

Modeling Rates/Proportions using Beta Regression with `rstanarm`

Imad Ali, Jonah Gabry and Ben Goodrich

2018-04-13

Contents

Introduction

Likelihood

Priors

Posterior

An Example Using Simulated Data

An Example Using Gasoline Data

References

Introduction

This vignette explains how to model continuous outcomes on the open unit interval using the `stan_betareg` function in the **`rstanarm`** package.

The four steps of a Bayesian analysis are

1. Specify a joint distribution for the outcome(s) and all the unknowns, which typically takes the form of a marginal prior distribution for the unknowns multiplied by a likelihood for the outcome(s) conditional on the unknowns. This joint distribution is proportional to a posterior distribution of the unknowns conditional on the observed data
2. Draw from posterior distribution using Markov Chain Monte Carlo (MCMC).
3. Evaluate how well the model fits the data and possibly revise the model.
4. Draw from the posterior predictive distribution of the outcome(s) given interesting values of the predictors in order to visualize how a manipulation of a predictor affects (a function of) the outcome(s).

Steps 3 and 4 are covered in more depth by the vignette entitled “How to Use the **`rstanarm`** Package” (`rstanarm.html`). This vignette focuses on Step 1 when the likelihood is the product of beta distributions.

Likelihood

Beta regression uses the beta distribution as the likelihood for the data,

$$f(y_i|a, b) = \frac{y_i^{(a-1)}(1-y_i)^{(b-1)}}{B(a, b)}$$

where $B(\cdot)$ is the beta function. The shape parameters for the distribution are a and b and enter into the model according to the following transformations,

$$\begin{aligned} a &= \mu \cdot \phi \\ b &= (1 - \mu) \cdot \phi \end{aligned}$$

Let $g_1(\cdot)$ be some link function. Then, in the specification of the shape parameters above, $\mu = g_1^{-1}(\mathbf{X}\boldsymbol{\beta})$, where \mathbf{X} is a $N \times K$ dimensional matrix of predictors, and $\boldsymbol{\beta}$ is a K dimensional vector of parameters associated with each predictor.

In the simplest case (with only one set of regressors), ϕ is a scalar parameter. Alternatively, it is possible to model ϕ using a second set of regressors \mathbf{Z} . In this context let $g_2(\cdot)$ be some link function that is not necessarily identical to $g_1(\cdot)$. Then $\phi = g_2^{-1}(\mathbf{Z}\boldsymbol{\gamma})$, where $\boldsymbol{\gamma}$ is a J dimensional vector of parameters associated with the $N \times J$ dimensional matrix of predictors \mathbf{Z} .

After substituting the shape parameter values in, the likelihood used in beta regression takes the following form,

$$f(y_i|\mu, \phi) = \frac{y_i^{(\mu\phi-1)}(1-y_i)^{((1-\mu)\phi-1)}}{B(\mu\phi, (1-\mu)\phi)}$$

Priors

A full Bayesian analysis requires specifying prior distributions $f(\boldsymbol{\beta})$ and $f(\phi)$ for the vector of regression coefficients and ϕ . When using `stan_betareg`, these distributions can be set using the `prior_intercept`, `prior`, and `prior_phi` arguments. The `stan_betareg` function supports a variety of prior distributions, which are explained in the **`rstanarm`** documentation (`help(priors, package = 'rstanarm')`).

When modeling ϕ with a linear predictor a full Bayesian analysis requires specifying the prior distributions $f(\boldsymbol{\beta})$ and $f(\boldsymbol{\gamma})$. In `stan_betareg` the prior distributions on $\boldsymbol{\gamma}$ can be set using the `prior_intercept_z` and `prior_z` arguments.

As an example, suppose we have K predictors and believe — prior to seeing the data — that β_1, \dots, β_K and ϕ are as likely to be positive as they are to be negative, but are highly unlikely to be far from zero. These beliefs can be represented by normal distributions with mean zero and a small scale (standard deviation). To give ϕ and each of the β s this prior (with a scale of 1, say), in the call to `stan_betareg` we would include the arguments `prior_intercept = normal(0,1)`, `prior = normal(0,1)`, and `prior_phi = normal(0,1)`.

If, on the other hand, we have less a priori confidence that the parameters will be close to zero then we could use a larger scale for the normal distribution and/or a distribution with heavier tails than the normal like the Student t distribution. **Step 1** in the “How to Use the `rstanarm` Package” vignette discusses one such example.

