

Zero-Inflated Models

General Principles

Zero-Inflated Regression models are used when the outcome variable is a count variable with an excess of zero counts. These models combine a count model (e.g., Poisson or Negative Binomial) with a separate model for predicting the probability of excess zeros.

Example

Below is an example code snippet demonstrating Bayesian Zero-Inflated Poisson regression using the Bayesian Inference (BI) package. The data represent the production of books in a monastery (y), which is affected by the number of days that individuals work, as well as the number of days individuals drink. This example is based on McElreath (2018).

Python

```
from BI import bi
import jax.numpy as jnp

# Setup device -----
m = bi('cpu', rand_seed = False) # Fixing seed for reproducibility

# Simulated data-----
prob_drink = 0.2 # 20% of days
rate_work = 1 # average 1 manuscript per day

# Sample one year of production
N = 365
drink = m.dist.binomial(1, prob_drink, shape = (N,), sample = True)
y = (1 - drink) * m.dist.poisson(rate_work, shape = (N,), sample = True)
```

```

# Setup device-----
m.data_on_model = dict(
  y=jnp.array(y)
)

# Define model -----
def model(y):
  al = m.dist.normal(1, 0.5, name='al')
  ap = m.dist.normal(-1.5, 1, name='ap')
  p = m.link.inv_logit(ap)
  lambda_ = jnp.exp(al)
  m.dist.zero_inflated_poisson(p, lambda_, obs=y)

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

# Summary -----
m.summary()

```

BI v 0.0.26 package loaded

E0525 10:53:40.812575 505287 cuda_dnn.cc:523] Loaded runtime CuDNN library: 9.1.0 but source
E0525 10:53:40.814261 505287 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 |
|----|-------|------|----------|-----------|-----------|---------|----------|----------|-------|
| al | 0.15 | 0.09 | 0.02 | 0.30 | 0.00 | 0.0 | 1485.52 | 1999.88 | 1.01 |
| ap | -0.74 | 0.24 | -1.12 | -0.38 | 0.01 | 0.0 | 1507.46 | 1700.08 | 1.01 |

R

```

library(BayesianInference)

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

# Simulate data -----

```

```

prob_drink = 0.2 # 20% of days
rate_work = 1 # average 1 manuscript per day
# sample one year of production
N = as.integer(365)
drink = bi.dist.binomial(total_count = as.integer(1), probs = prob_drink, shape = c(N), samp
y = (1 - drink) * bi.dist.poisson(rate_work, shape = c(N), sample = T)
data = list()
data$y = y
m$data_on_model = data

# Define model -----
model <- function(y){
  al = bi.dist.normal(1, 0.5, name='al', shape=c(1))
  ap = bi.dist.normal(-1, 1, name='ap', shape=c(1))
  p = m$link$inv_logit(ap)
  lambda_ = jnp$exp(al)
  bi.dist.zero_inflated_poisson(p, lambda_, obs=y)
}

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

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

```

Julia

```

using BayesianInference

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

# Simulated data -----
prob_drink = 0.2
rate_work = 1

# Sample one year of production
N = 365

# Note: Use lowercase 'true' for booleans in Julia

```

```

# 'drink' will be a Python/JAX object
drink = m.dist.binomial(1, prob_drink, shape=(N,), sample=true)

# Math works automatically because 'drink' is a Py object
# and we taught Julia how to handle Py arithmetic in the previous step
y = (1 - drink) * m.dist.poisson(rate_work, shape=(N,), sample=true)

# Send data to BI class object -----
m.data_on_model = pydict(Dict("y" =>y))

# Define model -----
@BI function model(y)
    al = m.dist.normal(1, 0.5, name="al")
    ap = m.dist.normal(-1.5, 1, name="ap")
    p = m.link.inv_logit(ap)
    lambda_ = jnp.exp(al)
    m.dist.zero_inflated_poisson(p, lambda_, obs=y)
end

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

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

```

Mathematical Details

We model the relationship between the independent variable X and the count outcome variable Y . The model assumes that the data-generating process has two states: a “zero” state that generates only zero counts, and a “Poisson” state that generates counts from a Poisson distribution.

The overall model has two main linear models (π_i and λ_i):

$$Y_i \sim \text{ZIPoisson}(\pi_i, \lambda_i)$$

Where the mass probability mass function $\text{ZIPoisson}(\pi_i, \lambda_i)$ is given by:

$$P(Y_i|\pi_i, \lambda_i) = \begin{cases} \pi_i + (1 - \pi_i) \times \text{Poisson}(0|\lambda_i) & \text{if } Y_i = 0 \\ (1 - \pi_i) \times \text{Poisson}(Y_i|\lambda_i) & \text{if } Y_i > 0 \end{cases}$$

1. π_i A logistic regression model to predict the probability of a structural zero.
2. λ_i is a Poisson regression conditional on the outcome not resulting from a structural zero.

Jointly these two parameters allow us to estimate the probability of getting a zero ($Y_i = 0$) as the sum of two processes: the probability of a structural zero (π_i) plus the probability of not being a structural zero ($1 - \pi_i$) and then getting a zero from the Poisson distribution anyway.

The parameters π_i and λ_i can be modeled as functions of an independent variable X_i :

$$\text{logit}(\pi_i) = \alpha_\pi + \beta_\pi X_i$$

$$\log(\lambda_i) = \alpha_\lambda + \beta_\lambda X_i$$

We can assign prior distributions to the model parameters with weakly informative Normal priors:

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

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

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

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

Reference(s)

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