

70%

71%

99%

100%

Il reverendo Bayes è tuo amico (anche se...)

Emmanuele Somma
esomma@ieee.org



PYCON CINQUE

Firenze — Towel Day 2014

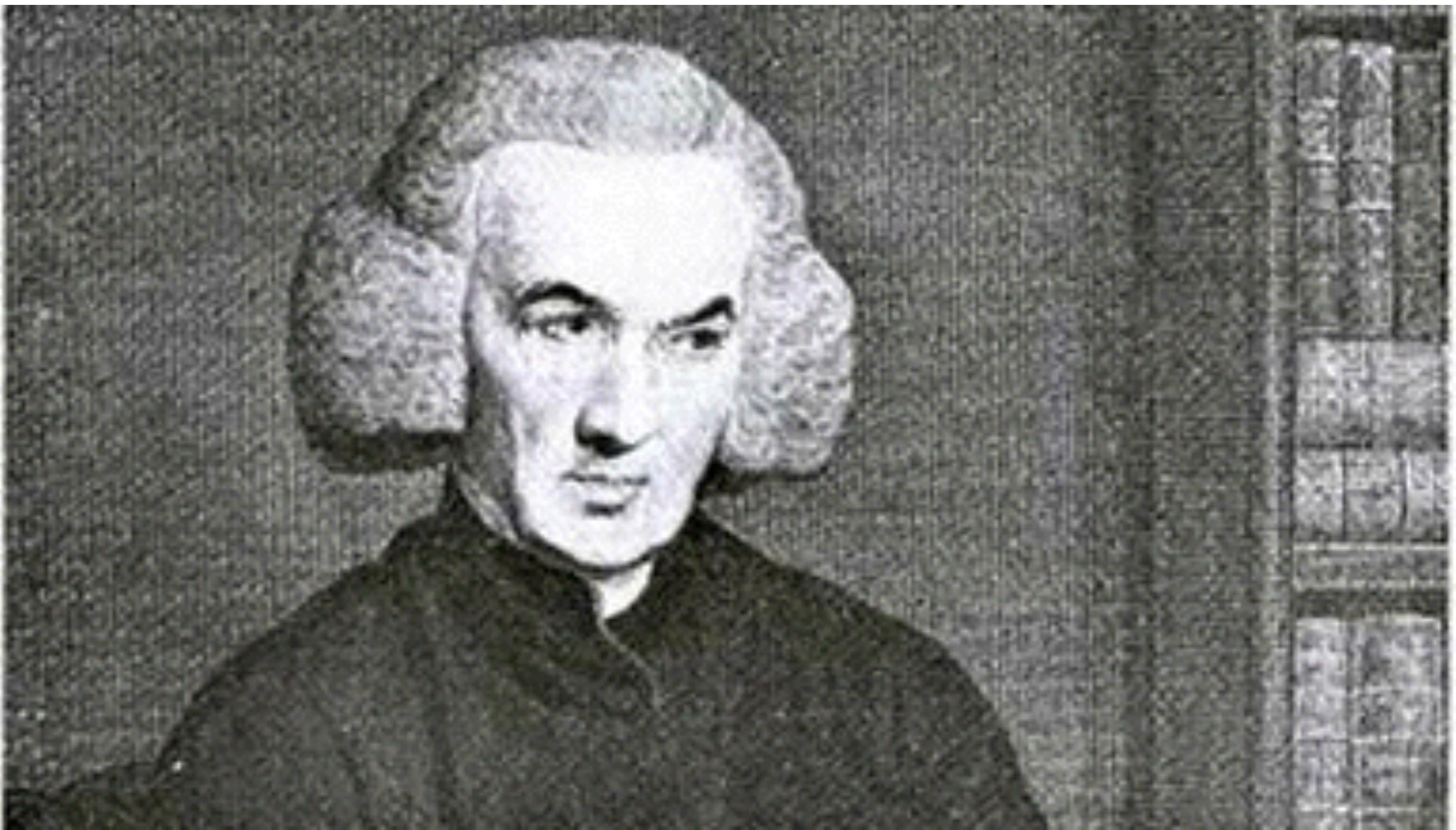


Londra, 1702 – Royal Tunbridge Wells, 17 aprile 1761

- Divine Benevolence, or an Attempt to Prove That the Principal End of the Divine Providence and Government is the Happiness of His Creatures (1731)
- An Introduction to the Doctrine of Fluxions, and a Defence of the Mathematicians Against the Objections of the Author of the Analyst (1736)

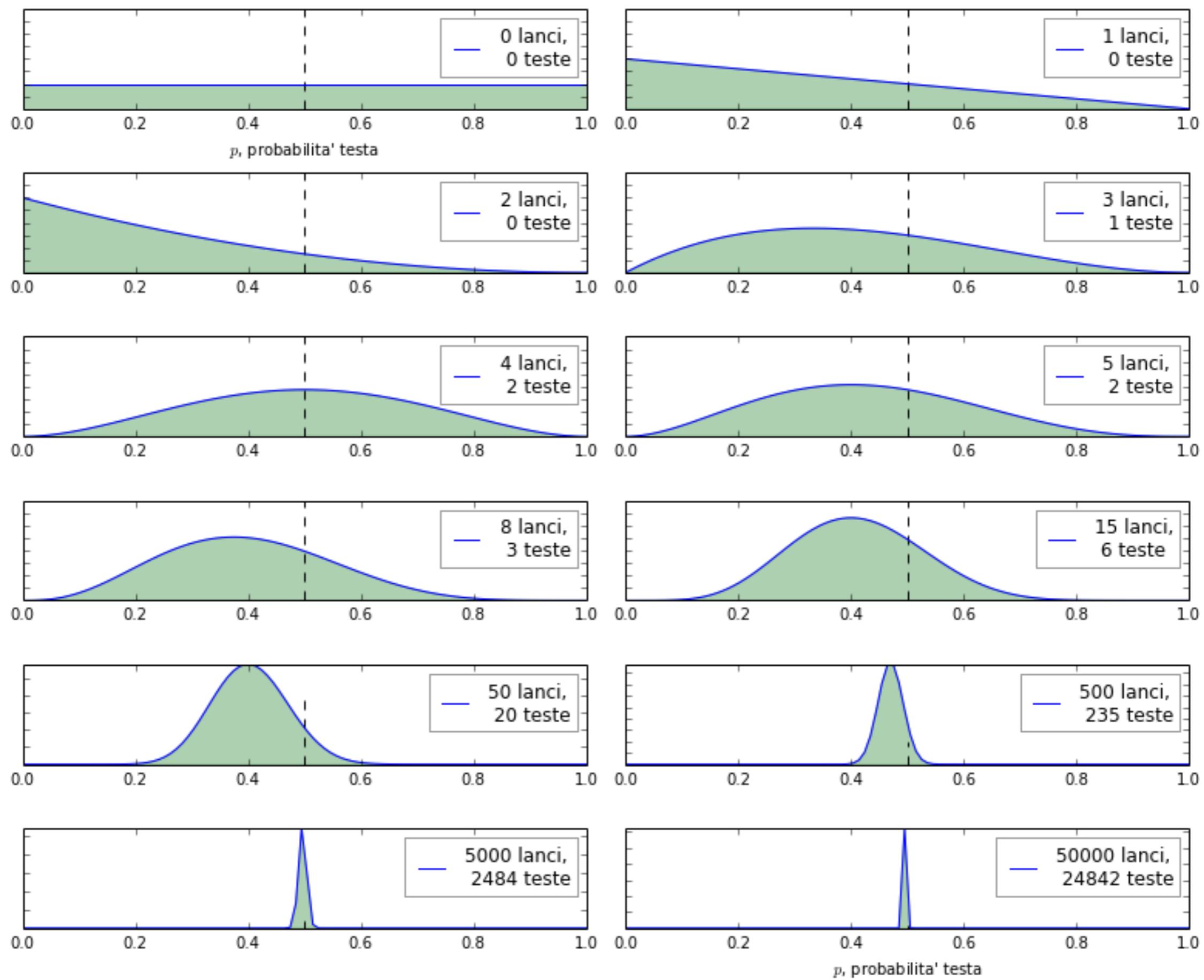
T. Bayes.

Richard Price (23 February 1723 – 19 April 1791)





probabilita' a posteriori bayesiane



$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Non ci sono bug nel programma se tutti i test passano?

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Non ci sono bug nel programma se tutti i test passano?

I programmi sono corretti?

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Non ci sono bug nel programma se tutti i test passano?

I programmi sono corretti?

I test passano se il programma è corretto?

Non ci sono bug nel programma se tutti i test passano?

I programmi sono corretti?

I test passano se il programma è corretto?

I test sono corretti?

