BLike

stimare il segnale con pochi fotoni

mario
astrosiesta
25 Giugno 2015

outline:

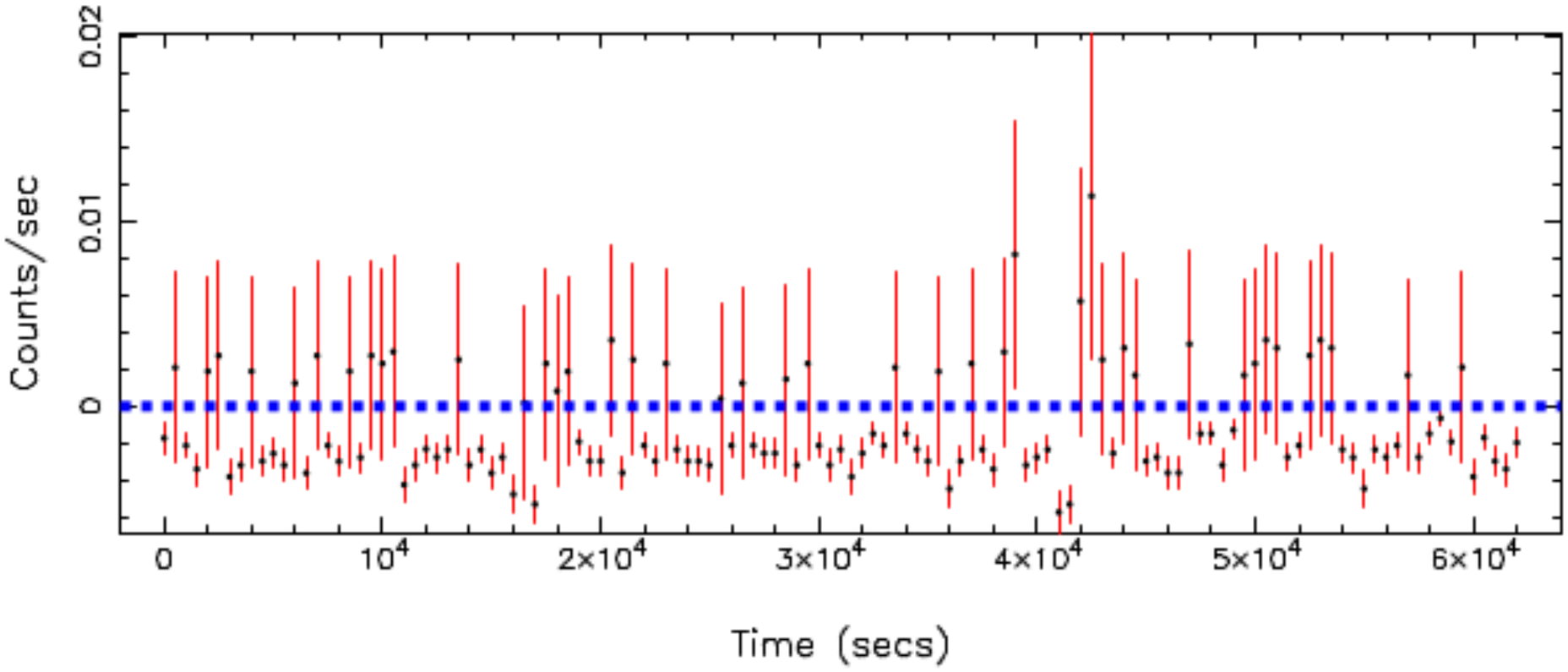
- il problema
- una soluzione
- la libreria
- le applicazioni

outline:

- il problema
- una soluzione
- la libreria
- le applicazioni

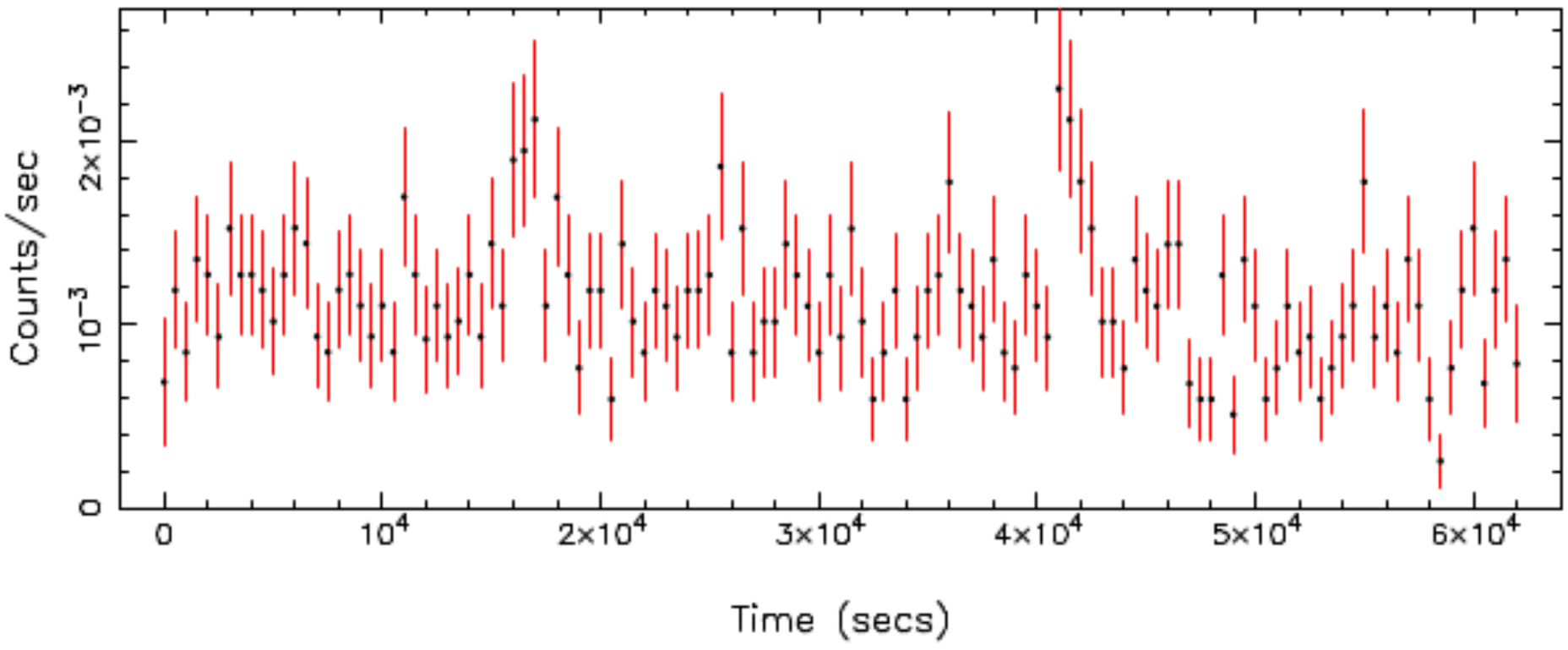
Background subtracted time series

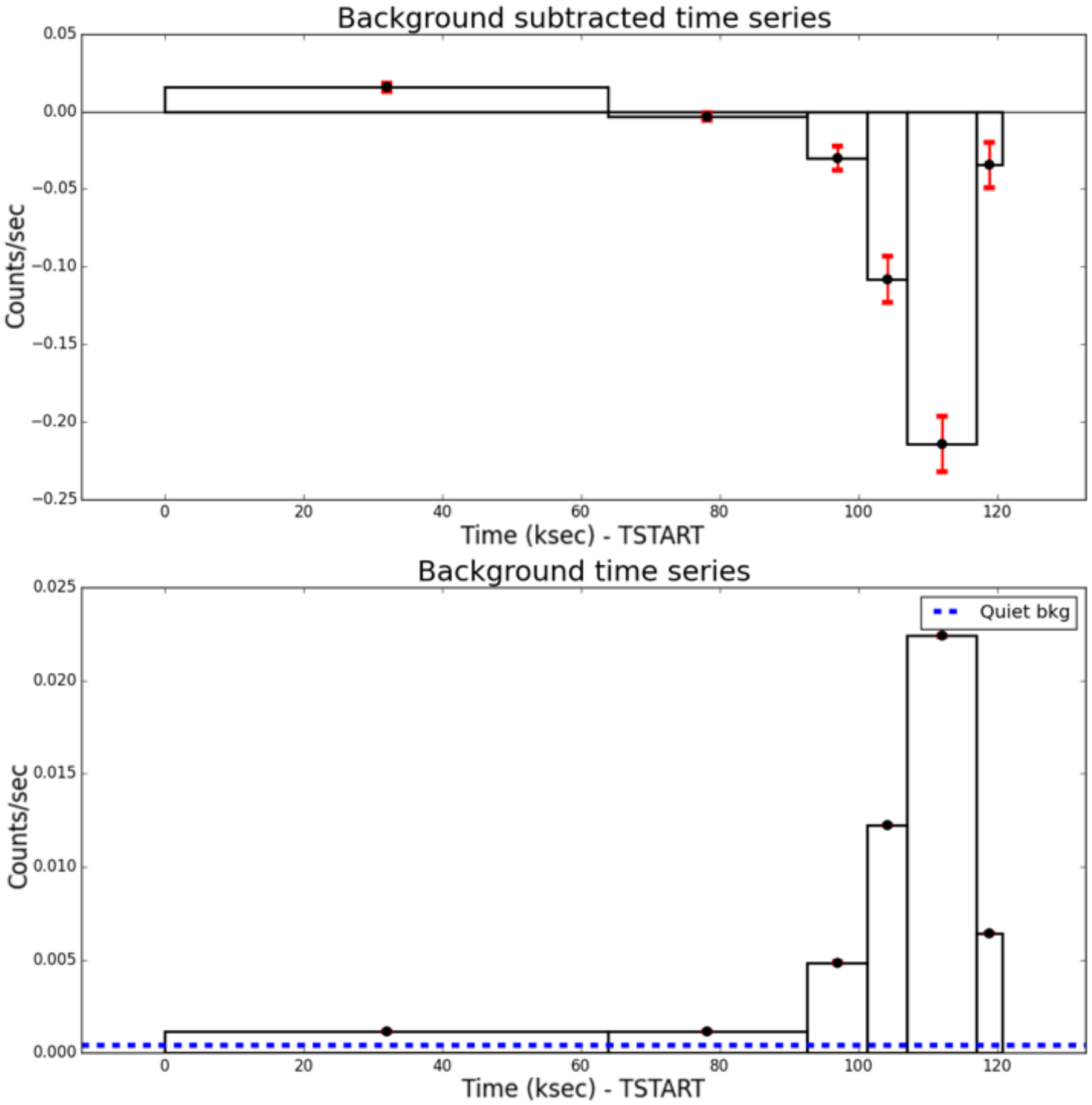
Mean Rate=-9.6011×10^{-4}



Background time series

Mean Rate= 1.12835×10^{-3}



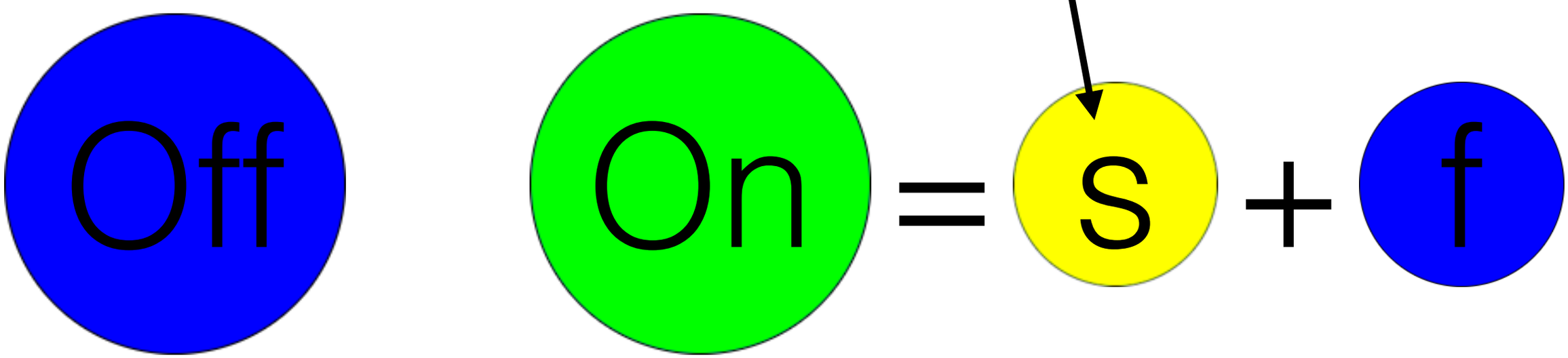

Background subtracted time series
Counts/sec
Time (ksec) - TSTART
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
0
20
40
60
80
100
120
Background time series
Quiet bkg
0.025
0.020
0.015
0.010
0.005
0.000

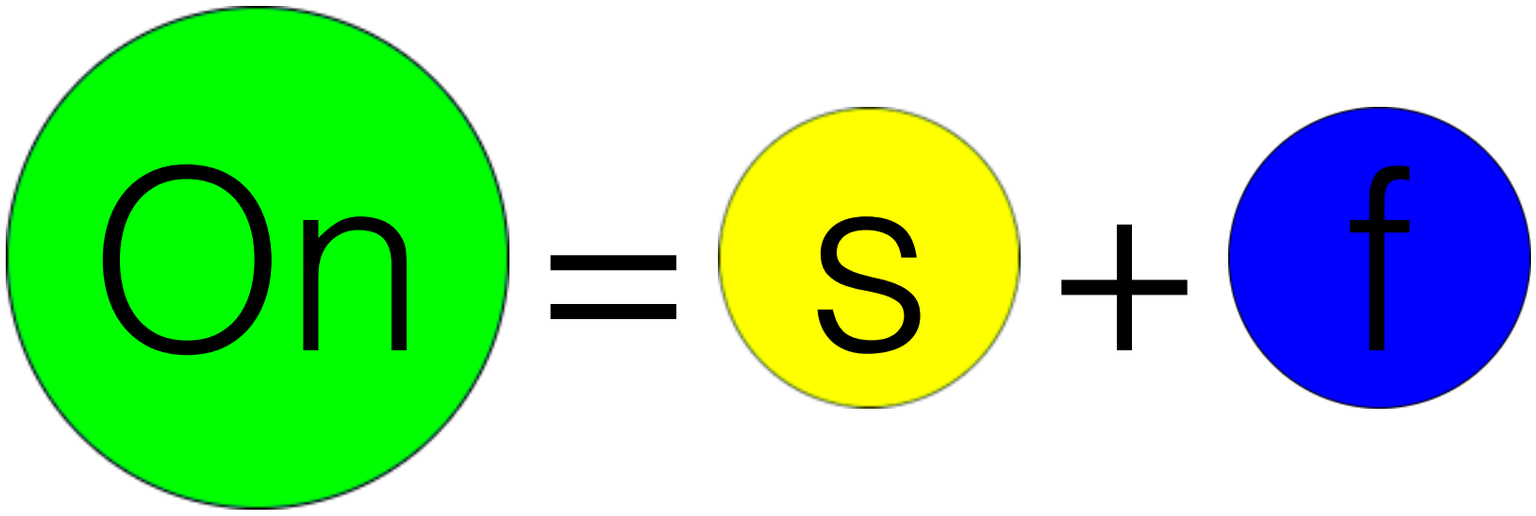
il problema

- la maggior parte delle sorgenti X ha pochi fotoni
- selezioni in tempo ed energia peggiorano le cose
- l’approssimazione Gaussiana non funziona piu’
- serve un metodo robusto di stima del segnale
- questo stimatore deve permettere analisi statistiche
- possiamo ridurci a considerare il problema On-Off

