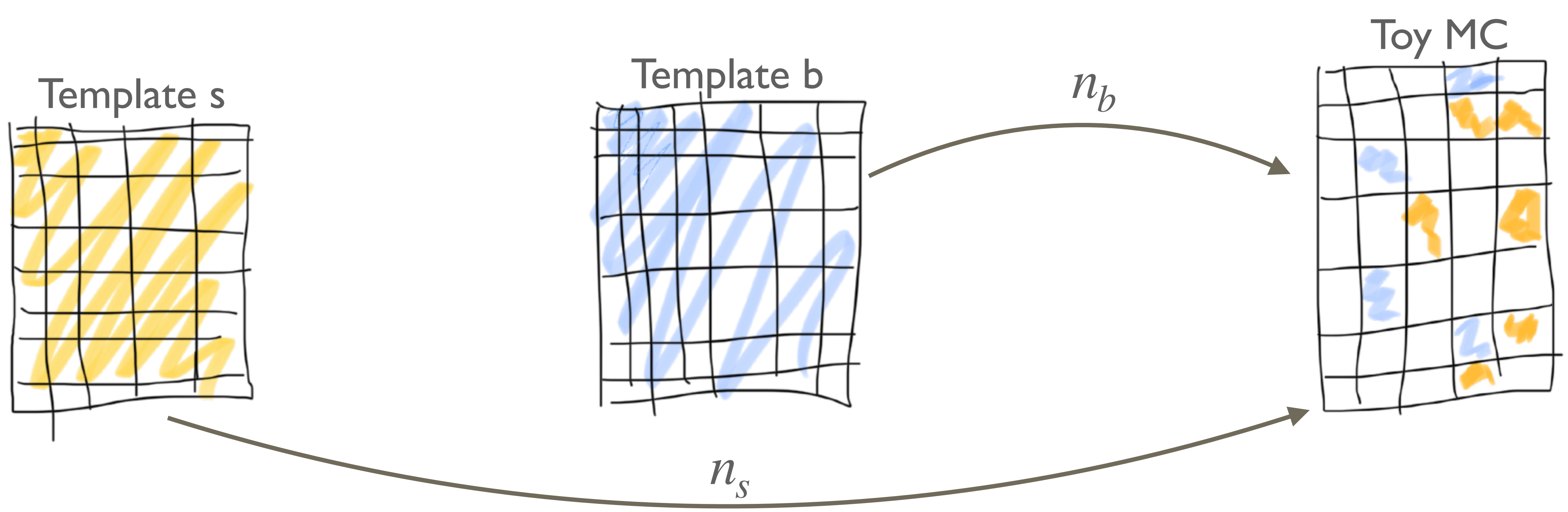






Toy-MC

- Toy-MC generated by:
  - Choosing an hypothesis of signal and background $\bar{N}_s, \bar{N}_b$
  - Extracting the actual values from a poissonian distribution n_s, n_b
  - Injecting n_s, n_b events, respectively from the signal and background templates, into an E-cos(θ) histogram



N.B. all the histograms have the same binning

- 50 toy MC for every combination of $\bar{N}_s$ and $\bar{N}_b$ have been generated:
  - $\bar{N}_b$ = 10,100,500,1000,10000
  - $\bar{N}_s$ = 5, 10, 20, 40, 60, 100, 200, 400, 600, 1000

How the code works

- Define the environment with the distributions, the data sample, and the parameters of the Markov-chain

```cpp
const std::string datafile_name(argv[1]);
const std::string  bkgfile_name(argv[2]);
const std::string  sigfile_name(argv[3]);
//const int bkg_amount = atoi(argv[4]);
const int run_index  = atoi(argv[4]);//5]);
const std::string addinfo(argv[5]);


int NIter = 30000;
int Nch   = 10;
```

```cpp
double sigMOD::LogLikelihood(const std::vector<double>& pars) {

    double LL = 0.;

    for(int i = 0; i<nbins; i++) {
        double lambda_i = pars[0] * (bkg[i] / bkgNorm) + pars[1] * (sig[i]/sigNorm);
        LL += BCMath::LogPoisson(int(data[i+2]), lambda_i);
    }
```

