# 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
# 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 = m.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
```

**R**

```
library(BI)

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

# Import data ------------------------------------------------
m$data(paste(system.file(package = "BI"),"/data/Sim dat Gamma poisson.csv", sep = ''), sep='
m$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$exp(log_days + alpha + beta * monastery)
  Lambda =  m.dist.gamma(rate = mu*phi, concentration = phi, name = 'Lambda')
  # Likelihood
  m$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 $i$.
- $\lambda_i$ is the rate parameter of the Poisson distribution for observation $i$, assuming that each Poisson count observation has its own $rate_i$.
- $\mu_i$ is the mean rate parameter.
-

- $\phi$ controls the level of overdispersion in the rates.
- $\alpha$ is the intercept term.
- $\beta$ 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)