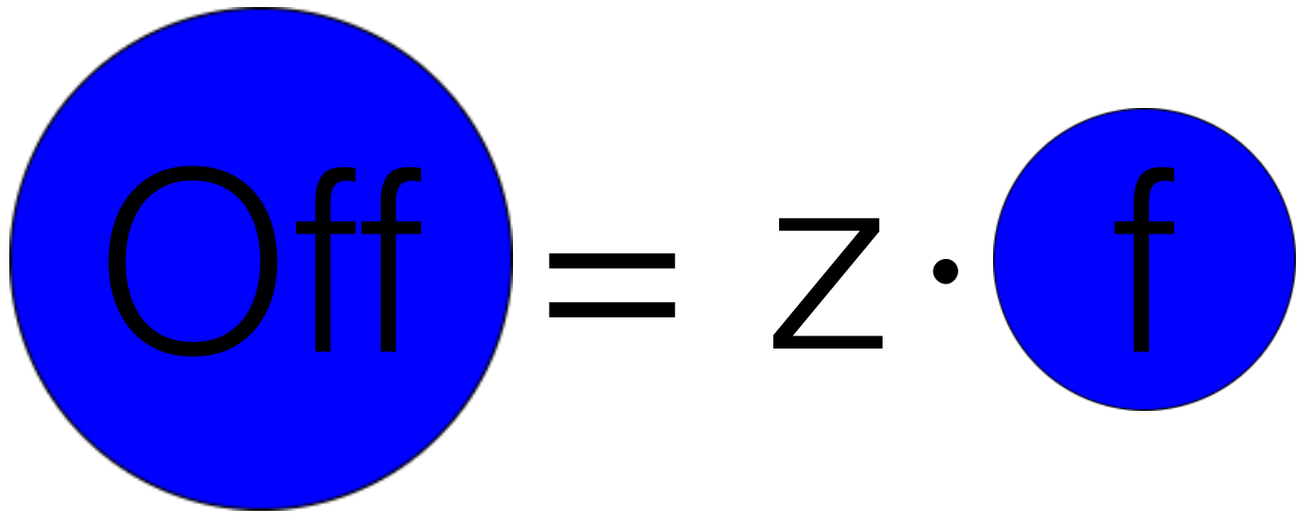
il problema On-Off

- stimo il fondo in una misura dedicata (Off)
- stimo segnale + fondo in un'altra misura (On)
- cerco di inferire il valore del segnale ripulito





$$N_{On} \sim \text{Poiss}(\mu_{On}) = \text{Poiss}(\mu_s + \mu_f)$$



$$N_{Off} \sim \text{Poiss}(\mu_{Off}) = \text{Poiss}(z \cdot \mu_f)$$

se fosse: $\mu, n \gg 0$

$$N \sim \text{Poiss}(\mu) \approx \text{Poiss}(n) \approx \text{Gauss}(n,n)$$

$$\mu_{Off} \approx n_{Off} \pm \sqrt{(n_{Off})}$$

$$\mu_f \approx n_{Off} / z \pm \sqrt{(n_{Off})} / z$$

$$\mu_{On} \approx n_{On} \pm \sqrt{(n_{On})}$$

$$\mu_S \approx (n_{On} - n_{Off} / z) \pm \sqrt{(n_{On} + (n_{Off} / z^2))}$$

outline:

- il problema
- una soluzione
- la libreria
- le applicazioni

c’e’ un segnale?

- proviamo a misurare la sua significativita’
- partiamo dalla Null Hypothesis di solo fondo
- il problema si riduce a quello di una ripartizione binomiale [Przyborowski & Wilenski, 1940]:

$$\text{p-val} = \frac{(n_{on}+n_{off})!}{(z+1)^{n_{on}+n_{off}}} \times \sum_{n=n_{on}}^{n_{on}+n_{off}} \frac{z^{n_{on}+n_{off}-n}}{n! \times (n_{on}+n_{off}-n)!}$$

- se p-val e’ molto piccolo possiamo rigettare la NH

come lo misuriamo?

- tentiamo un approccio Bayesiano
- usiamo dei prior fisici: $\mu_s \geq 0$ e $\mu_f \geq 0$
- marginalizziamo la distribuzione di μ_s su μ_f
- ricaviamo una distribuzione a posteriori di μ_s
- estraiamo il valore medio, incertezze, limiti superiori
- usiamo il suo profilo per valutare ipotesi complesse

in pratica, che si fa?

- partiamo dai profili di verosimiglianza (likelihood)

$$\mathcal{L}\left(\mu_s, \mu_f | n_{on}\right) = \mathrm{e}^{-(\mu_s+\mu_f)} \frac{\left(\mu_s + \mu_f\right)^{n_{on}}}{n_{on}!}$$

$$\mathcal{L}\left(\mu_f | n_{off}\right) = \mathrm{e}^{-z\times\mu_f} \frac{\left(z \times \mu_f\right)^{n_{off}}}{n_{off}!}$$

- e marginalizziamo su una distribuzione del fondo

$$\mathcal{L}\left(\mu_s | n_{on}, n_{off}\right) = \int_0^{+\infty} \mathcal{L}\left(\mu_s, \mu_f | n_{on}\right) \times \mathcal{L}\left(\mu_f | n_{off}\right)\ d\mu_f$$

cosa si ottiene?

- un profilo di verosimiglianza per il segnale ($\mu_s \geq 0$):

$$\mathcal{L}\left(\mu_s | n_{on}, n_{off}\right) = \frac{\mathrm{e}^{-\mu_s}}{K} \times \sum_{j=0}^{n_{on}} C_j\left(n_{on}, n_{off}\right) \times \mu_s^j$$

$$C_j\left(n_{on}, n_{off}\right) = \frac{(z+1)^j \times (n_{on} + n_{off} - j)!}{(n_{on} - j)! \times j!}$$

- o una distribuzione di probabilita' a posteriori se:

$$K = (z+1)^{n_{on}} \times \sum_{j=0}^{n_{on}} \frac{(n_{off} + j)!}{j! \times (z+1)^j}$$

quale prior?

- abbiamo assunto un prior piatto con fondo positivo:

$$\Pi\left(\mu_f\right) = \Theta\left(\mu_f\right) = \begin{cases} 1 & \mu_f \geq 0 \\ 0 & \mu_f < 0 \end{cases}$$

- si sarebbe potuto usare un prior esponenziale:

$$\Pi_\varepsilon\left(\mu_f\right) = \Theta\left(\mu_f\right) \times \varepsilon \times \mathrm{e}^{-\varepsilon \times \mu_f} = \begin{cases} \varepsilon \times \mathrm{e}^{-\varepsilon \times \mu_f} & \mu_f \geq 0 \\ 0 & \mu_f < 0 \end{cases}$$

$$\mathcal{L}_\varepsilon\left(\mu_s | n_{on}, n_{off}\right) = \int_0^{+\infty} \varepsilon \times \mathrm{e}^{-\varepsilon \times \mu_f} \times \mathcal{L}\left(\mu_s, \mu_f | n_{on}\right) \times \mathcal{L}\left(\mu_f | n_{off}\right)\ d\mu_f$$

outline:

- il problema
- una soluzione
- la libreria
- le applicazioni

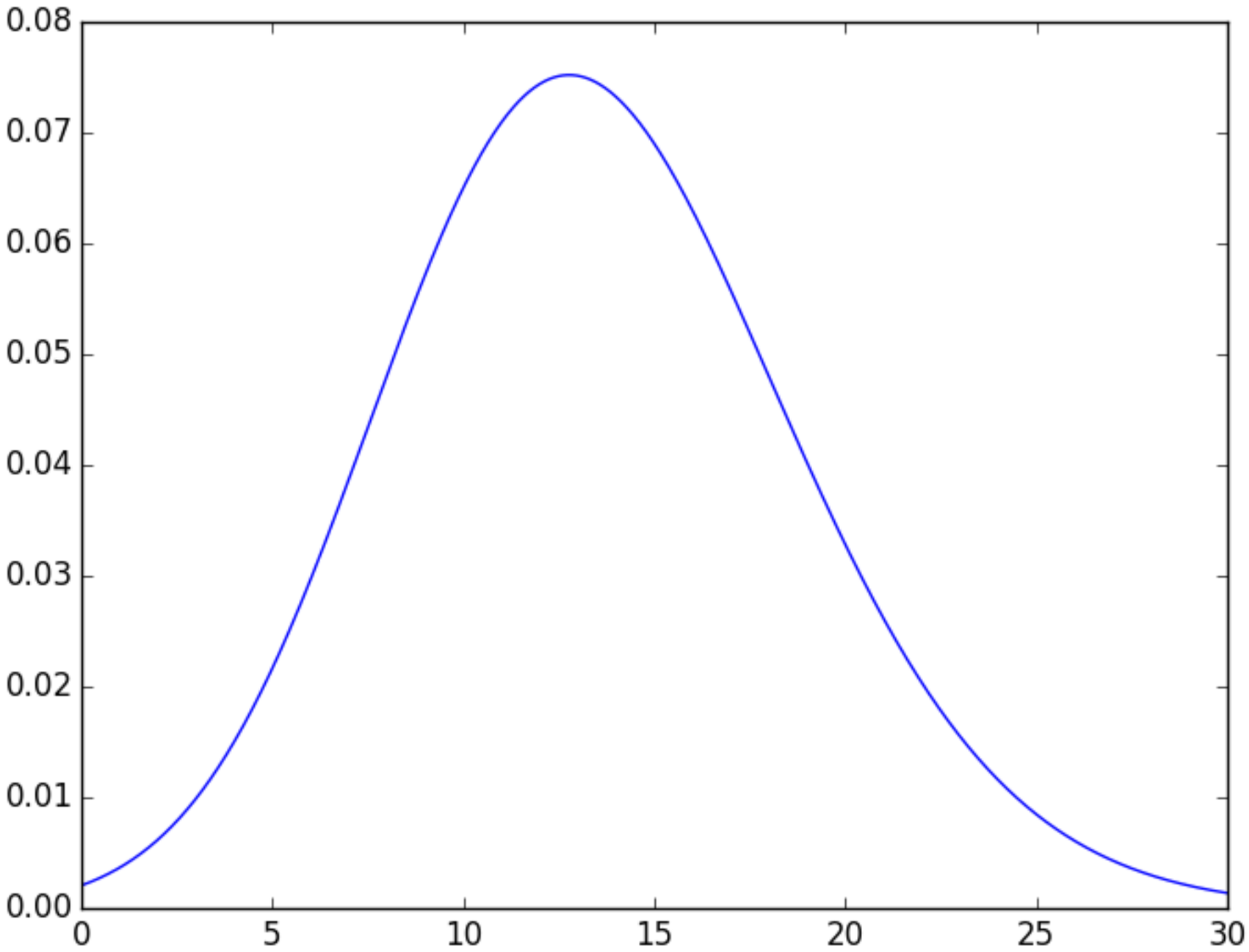
BLike

- sia pure tutto ok, ma e’ un po’ complicato…
- ho scritto un modulo Python, BLike che fa tutto
- e’ veloce e comodo da inserire in script complessi
- sto lavorando ora alla documentazione (+paper)
- non l’ho ancora inserito in github, ma lo faro’

come si usa?

```
ipython

In [1]: from BLike import BLike

In [2]: L=BLike(N_on=23, N_off=20, bkg_ratio=2.)

In [3]: print 'Detection at %.2f sigma (P-value=%.2e)' % (L.Significance(), L.Significance(True))
Detection at 2.80 sigma (P-value=5.07e-03)

In [4]: print 'Best fit %.3f +%.3f -%.3f' % (L.MostLikely(), L.ErrorBar(), L.ErrorBar(False))
Best fit 12.770 +5.249 -5.404

In [5]: import matplotlib.pyplot as plt

In [6]: import numpy as np

In [7]: X=np.arange(0., 30., .01)

In [8]: Y=[L.Get(x) for x in X]

In [9]: A=plt.plot(X, Y)

In [10]: plt.show()
```

come si usa?



cosa puo' fare?

```
In [11]: L.Integrate(15, 25)
Out[11]: 0.3518258952415475

In [12]: L.UpperLimit()
Out[12]: 23.07889473438263

In [13]: L.GetIntegral(_)
Out[13]: 0.9545003403538995 2

In [14]: L.UpperLimit(coverage=.99)
Out[14]: 27.220936238765717

In [15]: L.MostLikely()
Out[15]: 12.769500732421875

In [16]: L.Integrate(_ - L.ErrorBar(up=False), _ + L.ErrorBar(up=True))
Out[16]: 0.6826894813580610 7

In [17]: L.MaxLike()
Out[17]: 0.07520781276206861 5

In [18]: L.Get(12.37)
Out[18]: 0.07498987046952373 9

In [19]: L.TS(12.37)
Out[19]: 5.180804469947307 9
```

outline:

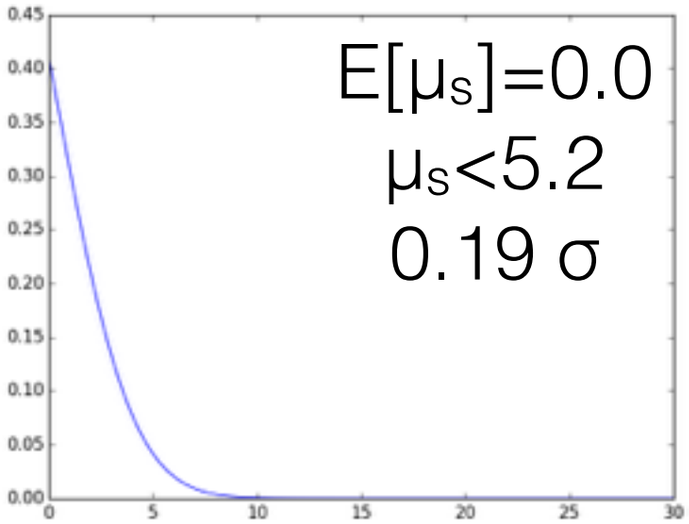
- il problema
- una soluzione
- la libreria
- le applicazioni

ma… funziona?