- Define the signal+background model with the log-likelihood and the log-priors

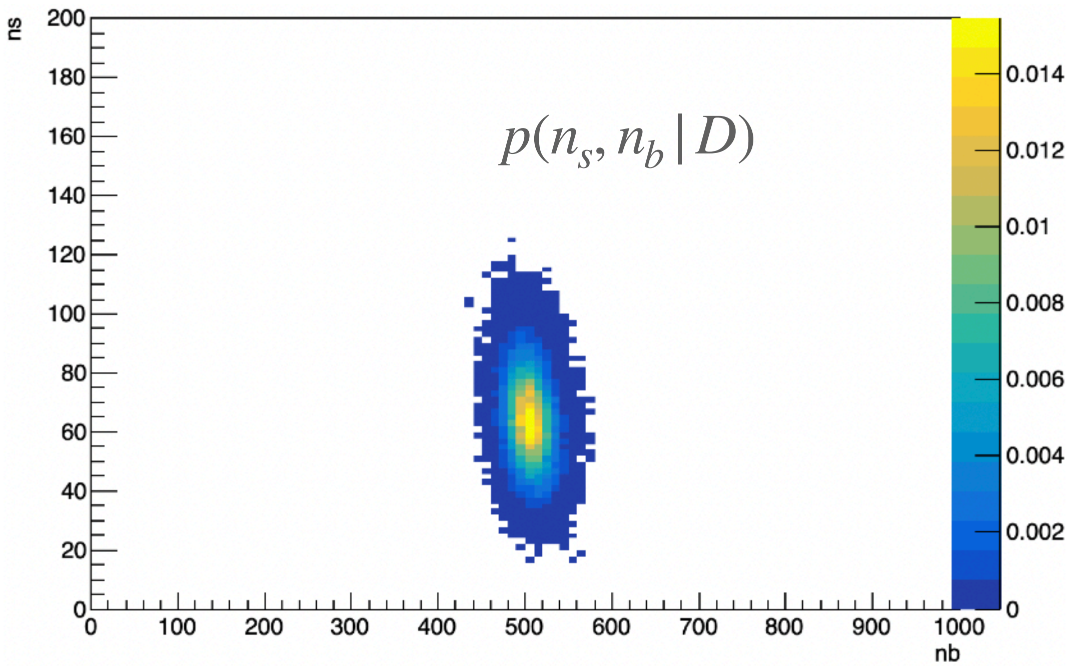
```cpp
double sigMOD::LogAPrioriProbability(const std::vector<double>& pars) {

    double LL = 0.;

    // prior on nb
    LL += BCMath::LogPoisson(int(pars[0]), bkg_prior_lambda);
    //LL += BCMath::LogGaus(int(pars[0]), bkg_prior_lambda, 10 * sqrt(bkg_prior_lambda))

    // uniform prior on ns
    if(pars[1]<0 || pars[1]>nsmax) {
        LL += log(0.0);
    } else {
        LL += log(1.0/nsmax);
    }
```

- Fit the likelihood, sample the posterior with the MCMC Metropolis-Hastings algorithm and get the distributions for signal and background

```cpp
m.SetMarginalizationMethod(BCIntegrate::kMargMetropolis);
m.SetPrecision(BCEngineMCMC::kMedium);
// run MCMC, marginalizing posterior
m.MarginalizeAll();//BCIntegrate::kMargMetropolis);
```