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

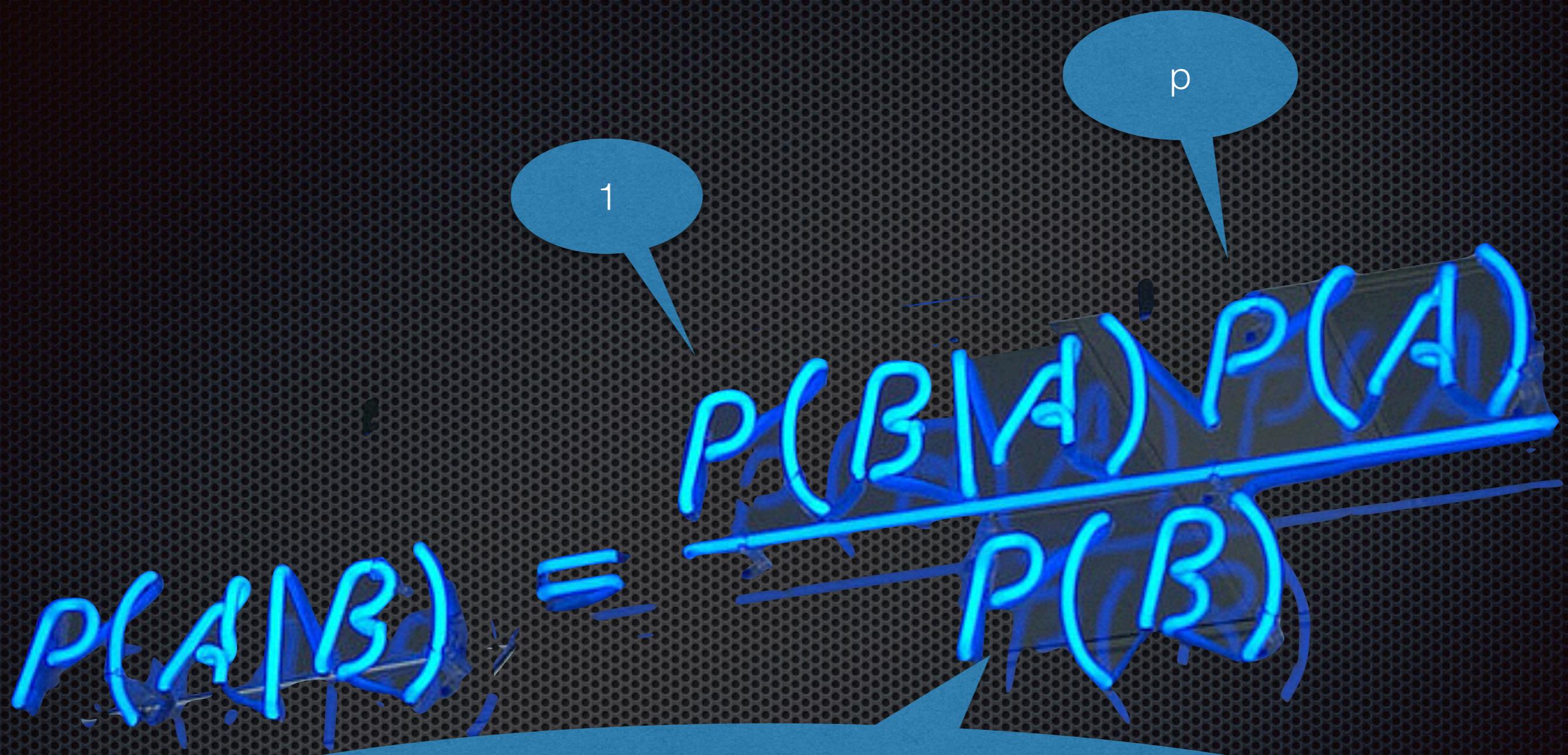
p

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

1

p



$$\begin{aligned} P(A|B) &= \frac{P(B|A)P(A)}{P(B)} \\ &= P(B \text{ and } A) + P(B \text{ and } \sim A) \\ &= P(B|A)P(A) + P(B|\sim A)P(\sim A) \\ &= P(B|A)p + P(B|\sim A)(1-p) \\ &= p + 0,5(1-p) \end{aligned}$$

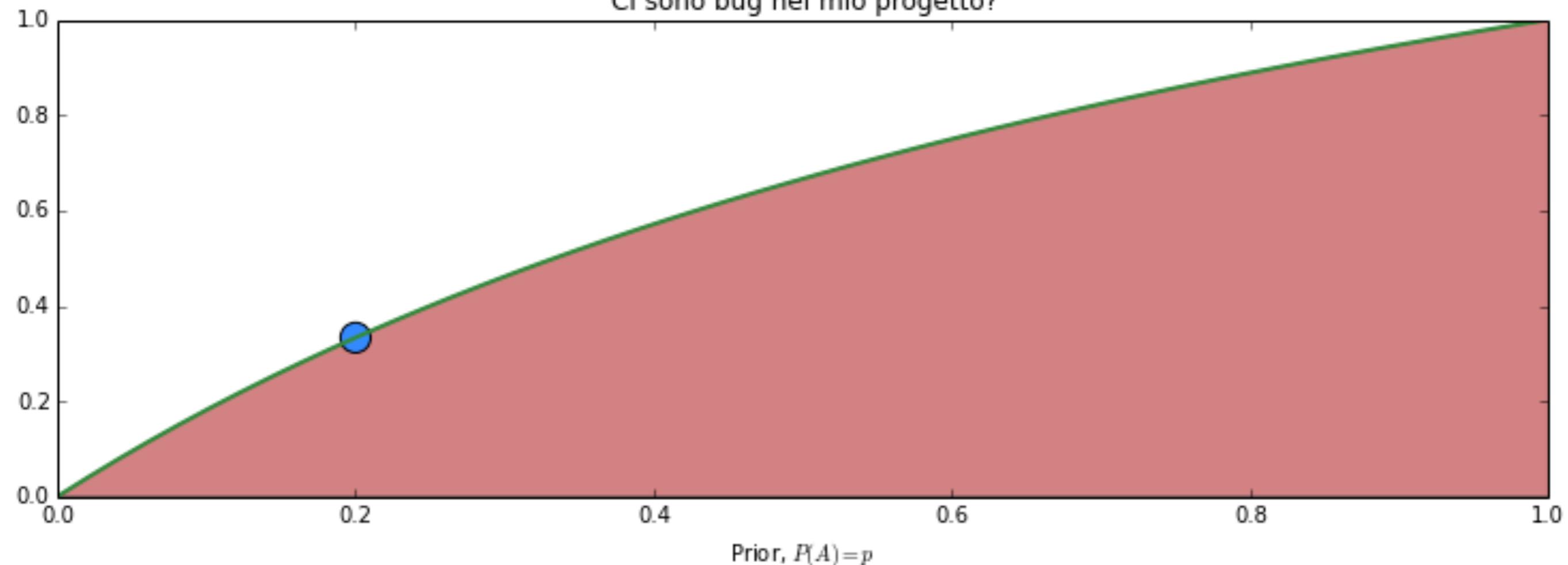
2p

1 + p

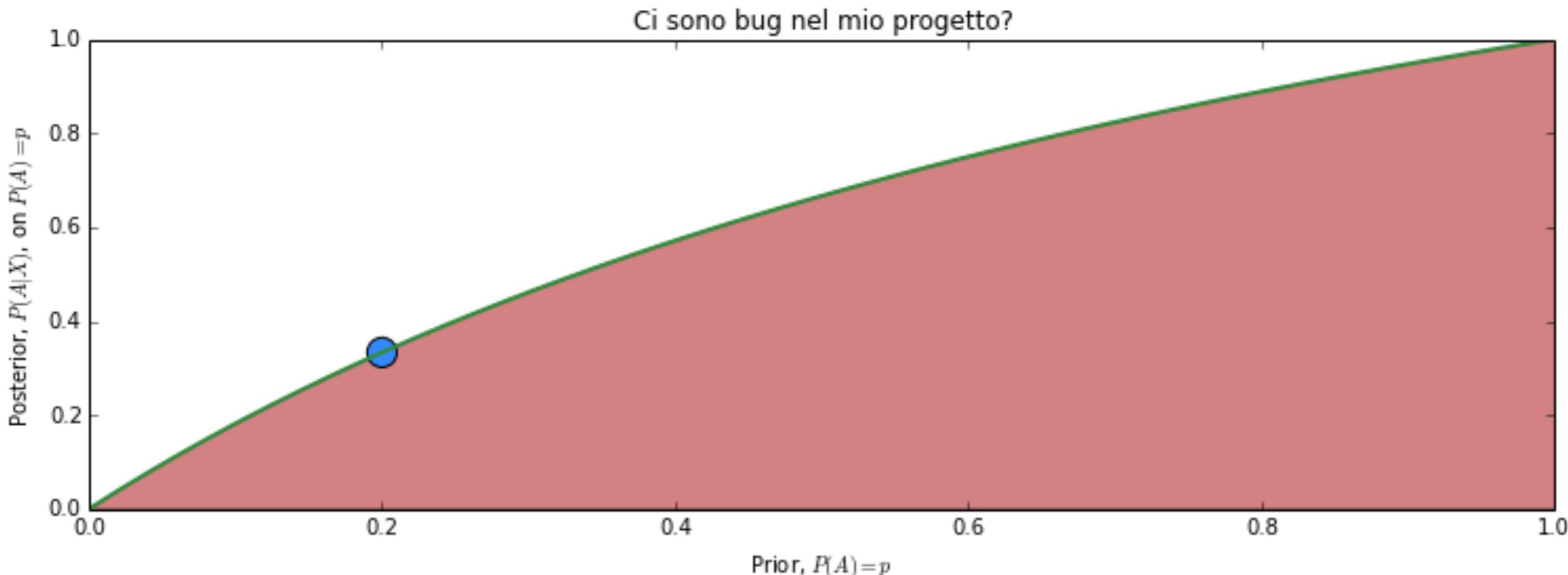
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

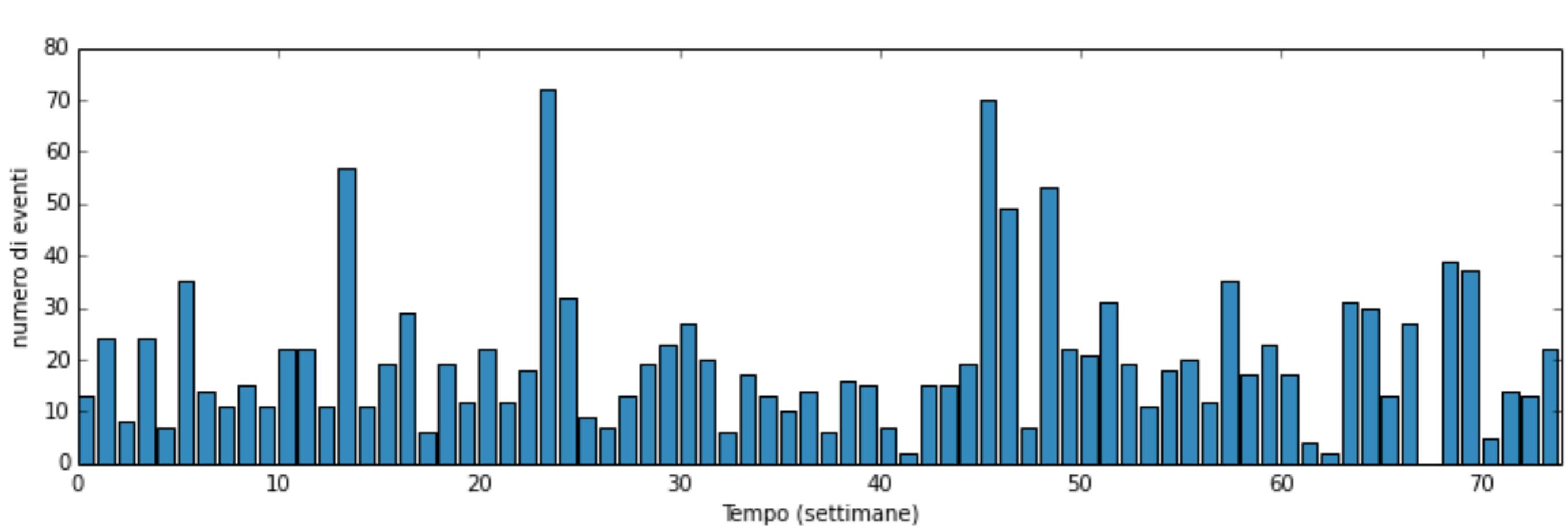
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