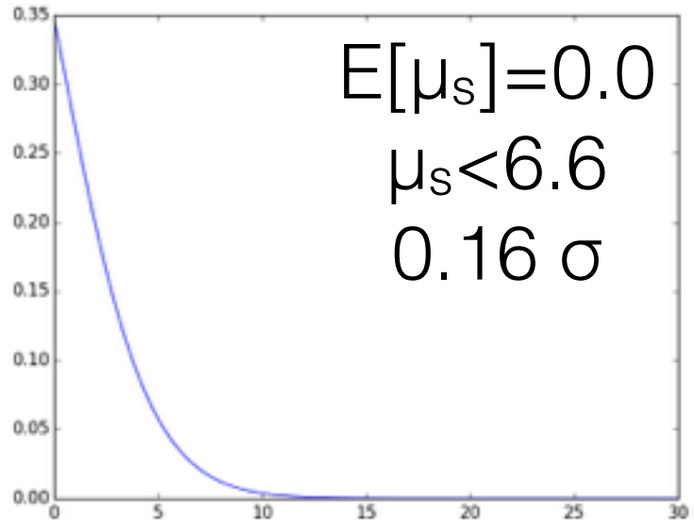
- 2 tipi di test: confronto con GLike e Montecarlo
- GLike usa l'approssimazione Gaussiana, troncata ai soli valori positivi di μ_s e normalizzata a 1
- GLike ha la stessa interfaccia di BLike e dovrebbe dare gli stessi risultati per $\mu_s \gg 1$
- via Montecarlo controllo le coperture dei limiti superiori e delle incertezze e la sensibilita'

