Poisson_regression-Notes2022

November 14, 2022

0.1 AST/STAT 5731

Claudia Scarlata

0.2 Poisson regression

Today we will review how to implement:

- Poisson regression models in PyMC,
- Negative Binomial models
- Zero Inflated models
- The effecto of truncating the data.

0.2.1 Poisson

When your data are counts of some sort you will most likely use a Poisson distribution. Remember that the Poisson distribution is used to model to describe the probability of a given number of events occurring on a fixed time/space interval. It assumes that the events occur indipendently of each other. The only parameter in the Poisson distribution is the mean λ.

$$p(y|\lambda) = \frac{e^{-\lambda}\lambda^y}{y!}$$

where x=0,1,2,…

Poisson regression, a generalized form of linear models, can be used when we suspect that the mean parameter is depends on another quantity.

In this case we assume that the response variable Y has a Poisson distribution, and we use the exponential function as the inverse link function, to make sure that the values returned by the linear models are always positive. In other words, we set:

$$\lambda = e^{\beta_0 + X\beta_1}$$

Data for this exercise are taken from the paper "X-ray luminosities of SDSS DR7 clusters from RASS" (Wang+, 2014) as well as the Globular Cluster data we showed in class.

During the lab time and left as HW to finish:

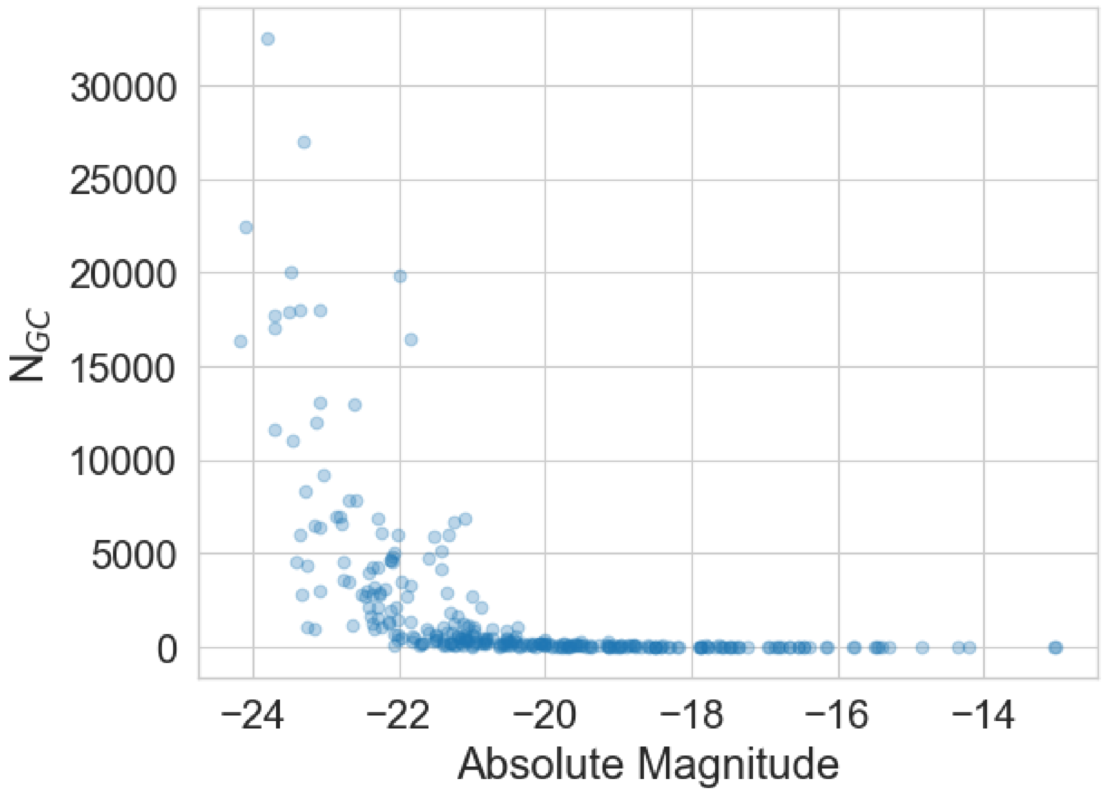
1- Using the file on canvas, extend the globular cluster analysis to simultaneously include:

absolute luminosity M_V [col10]

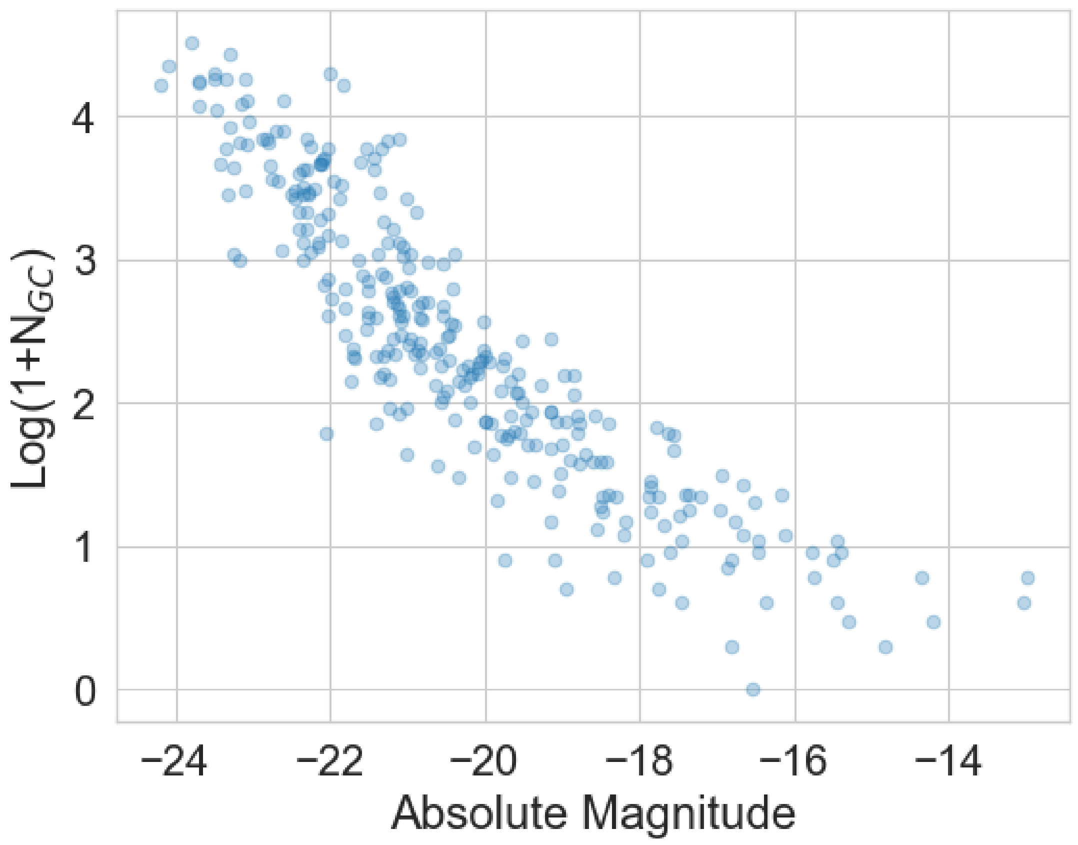
"Dynamical mass" of the galaxy [col21]

You may want to consider the Log of the Masses.

[2]:
```
from astropy.cosmology import Planck15
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from astropy.io import fits
import pandas as pd
import pymc as pm
from astropy.io import ascii
from astropy.table import Table
import arviz as az
sns.set_style("whitegrid", {"xtick.major.size": 0, "ytick.major.size": 0})
sns.set_context("paper", font_scale=1.8, rc={"lines.linewidth": 2.5})


#data = ascii.read('GCs.csv')
#x=data['MV_T'].data
#y=data['N_GC'].data

data = ascii.read('GC_allcolumns.txt')
data

x=data['MV_T'].data
y=data['N_GC'].data

plt.scatter(x,y,alpha=0.3)
#plt.yscale('log')
plt.xlabel(r'Absolute Magnitude')
plt.ylabel(r'N$_{GC}$')
plt.show()

plt.scatter(x,np.log10(1+y),alpha=0.3)
plt.xlabel(r'Absolute Magnitude')
plt.ylabel(r'Log(1+N$_{GC}$)')
```



```
[2]: Text(0, 0.5, 'Log(1+N$_{GC}$)')
```



We are going to see how to implement the Poisson model discussed in class, using PyMC. The number of GCs in each galaxy is the response variable, while the absolute magnitude is the predictor. We have 420 galaxies in total.

The Poisson model done in class for this problem is:

$$N_{GC,i} \sim \text{Poisson}(\lambda_i) log(\lambda_i) = \beta\, x_i \beta|\tau \sim N(0, \tau^{-1})\tau = 10^{-3}$$

Where we have assigned the broad reference prior on .

The implementation in PyMC is straightforward.