Output of the code

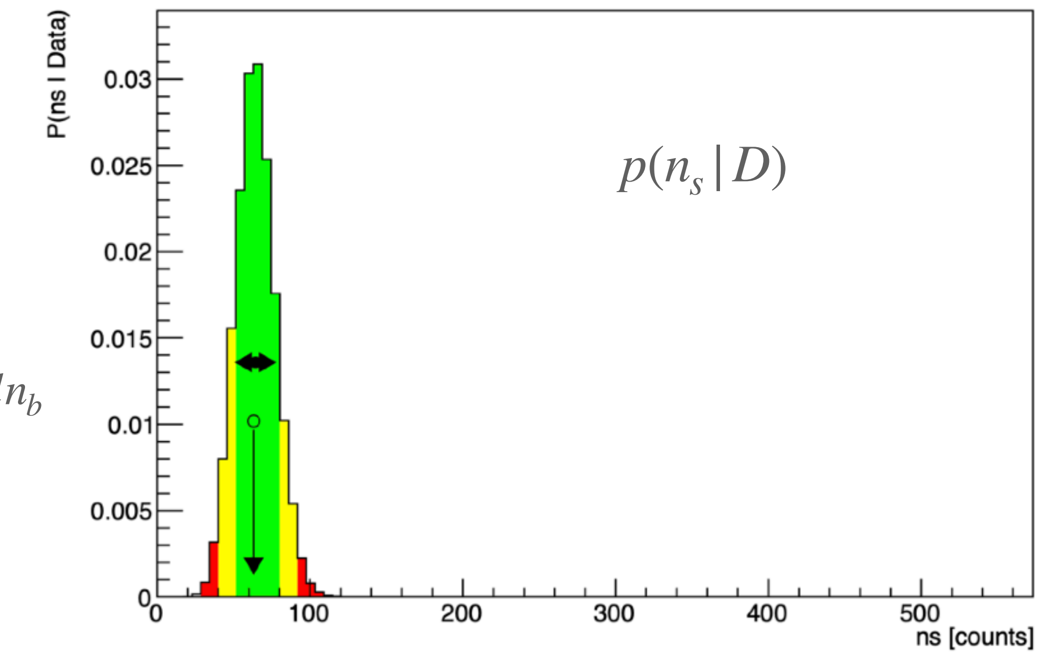
e.g. 60 events of signal over 500 events of background