After fitting the model we can use the `prior_summary` function to print information about the prior distributions used when fitting the model.

Posterior

When using only a *single set of regressors*, the posterior distribution of β and ϕ is proportional to the product of the likelihood contributions, the K priors on the β_k parameters, and ϕ ,

$$f(\beta, \phi | \mathbf{y}, \mathbf{X}) \propto \prod_{i=1}^N f(y_i | a, b) \times \prod_{k=1}^K f(\beta_k) \times f(\phi)$$

When using *two sets of regressors*, the posterior distribution of β and γ is proportional to the product of the likelihood contribution, the K priors on the β_k parameters, and the J priors on the γ_j parameters,

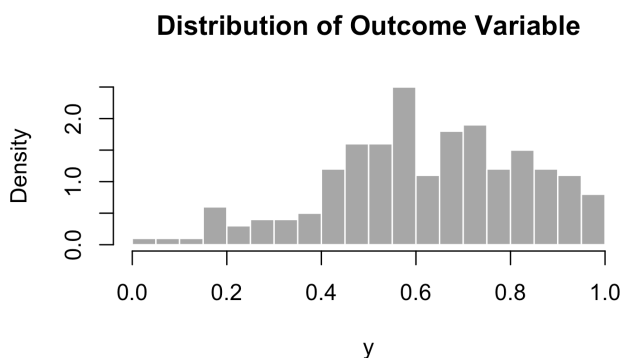
$$f(\beta, \gamma | \mathbf{y}, \mathbf{X}) \propto \prod_{i=1}^N f(y_i | a, b) \times \prod_{k=1}^K f(\beta_k) \times \prod_{j=1}^J f(\gamma_j)$$

An Example Using Simulated Data

In this example the outcome variable \mathbf{y} is simulated in a way that warrants the use of beta regression. It is worth mentioning that the data generation process is quite convoluted, which is apparent in the identification of the likelihood above.

The data simulated below uses the logistic link function on the first set of regressors and the log link function on the second set of regressors.

```
SEED <- 1234
set.seed(SEED)
eta <- c(1, -0.2)
gamma <- c(1.8, 0.4)
N <- 200
x <- rnorm(N, 2, 2)
z <- rnorm(N, 0, 2)
mu <- binomial(link = logit)$linkinv(eta[1] + eta[2]*x)
phi <- binomial(link = log)$linkinv(gamma[1] + gamma[2]*z)
y <- rbeta(N, mu * phi, (1 - mu) * phi)
dat <- data.frame(cbind(y, x, z))
hist(dat$y, col = "darkgrey", border = F, main = "Distribution of Outcome Variable", xlab = "y", br
```



The model can be fit by calling `stan_betareg`, using the appropriate link functions.

```
library(rstanarm)
fit1 <- stan_betareg (../reference/stan_betareg.html)(y ~ x | z, data = dat, link = "logit", link.p
chains = CHAINS, cores = CORES, seed = SEED, iter = ITER)
fit2 <- stan_betareg (../reference/stan_betareg.html)(y ~ -1 + x , data = dat, link = "logit", link
chains = CHAINS, cores = CORES, seed = SEED, iter = ITER)
round(coef(fit1), 2)
round(coef(fit2), 2)
```