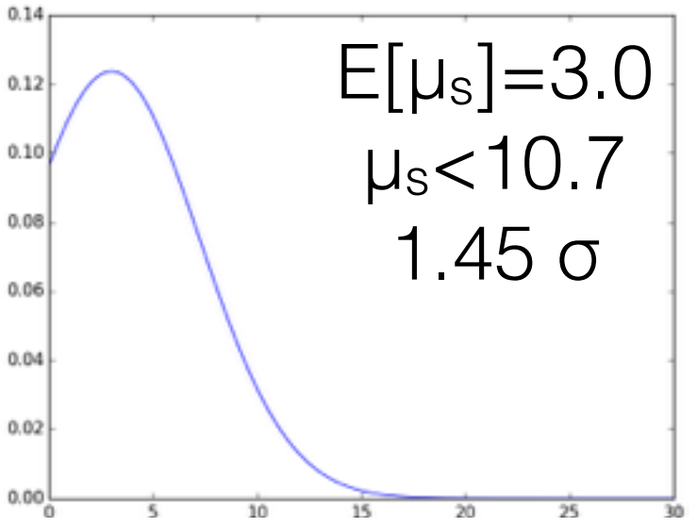
GLike

BLike

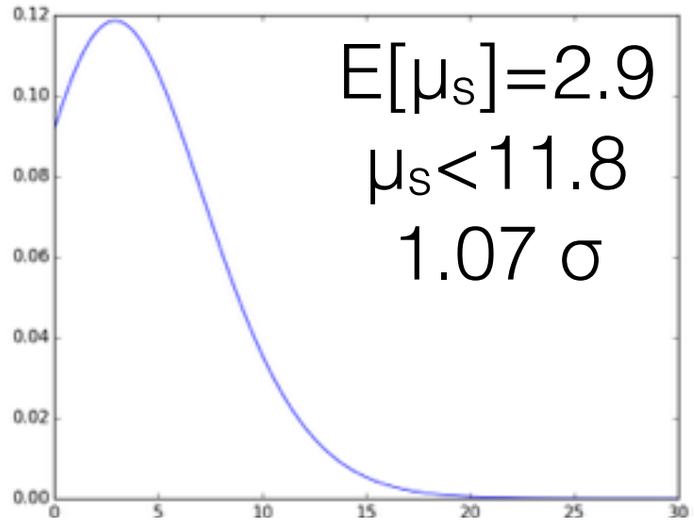


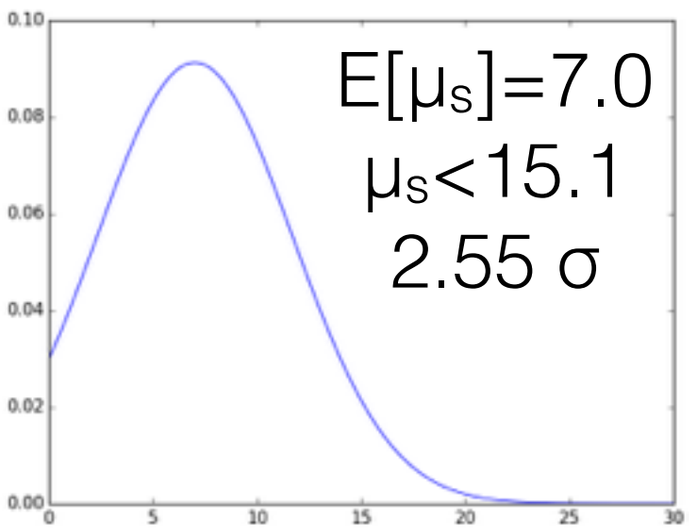
$n_{On}=7$ $n_{Off}=20$ $z=2$



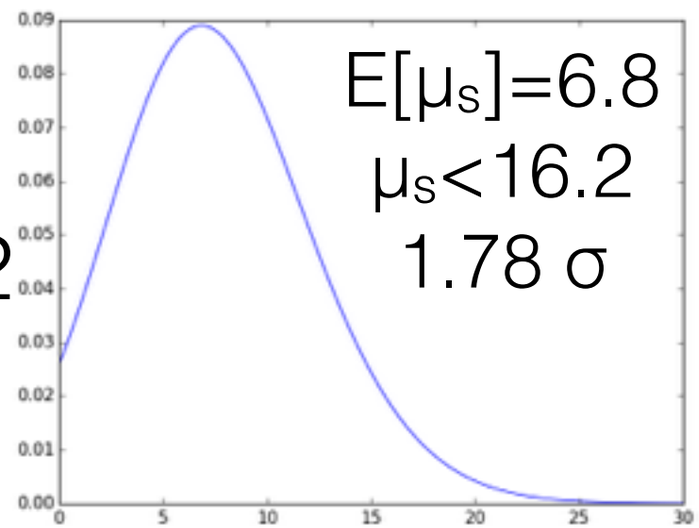


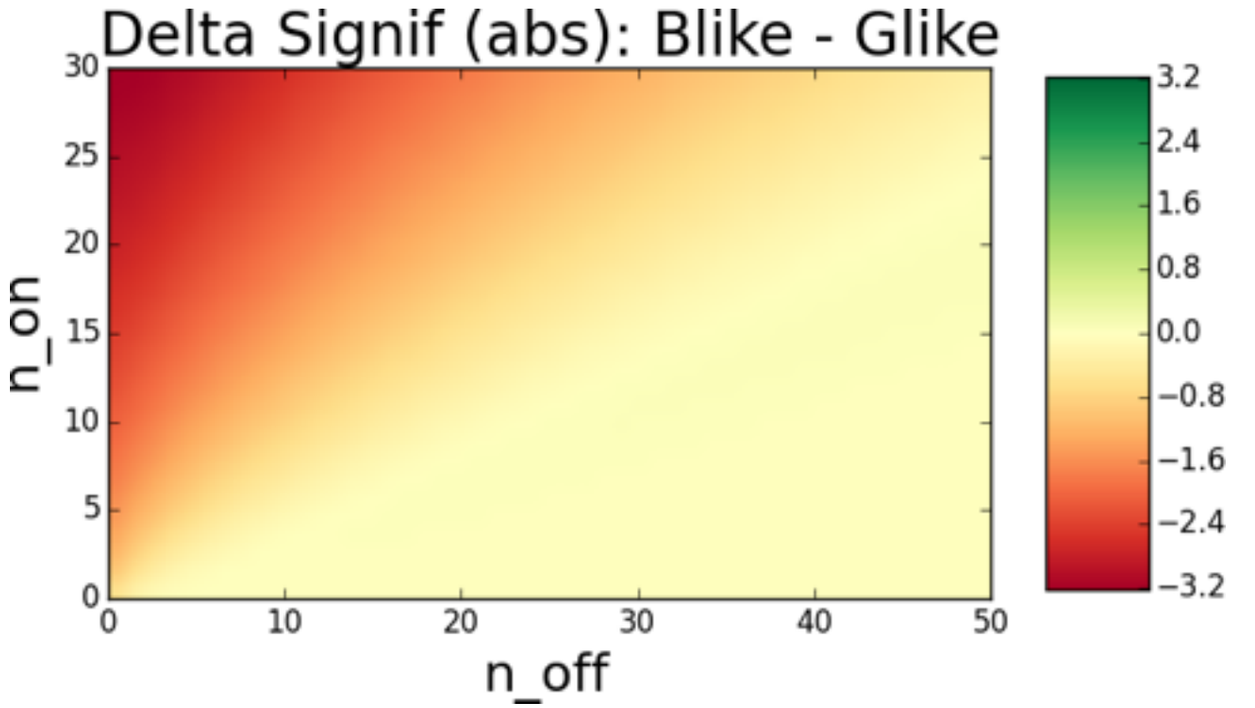
$n_{On}=13$ $n_{Off}=20$ $z=2$

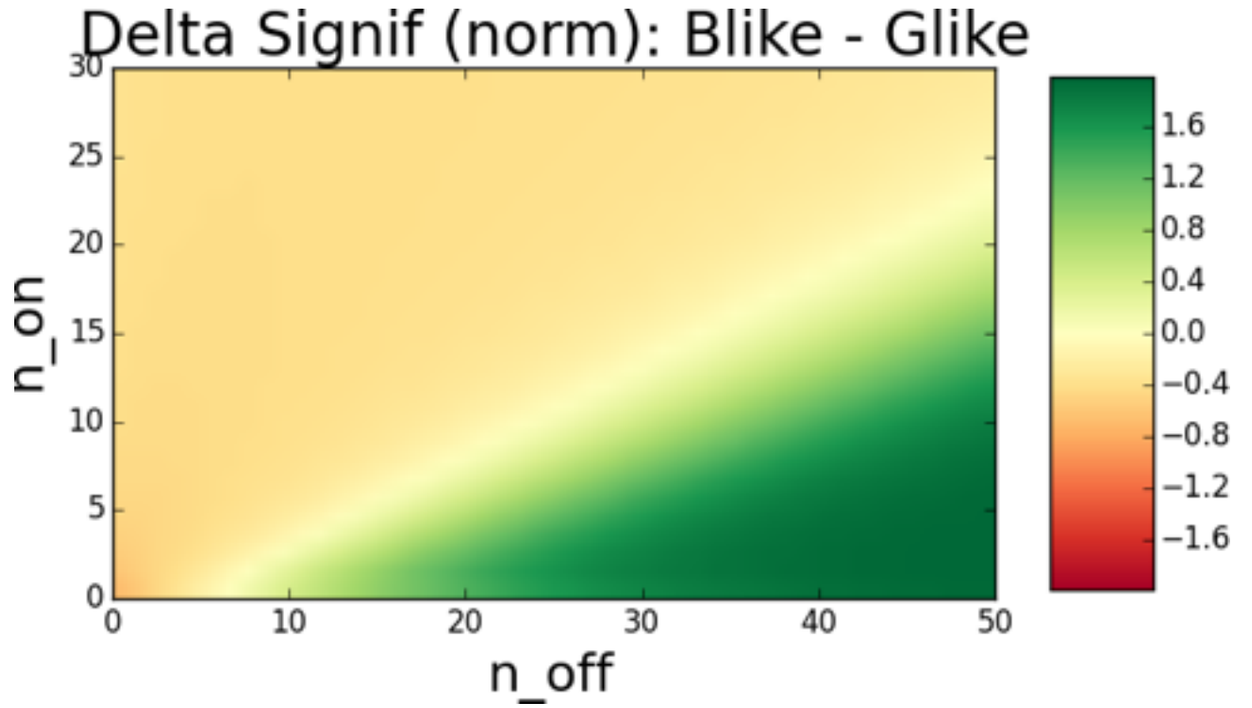


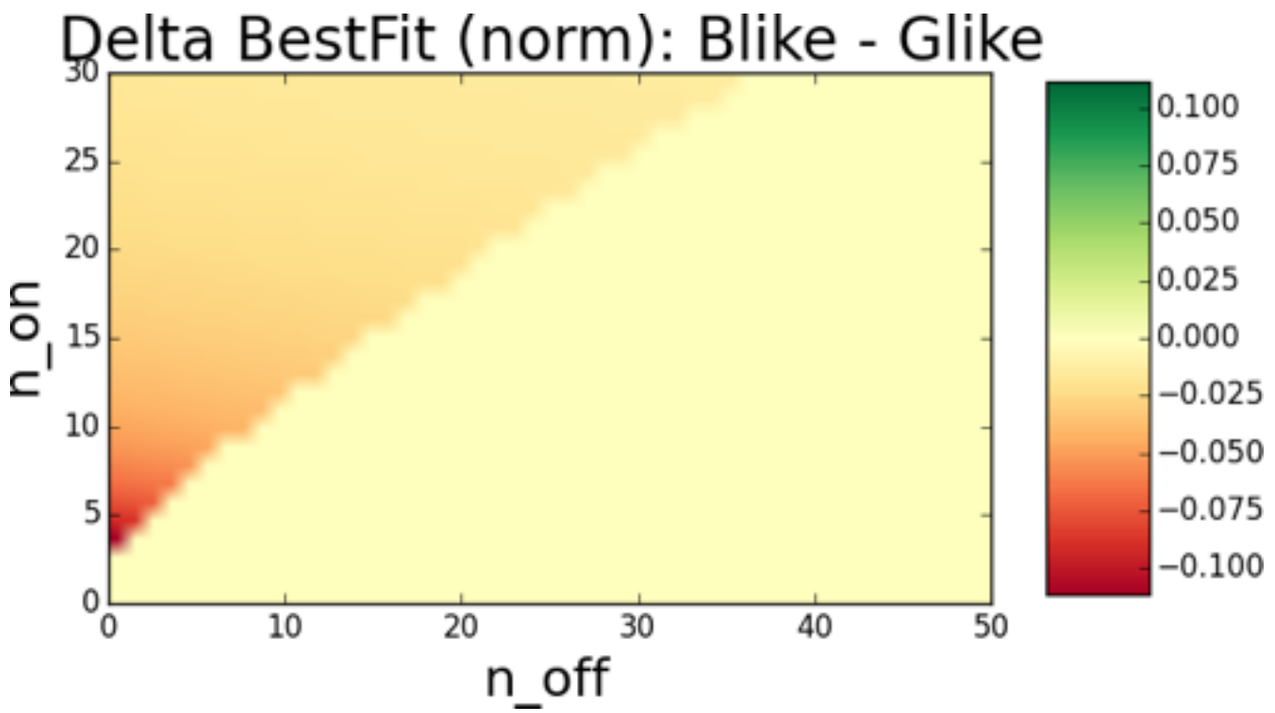


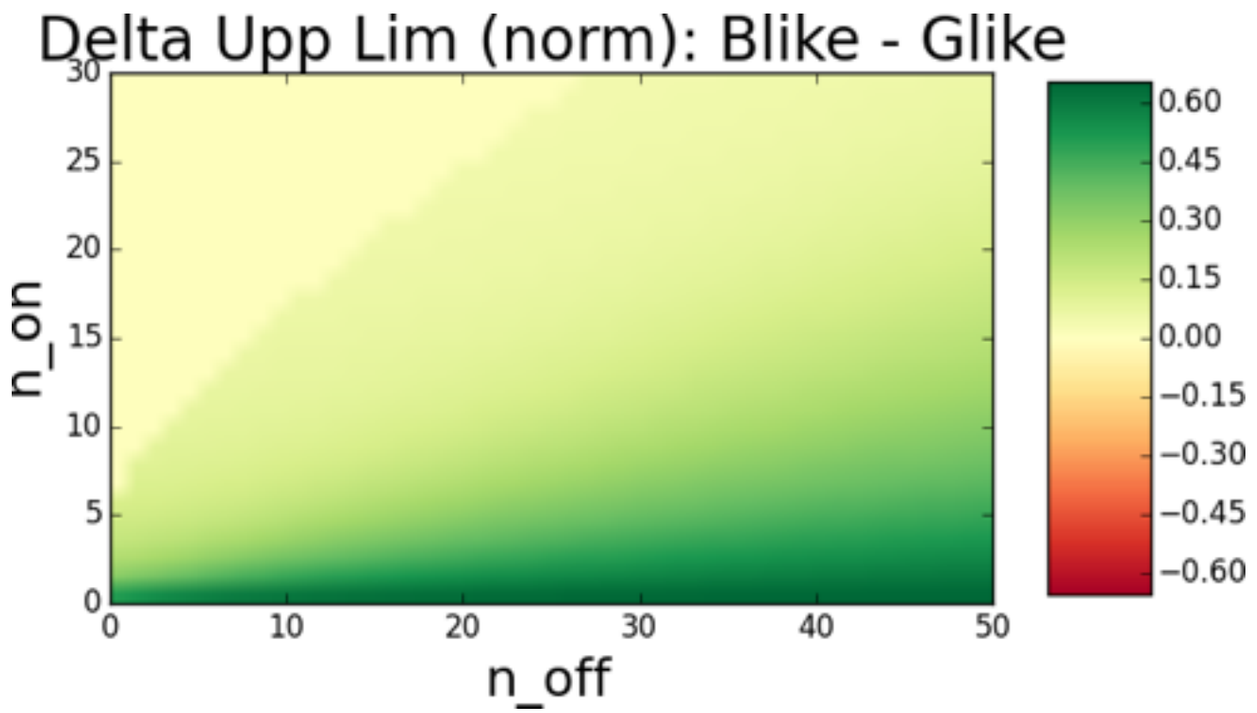
$n_{On}=17$ $n_{Off}=20$ $z=2$




Delta Signif (abs): Blike - Glike
n_on
n_off
0
5
10
15
20
25
30
0
10
20
30
40
50
3.2
2.4
1.6
0.8
0.0
−0.8
−1.6
−2.4
−3.2


Delta Signif (norm): Blike - Glike
n_on
n_off
0
5
10
15
20
25
30
0
10
20
30
40
50
1.6
1.2
0.8
0.4
0.0
−0.4
−0.8
−1.2
−1.6


Delta BestFit (norm): Blike - Glike
n_on
n_off
0
5
10
15
20
25
30
0
10
20
30
40
50
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100


Delta Upp Lim (norm): Blike - Glike
n_on
n_off
0
5
10
15
20
25
30
0
10
20
30
40
50
0.60
0.45
0.30
0.15
0.00
−0.15
−0.30
−0.45
−0.60

che me ne faccio?

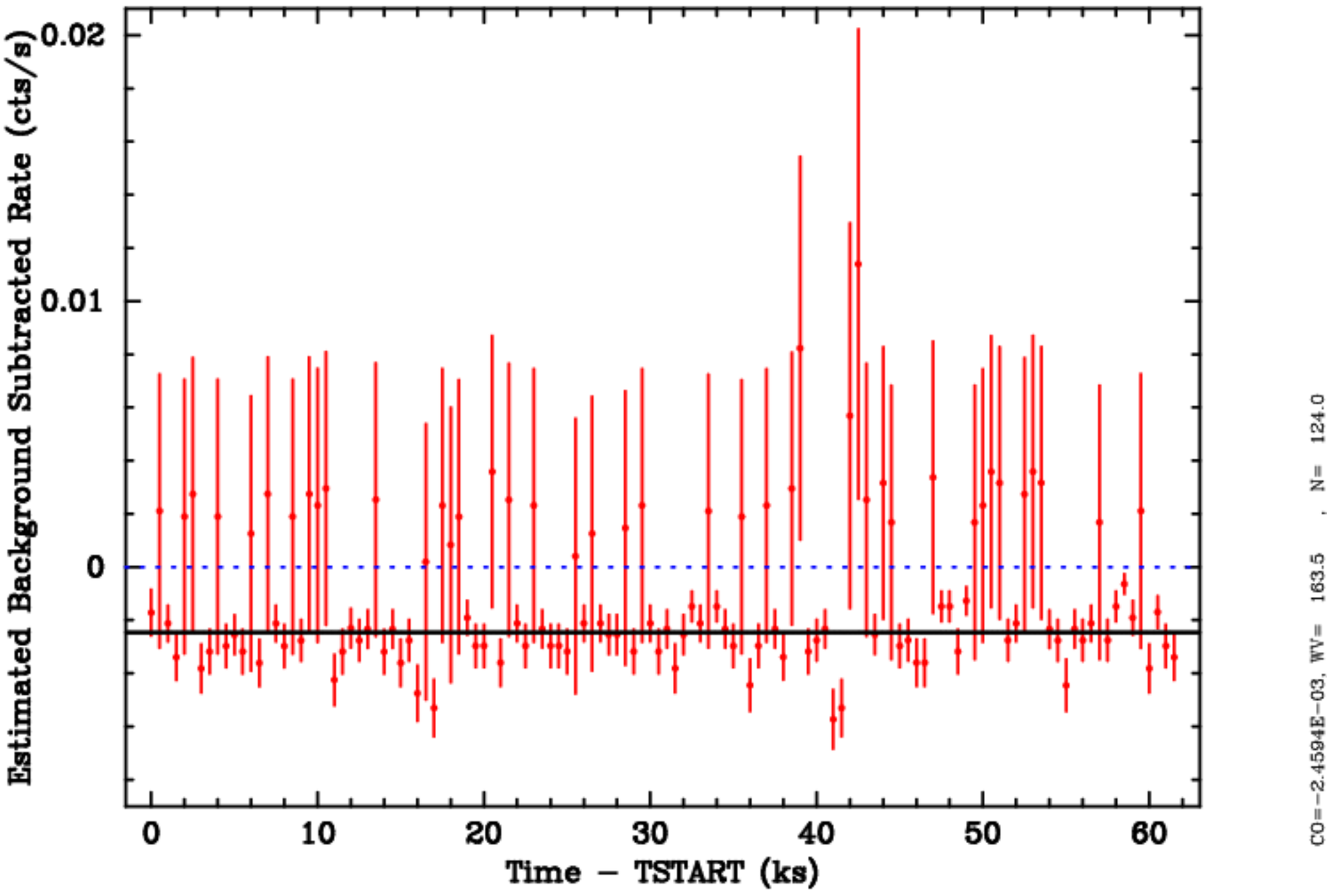
- devo avere dei problemi riconducibili a questo: conteggi, fondo da sottrarre, piccole statistiche

- puo' essere usato per curve di luce, spettri, studio di sorgenti estese o un mix di queste

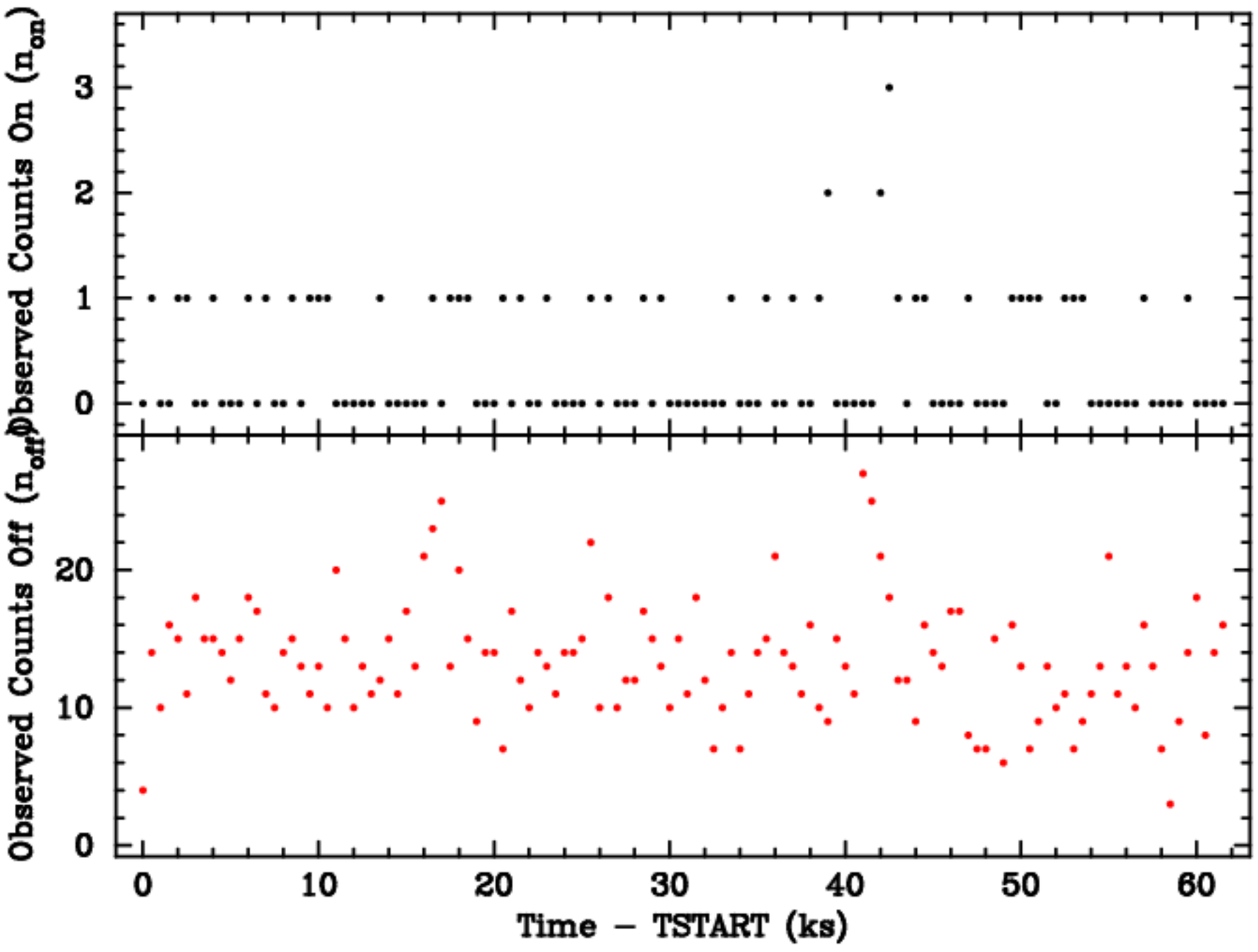
- se faccio bin abbastanza piccoli le statistiche diminuiscono, se no ricado in GLike (approssimato)

- la misura Off puo' venire anche da modelli molto piu' complessi (come in EXTraS, con BLike2)

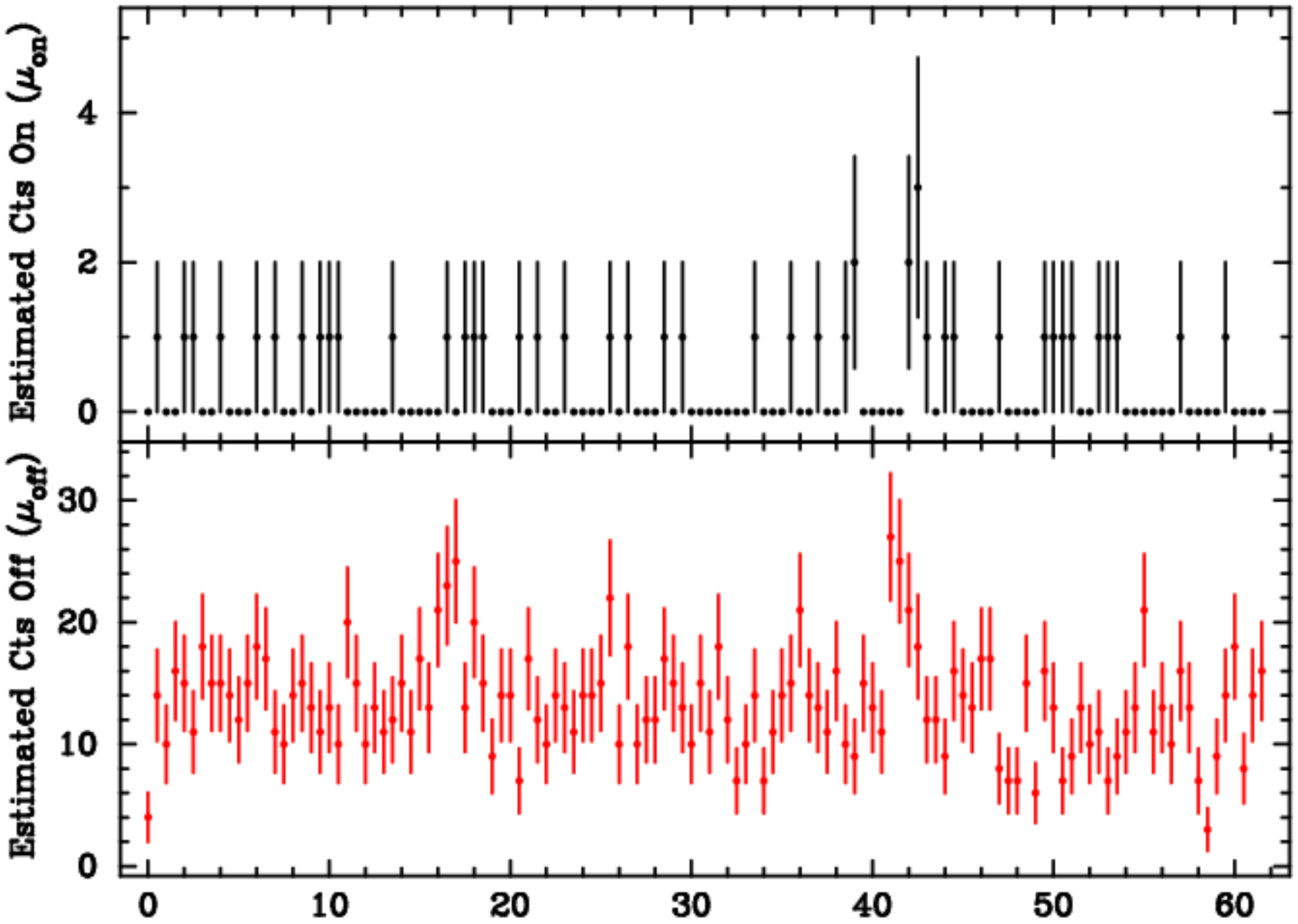
ha sistemato il problema?