[3]:
```
with pm.Model() as poisson_log:

    # define priors, weakly informative Normal
    b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=-5)
    b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1)

    # define linear model and exp link function
    theta = b0 + b1 * x
    ## Define Poisson likelihood
    yvar = pm.Poisson('yvar', mu=np.exp(theta), observed=y)
    trace = pm.sample(1000, return_inferencedata=True)
```

```
/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/3918687498.py:4
: FutureWarning: The `testval` argument is deprecated; use `initval`.
  b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=-5)
/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/3918687498.py:5
: FutureWarning: The `testval` argument is deprecated; use `initval`.
  b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1)
Auto-assigning NUTS sampler…
Initializing NUTS using jitter+adapt_diag…
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [b0, b1]
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

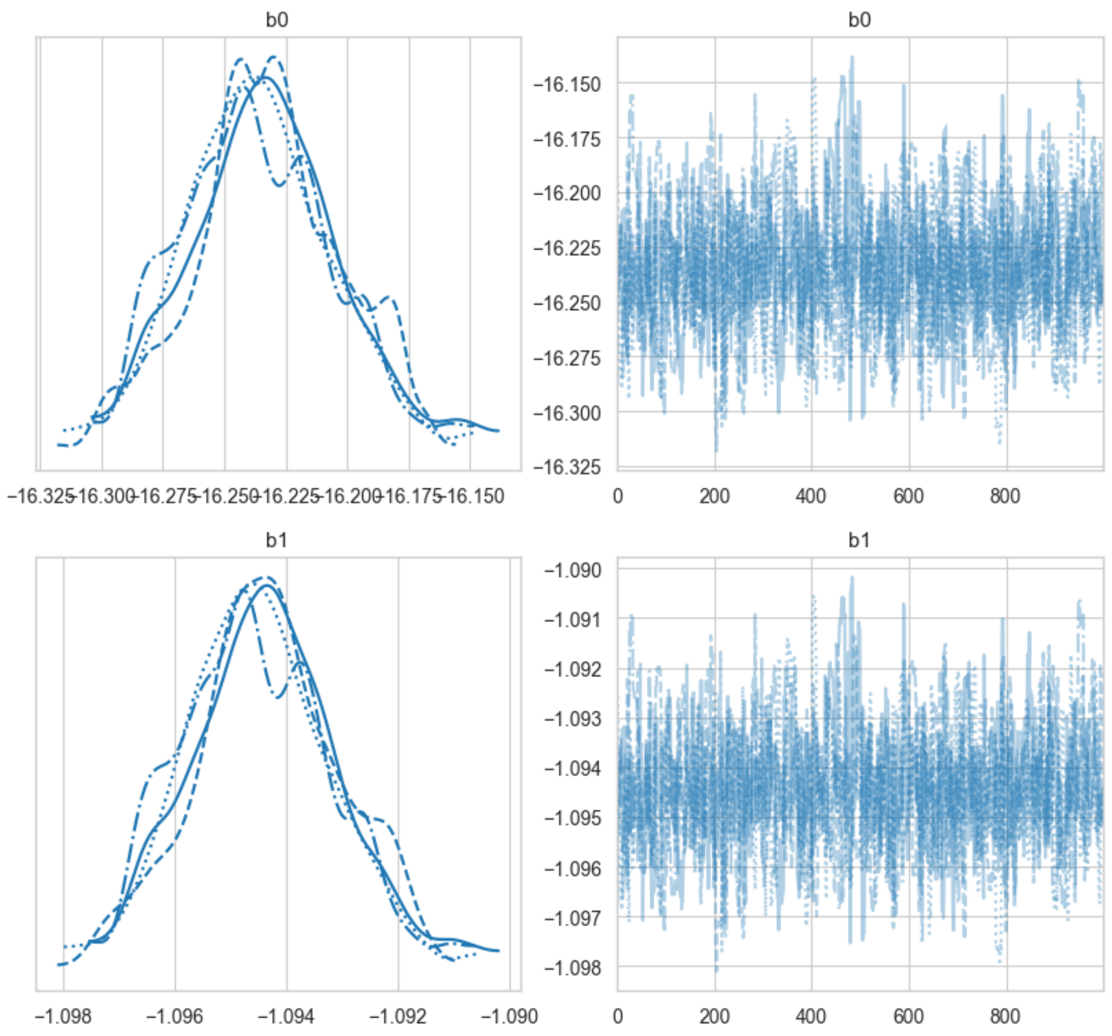
```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws
total) took 20 seconds.
```

[4]:
```
az.summary(trace)
```

[4]:
```
       mean     sd  hdi_3%  hdi_97%  mcse_mean  mcse_sd  ess_bulk  ess_tail  \
b0 -16.235  0.030 -16.287  -16.177      0.001    0.001     637.0     369.0
b1  -1.094  0.001  -1.097   -1.092      0.000    0.000     636.0     379.0

    r_hat
b0    1.0
b1    1.0
```

[5]:
```
az.plot_trace(trace,figsize=(10, 9))
```

[5]:
```
array([[<AxesSubplot: title={'center': 'b0'}>,
        <AxesSubplot: title={'center': 'b0'}>],
       [<AxesSubplot: title={'center': 'b1'}>,
        <AxesSubplot: title={'center': 'b1'}>]], dtype=object)
```



```
[6]: az.summary(trace, round_to=2)[['mean','hdi_3%','hdi_97%']]
```

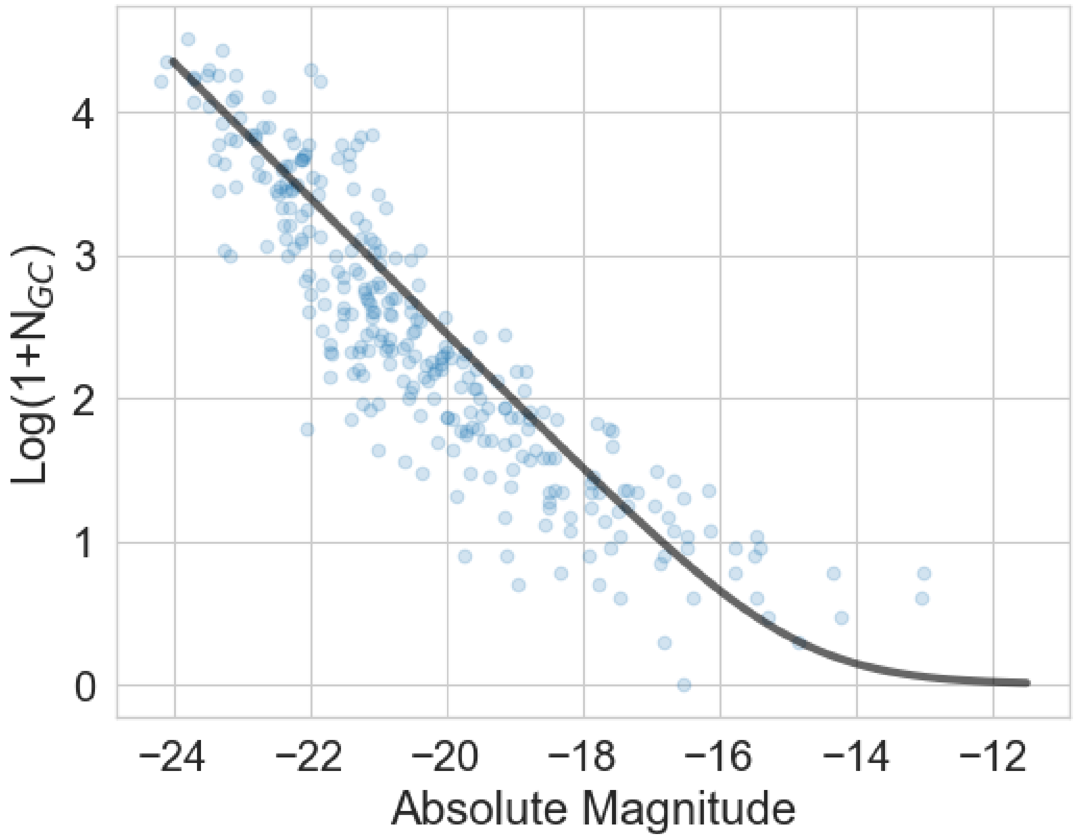
```
[6]:       mean   hdi_3%   hdi_97%
     b0 -16.23  -16.29   -16.18
     b1  -1.09   -1.10    -1.09
```

```
[7]: a = az.summary(trace)
     print(a)
     beta_0 = a['mean']['b0']
     beta_1 = a['mean']['b1']


     xt=np.linspace(-24,-11.5,100)

