Groups:

Group 1:
Amelia Camino
Jake Magee
Amanda Muratore
Julissa Sarmiento

Group 2:
Finian Ashmead
Francis Burk
Mykola Chernyashevskyy
Mohamed Ismail
Ehteshamul Karim

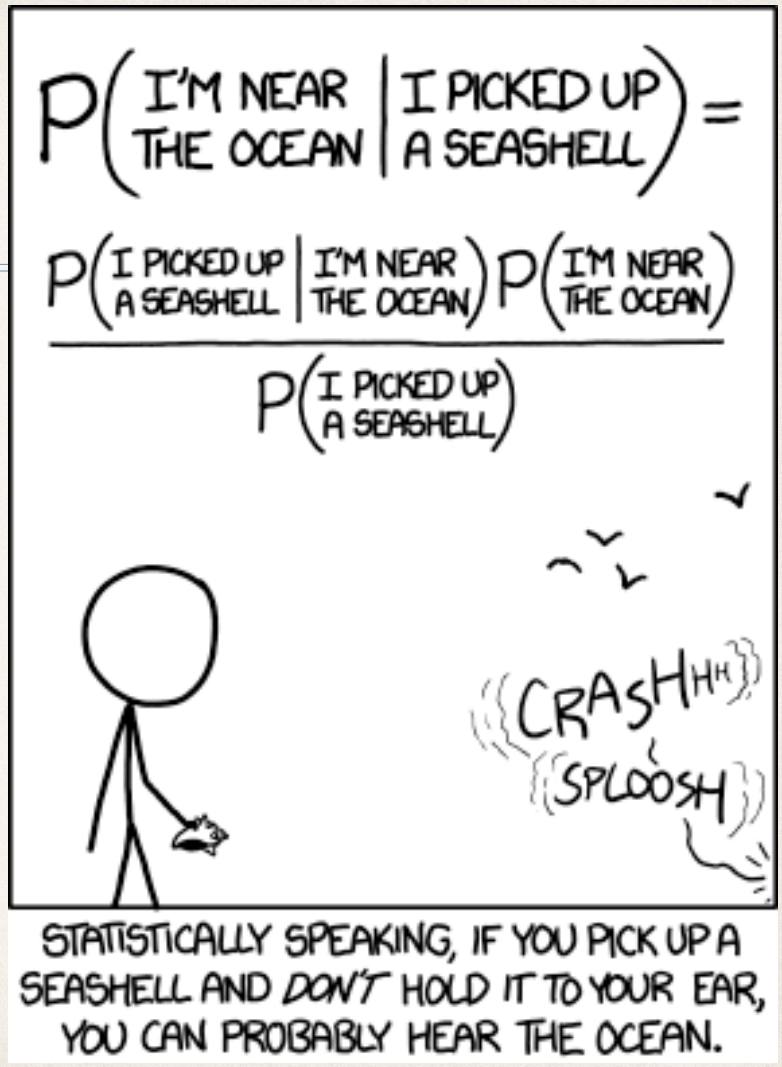
Group 3:
Cullen Abelson
Alia Dawood
Mira Salman
Yunchong Zhang

Group 4:
Nathalie Chicoine
Lauren Elicker
Yoki Salcedo
Marcos Tamargo-Arizmendi

Probability Distributions

Statistics and Data Science

Spring 2025



http://xkcd.com/1236/

Goals for today: you should be able to...

- **Lecture 6 notebook**: Perform Bayesian statistical inference with the binomial distribution
- Use np.where() and np.isfinite()
- Choose priors for a binomial distribution
- **Lecture 7 / 8 notebook: Use the Poisson distribution to describe real situations**

Review: the Binomial distribution

- In general, if there is a probability p of success, and we do N trials, then the probability of M successes is described by a binomial distribution:

$prob(M\ successes) = C(N,M)\ p^M\ (1-p)^{N-M}$

- The binomial distribution has mean $\mu = N\ p$ and variance $\sigma^2 = N\ p\ (1-p)$
- Non-informative priors:
  - Haldane's: $prob(p) \propto p^{-1}(1-p)^{-1}$
  - Jeffreys': $prob(p) \propto p^{-1/2}(1-p)^{-1/2}$
  - Uniform: $prob(p) \propto p^0$
- We will apply Bayes' theorem:
  - $prob(parameters \mid data) = prob(data \mid parameters)\ prob(parameters) / prob(data)$
  - i.e.: the ***posterior*** probability is equal to the ***likelihood*** times the ***prior***, divided by the ***evidence***

What difference does the prior make?