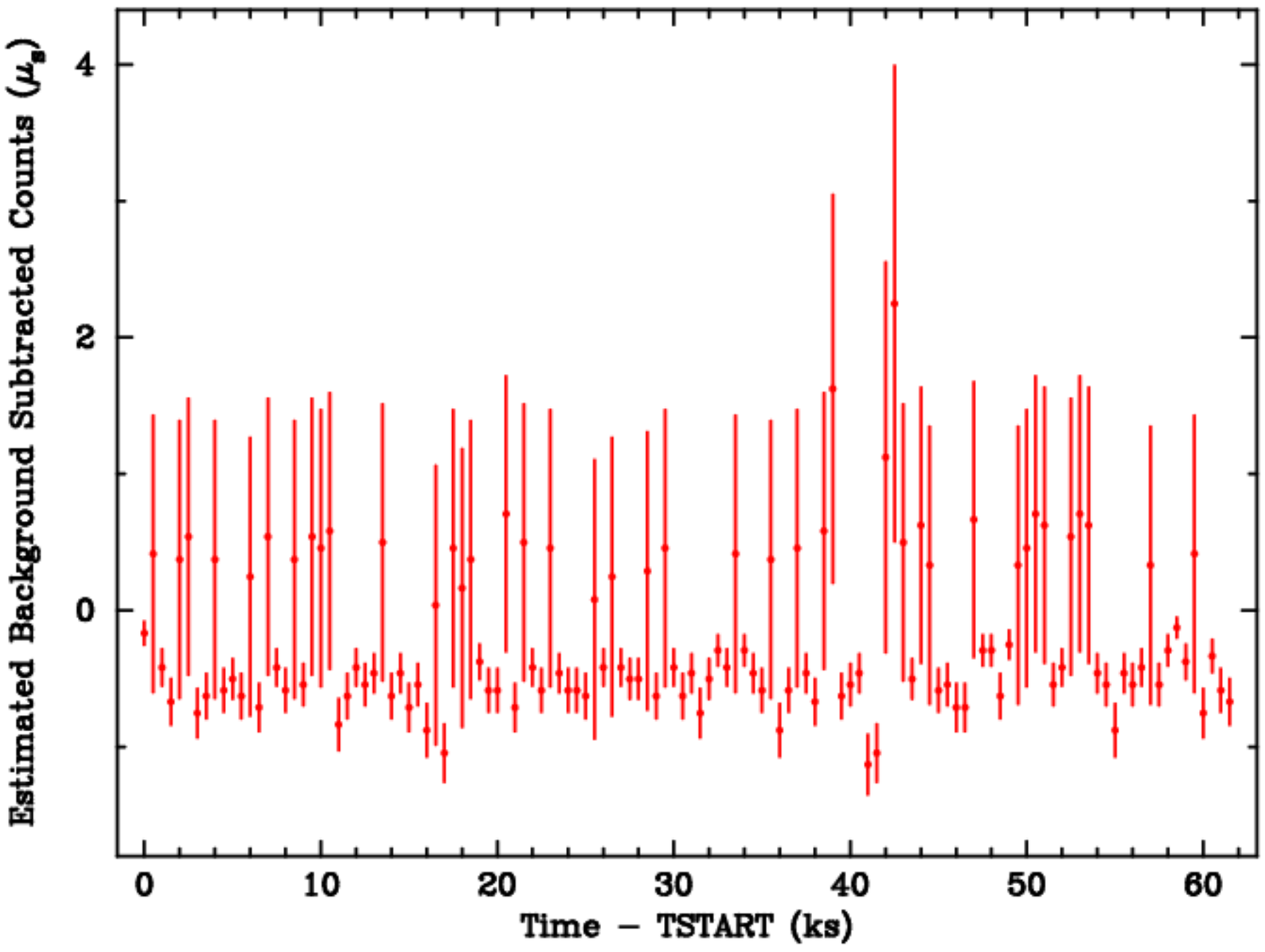
cosa avevamo davvero osservato?



questa e’ la stima in approssimazione Gaussiana

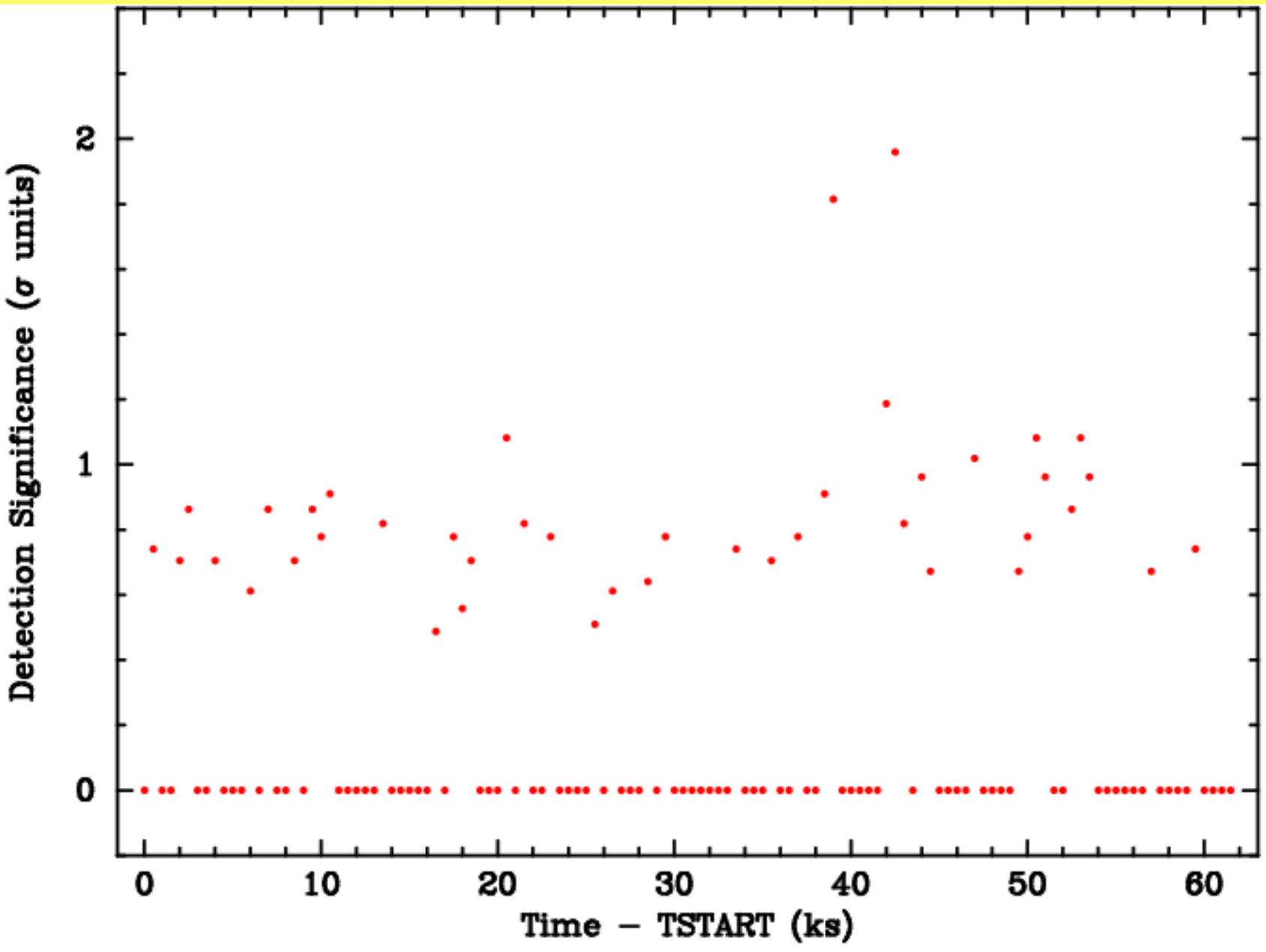


riscaliamo, sottraiamo, propaghiamo gli errori

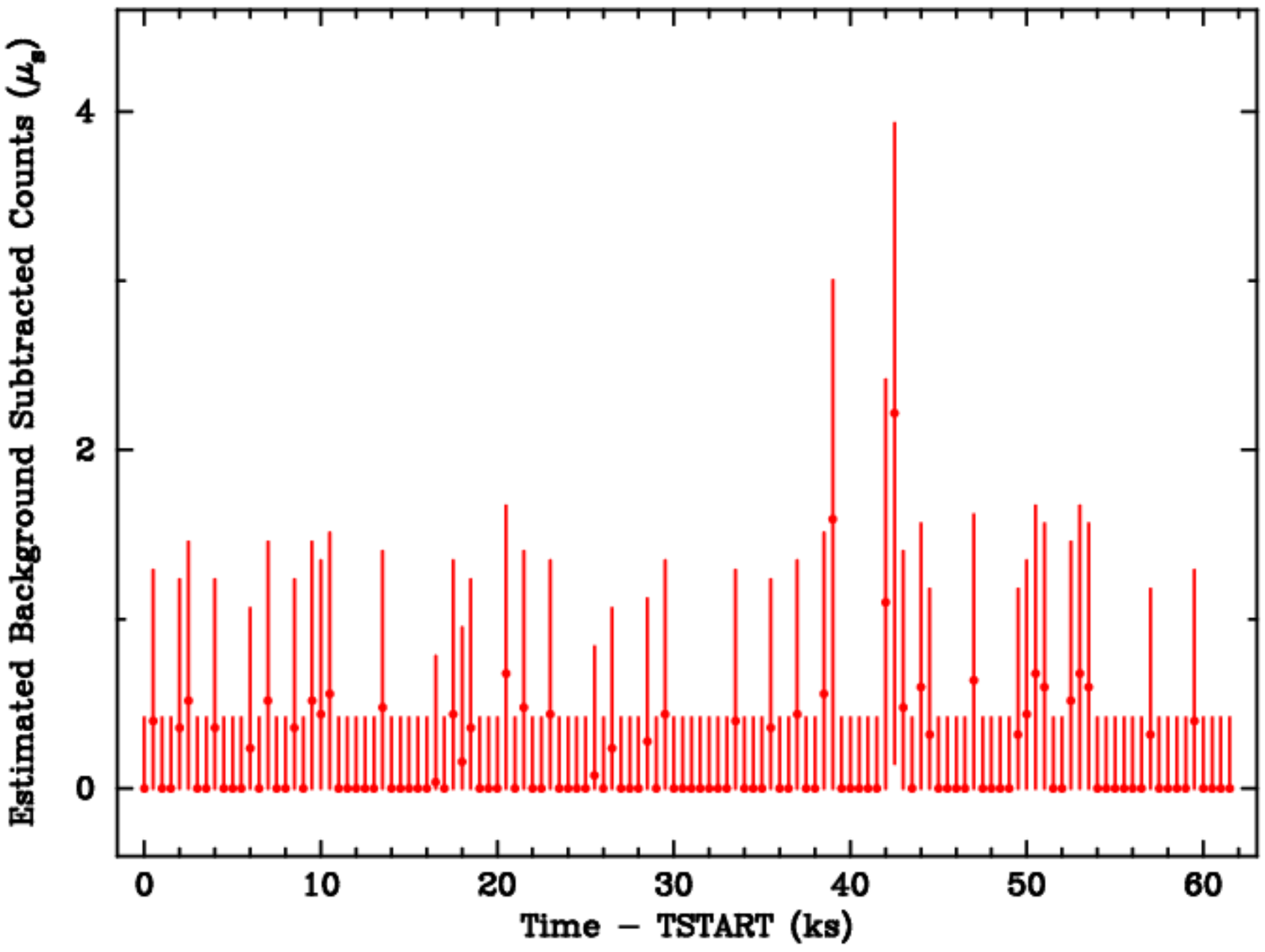


controlliamo se la sorgente si vede

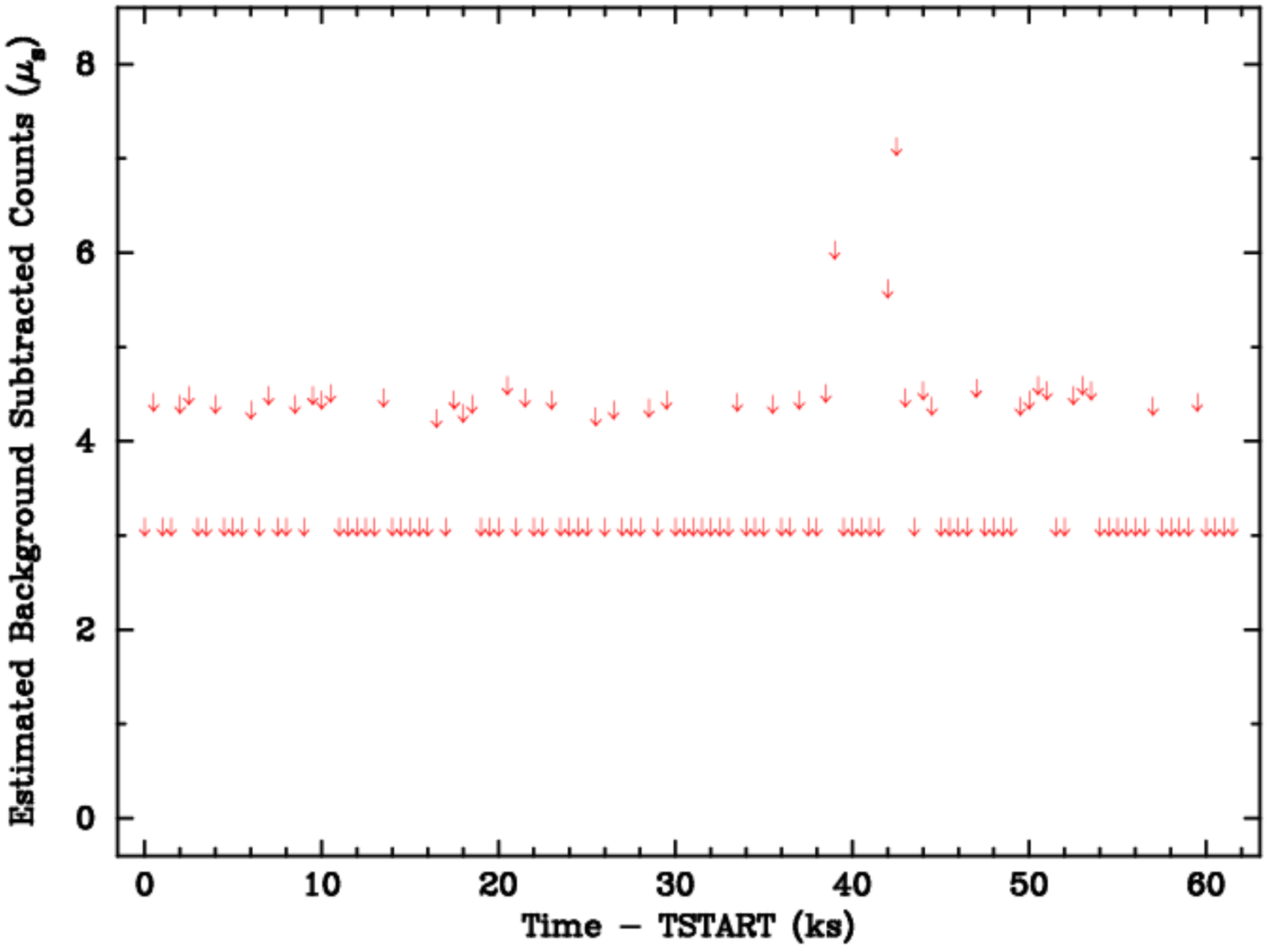
```
Like = [BLike(N_on=N_on[i], N_off=N_off[i], bkg_ratio=zeta[i]) for i in range(N_points)]
for i in range(N_points):
  print (time[i] - time[0])/1000., Like[i].Significance()
```



