

# Multinomial Model

## General Principles

To model the relationship between a vector outcome variable in which each element of the vector is a frequency from a set of more than two categories and one or more independent variables, we can use a *Multinomial* model.

## Considerations

### **i** Note

- We have the same considerations as for the [Categorical model](#).

## Example

Below is an example code snippet demonstrating a Bayesian multinomial model using the Bayesian Inference (BI) package. This example is based on McElreath (2018).

## Python

```
from BI import bi
import jax.numpy as jnp
import jax
# Setup device -----
m = bi('cpu')

# Import Data & Data Manipulation -----
# Import
data_path = m.load.sim_multinomial(only_path=True)
m.data(data_path, sep=',')
```

```

# Define model -----
def model(income, career):
    # Parameter prior distributions
    alpha = m.dist.normal(0, 1, shape=(2,), name='a')
    beta = m.dist.half_normal(0.5, shape=(1,), name='b')
    s_1 = alpha[0] + beta * income[0]
    s_2 = alpha[1] + beta * income[1]
    s_3 = [0]
    p = jnp.exp(jnp.stack([s_1[0], s_2[0], s_3[0]]))
    # Likelihood
    m.dist.multinomial(probs = p[career], obs=career)

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

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

```

/home/sosa/work/3.12venv/lib/python3.10/site-packages/tqdm/auto.py:21: TqdmWarning:

IProgress not found. Please update jupyter and ipywidgets. See <https://ipywidgets.readthedocs>

BI v 0.0.46 package loaded  
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]	0.00	1.01	-1.59	1.66	0.02	0.01	4105.89	2659.93	1.0
a[1]	3.98	1.00	2.29	5.48	0.02	0.01	3433.48	2848.04	1.0
b[0]	20.01	0.51	19.18	20.82	0.01	0.01	4522.26	2903.42	1.0

## R

```

library(BayesianInference)
jax = reticulate::import('jax')
# Setup platform-----
m=importBI(platform='cpu')

# import data -----

```

```

m$data(m$load$sim_multinomial(only_path=T), sep=',')
keys <- c("income", "career")
income = unique(m$df$income)
income = income[order(income)]
values <- list(jnp$array(as.integer(income)),jnp$array( as.integer(m$df$career)))
m$data_on_model = py_dict(keys, values, convert = TRUE)

# Define model -----
model <- function(income, career){
  # Parameter prior distributions
  alpha = bi.dist.normal(0, 1, name='alpha', shape = c(2))
  beta = bi.dist.normal(0.5, name='beta')

  s_1 = alpha[0] + beta * income[0]
  s_2 = alpha[1] + beta * income[1]
  s_3 = 0 # reference category

  p = jax$nn$softmax(jnp$stack(list(s_1, s_2, s_3)))

  # Likelihood
  m$dist$multinomial(probs=p[career], obs=career)
}

# Run sampler -----
m$fit(model)

# Summary -----
m$summary()

```

## Julia

```

using BayesianInference

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

# Import Data & Data Manipulation -----
# Import
data_path = m.load.sim_multinomial(only_path = true)
m.data(data_path, sep=',')

```

```

# Define model -----
@BI function model(income, career)
  # Parameter prior distributions
  alpha = m.dist.normal(0, 1, shape=(2,), name='a')
  beta = m.dist.half_normal(0.5, shape=(1,), name='b')
  s_1 = alpha[0] + beta * income[0]
  s_2 = alpha[1] + beta * income[1]
  # Use jnp.array to create a Python object, so [0] indexing works
  s_3 = jnp.array([0.0])
  p = jnp.exp(jnp.stack([s_1[0], s_2[0], s_3[0]]))
  # Likelihood
  m.dist.multinomial(probs = p[career], obs=career)
end

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

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

```

## Mathematical Details

We can model a vector of frequencies using a Dirichlet distribution. For an outcome variable  $Y_i$  with  $K$  categories, the *Dirichlet* likelihood function is:

$$Y_i \sim \text{Multinomial}(\theta_i) \theta_i = \text{Softmax}(\phi_i) \phi_{[i,1]} = \alpha_1 + \beta_1 X_i \phi_{[i,2]} = \alpha_2 + \beta_2 X_i \dots \phi_{[i,k]} = 0 \alpha_k \sim \text{Normal}(0, 1) \beta_k \sim \text{Normal}(0, 1)$$

Where:

- $Y_i$  is the outcome (i.e. the vector of frequencies for each  $k$  categories) for observation  $i$ .
- $\theta_i$  is a vector unique to each observation,  $i$ , which gives the probability of observing  $i$  in category  $k$ .
- $\phi_i$  give the linear model for each of the  $k$  categories. Note that we use the softmax function to ensure that that the probabilities  $\theta_i$  form a simplex .
- Each element of  $\phi_i$  is obtained by applying a linear regression model with its own respective intercept  $\alpha_k$  and slope coefficient  $\beta_k$ . To ensure the model is identifiable, one category,  $K$ , is arbitrarily chosen as a reference or baseline category. The linear predictor for this reference category is set to zero. The coefficients for the other categories

then represent the change in the log-odds of being in that category versus the reference category.

### **Reference(s)**

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