     BF = np.exp(beta_0+xt*beta_1)
```

```
plt.scatter(x,np.log10(1+y),alpha=0.2)
plt.plot(xt,np.log10(1+BF),alpha=0.6,lw=3,color='k')

plt.xlabel(r'Xray Luminosity [10$^{37}$ W]')
plt.ylabel('Number of galaxies')

plt.xlabel(r'Absolute Magnitude')
plt.ylabel(r'Log(1+N$_{GC}$)')
```

```
      mean     sd  hdi_3%  hdi_97%  mcse_mean  mcse_sd  ess_bulk  ess_tail  \
b0 -16.235  0.030 -16.287  -16.177      0.001    0.001     637.0     369.0
b1  -1.094  0.001  -1.097   -1.092      0.000    0.000     636.0     379.0

    r_hat
b0    1.0
b1    1.0
```

```
[7]: Text(0, 0.5, 'Log(1+N$_{GC}$)')
```



0.2.2 Negative Binomial

We saw that a different approach for data that overdispersed (i.e.,Var$< \lambda$) is to use the Negative Binomial distribution instead.

In PyMC3 the Negative Binomial is defined as:

$$p(y|\lambda,\alpha) = \binom{y+\alpha-1}{y}\left(\frac{\alpha}{\lambda+\alpha}\right)^{\alpha}\left(\frac{\lambda}{\lambda+\alpha}\right)^{y} \tag{1}$$

In this case, the model is:

$$N_{GC,i} \sim \text{NegBin}(\lambda_i,\alpha) log(\lambda_i) = \beta\, x_i \beta|\tau \sim N(0,\tau^{-1})\alpha = \text{Gamma(1e-3,1e-3)}\tau = 10^{-3}$$

The implementation in PyMC follows.

[8]:
```python
with pm.Model() as NegBin:

    # define priors, weakly informative Normal
    b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=41)
    b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1)
    alpha = pm.Gamma('alpha',1e-3,1e-3)
    # define linear model and exp link function
    eta = b0 + b1 * x
    lamb = np.exp(eta)
    ## Define Poisson likelihood
    yvar = pm.NegativeBinomial('yvar', mu=lamb, alpha=alpha, observed=y)
    trace_NB = pm.sample(1000, return_inferencedata=True)
```

```
/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/3222389970.py:4
: FutureWarning: The `testval` argument is deprecated; use `initval`.
  b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=41)
/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/3222389970.py:5
: FutureWarning: The `testval` argument is deprecated; use `initval`.
  b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1)
Auto-assigning NUTS sampler…
Initializing NUTS using jitter+adapt_diag…
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [b0, b1, alpha]
```

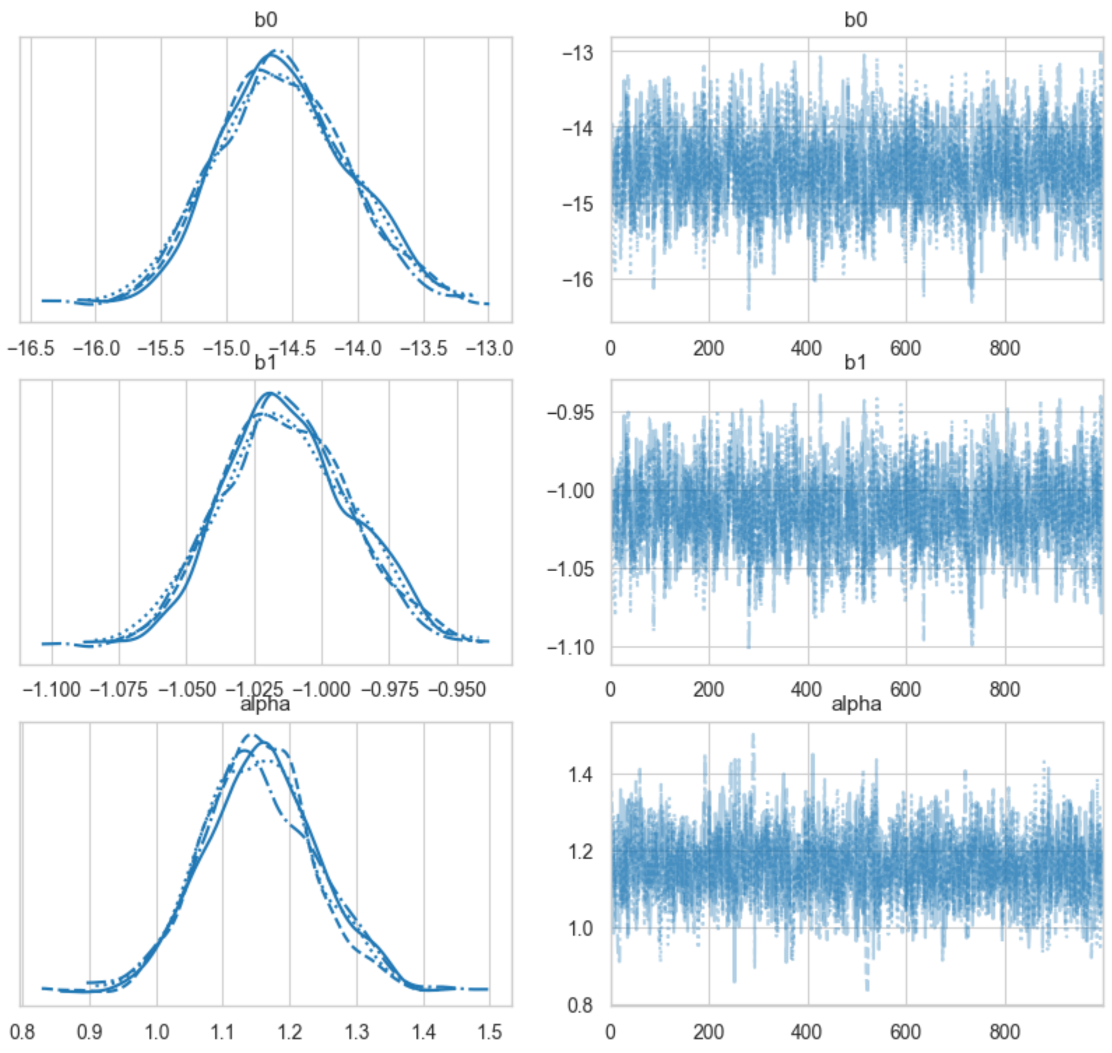
```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws
total) took 27 seconds.
```

[9]:
```python
az.plot_trace(trace_NB,figsize=(10, 9))
```

[9]:
```
array([[<AxesSubplot: title={'center': 'b0'}>,
        <AxesSubplot: title={'center': 'b0'}>],
       [<AxesSubplot: title={'center': 'b1'}>,
        <AxesSubplot: title={'center': 'b1'}>],
       [<AxesSubplot: title={'center': 'alpha'}>,
        <AxesSubplot: title={'center': 'alpha'}>]], dtype=object)
```



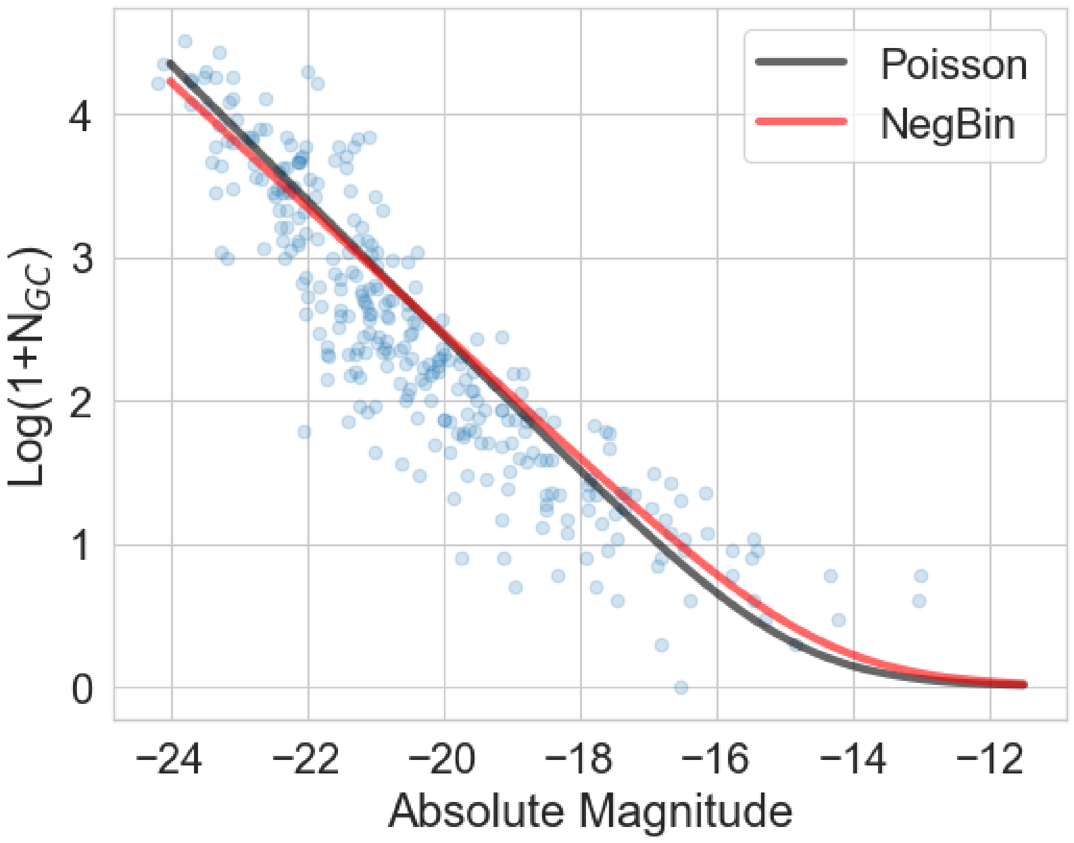
[11]:
```
az.summary(trace_NB)
```

[11]:
```
          mean     sd  hdi_3%  hdi_97%  mcse_mean  mcse_sd  ess_bulk  ess_tail  \
b0     -14.574  0.517 -15.511  -13.589      0.015    0.011    1179.0    1458.0
b1      -1.013  0.025  -1.057   -0.964      0.001    0.001    1185.0    1436.0
alpha    1.155  0.086   1.001    1.324      0.002    0.002    1444.0    1822.0