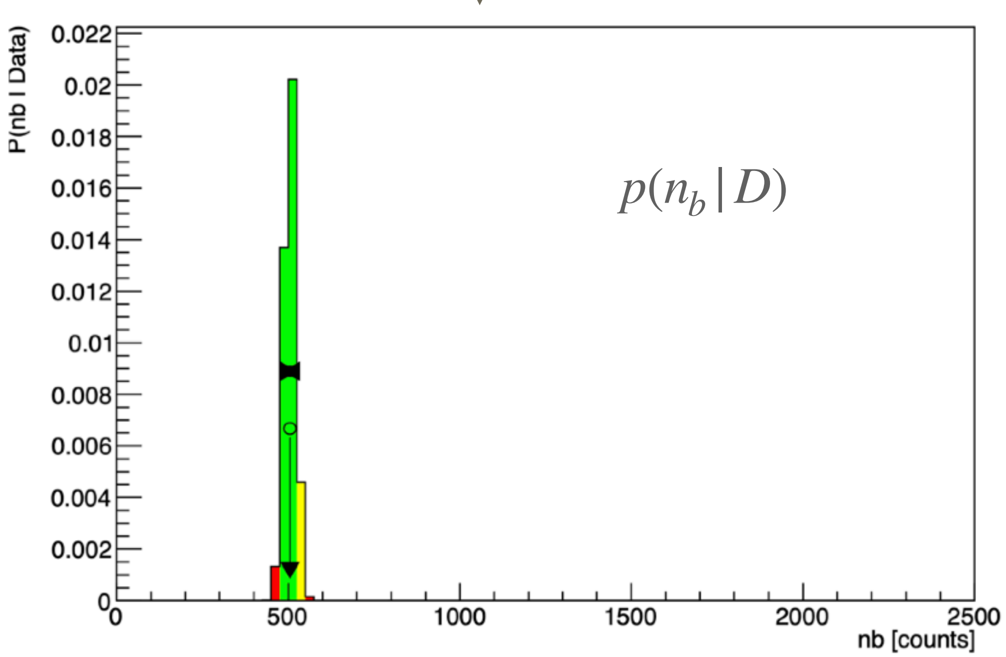
Marginalization

a.k.a. projection

a.k.a. $\int_{\Omega} p(n_s, n_b | D) dn_b$



Marginalization

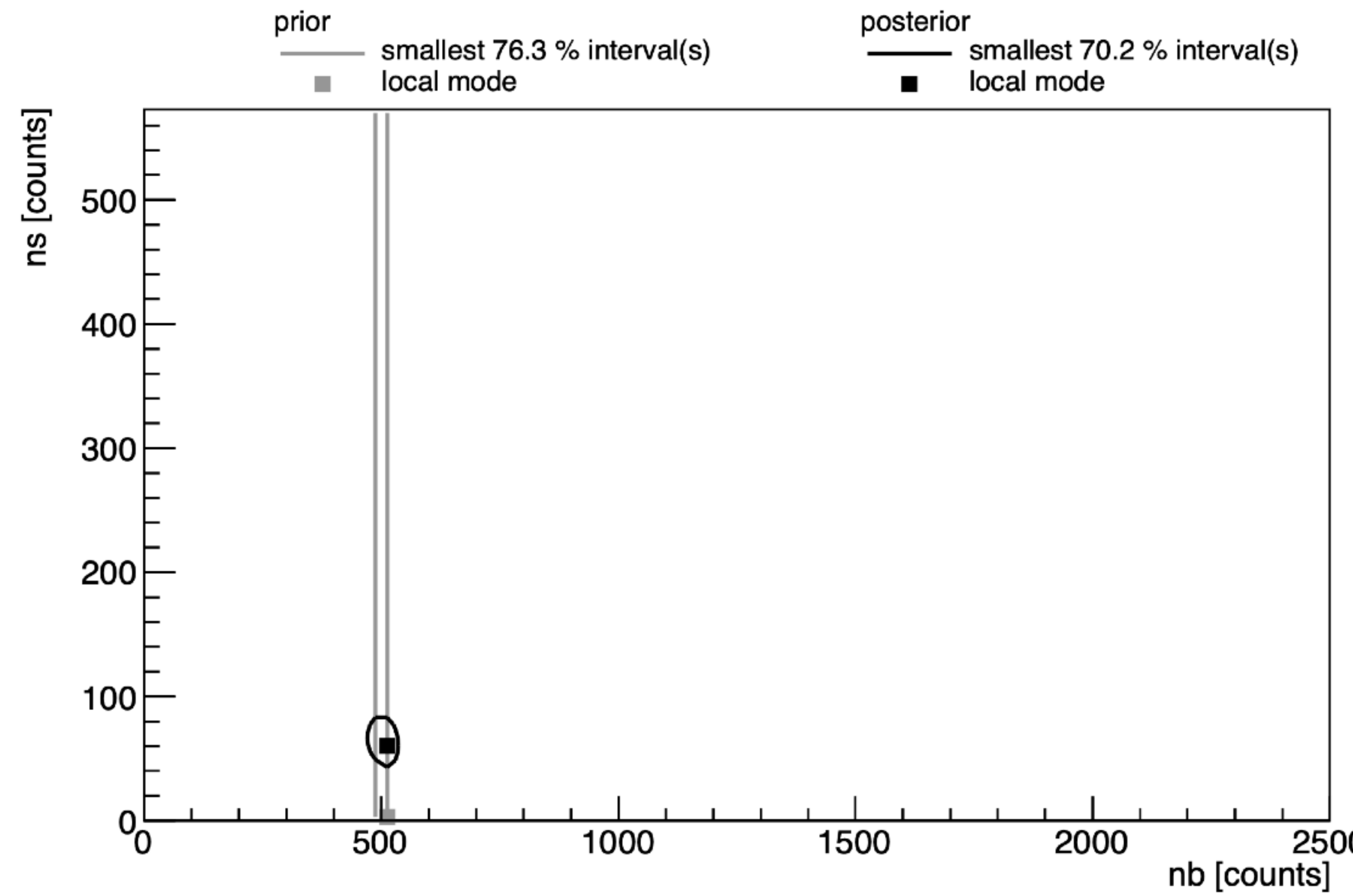