- Suppose you observe a set of 8 different early-type (red sequence / non-star-forming / elliptical/S0) galaxies and observe AGN signatures from three of them. What can we conclude about the probability p that a randomly chosen early-type galaxy has an AGN?
- The likelihood $prob(observation \mid p)$ should correspond to a binomial distribution, with 8 trials and 3 coming up 'AGN': $\propto p^3(1-p)^5$
- For the prior, $prob(p)$, we'll try all three possibilities for a binomial prior

$$prob(p \mid observation) = prob(observation \mid p)\, prob(p) / prob(observation)$$

Plotting a distribution in Python

- In looking at coin flips or dice rolls, we plotted functions of N, where N took whole-number values.
- Instead we want to consider values of the unknown probability p at equally-spaced values between 0 and 1, inclusive.

```
p=np.linspace(0.,1.,501)
```

- Ultimately, we want to plot up the ***probability distribution function*** for p.

What does prob(observation | p) look like?

```
likelihood=p**3 * (1-p)**5
plt.plot(likelihood)
```

- Now let's set up our priors: Haldane, Jeffreys, and uniform

```
prior_h=1/p/(1-p)
prior_j=np.sqrt(prior_h)
prior_u=p*0.+1.
```

- These are plotted in the notebook. Which one would you expect to have the greatest impact on the posterior probability density function?

So what does prob(p | observation) look like?

- Now, we can make plots of the relative probability of each value of *p*, as a function of *p:* our posterior distribution for *p*
- Use a y range from 0 to 0.03, and plot the likelihood and the posterior assuming each different prior, using different lines or plot symbols for each.

Normalizing probabilities

- Which of these was most strongly peaked?

```
plt.plot(p,likelihood*prior_h,'b-')
plt.plot(p,likelihood*prior_j,'r--')
plt.plot(p,likelihood*prior_u,'g-.')
```

- We can tell most easily if we normalize them all to give PDFs - i.e., to have integral 1.

- This will require us to do some calculus -- numerically.

Interpolation and calculus in Python

- We often want to calculate the integral or derivative of some function. There are many ways to do this in Python; a simple recipe:

1) create arrays of values of x and the evaluated function f(x), where f(x) is the function you want to integrate or differentiate.

2) Use `scipy.interpolate.interp1d()` to create a Python function (just like `np.sin(x)`, etc.) that interpolates between the tabulated values of x and f(x). E.g.:

```
import scipy.interpolate as interpol
x = np.linspace(-np.pi,3*np.pi,100)
# we want the interpolation table to extend beyond bounds we will use
x_fine = np.linspace(0,2*np.pi,10_000)
f = np.cos(x)
interp_f = interpol.interp1d(x,f,kind='cubic')
# interp_f is a new Python function!
plt.plot( x_fine,interp_f(x_fine),'r-' )
```

Interpolation and calculus

3) To integrate: one routine is `scipy.integrate.quad(function name, lower limit, upper limit)` . E.g.:

```
import scipy.integrate as integrate
print(f'{integrate.quad(interp_f,0,np.pi/2) = }')
```

- Note that `integrate.quad` returns a tuple (of the integral and its uncertainty) !

- To differentiate: one routine is `scipy.misc.derivative(function name,x value[s],dx=[dx value for calculations])`. E.g.:

```
import scipy.misc as misc
der = misc.derivative(interp_f,x_fine,dx=1E-3)
plt.plot(x_fine,der,'b--')
```

Normalizing probabilities (cont'd)

- So we can do:

```
posterior_u = interpol.interp1d(p,likelihood*prior_u,kind='cubic')
norm_u=(integrate.quad(posterior_u,0.,1.))[0]
```

- and similarly for priors h and j (Haldane & Jeffreys).

- We can then check how this worked:

```
print(f'{norm_u = } , {norm_j = } , {norm_h = }')
```

- Where did things go wrong?

Ways to avoid the problem

- 2 ways to proceed:

1) avoid division by zero via algebra:

```
prob_u=likelihood
prob_h=p**2*(1-p)**4
prob_j=p**2.5*(1-p)**4.5
posterior_u = interpol.interp1d(p,prob_u,kind='cubic')
norm_u=(integrate.quad(posterior_u,0.,1.))[0]
...
```

- and similarly for priors h and j (Haldane & Jeffreys).