possiamo visualizzare i valori piu’ probabili

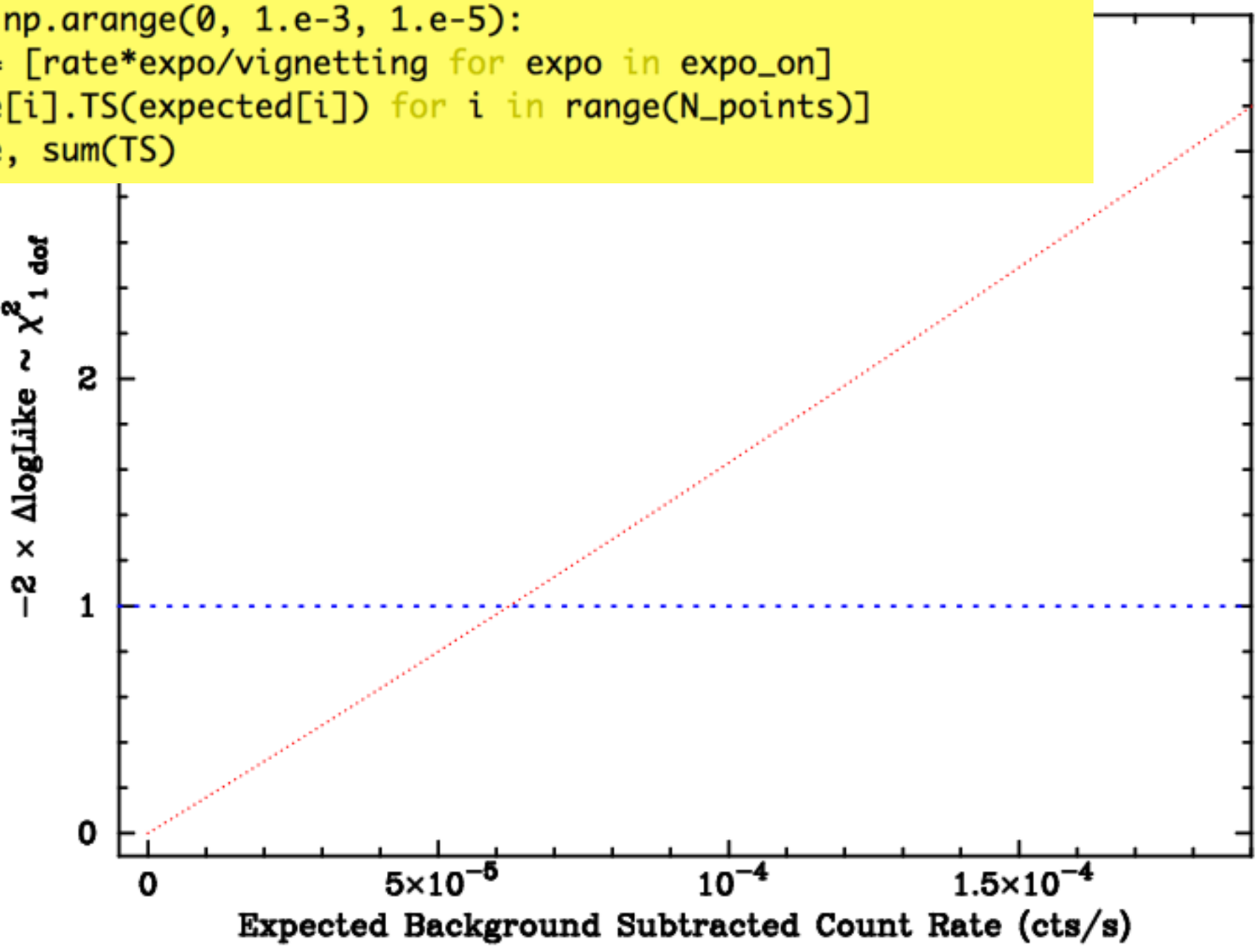


o forse meglio i limiti superiori

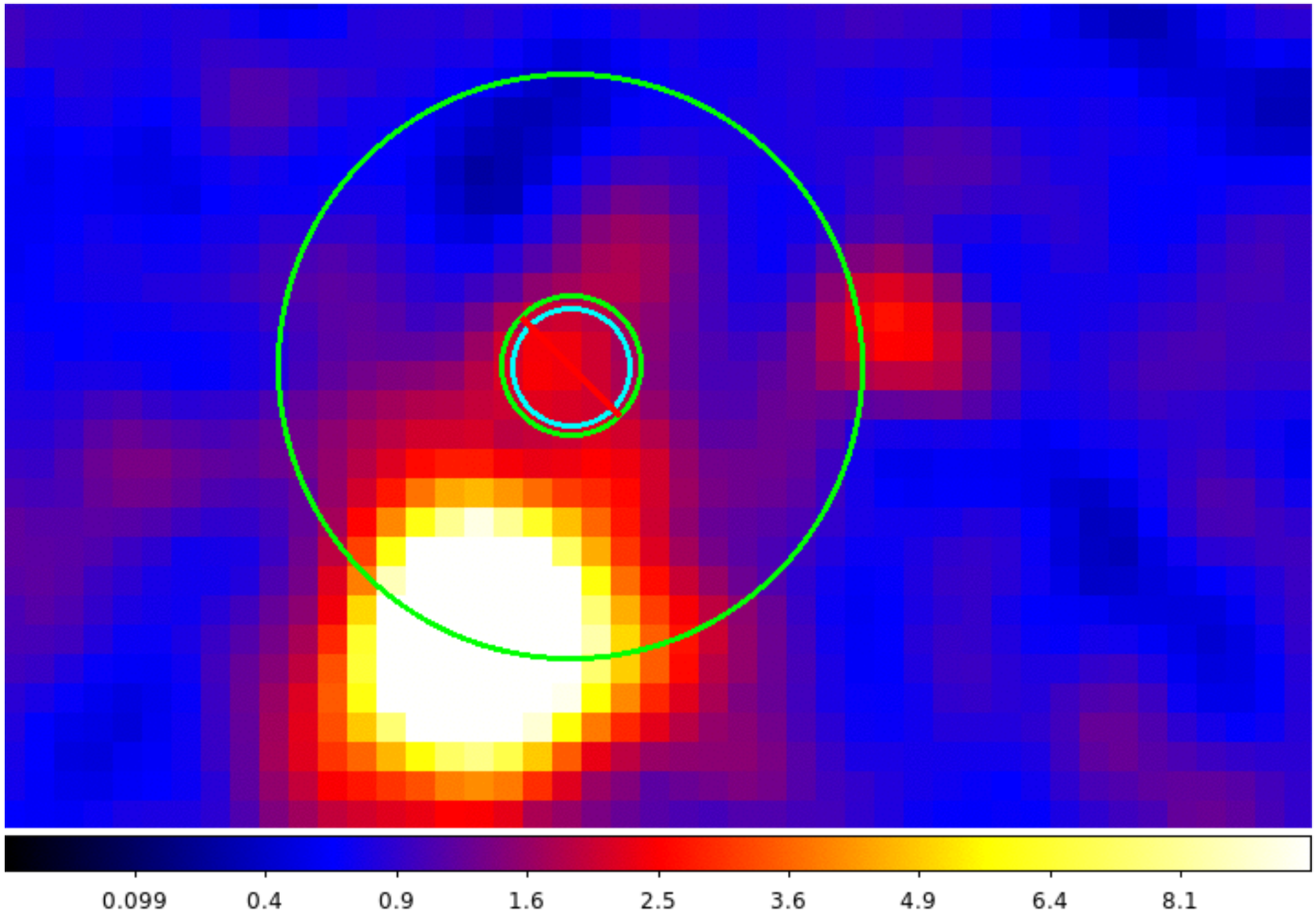


conserviamo i profili per fittare una costante

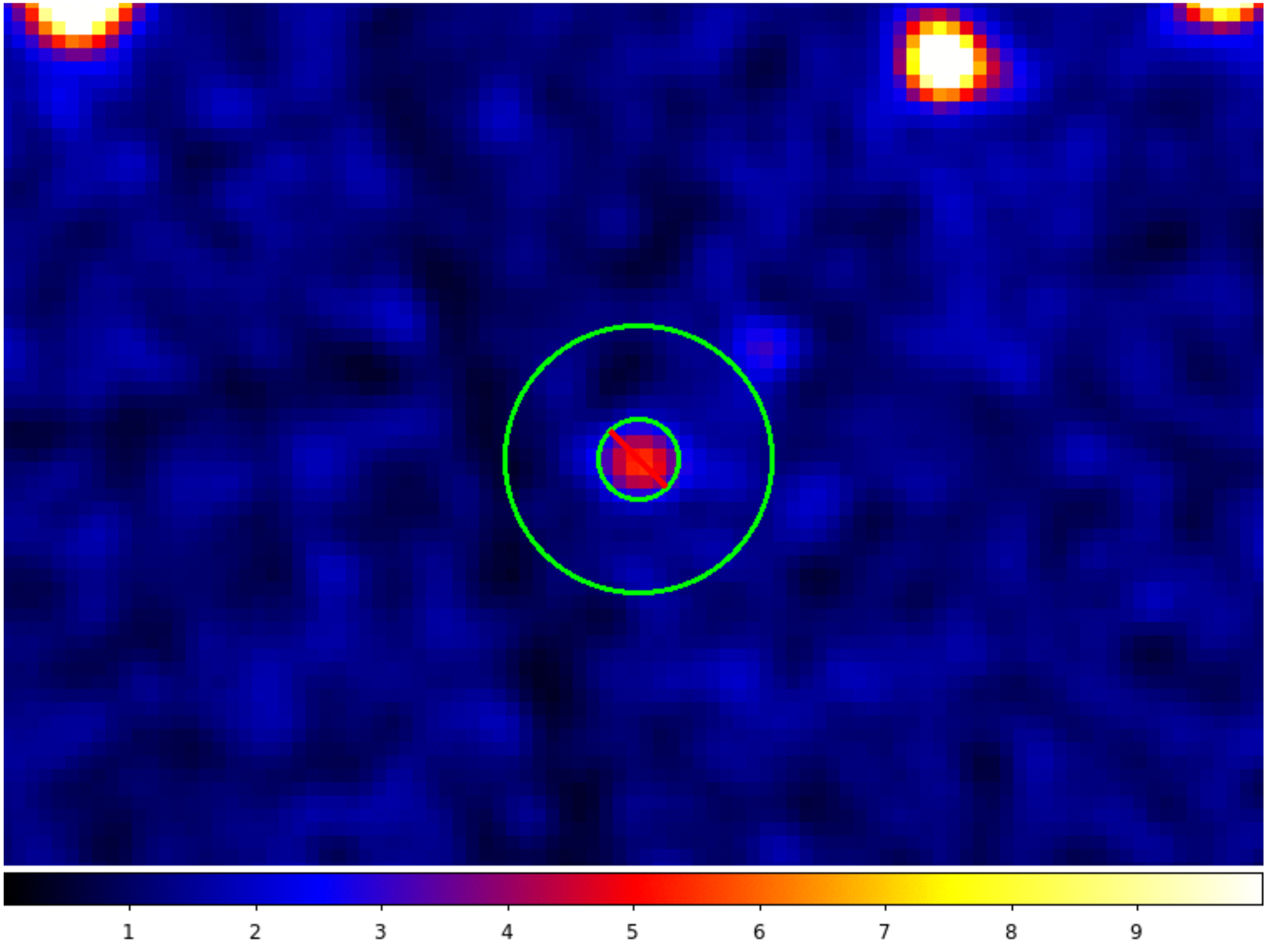
```
vignetting = 2.50487
for rate in np.arange(0, 1.e-3, 1.e-5):
  expected = [rate*expo/vignetting for expo in expo_on]
  TS = [Like[i].TS(expected[i]) for i in range(N_points)]
  print rate, sum(TS)
```