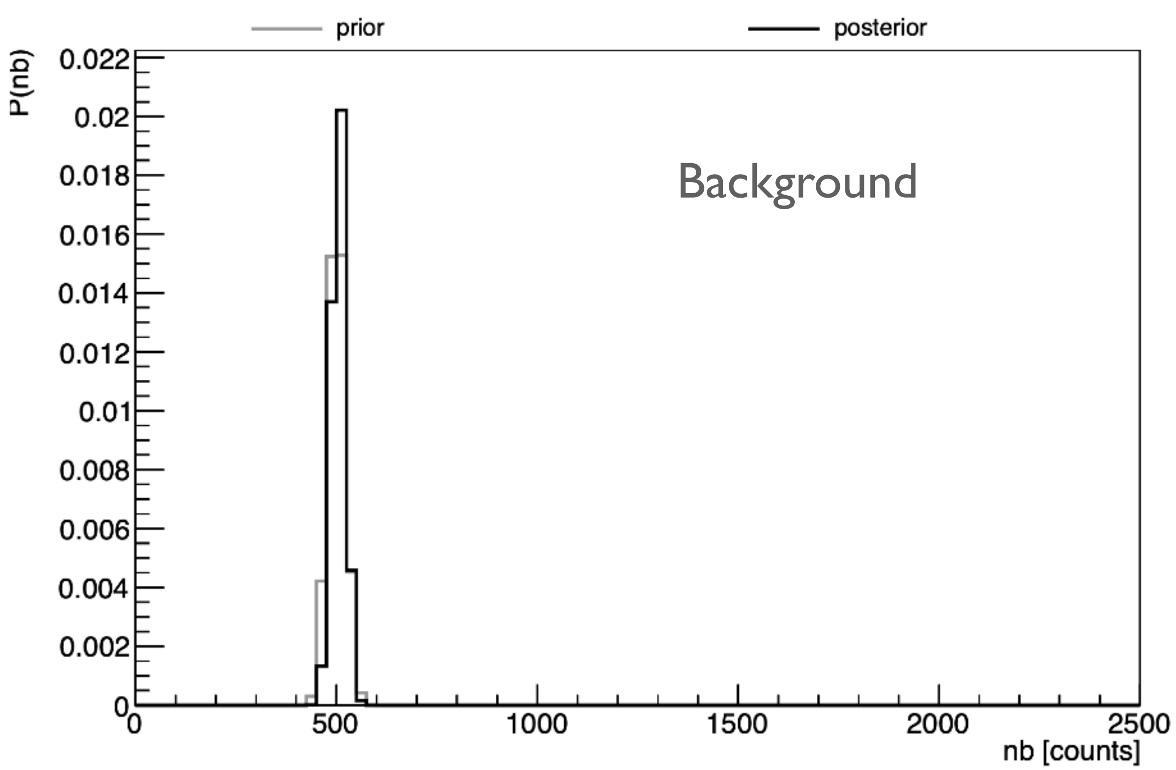

```
Parameter "nb" :
Mean +- sqrt(Variance):          +505.29 +- 16.59
Median +- central 68% interval: +506.13 + 16.813 - 21.903