       r_hat
```

```
b0       1.0
b1       1.0
alpha    1.0
```

[12]:
```
a_NB = az.summary(trace_NB)
beta_0_NB = a_NB['mean']['b0']
beta_1_NB = a_NB['mean']['b1']

BF_NB = np.exp(beta_0_NB+xt*beta_1_NB)

plt.scatter(x,np.log10(1+y),alpha=0.2)
plt.plot(xt,np.log10(1+BF),alpha=0.6,lw=3,color='k',label='Poisson')
plt.plot(xt,np.log10(1+BF_NB),alpha=0.6,lw=3,color='r',label='NegBin')


plt.xlabel(r'Xray Luminosity [10$^{37}$ W]')
plt.ylabel('Number of galaxies')

plt.xlabel(r'Absolute Magnitude')
plt.ylabel(r'Log(1+N$_{GC}$)')
plt.legend()
```

[12]:
```
<matplotlib.legend.Legend at 0x1c04d0d10>
```



0.2.3 Zero-Inflated Poisson Model

Sometimes we have reasons to believe that the probability of measuring a "null counts" is higher than what would be predicted by a simple Poissonian process. In this situation, we saw that we can define a Zero Inflated Probability (ZIP) model for the count variable Y.

The ZIP probability density is:

$$p(y) = (1-\pi) + \pi e^{-\theta}\ \ (for\ y=0) p(y) = \pi\frac{e^{-\theta}\theta^{y}}{y!}\ \ (for\ y>0)$$

This is a "mixture model" as zero counts are modeled as coming from two generating processes: a Poisson component with probability π, and a binary component with probability (1-π). PyMC has a built in ZIP pdf (look here).

Looking at the data, it does not look like a ZIP model is appropriate. But we can look at the results.

[14]:
```
with pm.Model() as ZIPmodel:

    # define priors, weakly informative Normal
    b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=-5)
    b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1)

    # define linear model and exp link function
    theta = b0 + b1 * x
    pi = pm.Beta('pi',1,1)
    ## Define Poisson likelihood
    yvar = pm.ZeroInflatedPoisson('yvar', psi=pi, mu=np.exp(theta), observed=y)
    trace_ZIP = pm.sample(1000, return_inferencedata=True)
```

```
/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/1162646053.py:4
: FutureWarning: The `testval` argument is deprecated; use `initval`.
  b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=-5)
/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/1162646053.py:5
: FutureWarning: The `testval` argument is deprecated; use `initval`.
  b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1)
Auto-assigning NUTS sampler…
Initializing NUTS using jitter+adapt_diag…
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [b0, b1, pi]
```

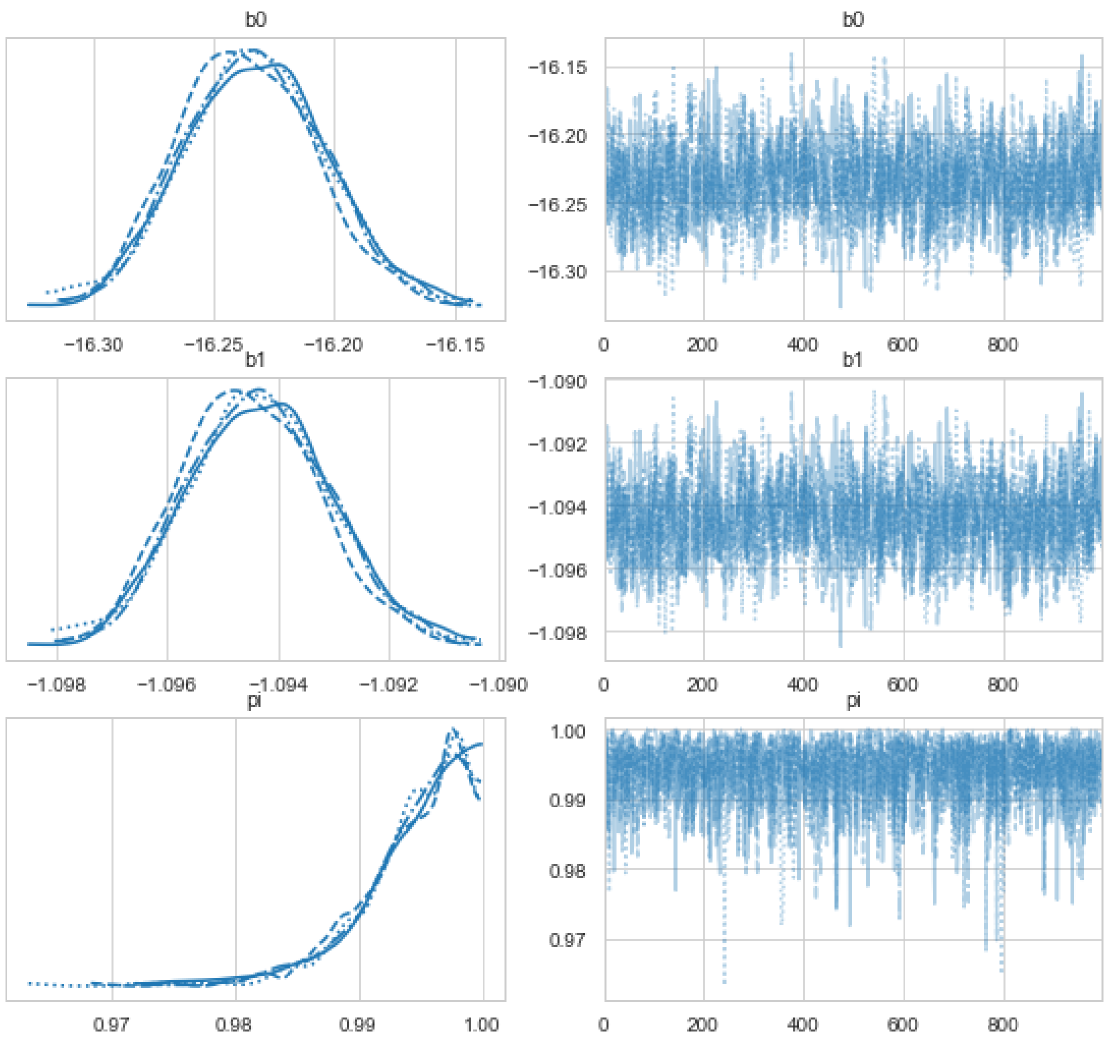
```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws
total) took 49 seconds.
```

[12]:
```
az.plot_trace(trace_ZIP,figsize=(10, 9))
```

[12]:
```
array([[<AxesSubplot:title={'center':'b0'}>,
        <AxesSubplot:title={'center':'b0'}>],
       [<AxesSubplot:title={'center':'b1'}>,
        <AxesSubplot:title={'center':'b1'}>],
       [<AxesSubplot:title={'center':'pi'}>,
        <AxesSubplot:title={'center':'pi'}>]], dtype=object)
```

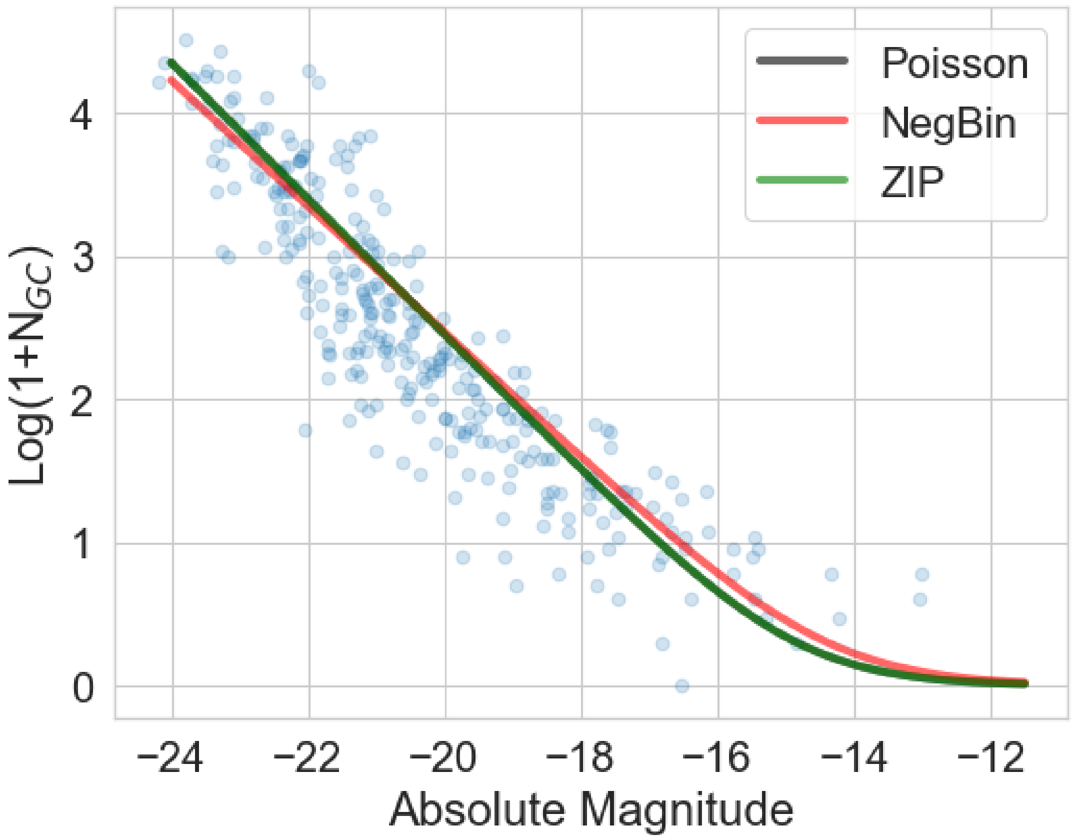