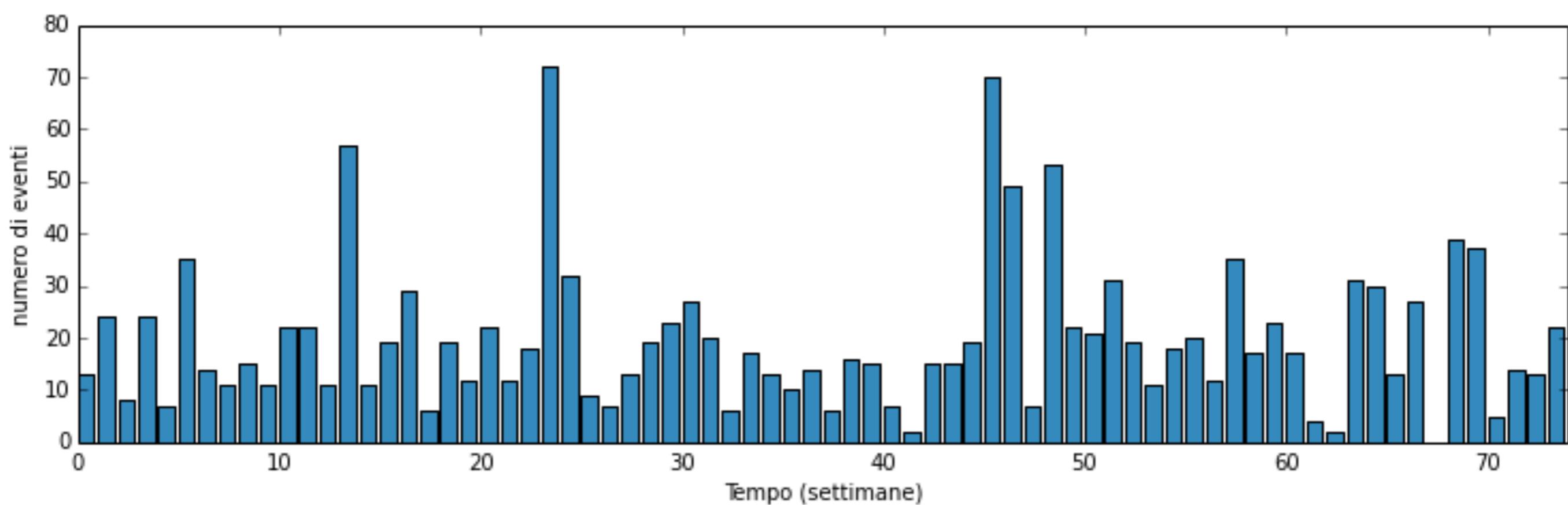
Ci sono bug nel mio progetto?



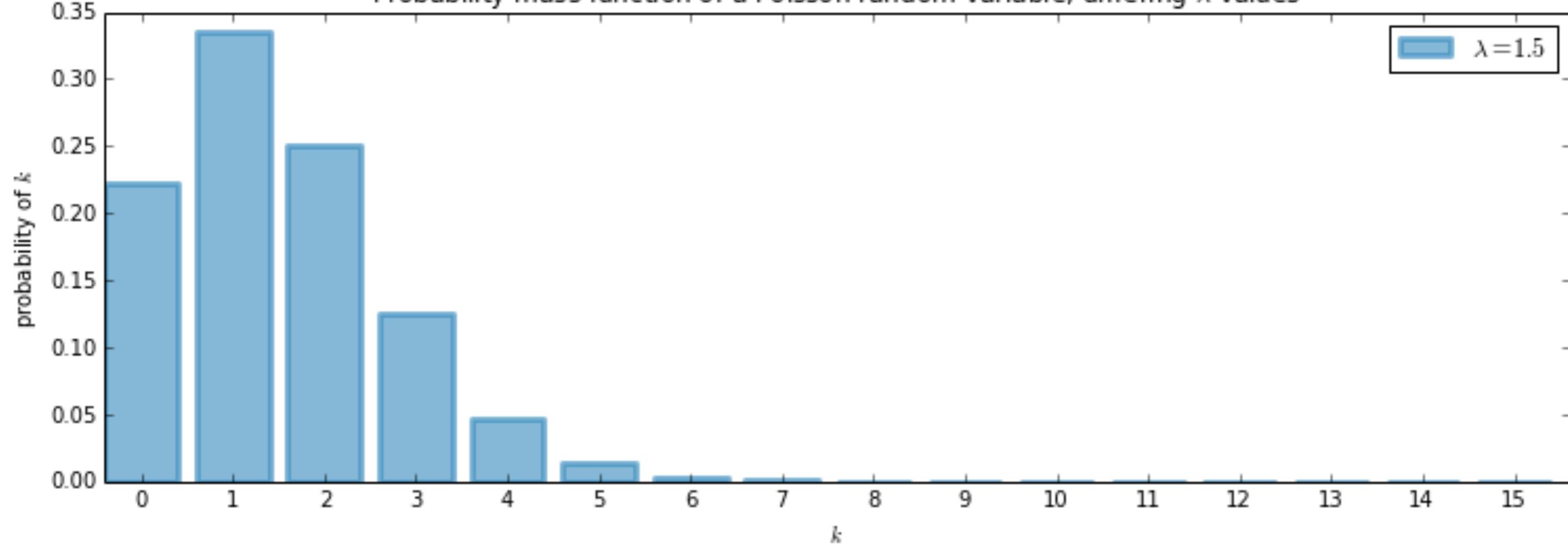
```
%matplotlib inline
from IPython.core.pylabtools import figsize
import numpy as np
from matplotlib import pyplot as plt
figsize(12.5, 4)
p = np.linspace(0, 1, 50)
plt.plot(p, 2 * p / (1 + p), color="#348A3D", lw=2)
plt.fill_between(p, 2*p/(1+p), alpha=.5, facecolor=[ "#A60608"])
plt.scatter(0.2, 2 * (0.2) / 1.2, s=200, c="#348AFD")
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Prior, $P(A) = p$")
plt.ylabel("Posterior, $P(A|X)$, su $P(A) = p$")
plt.title("Ci sono bug nel mio progetto?")
```

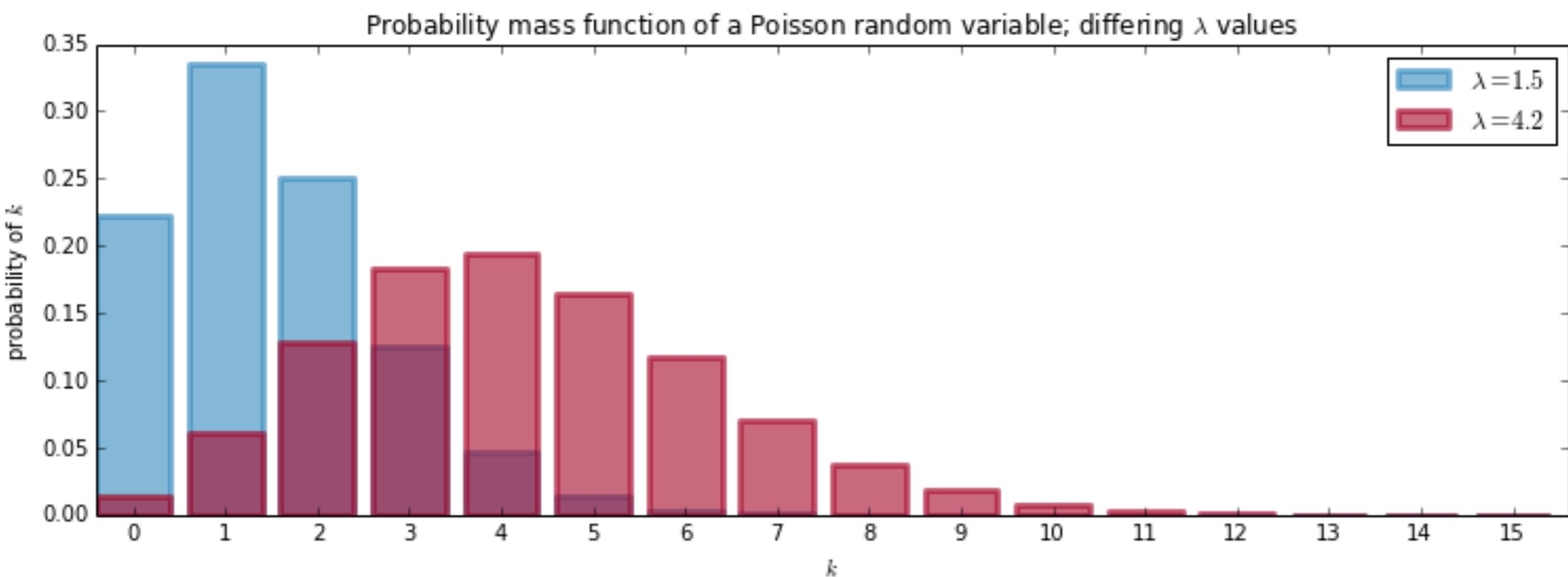
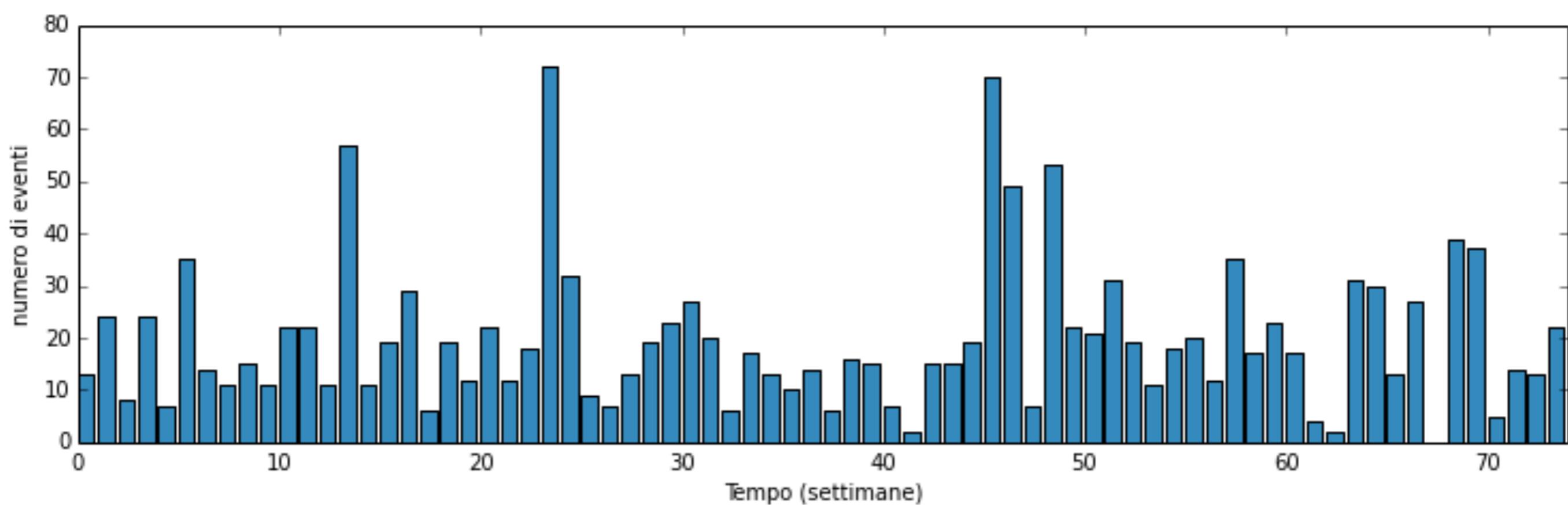


