

Poisson Model

General Principles

To model the relationship between a count outcome variable—e.g., counts of events occurring in a fixed interval of time or space—and one or more independent variables, we can use the *Poisson model*.

This is a special shape of the binomial distribution; it is useful because it models binomial events for which the number of trials n is unknown or uncountably large.

Considerations

Caution

- We have the same considerations as for [Regression for a continuous variable](#).
- We have the second link function : *log*. The *log* link ensures that is always positive.
- The dependent variable in a Poisson regression must be a non-negative count.
- To invert the log link function and linearly model the relationship between the predictor variables and the log of the mean rate parameter, we can apply the *exponential* function (see comment in code).
- A key assumption of the *Poisson* distribution is that the mean and variance of the count variable are equal. If the variance is greater than the mean, a condition known as overdispersion, a [Gamma-Poisson model](#) might be more appropriate.

Example

Below is an example code snippet demonstrating a Bayesian Poisson model using the BayesForge (BF) package. Data consist of:

- 1) A continuous dependent variable *total_tools*, which represents the number of tools produced by a civilization.
- 2) A continuous independent variable *population* representing population size.
- 3) A categorical independent variable *cid* representing different civilizations.

The goal is to estimate the production of tools based on population size, accounting for each civilization. This example is based on McElreath (2018).

Python

```
from BayesForge import bf
import jax.numpy as jnp
# Setup device-----
m = bf(platform='cpu')

# import data -----
# Import
from importlib.resources import files
data_path = m.load.kline(only_path = True)
m.data(data_path, sep=';')
m.scale(['population'])
m.df["cid"] = (m.df.contact == "high").astype(int)
#m.data_to_model(['total_tools', 'population', 'cid'])
def model(cid, population, total_tools):
    a = m.dist.normal(3, 0.5, shape=(2,), name='a')
    b = m.dist.normal(0, 0.2, shape=(2,), name='b')
    l = jnp.exp(a[cid] + b[cid]*population)
    m.dist.poisson(l, obs=total_tools)

# Run sampler -----
m.fit(model, progress_bar=False)

# Diagnostic -----
m.summary()
```

bf v 0.0.48 package loaded

E0526 16:17:11.136389 1063111 cuda_dnn.cc:523] Loaded runtime CuDNN library: 9.1.0 but source
E0526 16:17:11.139010 1063111 cuda_dnn.cc:523] Loaded runtime CuDNN library: 9.1.0 but source

jax.local_device_count 32

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a[0]	3.22	0.09	3.06	3.36	0.0	0.0	2811.53	2615.73	1.0
a[1]	3.64	0.09	3.49	3.78	0.0	0.0	2885.45	2745.74	1.0
b[0]	0.35	0.05	0.28	0.43	0.0	0.0	2786.25	2664.04	1.0
b[1]	0.04	0.19	-0.25	0.37	0.0	0.0	3041.64	2543.66	1.0

R

```
library(BayesForge)

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

# import data -----
m$data(m$load$kline(only_path = T), sep=';')
m$scale(list('population'))# Scale
m$df["cid"] = as.integer(ifelse(m$df$contact == "high", 1, 0)) # Manipulate
m$data_to_model(list('total_tools', 'population', 'cid' )) # Send to model (convert to jax ar

# Define model -----
model <- function(total_tools, population, cid){
  # Parameter prior distributions
  alpha = bf.dist.normal(3, 0.5, name='alpha', shape = c(2))
  beta = bf.dist.normal(0, 0.2, name='beta', shape = c(2))
  l = jnp$exp(alpha[cid] + beta[cid]*population)
  # Likelihood
  m.dist.poisson(l, obs=total_tools)
}

# Run MCMC -----
m$fit(model) # Optimize model parameters through MCMC sampling

# Summary -----
m$summary() # Get posterior distribution
```

Julia

```
using BayesForge

# Setup device-----
m = importBF(platform="cpu")

# Import Data & Data Manipulation -----
# Import
data_path = m.load.kline(only_path = true)
m.data(data_path, sep=';')
m.scale(["population"]) # Normalize
m.df["cid"] = m.df.contact.eq("high").astype("int")

# Define model -----
@BF function model(cid, population, total_tools)
    a = m.dist.normal(3, 0.5, shape=(2,), name="a")
    b = m.dist.normal(0, 0.2, shape=(2,), name="b")
    l = jnp.exp(a[cid] + b[cid]*population)
    m.dist.poisson(l, obs=total_tools)
end

# Run mcmc -----
m.fit(model) # Optimize model parameters through MCMC sampling

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

Mathematical Details

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

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

$$\log(\lambda_i) = \alpha + \beta X_i$$

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

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

Where:

- Y_i is the dependent variable for observation i .
- $\log()$ is the **log link function**. This function links the log of the mean of the response variable, λ_i , to the linear predictor, $\alpha + \beta X_i$. The logarithm is the canonical link function for the Poisson distribution. It ensures that the predicted mean, $\lambda_i = \exp(\alpha + \beta X_i)$, will always be positive, as required for a Poisson rate parameter.
- α and β are the intercept and regression coefficient, respectively, with their associated prior distributions.
- X_i is the value of the independent variable for observation i .

Notes

i Note

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

Reference(s)

McElreath, Richard. 2018. *Statistical Rethinking: A Bayesian course with examples in R and Stan*. Chapman; Hall/CRC.