- Then we can plot normalized posterior PDFs ... (e.g., `posterior_h / norm_h`)

Second option: `np.isfinite()`

2) The problem was that we divided by zero, so some elements of `prior_h` & `prior_j` were illegal numbers (NaN). We can test if a given number is finite with `np.isfinite(x);` it returns True if the number is finite, False if not.

```
print(np.sum(np.isfinite(p)==False))
print(np.sum(np.isfinite(prior_h)==False))
```

- Since, for a dataset that is not all one value, the posterior probability should be zero at both p=0 and p=1 (**WHY**?), we can safely make the prior 0 at those points. We could use logic to deduce the right element #s, but let's be lazy.

np.where()

- One of the most powerful functions in numpy is np.where().
- It returns the array indices where some condition is true. e.g.:

```
test=np.array([1,2,3,4,5])
print(np.where(test == 3))
print(np.where(test > 2))
```

- We can use the result just like we can specify any other set of array indices (e.g. [0:i]):

```
print(test[np.where(test > 2)])
```

- Alternatively, you can use a logical expression to slice the array, with similar results:

```
print(test[ test > 2 ])
```

Using `np.where` with `np.isfinite`

- So we can get the array indices where prior_h blows up with :

```
whbad=np.where( np.isfinite(prior_h)==False)
print(whbad)
```

- **Let's repair the problem:**

```
prior_h[whbad]=0
prior_j[whbad]=0
posterior_h = interpol.interp1d(p,likelihood*prior_h,kind='cubic')
posterior_j = interpol.interp1d(p,likelihood*prior_j,kind='cubic')

norm_h=integrate.quad(posterior_h,0.,1.)[0]
norm_j=integrate.quad(posterior_j,0.,1.)[0]
```

- and plot again:

```
plt.plot(p,likelihood*prior_u/norm_u)
...
```

Some things to discuss with your group

- **Where does the posterior peak in each case? How does this compare to the observed fraction (3/8)?**
- **For which prior do you get the tightest constraint on p?** (Hint: since the integral is one, a tighter distribution will have a higher peak, when they are normalized into PDFs)

What happens if we have more data?

- Montero-Dorta et al. 2008 found that, of 710 early-type galaxies, 213 were AGN and 497 were not. What is the probability distribution for the true probability an early-type galaxy is an AGN, given each prior?
- **What do we want for `likelihood`? Do the priors change?**
- **Redo the normalizations and plot the normalized posterior for the new likelihood**

$$prob(p \mid observation) = prob(observation \mid p)\, prob(p) \,/\, prob(observation)$$

Using previous results as priors

- The Uniform, Haldane, and Jeffreys priors are all uninformative priors - i.e., priors we might choose knowing absolutely nothing about p.
- Suppose we want to use our sample of 8 galaxies to estimate the AGN fraction
- Going in, we've read Montero-Dorta et al., and we can use their result as a prior:
  - $prob(p) \propto p^{213} (1-p)^{497}$

$$prob(p \mid observation) = prob(observation \mid p)\, prob(p) \,/\, prob(observation)$$

Results as priors

- Then: $prob(p \mid obs) = prob(obs \mid p)\, prob(p) / prob(obs)$

$$\propto (p^3 (1-p)^5) \times (p^{213} (1-p)^{497})$$

- Notice that:

1) This is the same result we'd get if we took the result of our observation as a prior, and then added the information from Montero-Dorta et al. - which is 'prior' and which is 'data' doesn't matter.

2) This is exactly the same result as if we had 216 AGN and 502 non-AGN; i.e., evaluated everything as one dataset

What difference does our sample make?

- Our prior is:

```
prior_obs=x**213.*(1-x)**497.
```

  - **Calculate its normalization to get a pdf!** No need to worry about checking for nonfinite elements here...

- Our likelihood is:

```
likelihood=x**3*(1-x)**5
```

  - **Calculate its normalization to get a pdf!**

```
posterior = likelihood*prior_obs
```

  - **Calculate its normalization too!**

- **Now plot up the normalized likelihood, the normalized prior, and the normalized posterior probability, with different linestyles and/or colors (solid, dashed, red, etc.)**

Coming up with your own priors

- One simple way to produce a prior for a binomial distribution is to produce a guess as to the right answer, and how many observations that guess is 'worth'.
- E.g. if in your experience about half of galaxies are AGN, but you're not too sure about that, you might consider it worth a few observations : prior $\propto p^{1.5}(1-p)^{1.5}$
- Or you might know that ~25% of galaxies overall are AGN, but be unsure whether that overall number is relevant for the particular sample you are working on : prior $\propto p^{1}(1-p)^{3}$
- Or you might be really sure that 25% is the right universal answer, whatever your sample tells you: prior $\propto p^{250}(1-p)^{750}$

Homework 2: Due Friday, Feb. 14

Homework is due next Friday, Feb 14 by midnight, on Canvas.