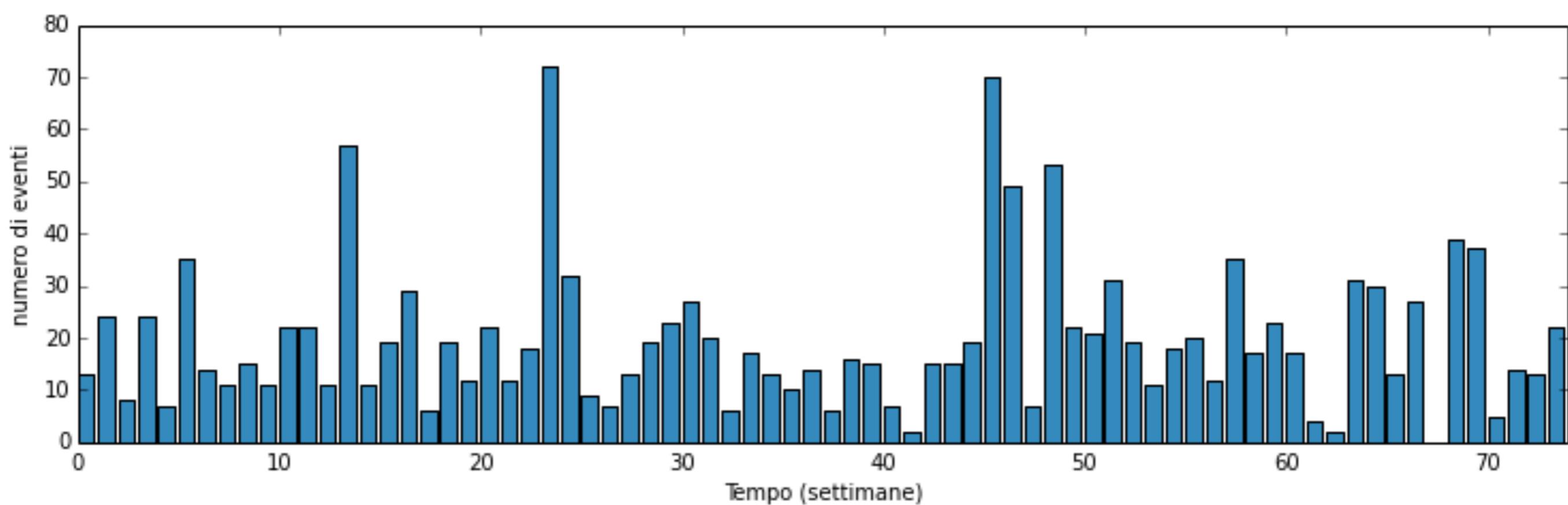




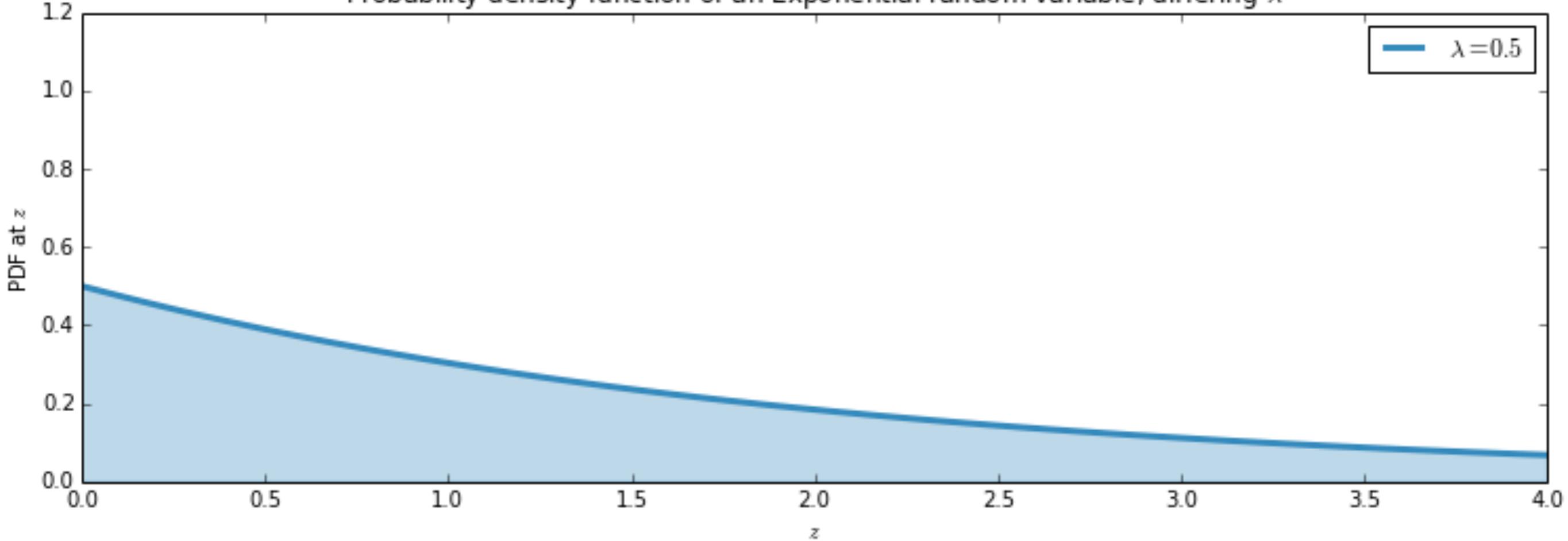
Probability mass function of a Poisson random variable; differing λ values