```
[16]: az.summary(trace_ZIP)[['mean','hdi_3%','hdi_97%']]
```

```
[16]:       mean  hdi_3%  hdi_97%
      b0 -16.234 -16.283  -16.177
      b1  -1.094  -1.097   -1.092
      pi   0.994   0.985    1.000
```

```
[17]: a_ZIP = az.summary(trace_ZIP)
      beta_0_ZIP = a_ZIP['mean']['b0']
      beta_1_ZIP = a_ZIP['mean']['b1']

      BF_ZIP = np.exp(beta_0_ZIP+xt*beta_1_ZIP)

      plt.scatter(x,np.log10(1+y),alpha=0.2)
      plt.plot(xt,np.log10(1+BF),alpha=0.6,lw=3,color='k',label='Poisson')
      plt.plot(xt,np.log10(1+BF_NB),alpha=0.6,lw=3,color='r',label='NegBin')
      plt.plot(xt,np.log10(1+BF_ZIP),alpha=0.6,lw=3,color='g',label='ZIP')


      plt.xlabel(r'Xray Luminosity [10$^{37}$ W]')
      plt.ylabel('Number of galaxies')

      plt.xlabel(r'Absolute Magnitude')
      plt.ylabel(r'Log(1+N$_{GC}$)')
      plt.legend()
```

```
[17]: <matplotlib.legend.Legend at 0x1bf7d1ad0>
```



0.3 Quadratic or linear in Absolute Magnitude?

From the plot above,we can see that none of the three models really does a good job at predicting the numbers of globulare clusters in galaxies with $M_V \approx -20$.

We can see if a quadratic function in the predictor works better. We will still assume a Poissonian model for the number of globular clusters. Now our model is:

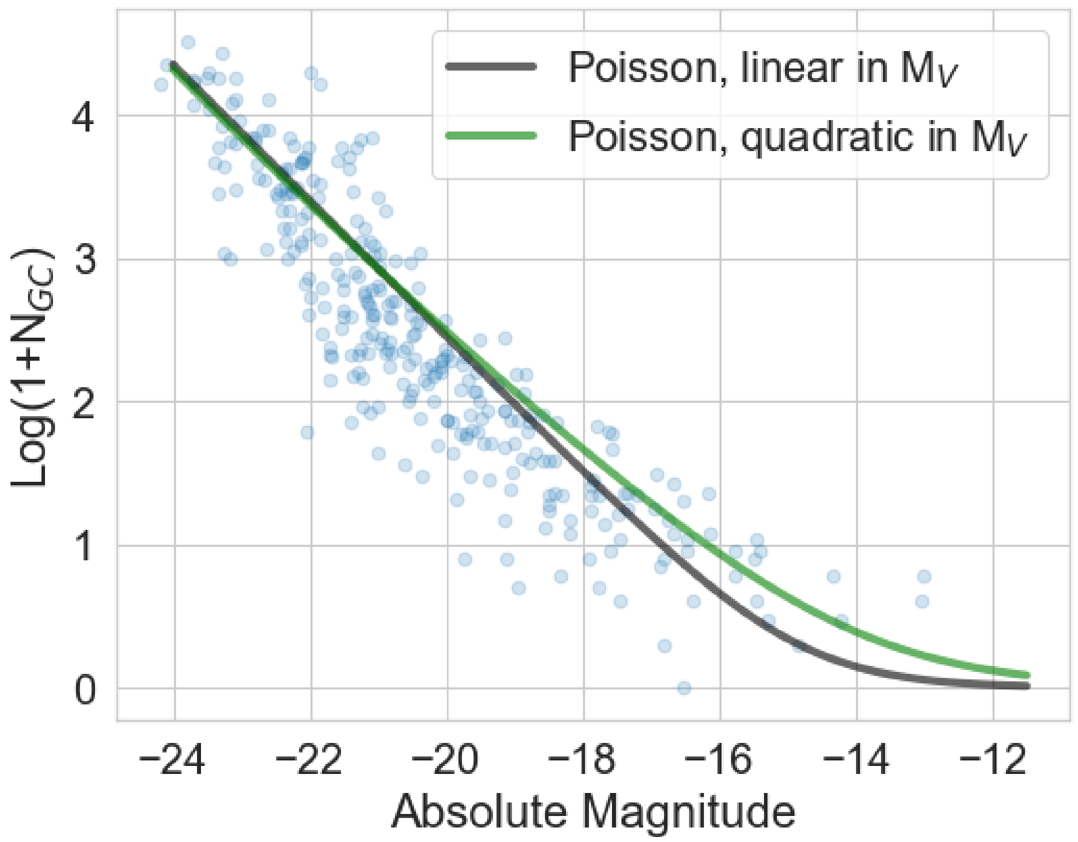
$$N_{GC,i} \sim \text{Poisson}(\lambda_i) log(\lambda_i) = \beta_0 + \beta_1\, x_i + \beta_2\, x_i^2 \beta | \tau \sim N(0, \tau^{-1}) \tau = 10^{-3}$$

[18]:
```python
order = 2
with pm.Model() as quadratic:

    # define priors, weakly informative Normal
    b0 = pm.Normal('b0', mu=0, tau=1e-3)
    b1 = pm.Normal('b1', mu=0, tau=1e-3)
    b2 = pm.Normal('b2', mu=0, tau=1e-3)

    # define linear model and exp link function
    theta = b0 + b1 * x + b2 * pm.math.sqr(x)

    ## Define Poisson likelihood
    yvar = pm.Poisson('yvar', mu=np.exp(theta), observed=y)
    trace_QD = pm.sample(100000, return_inferencedata=True,step=pm.Metropolis())
```

```
Multiprocess sampling (4 chains in 4 jobs)
CompoundStep
>Metropolis: [b0]
>Metropolis: [b1]
>Metropolis: [b2]

<IPython.core.display.HTML object>

/Users/claudia/anaconda3/anaconda3/envs/pymc_env/lib/python3.11/site-
packages/pymc/step_methods/metropolis.py:296: RuntimeWarning: overflow
encountered in exp
  "accept": np.mean(np.exp(self.accept_rate_iter)),

<IPython.core.display.HTML object>

/Users/claudia/anaconda3/anaconda3/envs/pymc_env/lib/python3.11/site-
packages/pymc/step_methods/metropolis.py:296: RuntimeWarning: overflow
encountered in exp
  "accept": np.mean(np.exp(self.accept_rate_iter)),
/Users/claudia/anaconda3/anaconda3/envs/pymc_env/lib/python3.11/site-
packages/pymc/step_methods/metropolis.py:296: RuntimeWarning: overflow
encountered in exp
  "accept": np.mean(np.exp(self.accept_rate_iter)),
/Users/claudia/anaconda3/anaconda3/envs/pymc_env/lib/python3.11/site-
packages/pymc/step_methods/metropolis.py:296: RuntimeWarning: overflow
encountered in exp
  "accept": np.mean(np.exp(self.accept_rate_iter)),
Sampling 4 chains for 1_000 tune and 100_000 draw iterations (4_000 + 400_000
draws total) took 74 seconds.
```