Problem 1:

A) You wake up in the morning and can't remember where you are, as your mind's still foggy. What is the probability you are in Pittsburgh where you normally live, back home with your family, or on vacation? (Feel free to make informed guesses for any unknown numbers).

B) You wake up a little more and notice that you can hear the sounds of the ocean. What would you estimate the probabilities to be now?

Approach this problem from a Bayesian framework; be quantitative at each step.

Homework 2: Due Friday, Feb. 10

Problem 2:

- In a recent poll by Ipsos, out of 1035 American interviewees, 44% (455) said they were fans of professional (NFL) football and 17% (176) were fans of soccer.

- Choose one of these sports, and calculate the probability that the true fraction of people in America who are fans of that sport is within +/-1%, 2%, 3%, 4%, 5%, or 10% of the percentage who said they were fans in the interviews. If you apply a prior, be sure to describe it explicitly. You should implement this as an Python (or other-language) function which takes as input the number of interviewees, the number of positive results, and the percentage range about the result to consider.

Note: you can consider this a binomial process: N people were interviewed about their preferences, with M successes for a given sport, and we want to draw conclusions about the probability a random person would be a fan of that sport, which we can label as p.

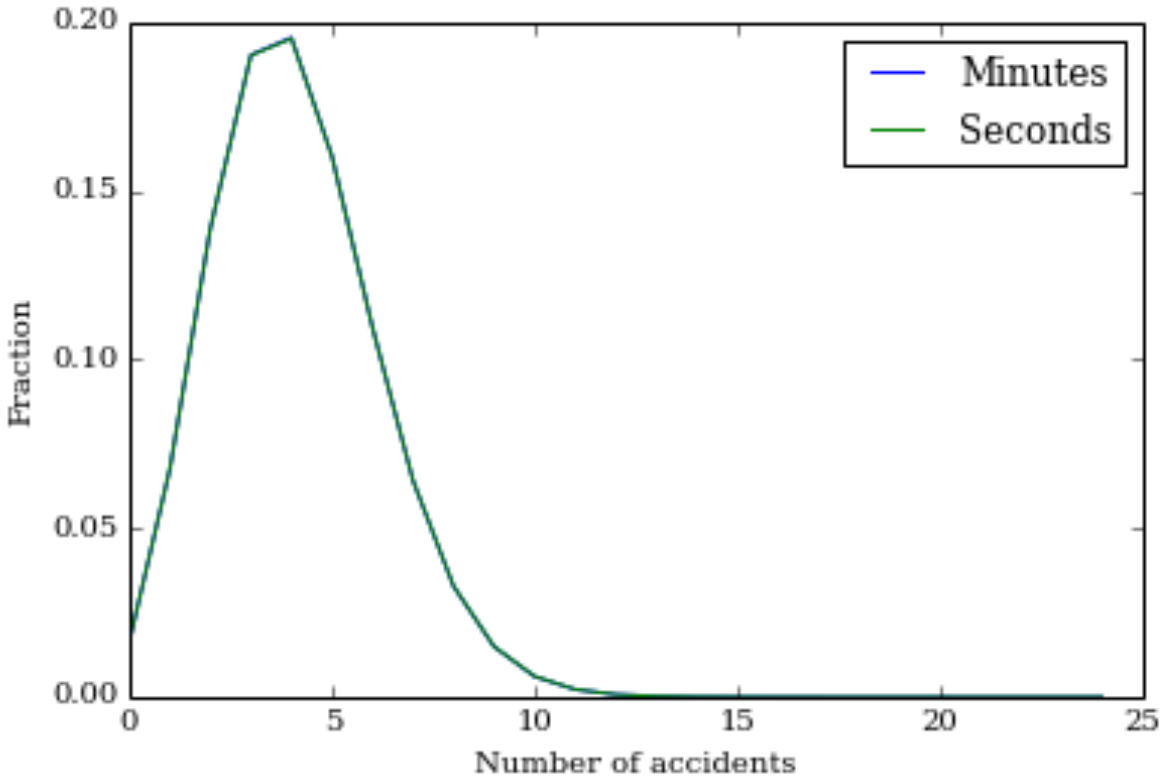
Binomial distributions with $p \sim 0$

- Recall that the binomial distribution ($\propto C(N,M)p^M(1-p)^{N-M}$) has mean $\mu = N\,p$ and variance $\sigma^2 = N\,p\,(1-p)$
- If p is small, $1-p \cong 1$, and so the binomial distribution will have equal mean and variance. Does this case ever occur?
- Consider an event that can only occur integer numbers of times: e.g., the number of car accidents in Pittsburgh in a day. There are about 1500 accidents in Pittsburgh per year, or a little more than 4.1 per day, 0.171 per hour, 2.85×10^{-3} per minute, or 4.75×10^{-5} per second.
- If we ignore any seasonal, hourly, etc. variation, we'd expect to get the same distribution for the number of accidents per day whether we break it up into seconds, minutes, etc.