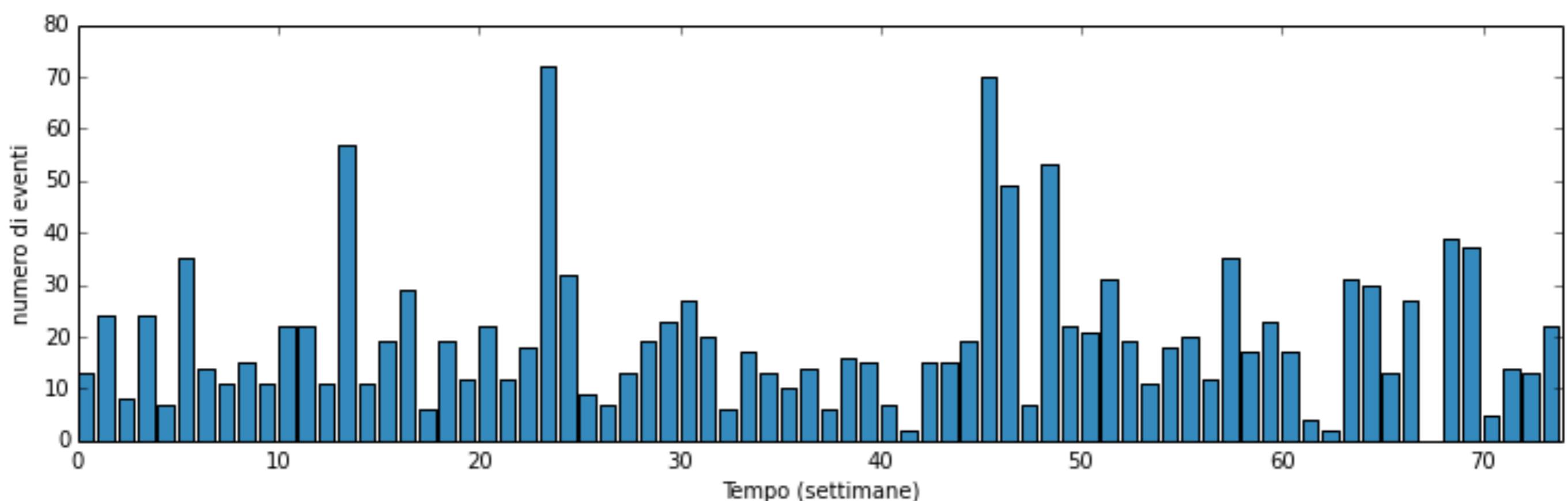




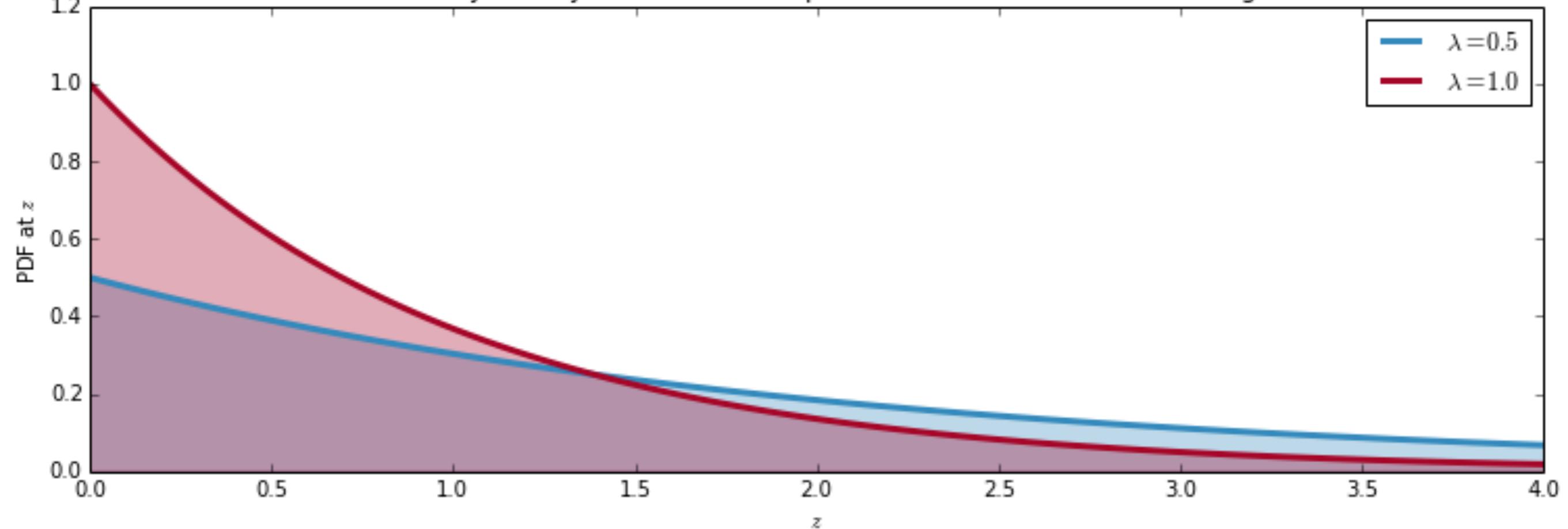


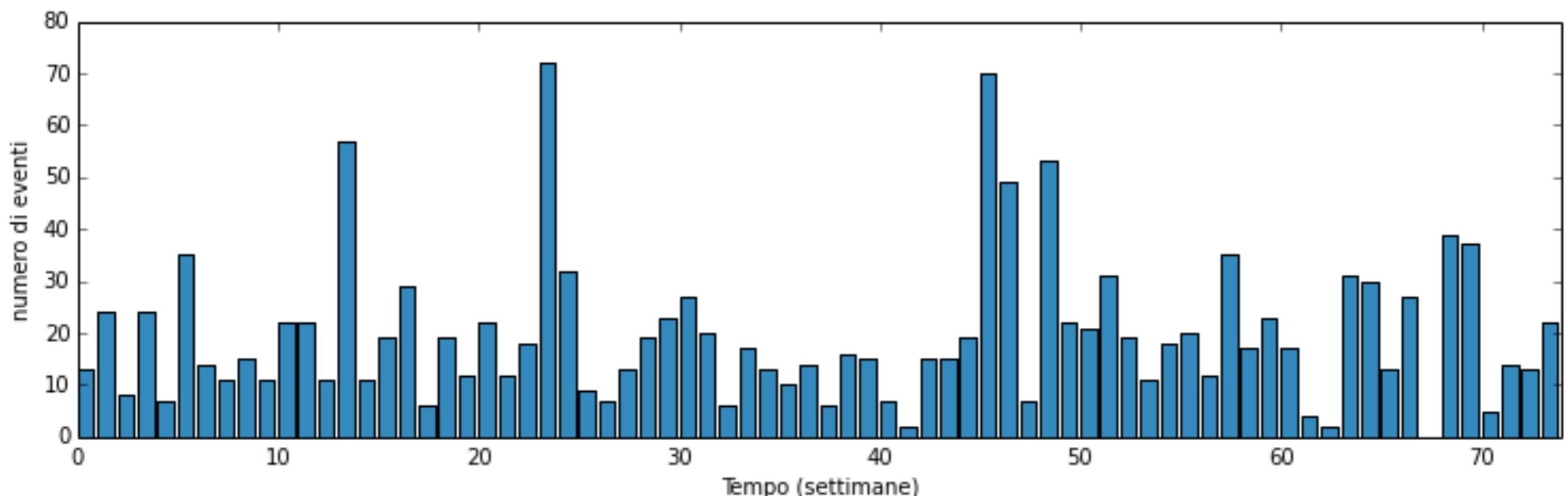
Probability density function of an Exponential random variable; differing λ

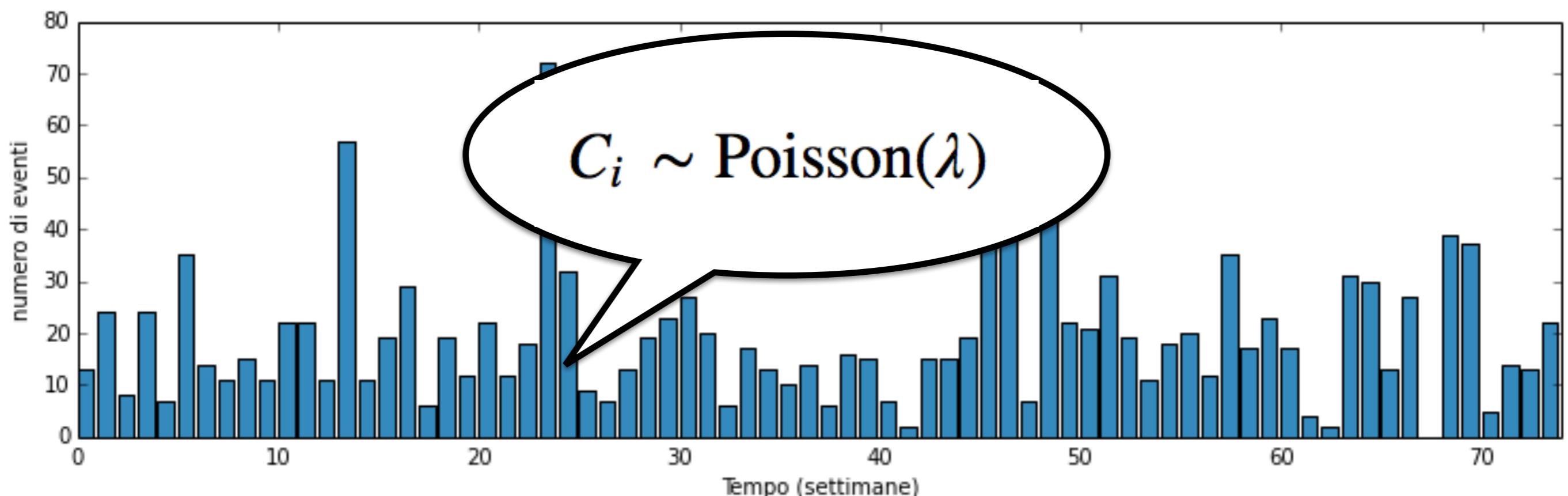


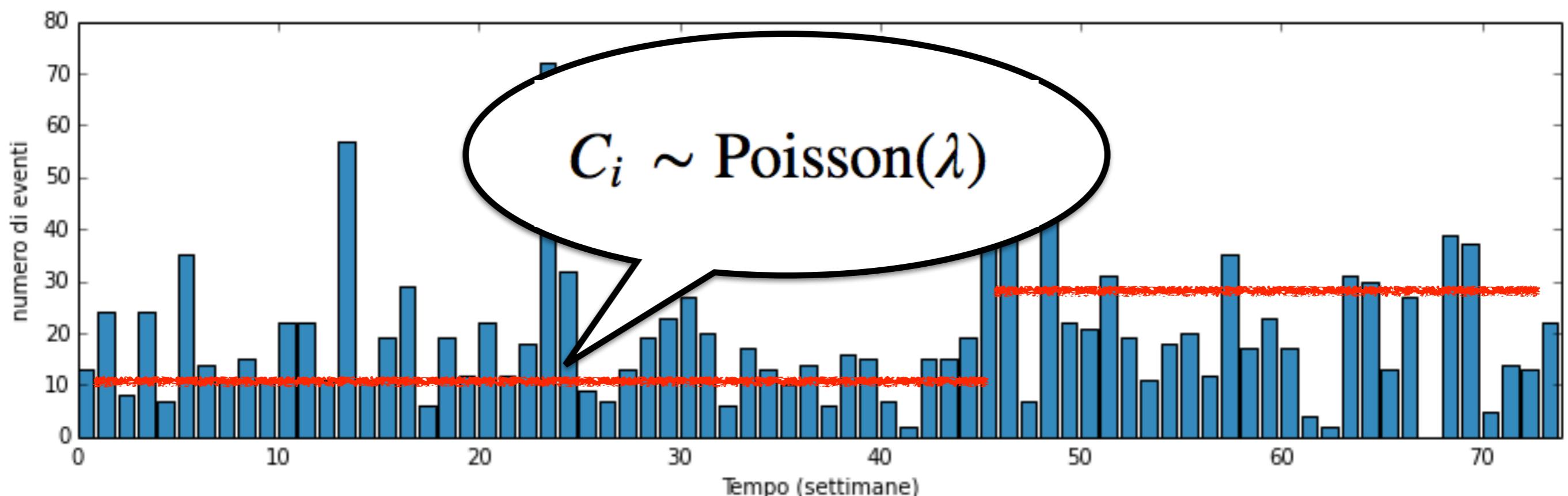


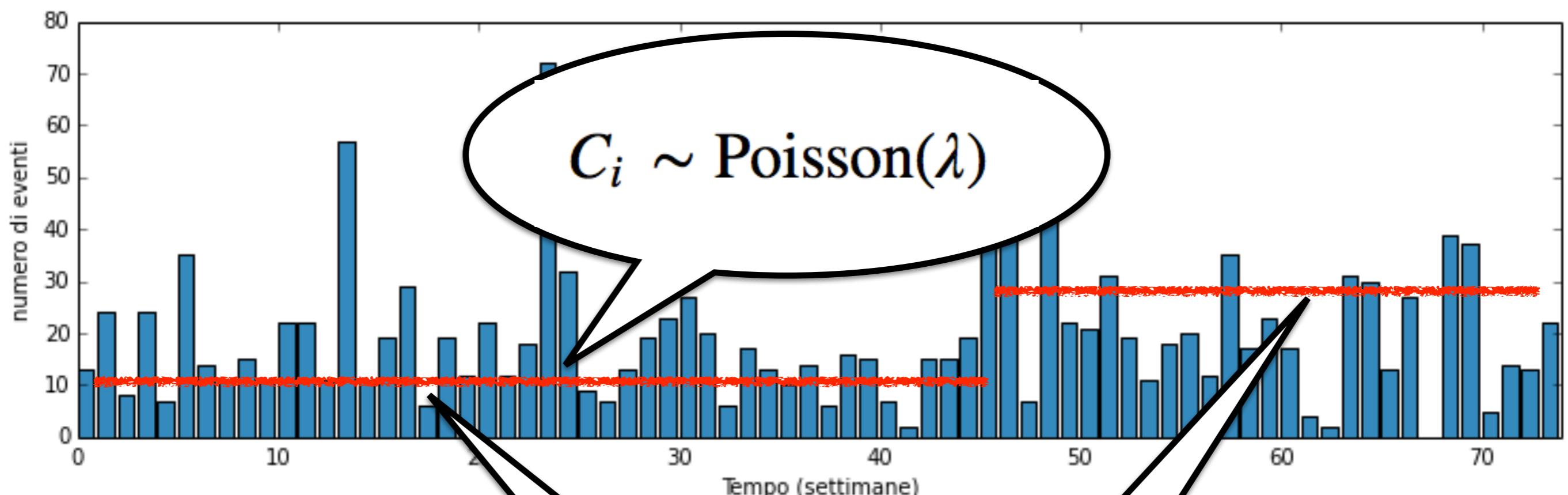
Probability density function of an Exponential random variable; differing λ