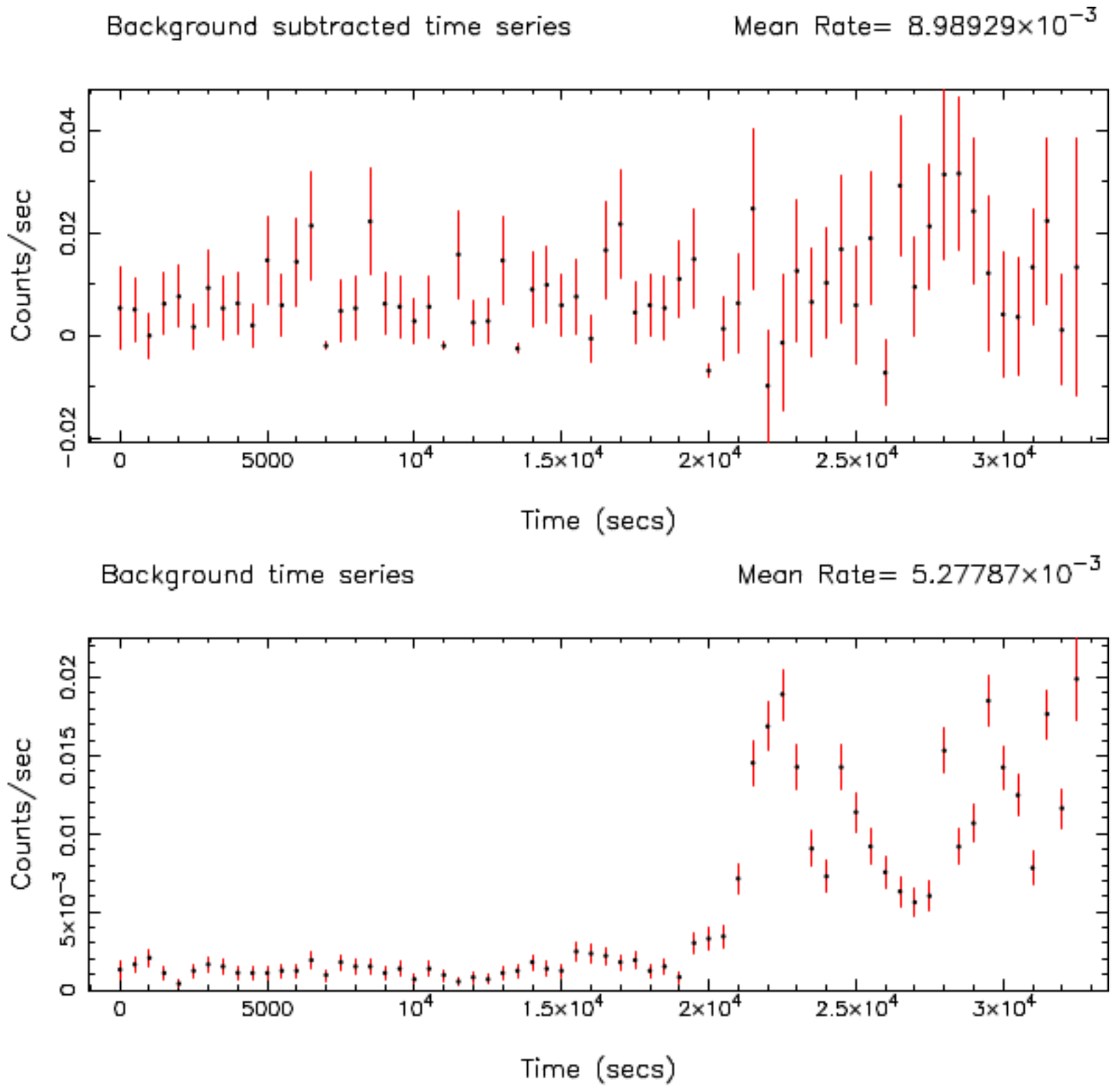
in realta’ questa era una sorgente sciagurata