Binomial distributions with $p \sim 0$

- So the chance we have an accident each minute, p, is 2.85×10^{-3}. Since there are 1440 minutes per day, the probability of M accidents is $C(1440,M)p^M(1-p)^{1440-M}$, which has mean $\mu=4.1$ and variance $\sigma^2=4.1$ accidents in a day.
- Alternatively, the chance we have an accident each second, q, is 4.75×10^{-5}. Since there are 86,400 seconds per day, the probability of M accidents is $C(86400,M)q^M(1-q)^{86400-M}$, which has mean $\mu=4.1$ and variance $\sigma 2=4.1$ accidents in a day
- Notice that these are identical means and variances! If we sliced the day up into finer and finer independent time intervals, we **still** should get the same distribution of accidents per day.

Binomial distributions with $p \sim 0$



The Poisson distribution

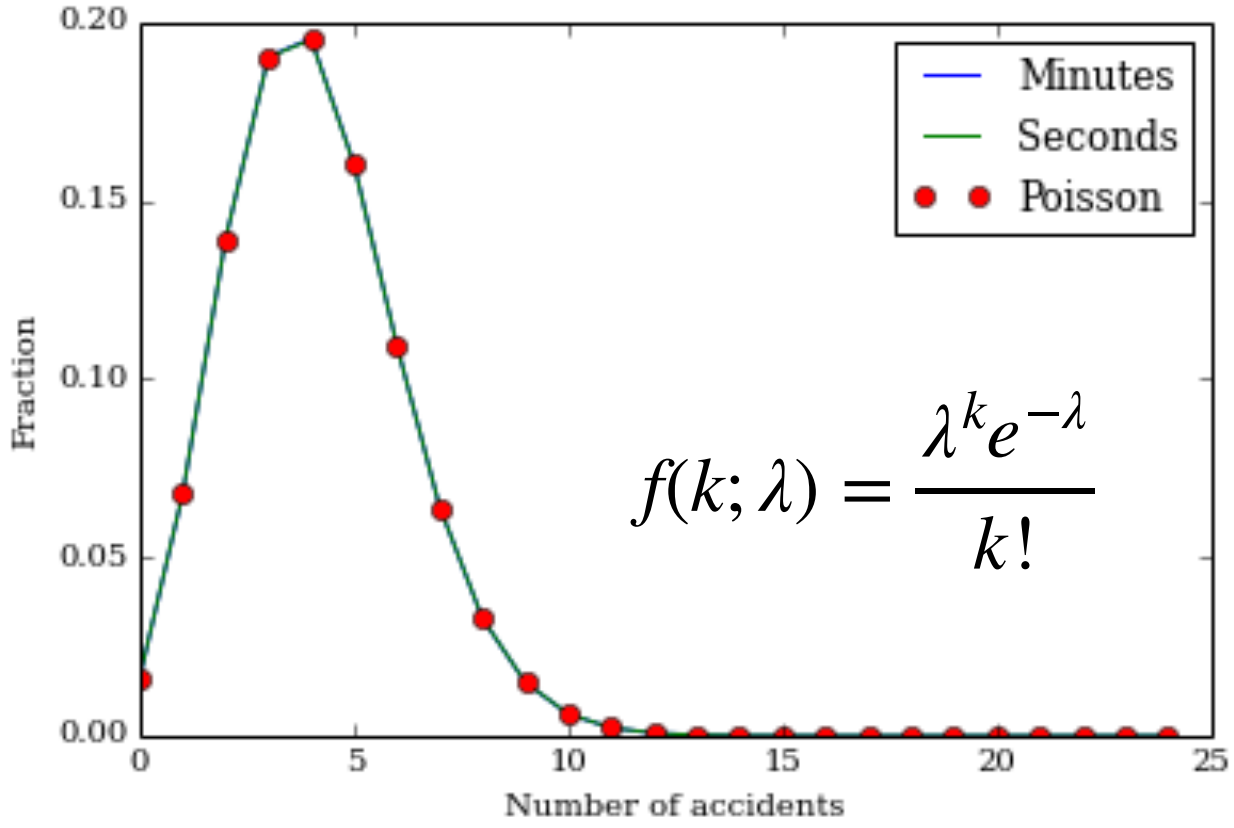
- If we have a scenario where we have a process that occurs at a constant rate per unit time (so that it would be described by a binomial distribution with N large and p small, if we choose a small enough time interval), the number of times that process occurs, k, will follow a Poisson distribution:

$$f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

- where λ is the mean total number of times that event will occur, integrating over the whole time interval (e.g., 4.1 for traffic accidents in Pittsburgh)

- The Poisson distribution has mean $\mu=\lambda$ and variance $\sigma^2=\lambda$

The Poisson distribution matches the binomial result



Why do physicists and astronomers care about Poisson distributions?