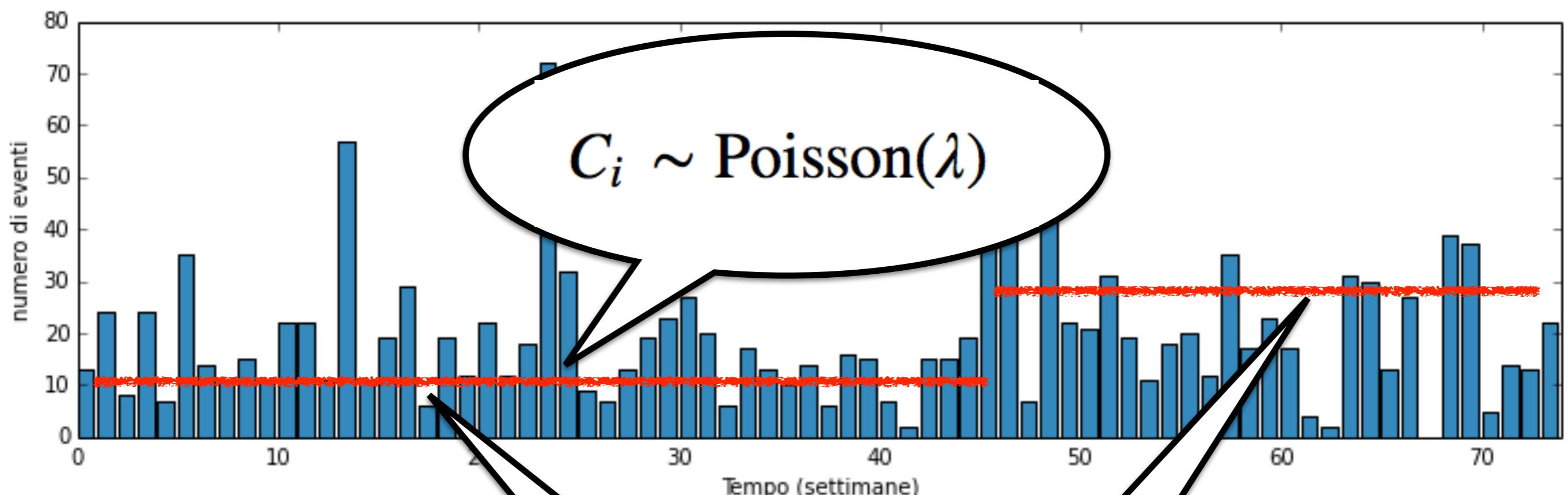






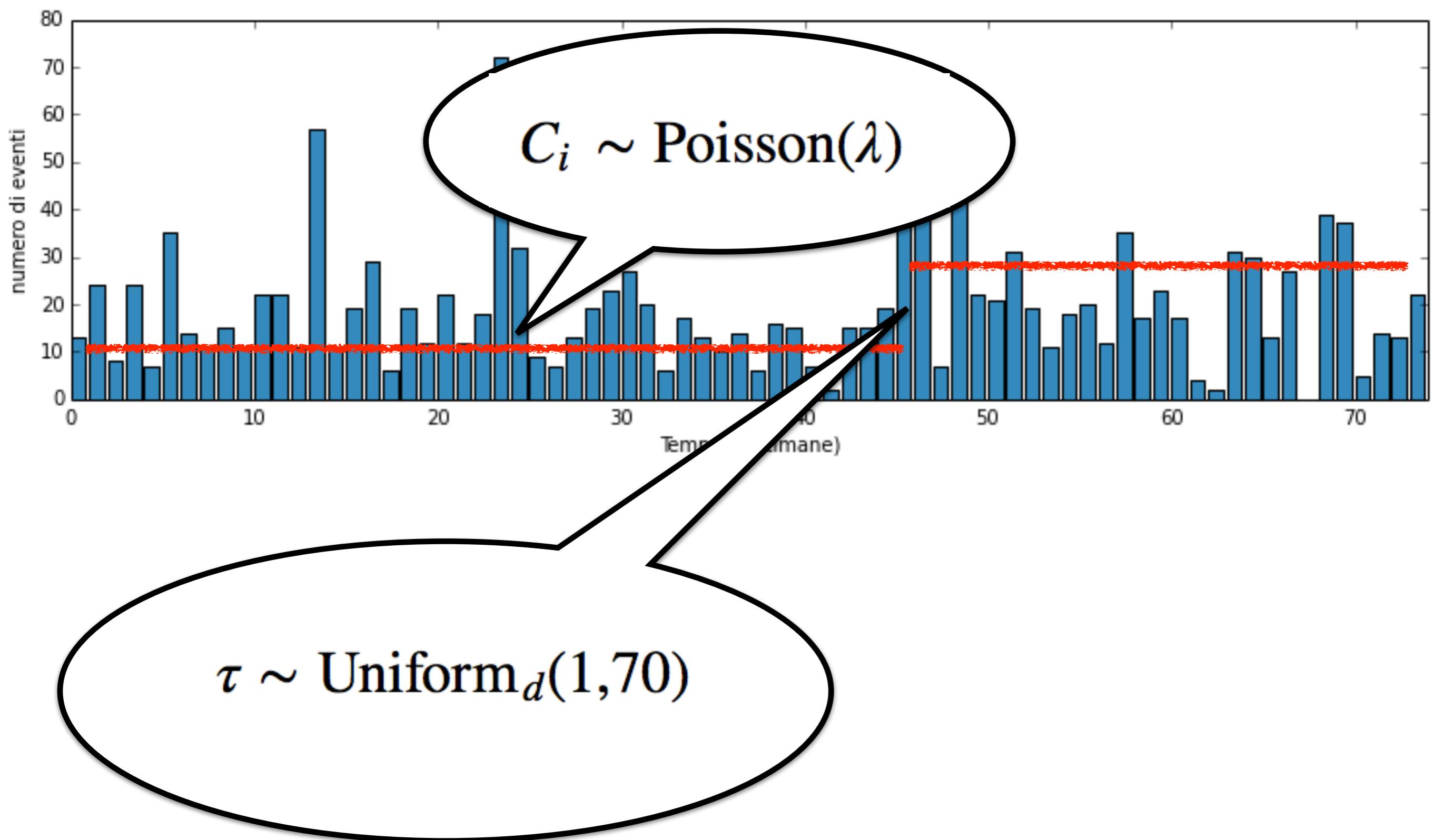


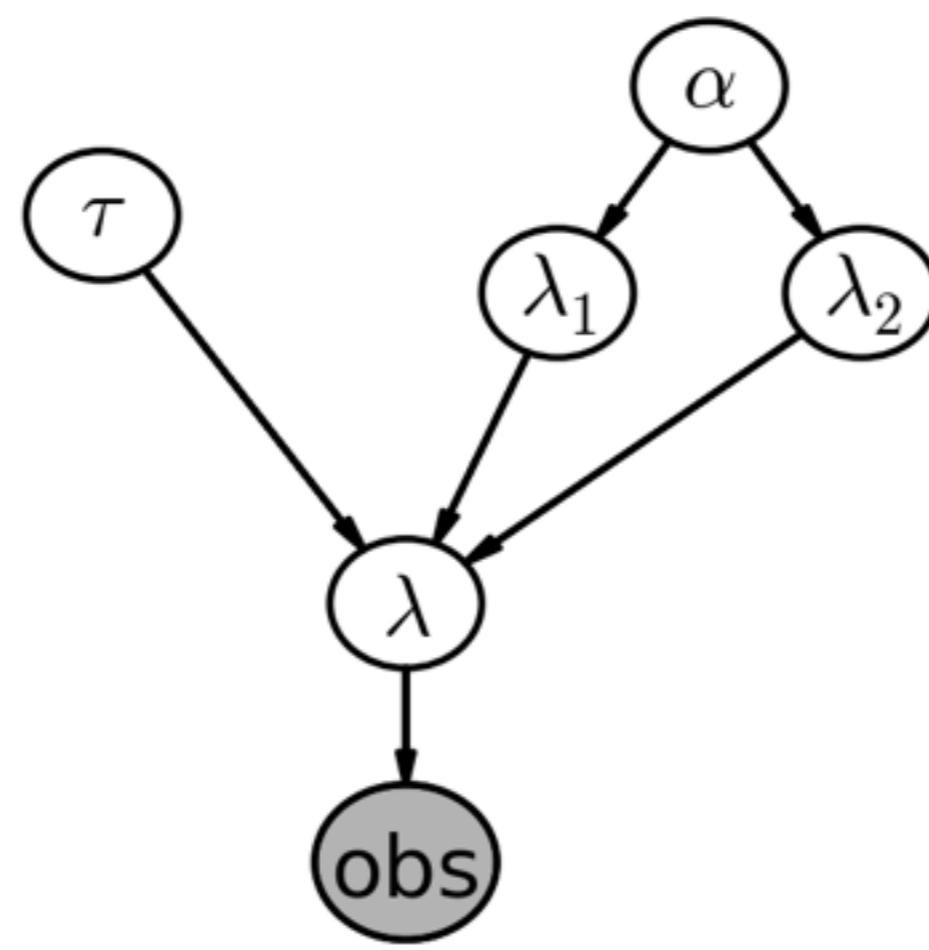
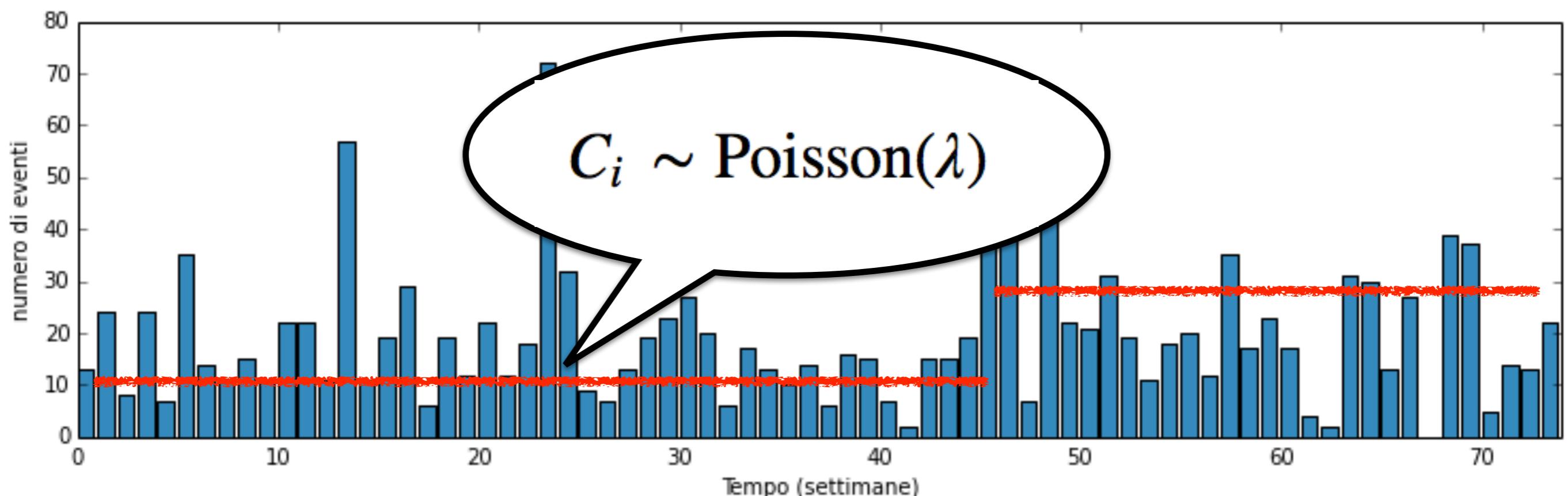
$$\lambda = \begin{cases} \lambda_1 & \text{if } t < \tau \\ \lambda_2 & \text{if } t \geq \tau \end{cases}$$