```
[19]: a_QD = az.summary(trace_QD)
      beta_0_QD = a_QD['mean']['b0']
      beta_1_QD = a_QD['mean']['b1']
      beta_2_QD = a_QD['mean']['b2']
      BF_QD = np.exp(beta_0_QD+xt*beta_1_QD+xt**2*beta_2_QD)

      plt.scatter(x,np.log10(1+y),alpha=0.2)
      plt.plot(xt,np.log10(1+BF),alpha=0.6,lw=3,color='k',label=r'Poisson, linear in␣
       ↪M$_V$')
      plt.plot(xt,np.log10(1+BF_QD),alpha=0.6,lw=3,color='g',label=r'Poisson,␣
       ↪quadratic in M$_V$')

      plt.xlabel(r'Xray Luminosity [10$^{37}$ W]')
      plt.ylabel('Number of galaxies')

      plt.xlabel(r'Absolute Magnitude')
      plt.ylabel(r'Log(1+N$_{GC}$)')
      plt.legend()
```

```
[19]: <matplotlib.legend.Legend at 0x1beb21ad0>
```



[20]:
```
az.summary(trace_QD)
```

[20]:
```
      mean     sd  hdi_3%  hdi_97%  mcse_mean  mcse_sd  ess_bulk  ess_tail  \
b0 -7.649  1.896  -9.641   -4.813      0.911    0.699       4.0      11.0
b1 -0.349  0.158  -0.504   -0.108      0.079    0.060       4.0      11.0
b2  0.016  0.004   0.013    0.022      0.002    0.001       4.0      11.0

    r_hat
b0   3.35
b1   4.28
b2   3.52
```

The quadratic fit seems to be doing a better job, but on the faint side it seems a little worse. We can use PyMC to look at the posterior predictive samples for both models.

We can also perform a quantitative comparison using the LOO method (and others) we discussed in class. (See also Vehtari, A., Gelman, A. & Gabry, J. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. Stat Comput 27, 1413–1432 (2017)).

Also read the arviz explanation here.

0.4 Truncated data – Poissonian example

Normal example here.

Next, we look at an example in which we know that the data are truncated in some ways. Suppose we have a statistical model to generate the data Y. If we now that there are some values of Y that are impossible to observe, then we talk about truncation.

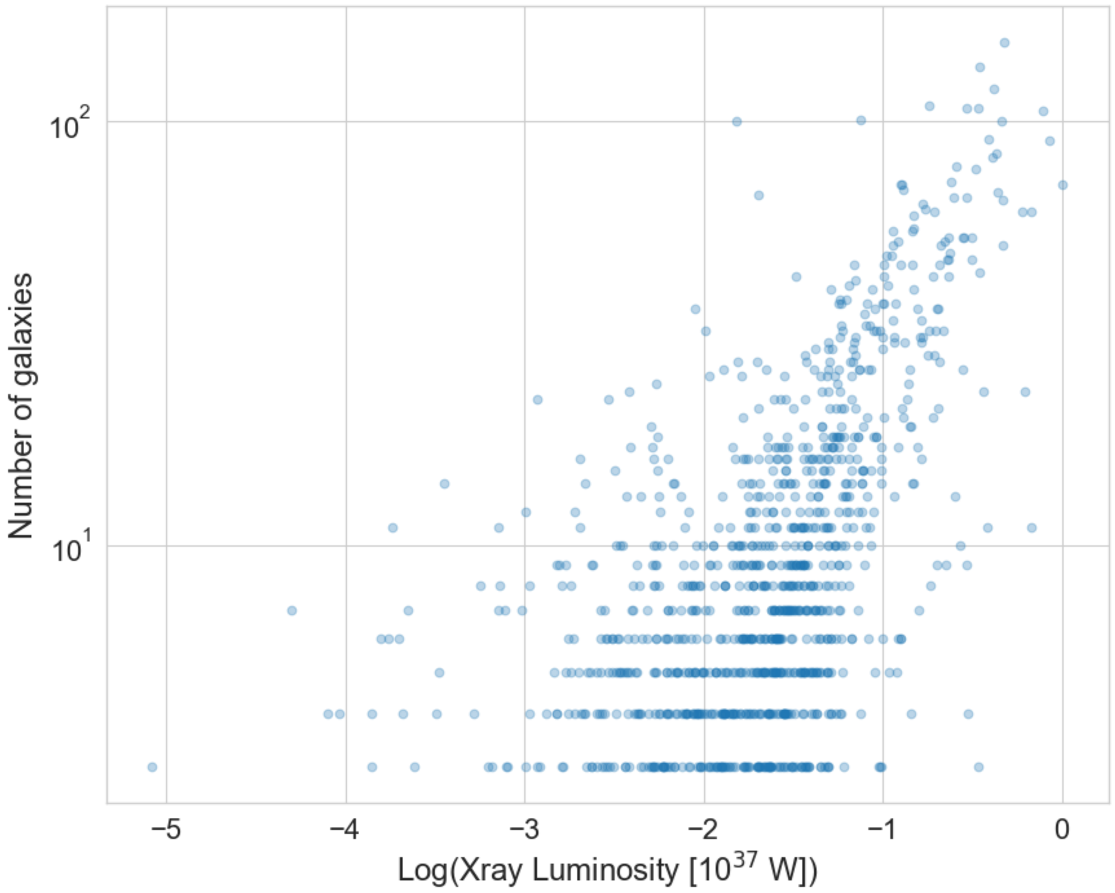
As an exaple you can think of the measured sizes of a galaxy. Either the instrumental point-spread-function or the seeing provide a minimum size we can measure. So we know that sizes smaller than the PSF have probability of zeros.

In the next example we look at the number of galaxy in groups and clusters as a function of their X-ray luminosity. This latter quantity is proportional to the total mass of the cluster (DM+baryonic) so it is expected to correlate with the number of galaxies, to first approximation.

The next few cells show the data, and the Poisson model using all the data.

[33]:
```
import pandas as pd

plt.figure(figsize=(10,8))

data = pd.read_csv('clusters_xrays.csv')
x=np.log10(data['LX'])
y=data['Ngal']

plt.scatter(x,y,alpha=0.3)
plt.yscale('log')
plt.xlabel(r'Log(Xray Luminosity [10$^{37}$ W])')
plt.ylabel('Number of galaxies')
```

```
[33]: Text(0, 0.5, 'Number of galaxies')
```



```
[34]: with pm.Model() as poisson_log:

          # define priors, weakly informative Normal
          b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=4.)
          b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1.)

          # define linear model and exp link function
          theta = b0 + b1 * x

          ## Define Poisson likelihood
          yvar = pm.Poisson('yvar', mu=np.exp(theta), observed=y)
          trace = pm.sample(4000, return_inferencedata=True)
```

```
/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/1714256894.py:4
: FutureWarning: The `testval` argument is deprecated; use `initval`.
```