- In the UV, optical, near-IR, X-ray, and γ-ray parts of the spectrum, our detectors respond to individual photons .
- If observing conditions are constant and the source is not variable, we'd expect a constant number of photons per hour (or minute, or second) to be detected.
- As a result, the total number of photons we detect (e.g., in some pixel of a detector) after some amount of integration time will follow a Poisson distribution (presuming our detector returns a number of counts proportional to the number of photons entering).
- Similarly, the number of events of a certain type we observe in a particle collider will be Poisson-distributed, since we have many events total but only a small fraction that are of a given type

Poisson distributions for λ=1,4,10

- Let's plot a few Poisson distributions.
- We can only get integer results, so we can do:

```
k=np.arange(25)
```

- to set up an array from 0 to 25.

- Then we can plot the distribution for any choice of lambda:

```
plt.plot(k,lambda**k * ??? (-lambda)/factorial(k) )
```

- Like most Python mathematical functions, factorial() works fine when given an array as input.
- **Plot Poisson PMFs for lambda=1, lambda=4, and lambda=10.**

Creating legends

- Putting legends in plots is easy in Python.

1) In all your plot commands, use the `label=` keyword to provide a string label for each set of points/curve in the plot; e.g.:

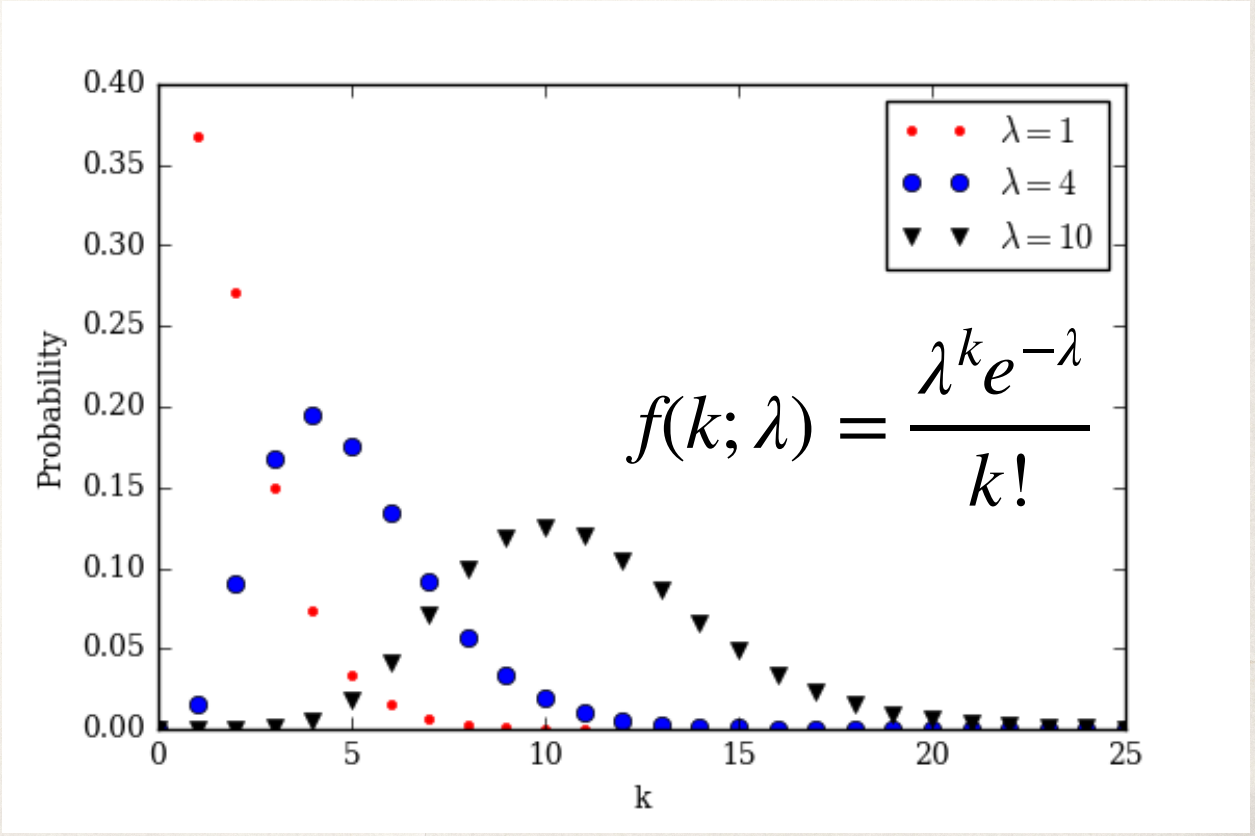
```
plt.plot(x,events_from_minutes, label='Minutes')
```