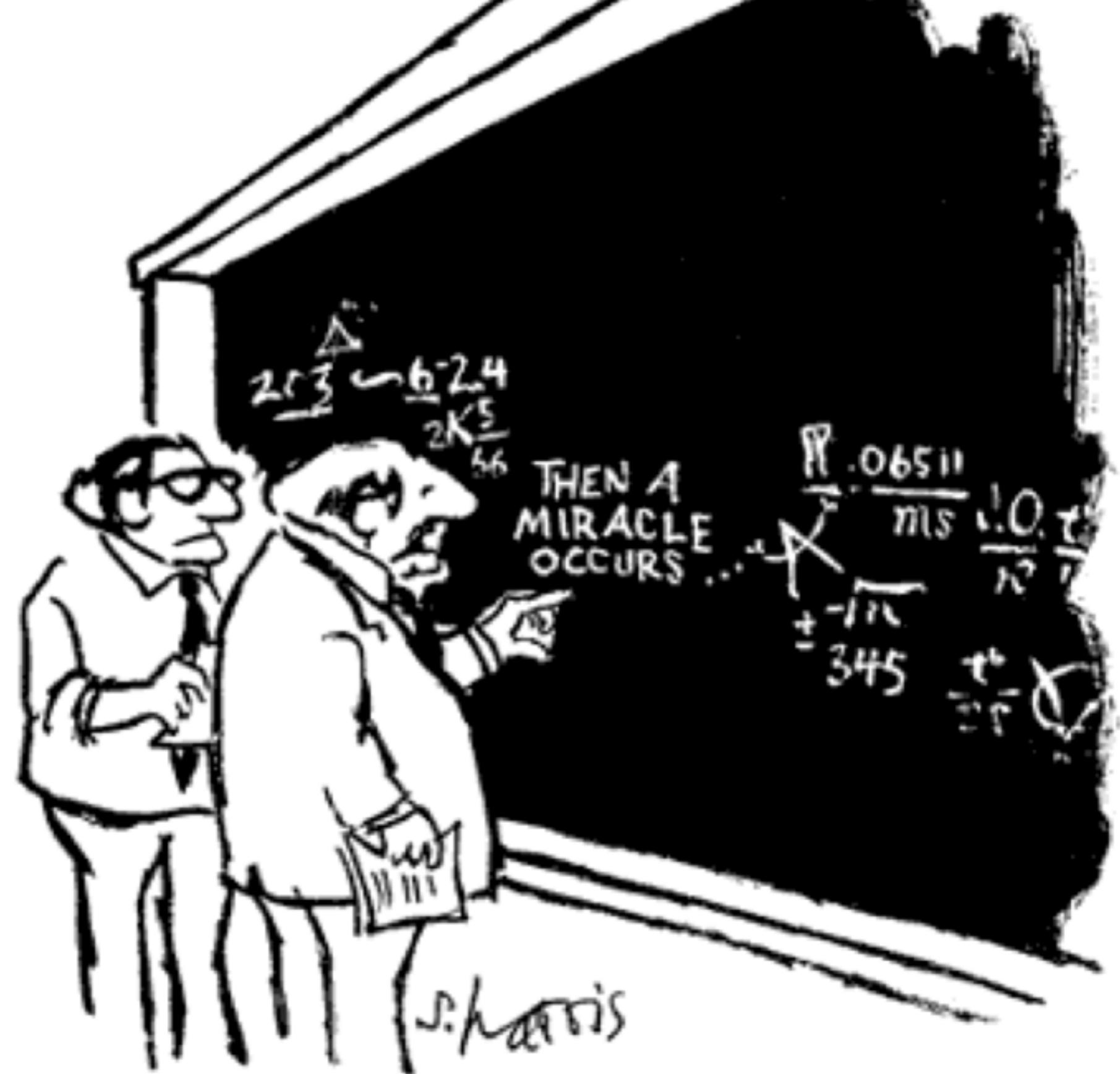


An oval annotation contains the following text:

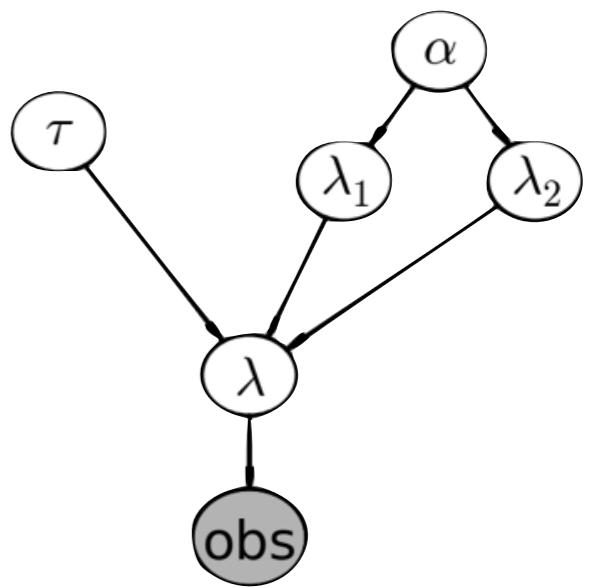
$$\lambda_1 \sim \text{Exp}(\alpha)$$
$$\lambda_2 \sim \text{Exp}(\alpha)$$







"I think you should be more explicit here in step two."



```

import pymc as pm
import numpy as np

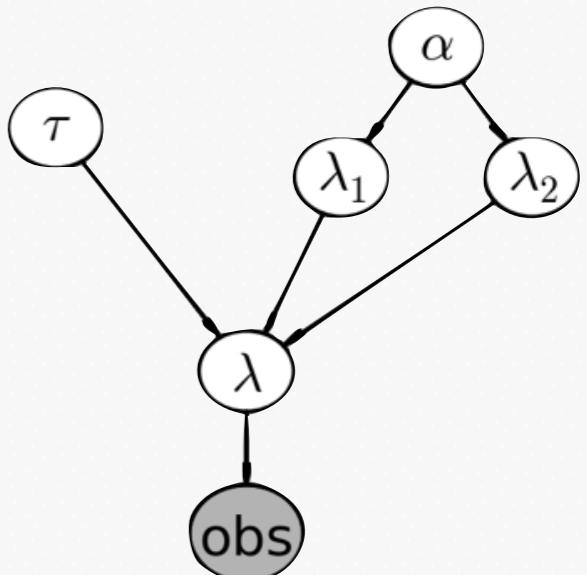
count_data = np.loadtxt("data/txtdata.csv")
n_count_data = len(count_data)

alpha = 1.0 / count_data.mean()

lambda_1 = pm.Exponential("lambda_1", alpha)
lambda_2 = pm.Exponential("lambda_2", alpha)

tau = pm.DiscreteUniform("tau", lower=0, upper=n_count_data)

```



```

import pymc as pm
import numpy as np

count_data = np.loadtxt("data/txtdata.csv")
n_count_data = len(count_data)

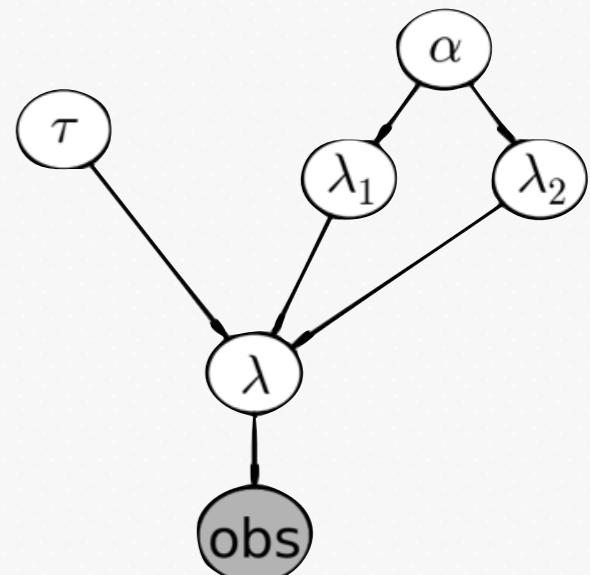
alpha = 1.0 / count_data.mean()

lambda_1 = pm.Exponential("lambda_1", alpha)
lambda_2 = pm.Exponential("lambda_2", alpha)

tau = pm.DiscreteUniform("tau", lower=0, upper=n_count_data)

@pm.deterministic
def lambda_(tau=tau, lambda_1=lambda_1, lambda_2=lambda_2):
    out = np.zeros(n_count_data)
    out[:tau] = lambda_1
    out[tau:] = lambda_2
    return out

```



```

import pymc as pm
import numpy as np

count_data = np.loadtxt("data/txtdata.csv")
n_count_data = len(count_data)

alpha = 1.0 / count_data.mean()

lambda_1 = pm.Exponential("lambda_1", alpha)
lambda_2 = pm.Exponential("lambda_2", alpha)

tau = pm.DiscreteUniform("tau", lower=0, upper=n_count_data)

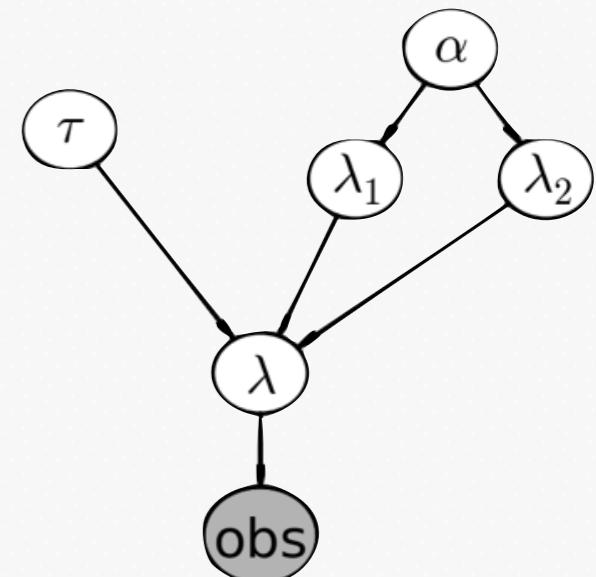
```

```

@pm.deterministic
def lambda_(tau=tau, lambda_1=lambda_1, lambda_2=lambda_2):
    out = np.zeros(n_count_data)
    out[:tau] = lambda_1
    out[tau:] = lambda_2
    return out

observation = pm.Poisson("obs", lambda_, value=count_data, observed=True)
model = pm.Model([observation, lambda_1, lambda_2, tau])