```

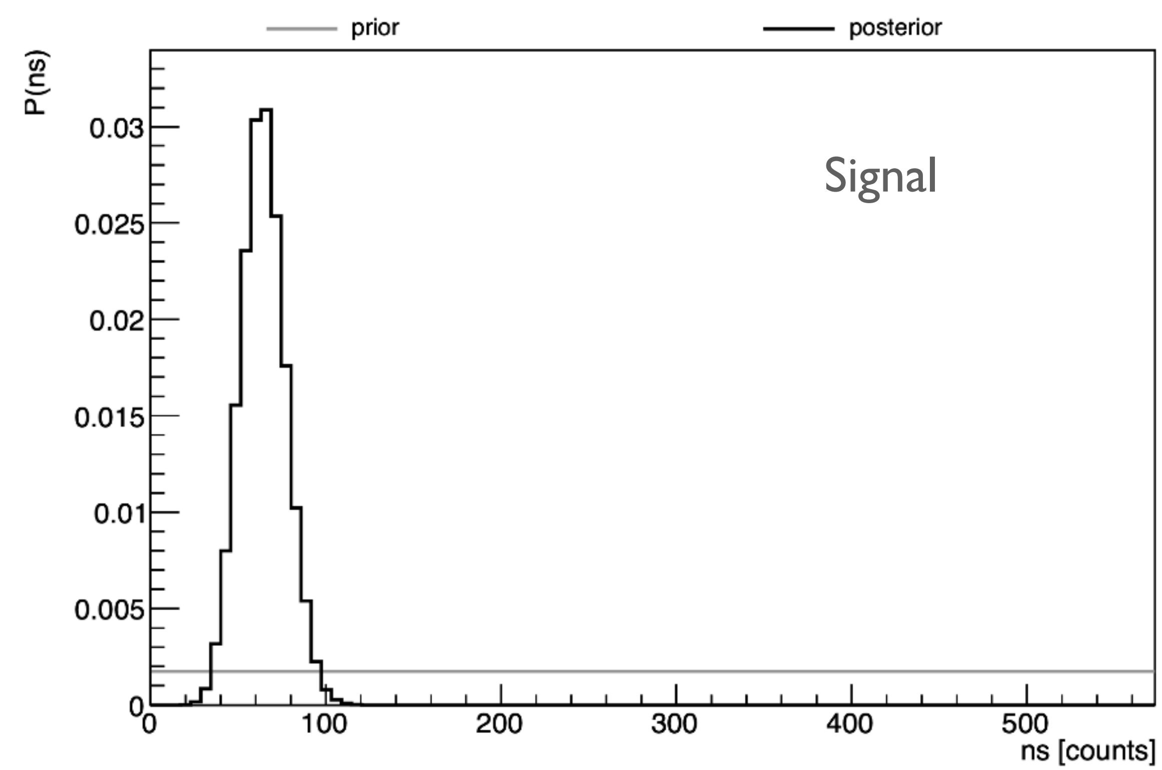
```
Parameter "ns" :
Mean +- sqrt(Variance):          +64.362 +- 12.766
Median +- central 68% interval: +64.068 + 13.263 - 12.459
```