2) Use the pyplot command `plt.legend()` to draw the legend on your plot. Take a look at the help information for it; you can set legend positions, etc. with keywords.

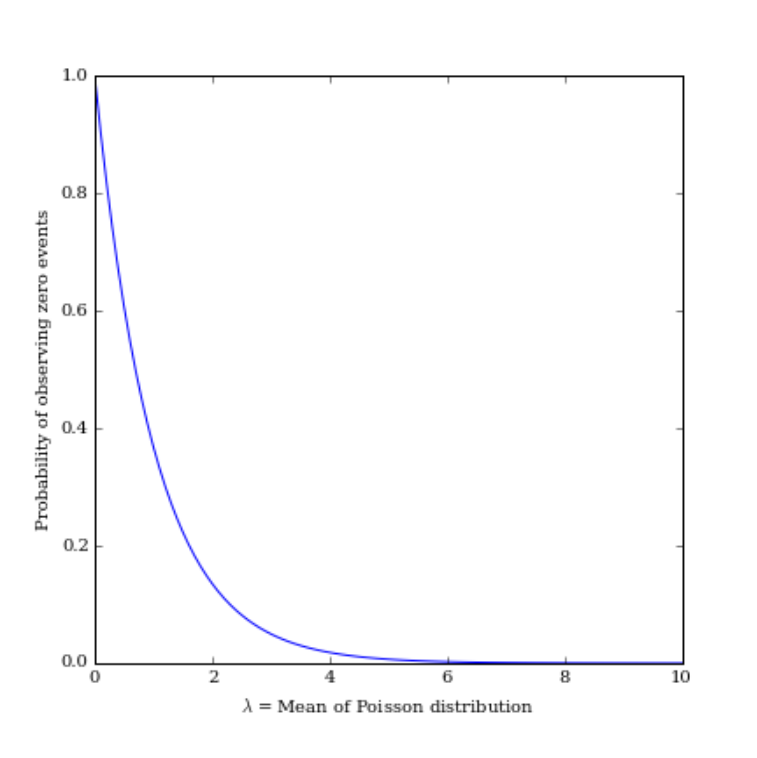
```
plt.legend()
```

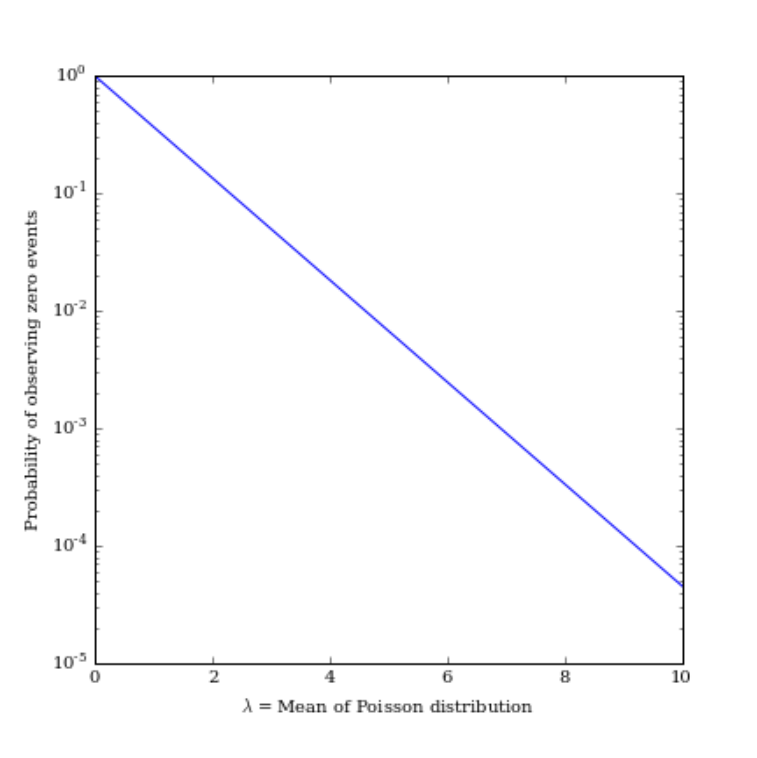
Poisson distributions for λ=1,4,10

Even for λ=10, the Poisson distribution is a bit asymmetric - as <0 events is impossible.