```
  b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=4.)
/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/1714256894.py:5
: FutureWarning: The `testval` argument is deprecated; use `initval`.
  b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1.)
Auto-assigning NUTS sampler…
Initializing NUTS using jitter+adapt_diag…
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [b0, b1]
```

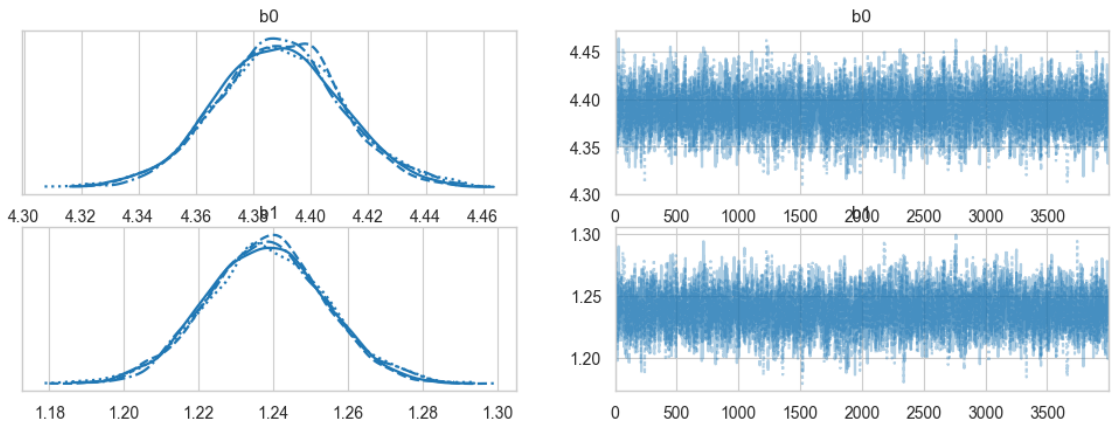
```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Sampling 4 chains for 1_000 tune and 4_000 draw iterations (4_000 + 16_000 draws
total) took 17 seconds.
```

[35]:
```python
az.plot_trace(trace)
az.summary(trace, round_to=2)[['mean','hdi_3%','hdi_97%']]
```

[35]:
```
    mean  hdi_3%  hdi_97%
b0  4.39    4.35     4.43
b1  1.24    1.21     1.27
```



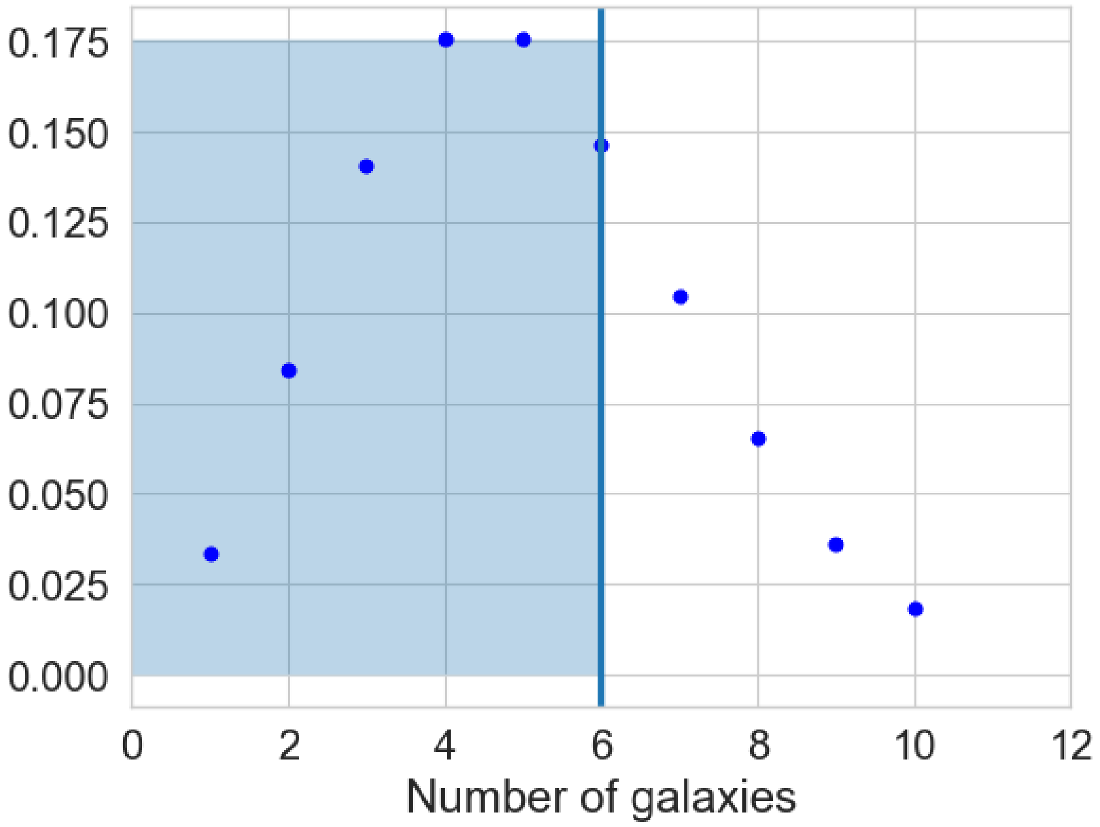
But is the above correct?

Let's consider the situation in which I want to limit the analysis to only systems with at least N galaxies (for example, if I want to mesure a velocity dispersion I need to have more than a few members).

This means that for a given X-ray luminosity, some parts of the Poisson distribution cannot be observed. As an example, suppose that for Log(LX)=-2 we have =5. The figure below shows the portion of the Poisson distribution that we are sampling by truncating the number of galaxies to be equal or greater than `limit`.

```
[36]: from scipy.stats import poisson
      limit=6

      mu=5
      xx = np.arange(poisson.ppf(0.01, mu),poisson.ppf(0.99, mu))
      plt.plot(xx, poisson.pmf(xx, mu), 'bo', label='poisson pmf')
      plt.axvline(limit)
      plt.fill_between([0,limit,limit,0],[0,0,np.max(poisson.pmf(xx, mu)),np.
       ↪max(poisson.pmf(xx, mu))],alpha=0.3)
      plt.xlim(0,12)
      plt.xlabel('Number of galaxies')
```

```
[36]: Text(0.5, 0, 'Number of galaxies')
```



```
[37]: use = y>limit
      with pm.Model() as poisson_log_truncated:

          # define priors, weakly informative Normal
          b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=4.)
          b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1.5)
```

```
    # define linear model and exp link function
    theta = b0 + b1 * x[use]

    ## Define Poisson likelihood
    yvar = pm.Poisson('yvar', mu=np.exp(theta), observed=y[use])
    trace_truncated = pm.sample(4000, return_inferencedata=True)
```

```
/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/1031980758.py:5
: FutureWarning: The `testval` argument is deprecated; use `initval`.
  b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=4.)
/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/1031980758.py:6
: FutureWarning: The `testval` argument is deprecated; use `initval`.
  b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1.5)
Auto-assigning NUTS sampler…
Initializing NUTS using jitter+adapt_diag…
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [b0, b1]
```

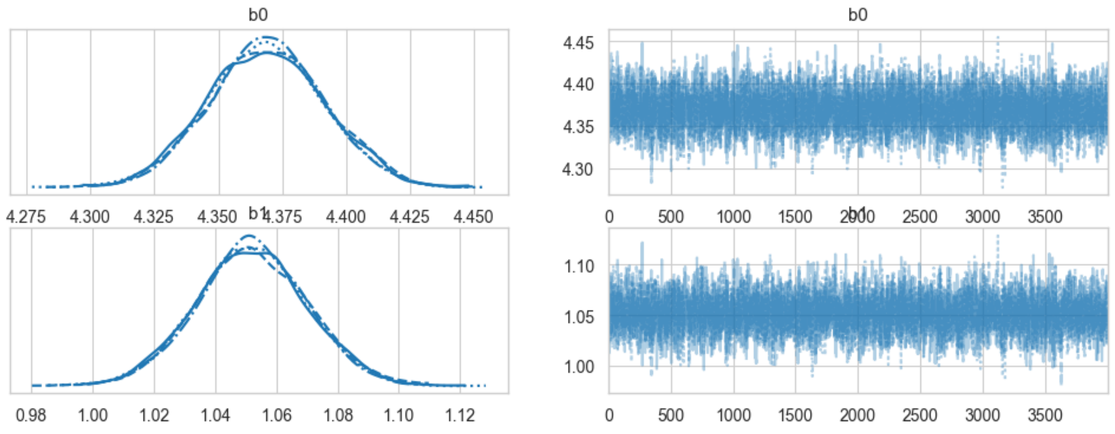
```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Sampling 4 chains for 1_000 tune and 4_000 draw iterations (4_000 + 16_000 draws
total) took 16 seconds.
```

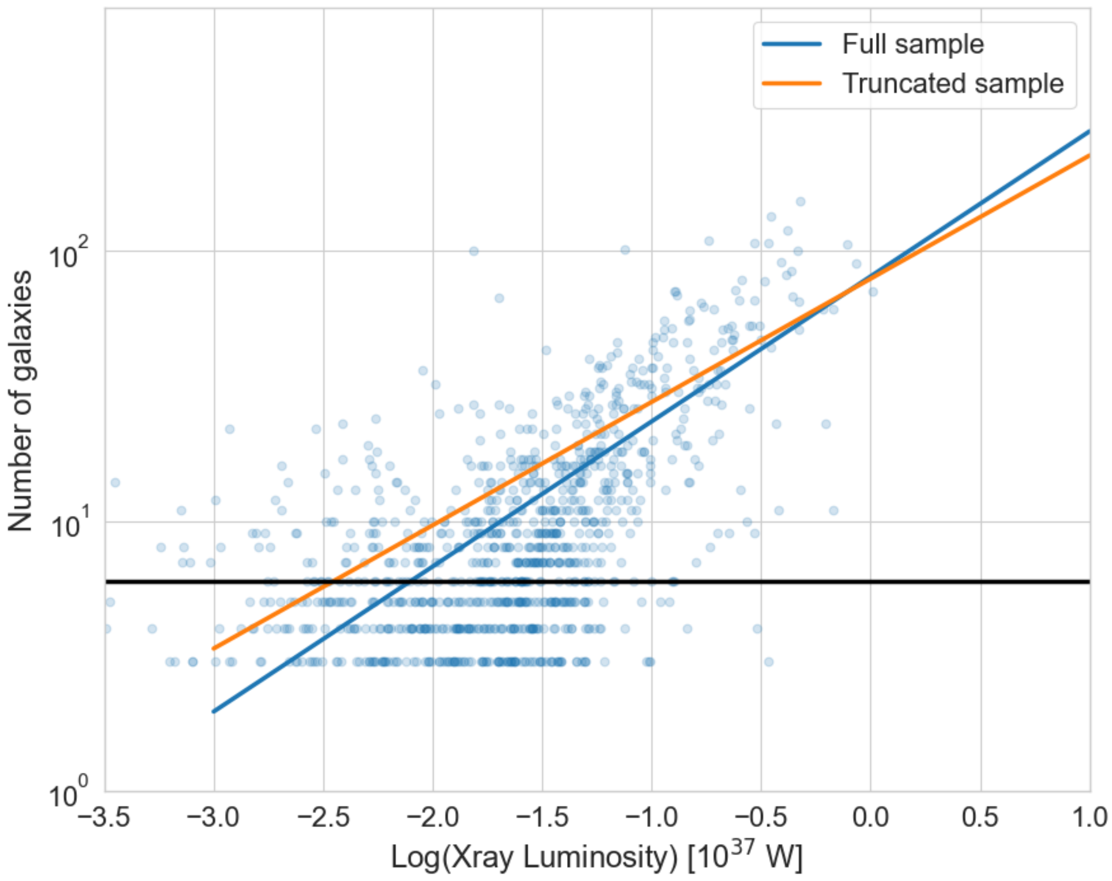
[38]:
```
az.plot_trace(trace_truncated)
az.summary(trace_truncated)[['mean','hdi_3%','hdi_97%']]
```

[38]:
```
     mean  hdi_3%  hdi_97%
b0  4.369   4.328    4.413
b1  1.052   1.017    1.085
```



[39]:
```
xt=np.linspace(-3,1,1000)
plt.figure(figsize=(10,8))


