

Gamma-Poisson Model

General Principles

To model the relationship between a count outcome variable and one or more independent variables with overdispersion , we can use the *Negative Binomial model*.

Considerations



Caution

- We have the same considerations as for the [Poisson model](#).
- Overdispersion is handled because the Gamma-Poisson model assumes that each Poisson count observation has its own rate. This is an additional parameter specified in the model (in the code, it is `log_days`).

Example

Below is an example code snippet demonstrating a Bayesian Gamma-Poisson model using the Bayesian Inference (BI) package.

Python

```
from BI import bi, jnp

# Setup device -----
m = bi(platform='cpu') # Import

# Import Data & Data Manipulation -----
# Import
```

```

from importlib.resources import files
data_path = files('BI.resources.data') / 'Sim dat Gamma poisson.csv'
m.data(data_path, sep=',')
m.data_to_model(['log_days', 'monastery', 'y']) # Send to model (convert to jax array)

# Define model -----
def model(log_days, monastery, y):
    a = m.dist.normal(0, 1, name = 'a', shape=(1,))
    b = m.dist.normal(0, 1, name = 'b', shape=(1,))
    phi = m.dist.exponential(1, name = 'phi', shape=(1,))
    mu = jnp.exp(log_days + a + b * monastery)
    Lambda = m.dist.gamma(rate = mu*phi, concentration = phi, name = 'Lambda')
    m.dist.poisson(rate = Lambda, obs=y)
# Run MCMC -----
m.fit(model) # Optimize model parameters through MCMC sampling

# Summary -----
m.summary() # Get posterior distributions

```

jax.local_device_count 16

0%| 0/1000 [00:00<?, ?it/s] warmup: 0%| 1/1000 [00:01<24:25, 1.47s]
arviz - WARNING - Shape validation failed: input_shape: (1, 500), minimum_shape: (chains=2, 0)

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
Lambda[0]	1.39	0.36	0.83	1.91	0.01	0.02	537.43	299.02	NaN
Lambda[1]	1.76	0.40	1.13	2.34	0.01	0.02	713.23	285.08	NaN
Lambda[2]	1.57	0.42	0.92	2.17	0.01	0.02	826.04	397.99	NaN
Lambda[3]	1.47	0.36	0.86	1.97	0.01	0.02	850.98	414.15	NaN
Lambda[4]	1.53	0.38	0.95	2.14	0.01	0.02	630.33	428.36	NaN
...
Lambda[3398]	3.04	0.77	1.93	4.19	0.03	0.04	854.46	465.41	NaN
Lambda[3399]	2.86	0.73	1.76	3.95	0.03	0.04	341.11	200.88	NaN
a[0]	-0.42	0.01	-0.44	-0.40	0.00	0.00	51.13	111.55	NaN
b[0]	-2.78	0.03	-2.83	-2.73	0.00	0.00	63.60	188.73	NaN
phi[0]	16.10	1.92	13.30	19.15	0.55	0.14	12.83	101.44	NaN

R

```

library(BayesianInference)

# Setup platform-----
m=importBI(platform='cpu')

# Import data -----
m$data(paste(system.file(package = "BayesianInference"),"/data/Sim dat Gamma poisson.csv", s
m$data_to_model(list('log_days', 'monastery', 'y')) # Send to model (convert to jax array)

# Define model -----
model <- function(log_days, monastery, y){
  # Parameter prior distributions
  alpha = bi.dist.normal(0, 1, name='alpha', shape=c(1))
  beta = bi.dist.normal(0, 1, name='beta', shape=c(1))
  phi = bi.dist.exponential(1, name='phi', shape=c(1))
  mu = jnp$exp(log_days + alpha + beta * monastery)
  Lambda = bi.dist.gamma(rate = mu*phi, concentration = phi, name = 'Lambda')
  # Likelihood
  bi.dist.poisson(rate=Lambda, obs=y)
}
# Run MCMC -----
m$fit(model) # Optimize model parameters through MCMC sampling

# Summary -----
m$summary() # Get posterior distributions

```

Mathematical Details

Bayesian model

In the Bayesian formulation, we define each parameter with priors . We can express the Bayesian regression model accounting for prior distributions as follows:

$$Y_i \sim \text{Poisson}(\lambda_i)$$

$$\lambda_i \sim \text{Gamma}(\mu_i\phi, \phi)$$

$$\log(\mu_i) = \text{rates}_i + \alpha + \beta X_i$$

$$\alpha \sim \text{Normal}(0, 1)$$

$$\beta \sim \text{Normal}(0, 1)$$

$$\phi \sim \text{Exponential}(1)$$

Where:

- Y_i is the dependent variable for observation i .
- λ_i is the rate parameter of the Poisson distribution for observation i , assuming that each Poisson count observation has its own $rate_i$.
- μ_i is the mean rate parameter.
- ϕ controls the level of overdispersion in the rates.
- α is the intercept term.
- β is the regression coefficient.
- X_i is the value of the predictor variable for observation i .

Notes

Note

- We can apply multiple variables similarly as in [chapter 2](#).
- We can apply interaction terms similarly as in [chapter 3](#).
- We can apply categorical variables similarly as in [chapter 4](#).

Reference(s)