As a rule of thumb, it's OK to ignore this asymmetry if λ>5





Here we plot the probability of getting 0 as a function of lambda:

```
plt.plot(x,x**0*np.exp(-x)/factorial(0))
plt.yscale('log')
```

$$f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Priors for the Poisson distribution

- It turns out the Jeffreys prior for a Poisson distribution is $prob(\lambda) \propto \lambda^{-1/2}$. So if we measure N counts from a Poisson process, the resulting posterior for λ is:

  $prob(\lambda \mid N) \propto prob(N \mid \lambda)\, prob(\lambda) \propto \lambda N e^{-\lambda}\, \lambda^{-1/2} / N! \propto \lambda^{N-1/2} e^{-\lambda}$

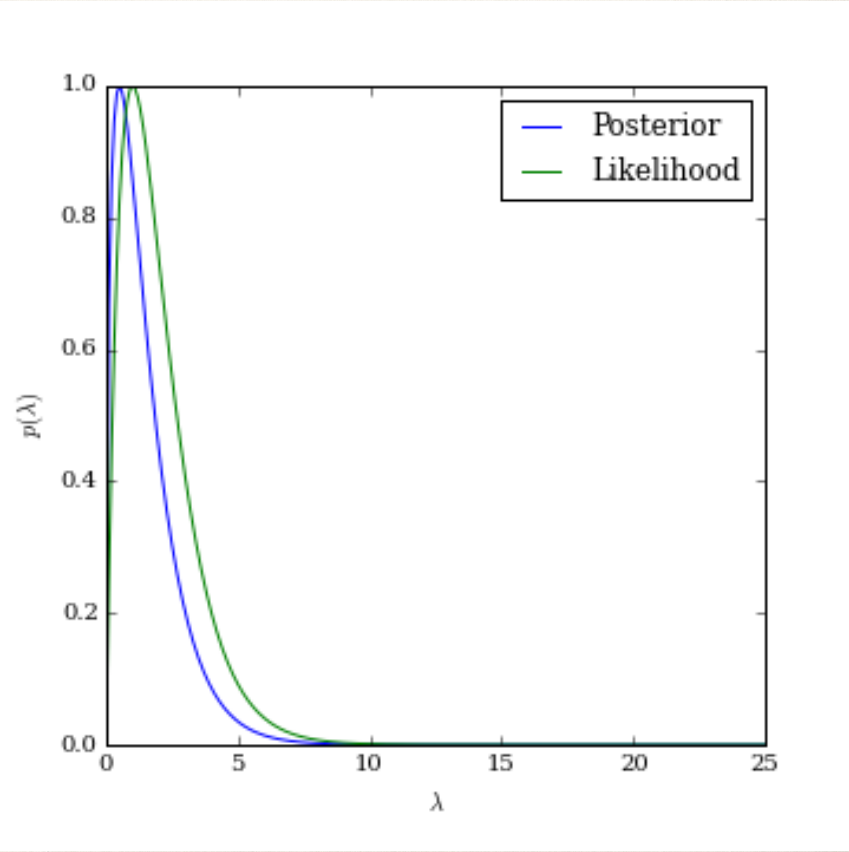
- Let's plot up the distribution, for N=1,3,5,10, with and without the prior:

```
lam_arr=np.linspace(0,25,251)
N=1
likelihood = lam_arr**(N)*np.exp(-lam_arr)
posterior = lam_arr**(N-0.5)*np.exp(-lam_arr)
```

- now try what happens with N=3,5,10...

Again, as N gets larger, the prior has less impact

- Here are the results for N=1, normalized to have the same maximum



A few more Poisson facts

- If we have two measurements, x_1 and x_2, drawn from Poisson distributions with means μ_1 and μ_2, then their sum, $x_1 + x_2$, will follow a Poisson distribution with mean $\mu_1 + \mu_2$
- E.g., if we have two images with just Poisson noise, we can just add the two images and get a result equivalent to a single image with a longer exposure time (given by the sum of the 2 exposure times).
- One thing to beware of: the Poisson distribution controls the values of actual counts. If we converted to counts per hour, say (or otherwise multiply x by some constant c), the new variable cx will have mean $c\lambda$, but variance $c^2\lambda$, so **the mean and variance are no longer equal**. Similar caution is needed when averaging, instead of summing, multiple Poisson-distributed measurements.