Output of the code

Comparison of
prior and posterior

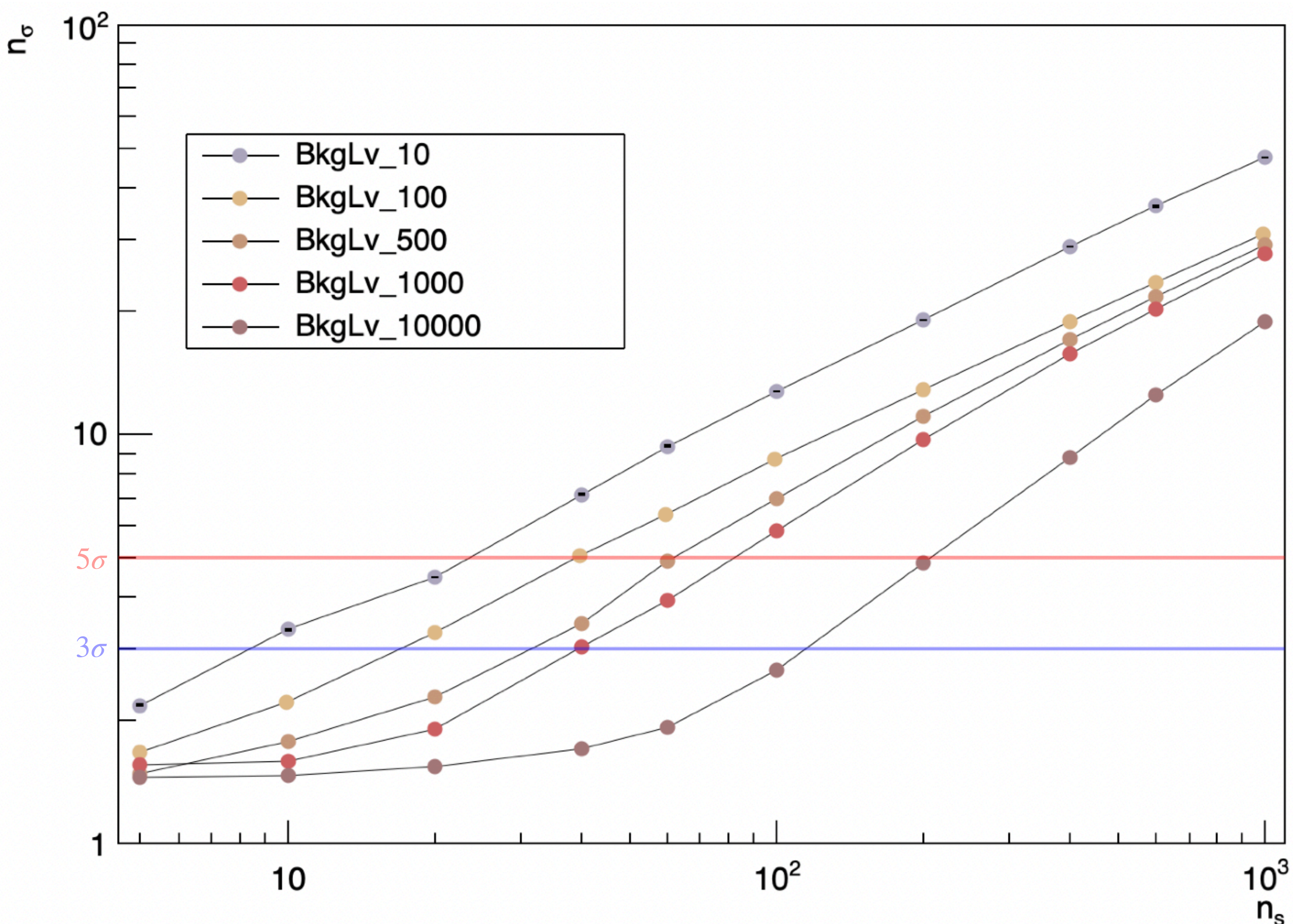


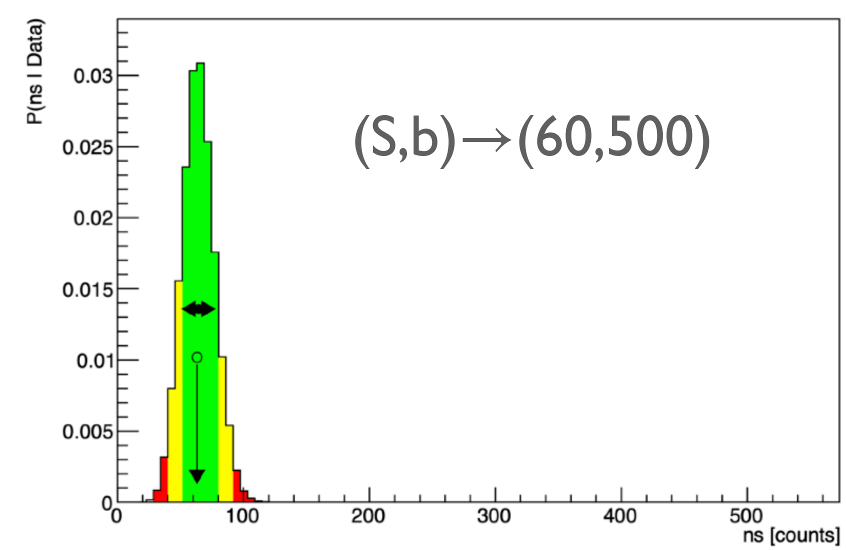




Nsigma sensitivity

- Plot of $\frac{\bar{n}_s - 0}{\sigma}$ as a function of n_s under different background hypotheses
- Each point is the mean of the 50 MC generated and analyzed