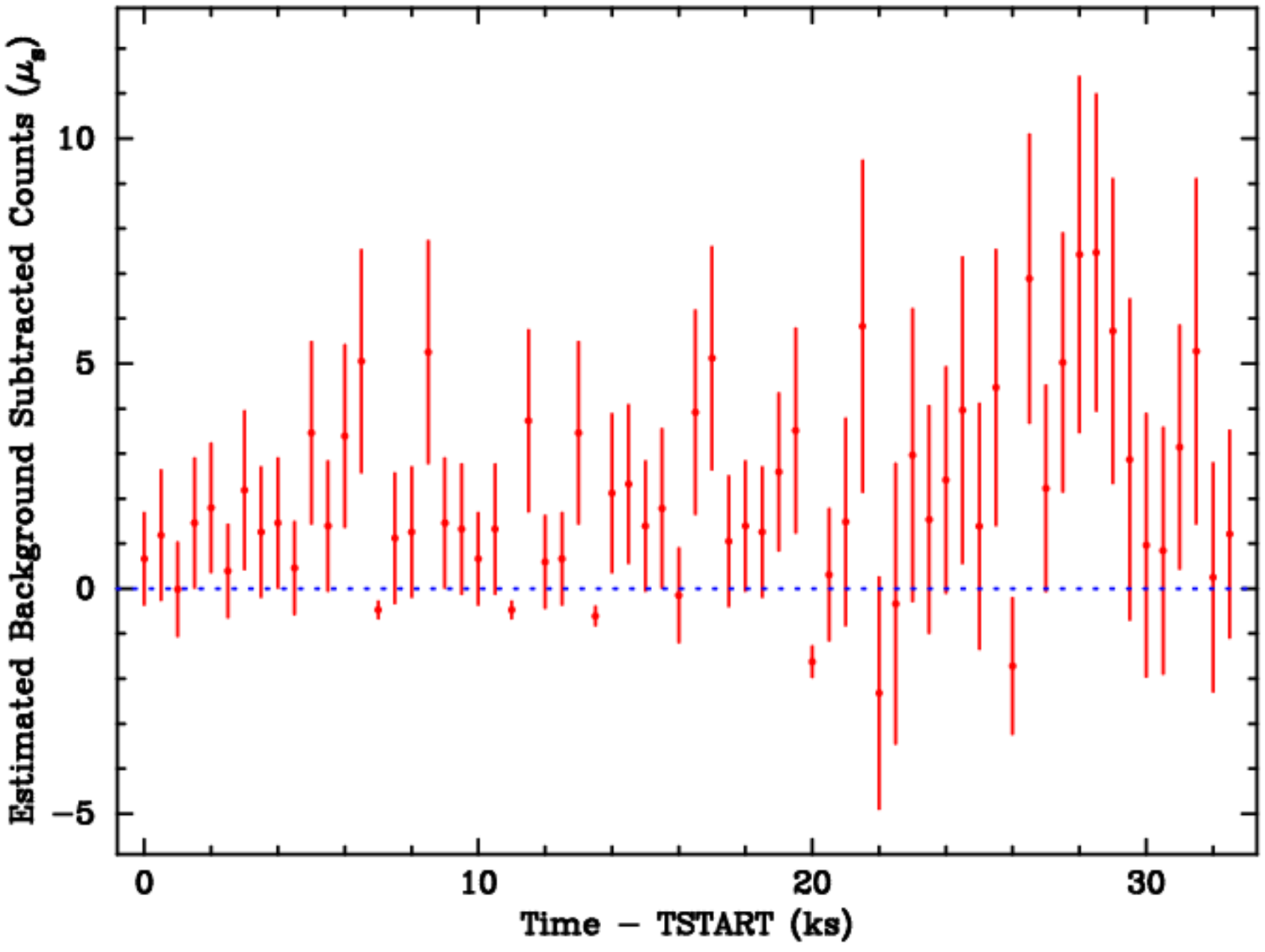


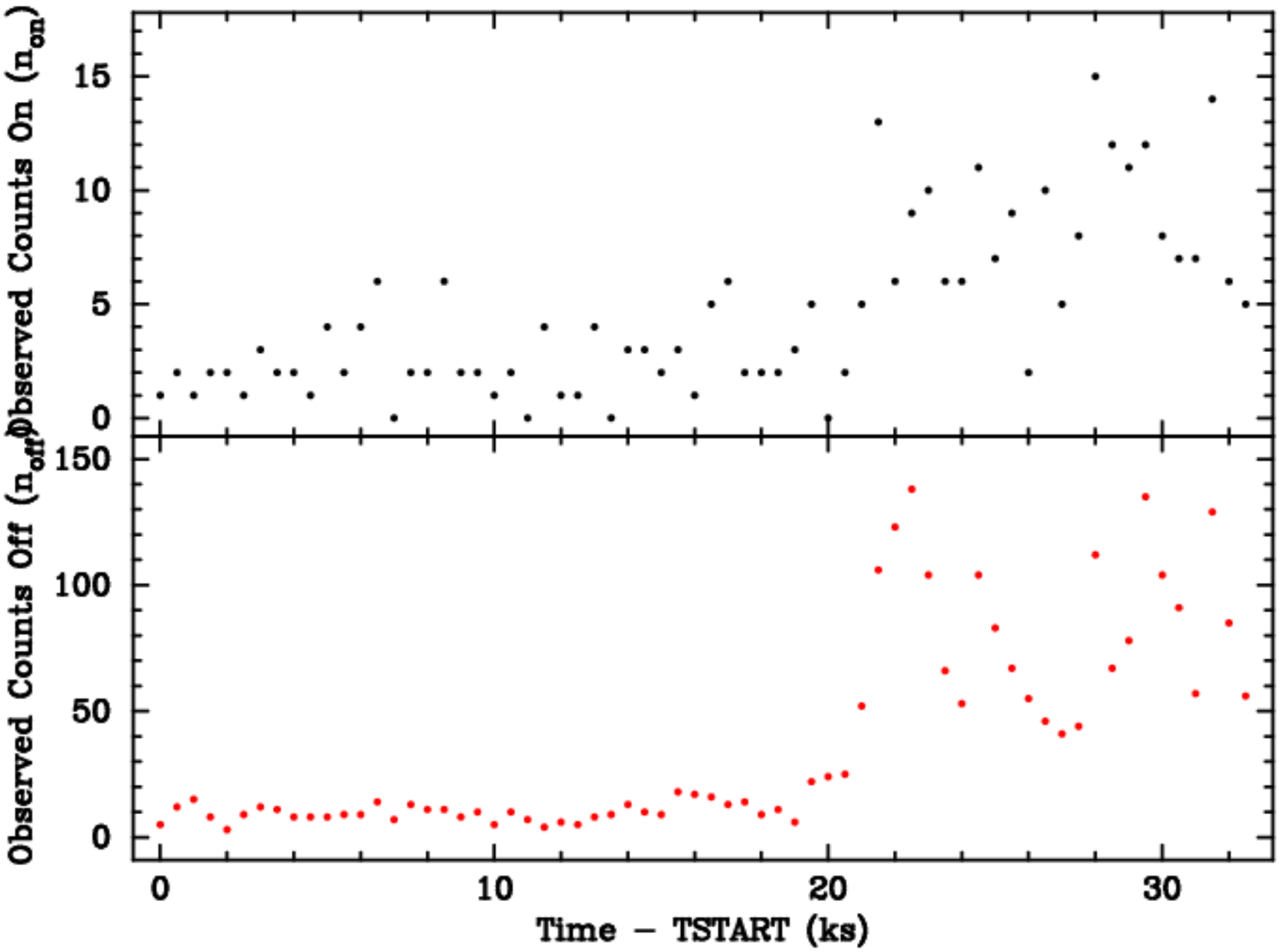
un caso piu’ sano

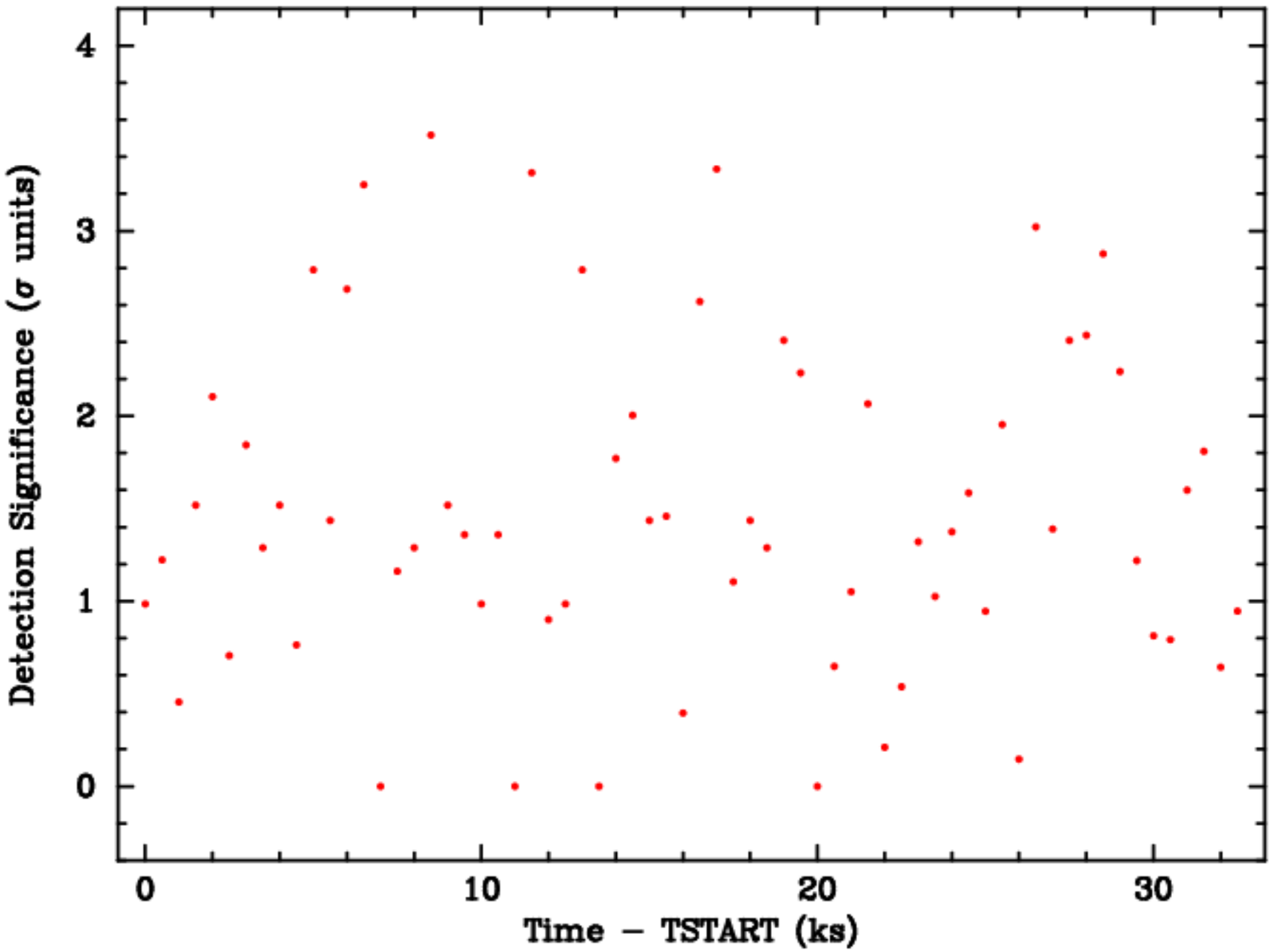


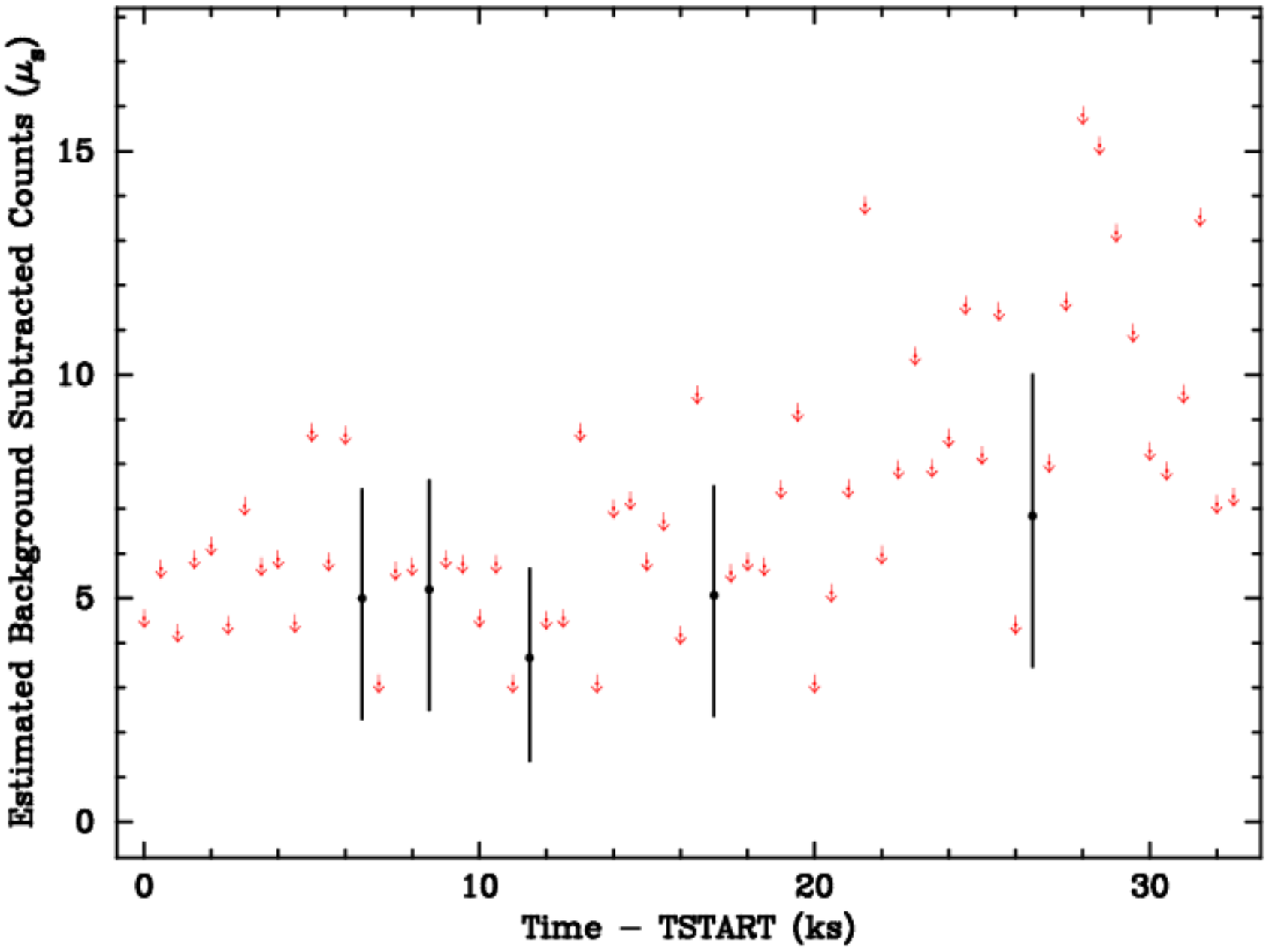
ma con i suoi problemi

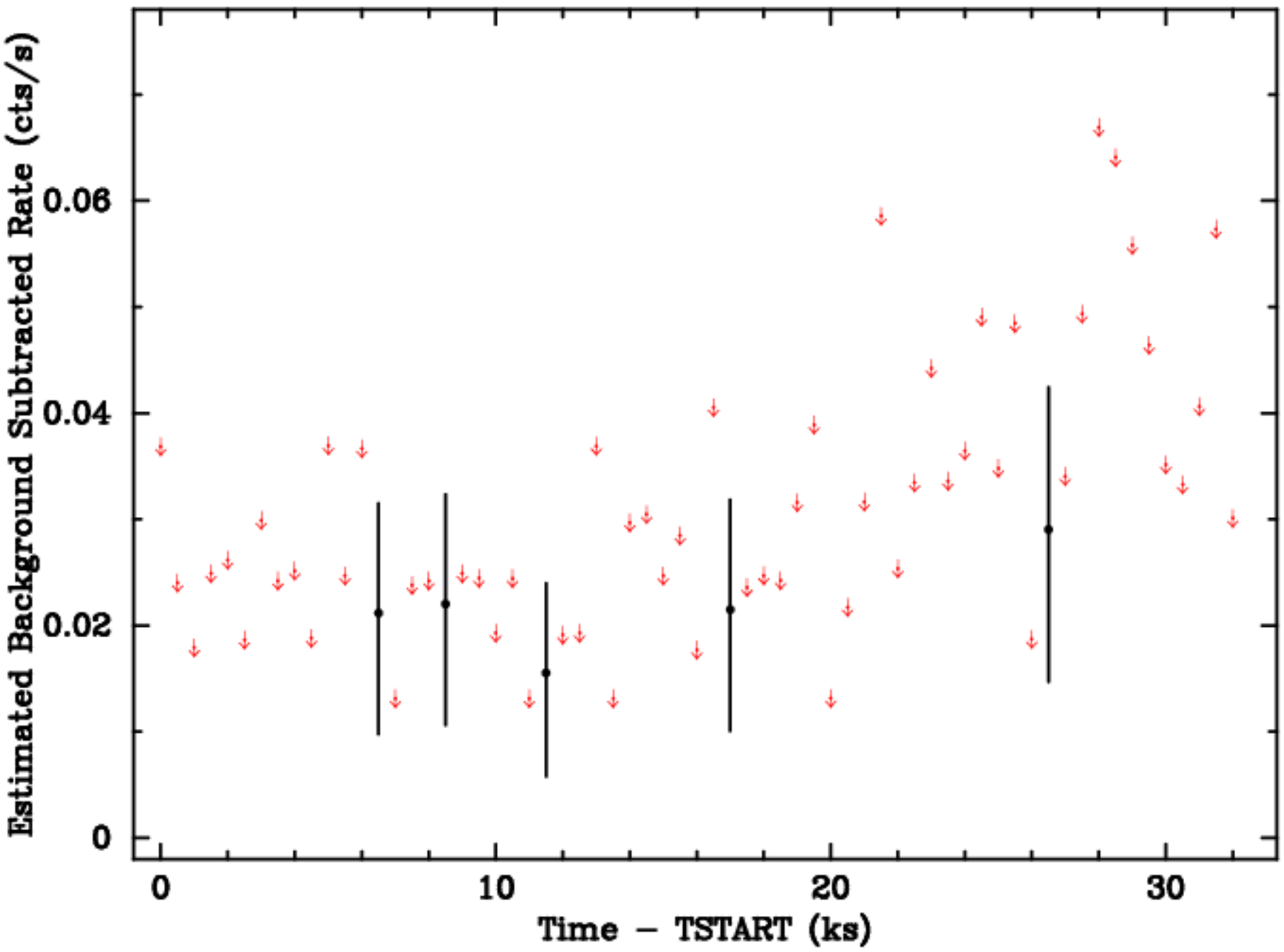


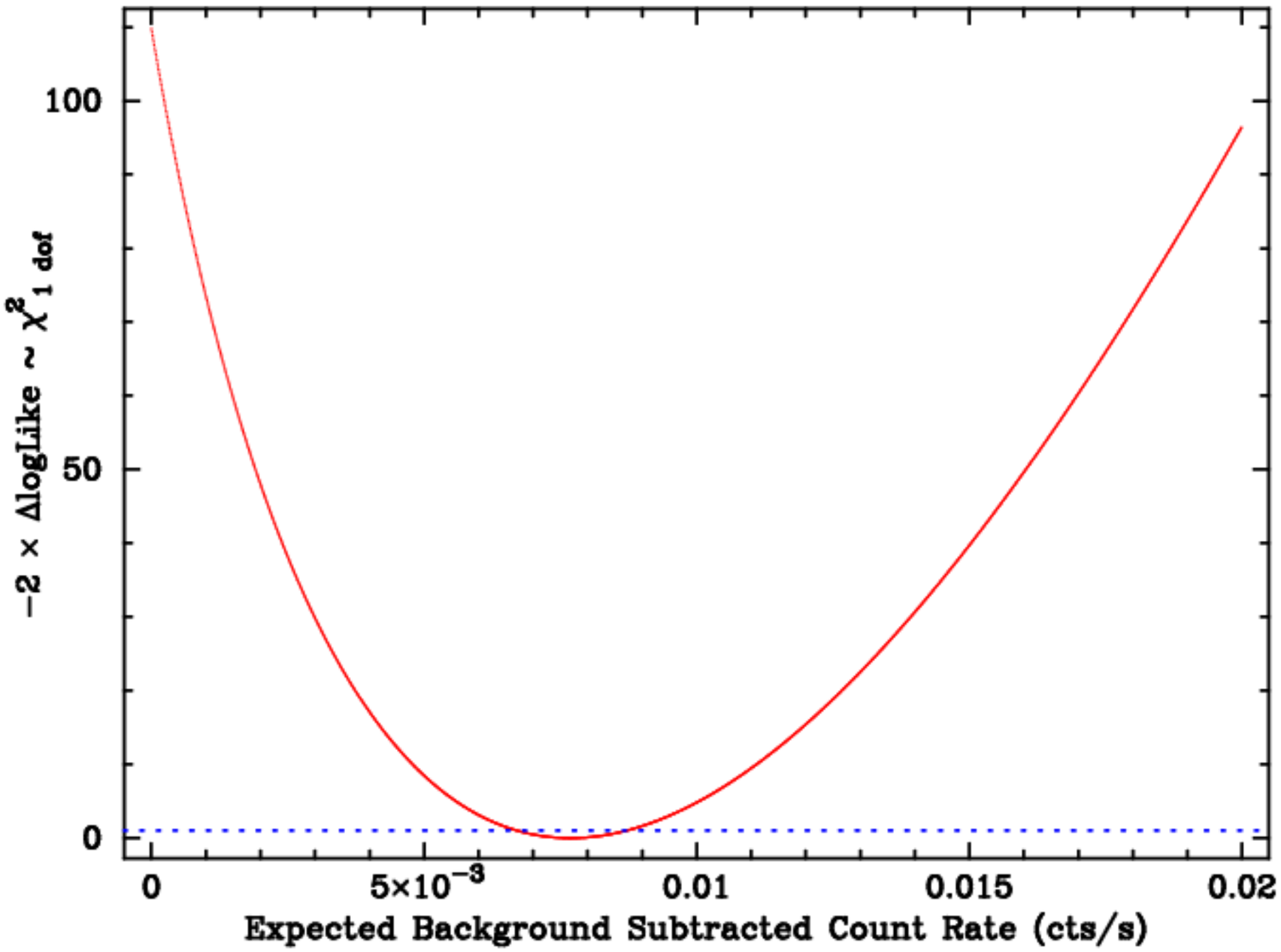

Estimated Background Subtracted Counts (μ_s)
10
5
0
−5
0
10
20
30
Time − TSTART (ks)


Observed Counts On (n_{on})
15
10
5
0
Observed Counts Off (n_{off})
150
100
50
0
0
10
20
30
Time − TSTART (ks)

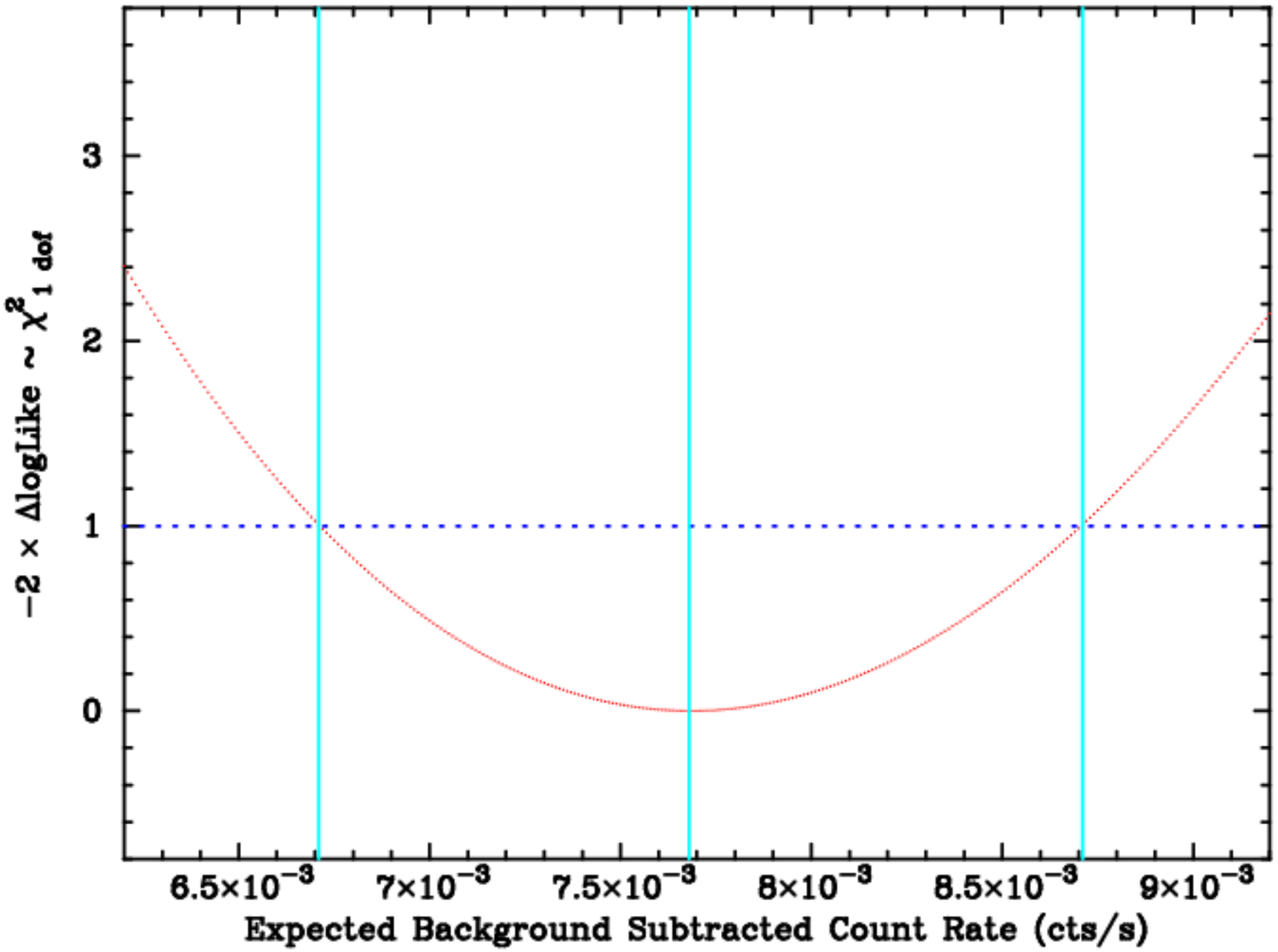

Detection Significance (σ units)
0
1
2
3
4
0
10
20
30
Time − TSTART (ks)


Estimated Background Subtracted Counts (μ_s)
0
5
10
15
0
10
20
30
Time − TSTART (ks)


Estimated Background Subtracted Rate (cts/s)
0.06
0.04
0.02
0
0
10
20
30
Time − TSTART (ks)


100
50
0
−2 × ΔlogLike ~ χ²₁ dof
0
5×10⁻³
0.01
0.015
0.02
Expected Background Subtracted Count Rate (cts/s)

$\rho_S = 7.68^{+1.03}_{-.97}$ e-3 cts/s



grazie per l’attenzione

buone vacanze!