Beta-Binomial Model

General Principles

To model the relationship between a binary outcome variable representing success counts and one or more independent variables with overdispersion , we can use the *Beta-Binomial model*.

Considerations

> **Note**
>
> - We have the same considerations as for Binomial regression.
> - A Beta-Binomial model assumes that each binomial count observation has its own probability of success. The model estimates the distribution of probabilities of success across cases, instead of a single probability of success.
> - A Beta distribution is a continuous probability distribution defined on the interval. It is characterized by two positive shape parameters, commonly denoted as and , which control the shape of the distribution. In the context of the provided equations, γ and η serve as these shape parameters. These parameters determine the shape of the distribution, allowing it to model a wide variety of random variables representing proportions or probabilities. How the γ and η parameters influence the distribution's shape can be summarized as follows:
> - When $\gamma > 1$ and $\eta > 1$, the distribution is unimodal (bell-shaped), with the peak becoming sharper as the values of γ and η increase. If γ and η are equal and greater than 1, the distribution is symmetrical and centered around 0.5.
> - If $\gamma < 1$ and $\eta < 1$, the distribution is U-shaped, with peaks at both 0 and 1.
> - The skewness of the distribution is determined by the relative values of γ and η. If $\gamma > \eta$, the distribution is skewed toward 1 (left-skewed), meaning more of the probability mass is concentrated on higher values. Conversely, if $\eta > \gamma$, the

distribution is skewed toward 0 (right-skewed), with more probability mass on lower values. The mean of the distribution is given by γ / (γ + η).

Therefore, by adjusting the shape parameters γ and η, the Beta distribution offers significant flexibility in modeling various types of prior beliefs about probabilities.

Example

Below is an example code snippet demonstrating Bayesian Beta-Binomial regression using the Bayesian Inference (BI) package. The data consist of:

1) One binary dependent variable (admit), which represents candidates' admission status.
2) One independent categorical variable representing individuals' gender (gid).
3) Additionally, we have the number of applications (applications) per gender, which will be used to account for independent rates.

The goal is to evaluate whether the probability of admission is different between genders, while accounting for differences in the number of applications between genders. This example is based on McElreath (2018).

Python

```
from BI import bi

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

# Import Data & Data Manipulation ------------------------------------------------
# Import
from importlib.resources import files
data_path = m.load.ucbadmit(only_path = True)
m.data(data_path, sep=';')
m.df["gid"] = (m.df["applicant.gender"] != "male").astype(int)

# Define model -------------------------------------------------
def model(gid, applications, admit):
    # Prior for overall concentration scaling (positive, via exponential)
    phi = m.dist.exponential(1, name='phi')
```

```
    # Priors for group-level intercepts (two groups, normal-distributed)
    alpha = m.dist.normal(0., 1.5, shape=(2,), name='alpha')

    # Shifted concentration scale (avoids too small values)
    theta = phi + 2

    # Group-specific mean success probability (mapped to [0,1] with sigmoid)
    pbar = m.link.inv_logit(alpha[gid])

    # Beta distribution parameter for "successes"
    concentration1 = pbar * theta

    # Beta distribution parameter for "failures"
    concentration0 = (1 - pbar) * theta

    # Likelihood: admissions modeled with Beta-Binomial
    m.dist.beta_binomial(
        total_count=applications,
        concentration1=concentration1,
        concentration0=concentration0,
        obs=admit
    )

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

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

```
jax.local_device_count 32
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]warmup:   0%|          | 1/1000 [00:00<13:58,  1.19it
arviz - WARNING - Shape validation failed: input_shape: (1, 500), minimum_shape: (chains=2, d
```

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
alpha[0]	-0.42	0.39	-1.00	0.19	0.02	0.03	470.21	230.54	NaN
alpha[1]	-0.30	0.42	-0.96	0.34	0.02	0.02	431.00	297.75	NaN
phi	1.00	0.78	0.00	2.04	0.04	0.04	284.63	189.52	NaN

R

```r
library(BayesianInference)

# setup platform-----------------------------------------------
m=importBI(platform='cpu')
jnp = reticulate::import("jax.numpy")
# import data -------------------------------------------------
m$data(m$load$ucbadmit(only_path = T), sep=';')
m$data_on_model$gid = jnp$array(as.integer(ifelse(m$df["applicant.gender"] == "male", 0, 1)))
m$data_on_model$applications = jnp$array(as.integer(unlist(m$df["applications"])))
m$data_on_model$admit = jnp$array(as.integer(unlist(m$df["admit"])))
# Define model ------------------------------------------------
model <- function(gid, applications, admit){
  # Parameter prior distributions
  phi = bi.dist.exponential(1, name = 'phi',shape=c(1))
  alpha = bi.dist.normal(0., 1.5, shape= c(2), name='alpha')
  t = phi + 2
  pbar = m$link$inv_logit(alpha[gid])
  gamma = pbar * t
  eta = (1 - pbar) * t
  # Likelihood

  m$dist$beta_binomial(total_count=applications, concentration1=gamma, concentration0=eta, ob
}

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

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

m$data_on_model
```

Julia

```julia
using BayesianInference

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

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

m.df["gid"] = m.df["applicant.gender"].ne("male").astype("int")

# Define model -------------------------------------------------
@BI function model(gid, applications, admit)
    # Prior for overall concentration scaling (positive, via exponential)
    phi = m.dist.exponential(1, name="phi")

    # Priors for group-level intercepts (two groups, normal-distributed)
    alpha = m.dist.normal(0., 1.5, shape=(2,), name="alpha")

    # Shifted concentration scale (avoids too small values)
    theta = phi + 2

    # Group-specific mean success probability (mapped to [0,1] with sigmoid)
    pbar = m.link.inv_logit(alpha[gid])

    # Beta distribution parameter for "successes"
    concentration1 = pbar * theta

    # Beta distribution parameter for "failures"
    concentration0 = (1 - pbar) * theta

    # Likelihood: admissions modeled with Beta-Binomial
    m.dist.beta_binomial(
        total_count=applications,
        concentration1=concentration1,
        concentration0=concentration0,
        obs=admit
    )
end

# 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:

$$Y_i \sim \text{BetaBinomial}(N_i, \gamma_i, \eta_i)$$

$$\gamma_i = p_i \tau$$

$$\eta_i = (1 - p_i)\tau$$

$$p_i = \text{logit}^{-1}(\alpha + \beta * X_i)$$

$$\tau = \phi + 2$$

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

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

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

Where:

- Y_i is the count of successes for the i-th observation, which follows a Beta-binomial distribution with N_i trials.
- γ_i represents the concentration parameter for the number of successes, derived from the probability of success, p_i, and scaled by τ.
- η_i represents the concentration parameter for failures, derived from the probability of failure $(1 - p_i)$ and also scaled by τ.
- p_i is the probability of success for the i-th observation. The logit function transforms the linear predictor (which can take any real value) into a probability value between 0 and 1.

- τ is derived from and is used as a scaling factor for the shape parameters and .
- β and α are the regression coefficient and intercept, respectively.
- is a random variable following an Exponential distribution with a rate of 1.

Reference(s)

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