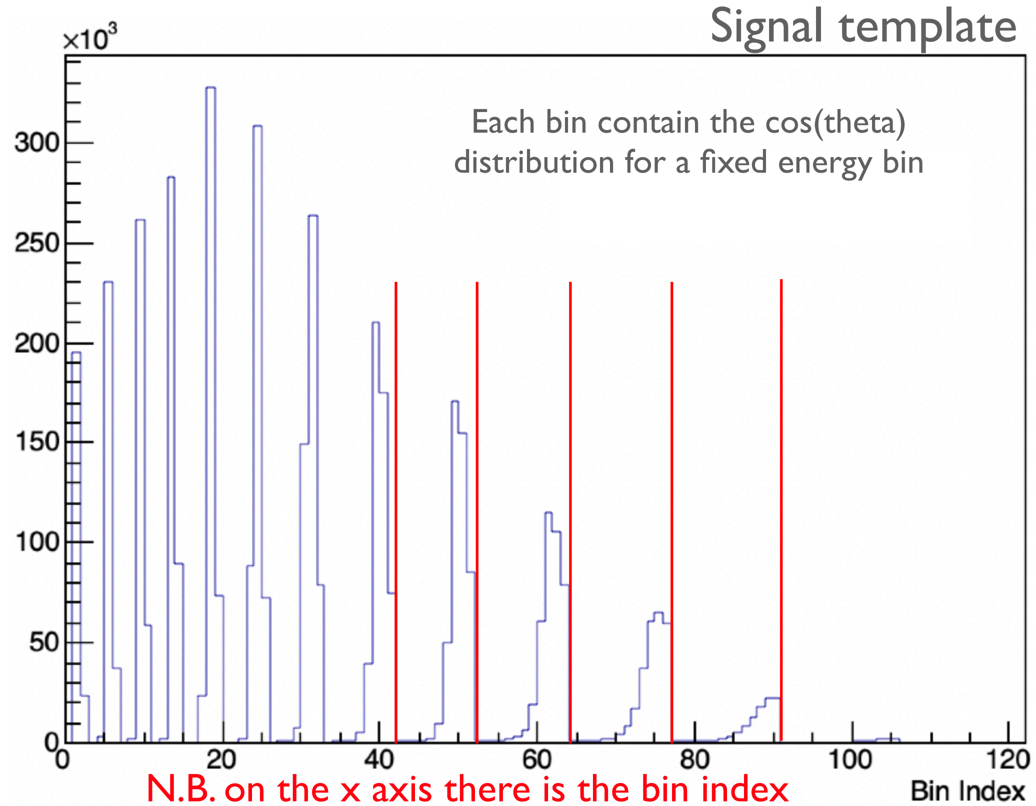


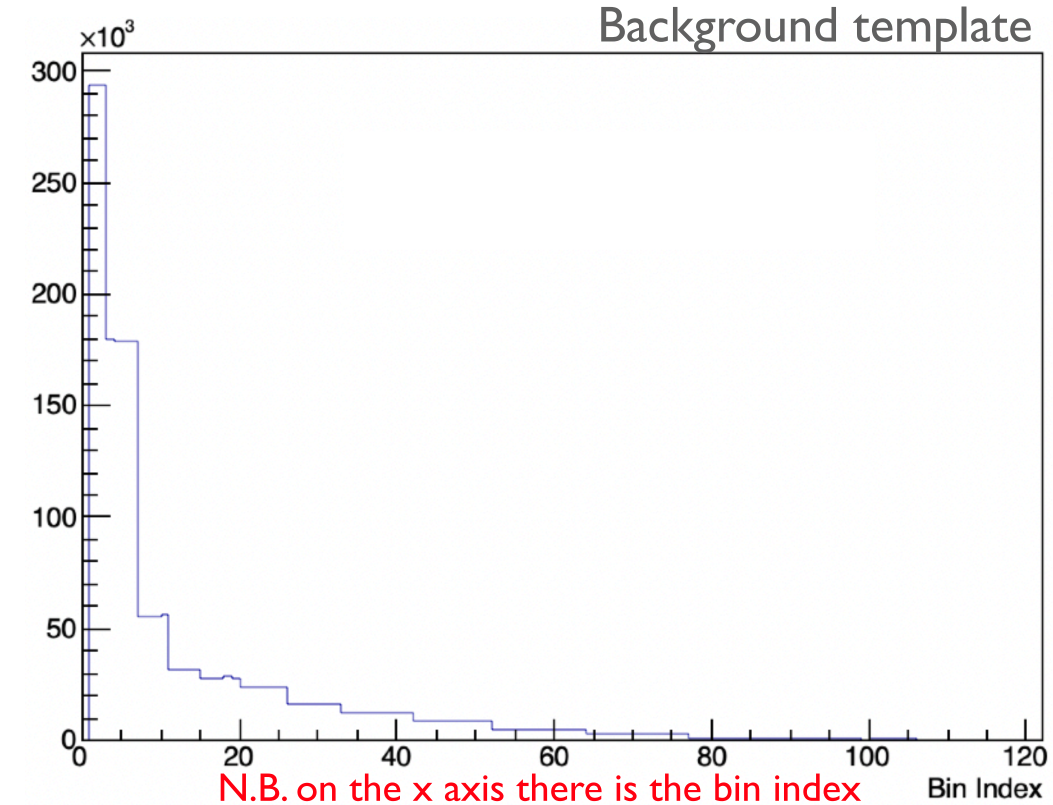



```
Parameter "ns" :
Mean +- sqrt(Variance):          +64.362 +- 12.766
Median +- central 68% interval: +64.068 + 13.263 - 12.459
```

$$n_\sigma = 64{,}3/12{,}7 = 5{,}1$$

Can we trust it?





Following plots obtained filling two histogram with the same bkg, and adding signal to one → (B), (S+B)