plt.scatter(x,y,alpha=0.2)
plt.yscale('log')

a = az.summary(trace)
b0 = a['mean']['b0']
b1 = a['mean']['b1']

BF =  np.exp(b0 + b1*xt)

aT = az.summary(trace_truncated)
b0T = aT['mean']['b0']
b1T = aT['mean']['b1']

BFT = np.exp(b0T + b1T*xt)

plt.xlabel(r'Log(Xray Luminosity) [10$^{37}$ W]')
plt.ylabel('Number of galaxies')

plt.plot(xt,BF,label='Full sample')
plt.plot(xt,BFT,label='Truncated sample')
plt.axhline(limit,color='k')
plt.legend()
plt.xlim(-3.5,1)
plt.ylim(1,800)
```

[39]:
```
(1, 800)
```



The way to proceed in this case is to change the likelihood of the data, as we know that the likelihood of observing any number smaller than the limit we set is 0.

We need to use a truncated Poisson distribution.

The Poisson is:

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

The truncated Poisson g(y) is given by:

$$f(y|Y > a) = \frac{g(y)}{F(\infty) - F(a)}$$

Where F(y) is the cumulative distribution function for p(y), and g(x) is the modified Poisson PDF which is zero for y≤a. (Basically after “cutting” the PDF we need to renormalize it to account for the missing probability). F(∞)=1, so we easily find (e.g., here):

$$g(y) = \frac{f(y|\lambda)}{F(\infty) - F(a|\lambda)} = \frac{\lambda^y}{y!(e^\lambda - \sum_{i=0}^{a} \frac{\lambda^i}{i!})}$$

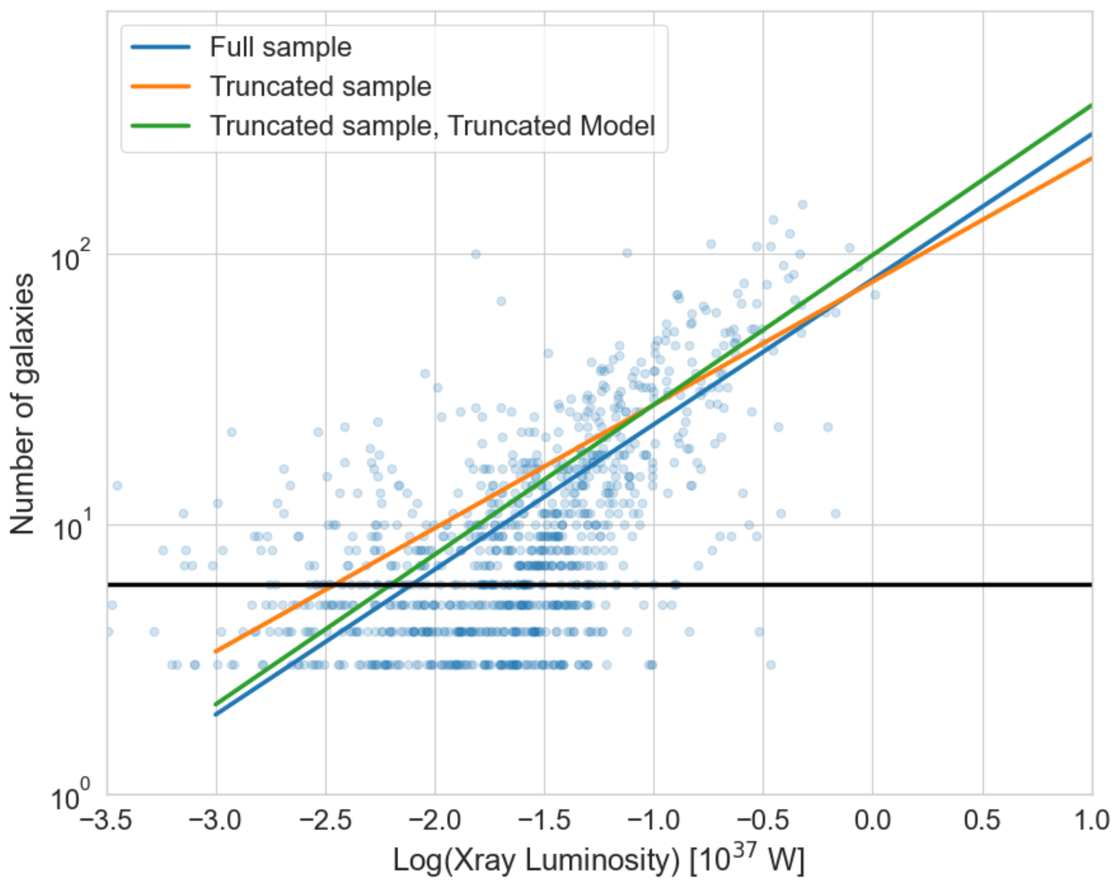
So we can implement the log distribution in PyMC

[45]:
```
print(x[use].shape)
import multiprocessing as mp


with pm.Model() as poisson_pdf_tr:

    # define priors, weakly informative Normal
    b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=4.370)
    b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1.2)

    # define linear model and exp link function
    theta = b0 + b1 * x[use]

    ## Define Poisson likelihood
    #step = pm.Metropolis()
    Poisson_dist = pm.Normal.dist(mu=np.exp(theta))
    yvar = pm.Truncated("yvar", Poisson_dist, lower=6,observed=y[use])
    trace_tt = pm.sample(5000,return_inferencedata=True,␣
 ↪cores=1,mp_ctx="forkserver")
```

```
(586,)

/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/3363363298.py:8
: FutureWarning: The `testval` argument is deprecated; use `initval`.
  b0 = pm.Normal('b0', mu=0, tau=1e-3, testval=4.370)
/var/folders/mt/j5_ht1ns57zcwpnlm45sqzwr0000gn/T/ipykernel_35308/3363363298.py:9
: FutureWarning: The `testval` argument is deprecated; use `initval`.
  b1 = pm.Normal('b1', mu=0, tau=1e-3, testval=1.2)
Auto-assigning NUTS sampler…
Initializing NUTS using jitter+adapt_diag…
Sequential sampling (2 chains in 1 job)
NUTS: [b0, b1]

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Sampling 2 chains for 1_000 tune and 5_000 draw iterations (2_000 + 10_000 draws
total) took 16 seconds.
```

[46]:
```
plt.figure(figsize=(10,8))

plt.scatter(x,y,alpha=0.2)
plt.yscale('log')

a_tt = az.summary(trace_tt)
b0 = a_tt['mean']['b0']
b1 = a_tt['mean']['b1']

BFTM =  np.exp(b0 + b1*xt)


plt.xlabel(r'Log(Xray Luminosity) [10$^{37}$ W]')
plt.ylabel('Number of galaxies')

plt.plot(xt,BF,label='Full sample')
plt.plot(xt,BFT,label='Truncated sample')
plt.plot(xt,BFTM,label='Truncated sample, Truncated Model')

plt.axhline(limit,color='k')
plt.legend()
plt.xlim(-3.5,1)
plt.ylim(1,800)
```

[46]:
```
(1, 800)
```


Full sample
Truncated sample
Truncated sample, Truncated Model
Number of galaxies
10^0
10^1
10^2
−3.5
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
Log(Xray Luminosity) [10^{37} W]

```
[ ]:
```