```



```

import pymc as pm
import numpy as np

count_data = np.loadtxt("data/txtdata.csv")
n_count_data = len(count_data)

alpha = 1.0 / count_data.mean()

lambda_1 = pm.Exponential("lambda_1", alpha)
lambda_2 = pm.Exponential("lambda_2", alpha)

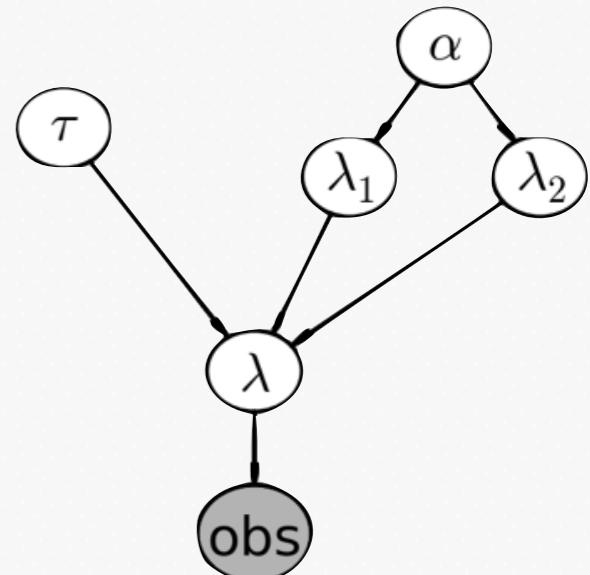
tau = pm.DiscreteUniform("tau", lower=0, upper=n_count_data)

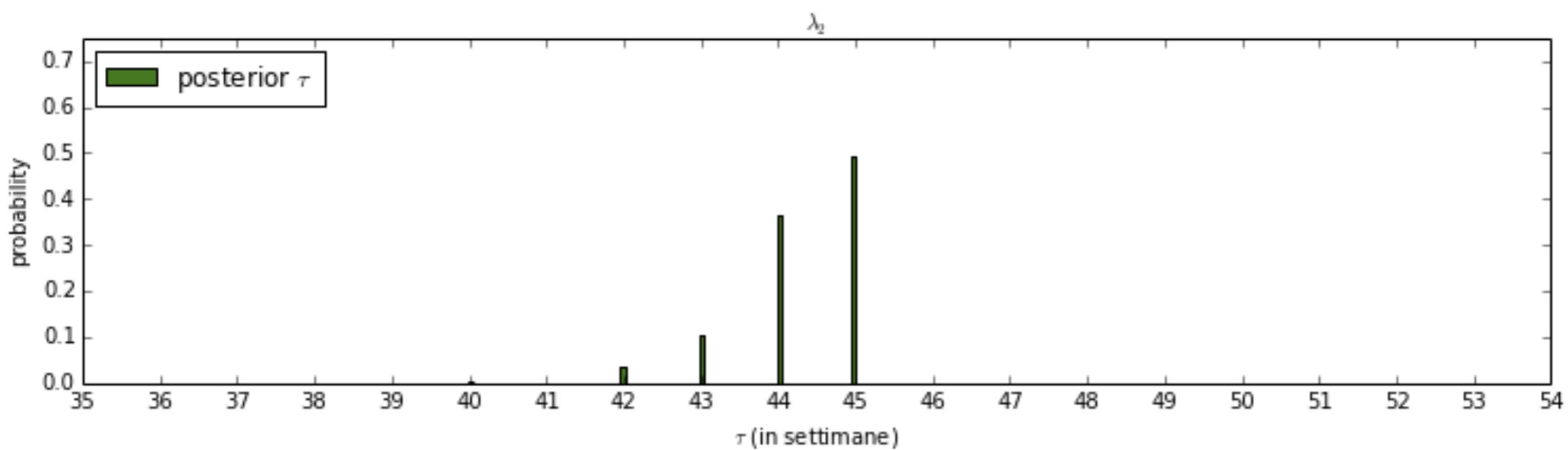
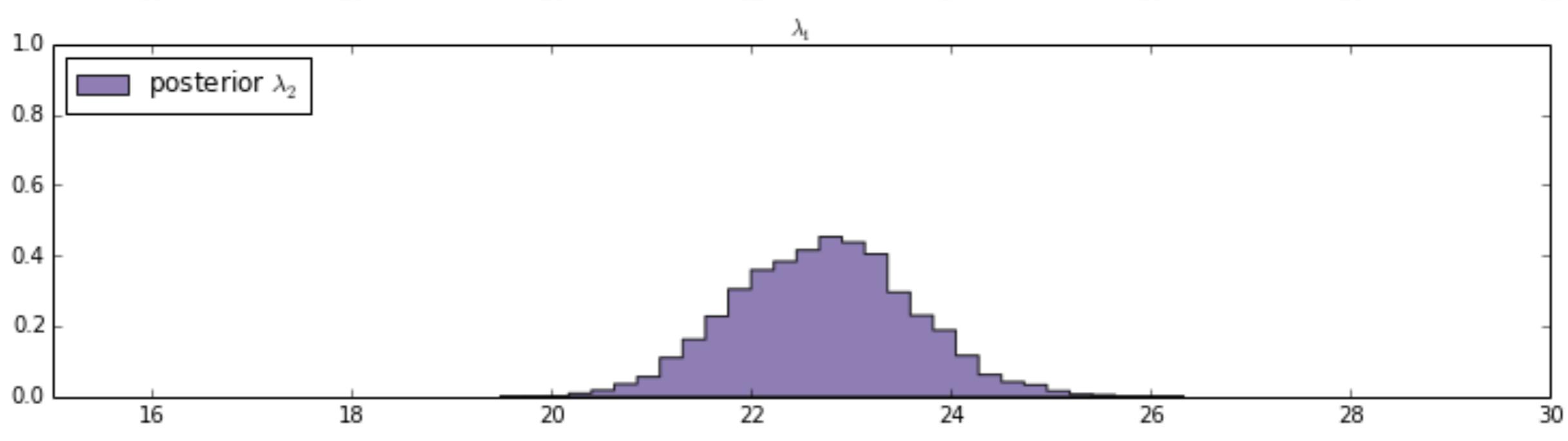
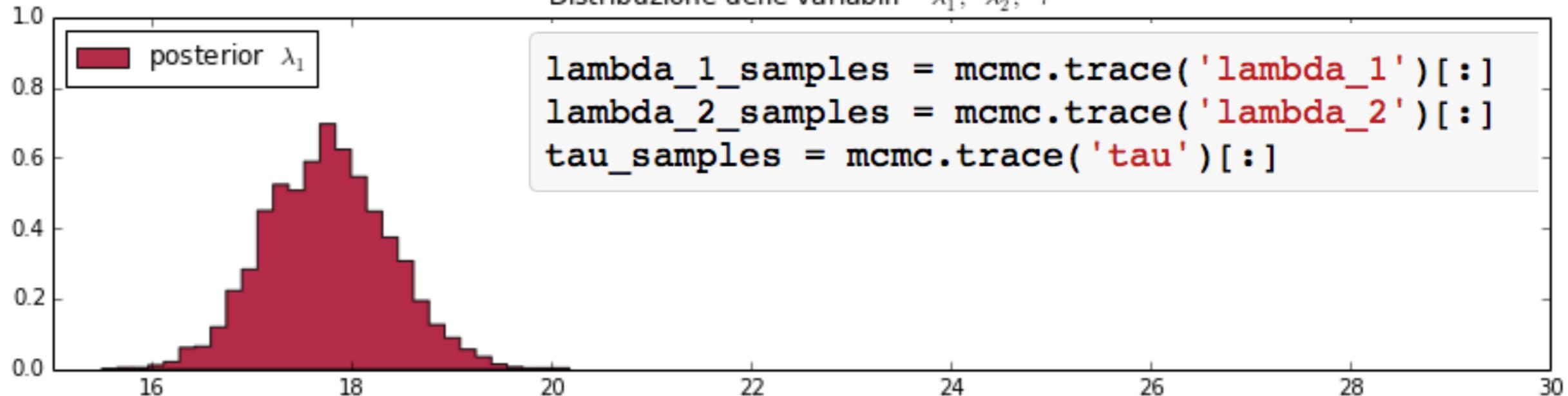
@pm.deterministic
def lambda_(tau=tau, lambda_1=lambda_1, lambda_2=lambda_2):
    out = np.zeros(n_count_data)
    out[:tau] = lambda_1
    out[tau:] = lambda_2
    return out

observation = pm.Poisson("obs", lambda_, value=count_data, observed=True)
model = pm.Model([observation, lambda_1, lambda_2, tau])

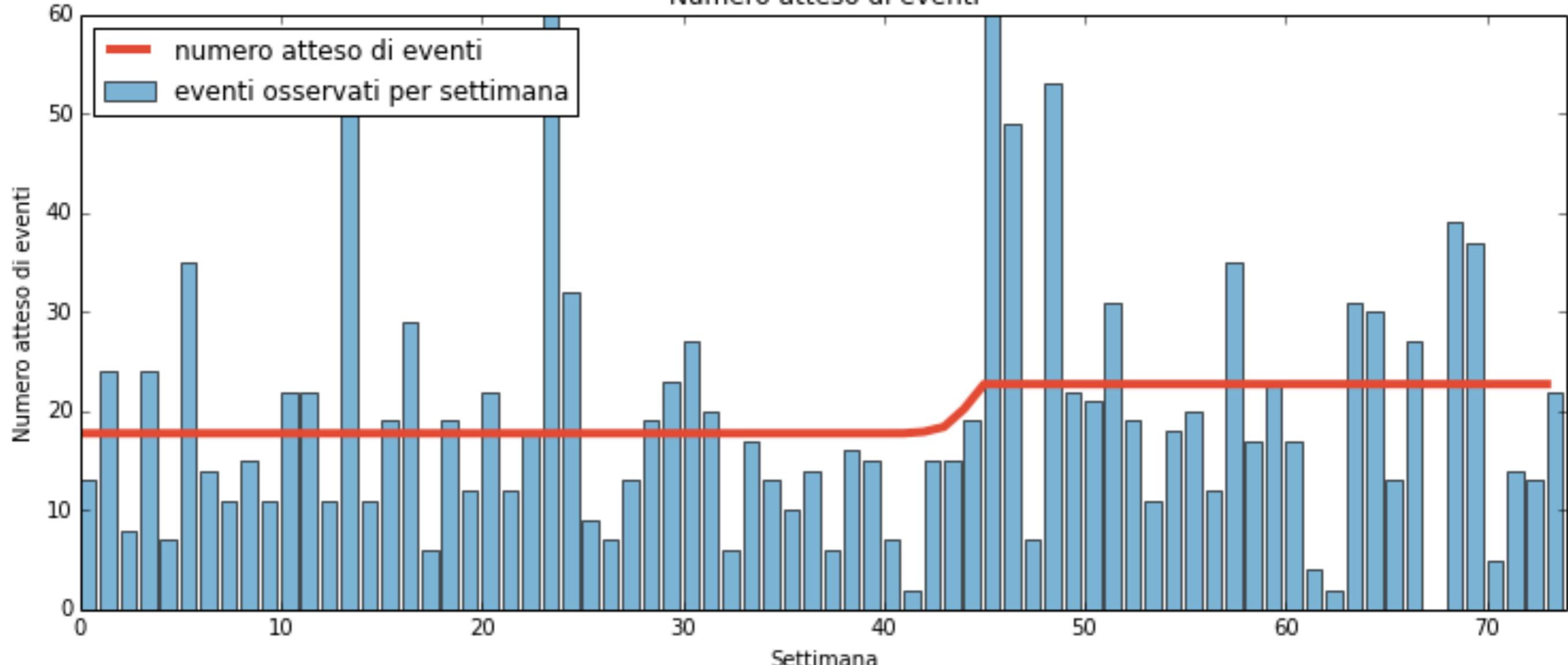
mcmc = pm.MCMC(model)
mcmc.sample(40000, 10000, 1)

```



Distribuzione delle variabili λ_1 , λ_2 , τ 

Numero atteso di eventi



$$\lambda = \begin{cases} \lambda_1 & = 17.7545366 \\ \lambda_2 & = 22.71789065 \end{cases}$$

Think Bayes

Bayesian Statistics Made Simple

Allen B. Downey

Bayesian Statistics in Python

Think Bayes



O'REILLY®

Allen B. Downey

<http://www.greenteapress.com/thinkbayes/>

Bayesian Methods for Hackers

Cam Davidson-Pilon



<http://camdavidsonpilon.github.io/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/>

Grazie



esomma@ieee.org