(Intercept)	x (phi)_(Intercept)	(phi)_z
0.94	-0.20	1.85
		0.32

x (phi)_(Intercept)
0.01
1.09

For clarity we can use `prior_summary` to print the information about the prior distributions used to fit the models. The priors used in `fit1` are provided below.

```
prior_summary (../reference/prior_summary.stanreg.html)(fit1)
```

```
Priors for model 'fit1'
-----
Intercept (after predictors centered)
~ normal(location = 0, scale = 10)

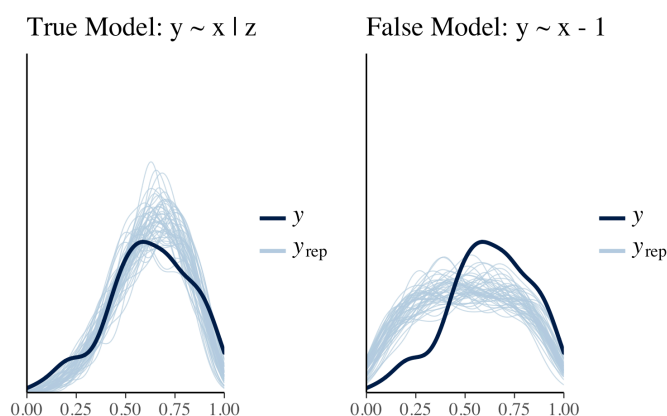
Coefficients
~ normal(location = 0, scale = 2.5)
**adjusted scale = 1.22

Intercept_z (after predictors centered)
~ normal(location = 0, scale = 10)

Coefficients_z
~ normal(location = 0, scale = 2.5)
**adjusted scale = 1.24
-----
See help('prior_summary.stanreg') for more details
```

The usual posterior analyses are available in **rstanarm**. The plots below illustrate simulated values of the outcome variable. The incorrect model noticeably fails to capture the top of the distribution consistently in comparison to the true model.

```
library(ggplot2)
library(bayesplot)
bayesplot_grid (http://www.rdocumentation.org/packages/bayesplot/topics/bayesplot_grid)(
  pp_check (../reference/pp_check.stanreg.html)(fit1), pp_check (../reference/pp_check.stanreg.html
xlim = c(0,1),
ylim = c(0,4),
titles = c("True Model: y ~ x | z", "False Model: y ~ x - 1"),
grid_args = list(ncol = 2)
)
```



We can also compare models by evaluating the expected log pointwise predictive density (`elpd`), which can be calculated using the `loo` method, which provides an interface for **rstanarm** models to the functionality in the **loo** package.

```
loo1 <- loo (../reference/loo.stanreg.html)(fit1)
loo2 <- loo (../reference/loo.stanreg.html)(fit2)
compare_models (../reference/loo.stanreg.html)(loo1, loo2)
```

Model comparison:
(negative 'elpd_diff' favors 1st model, positive favors 2nd)

```
elpd_diff      se
-80.2         12.0
```

The difference in `elpd` is negative indicating that the expected predictive accuracy for the first model is higher.

An Example Using Gasoline Data

In some applied contexts it may be necessary to work with an outcome variable that is a proportion. If the proportion is bound on the open unit interval then beta regression can be considered a reasonable estimation method. The `betareg` package provides a dataset on the proportion of crude oil converted to gasoline after distillation and fractionation. This variable is defined as `yield`. Below `stan_betareg` is used to model `yield` as a function of temperature, pressure, and the batch of conditions.

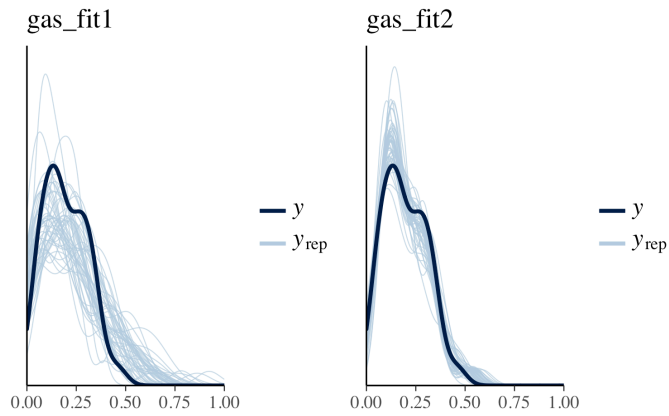
```
library(rstanarm)
data("GasolineYield", package = "betareg")
gas_fit1 <- stan_betareg (../reference/stan_betareg.html)(yield ~ temp + batch, data = GasolineYield,
  chains = CHAINS, cores = CORES, seed = SEED, iter = ITER)
gas_fit2 <- stan_betareg (../reference/stan_betareg.html)(yield ~ temp + batch | pressure,
  data = GasolineYield, link = "logit",
  chains = CHAINS, cores = CORES, seed = SEED, iter = ITER)
round(coef(gas_fit1), 2)
round(coef(gas_fit2), 2)
```

(Intercept)	temp	batch1	batch2	batch3	batch4
-5.12	0.01	1.33	0.94	1.11	0.71
batch5	batch6	batch7	batch8	batch9	(phi)
0.78	0.72	0.27	0.24	0.11	12.42

(Intercept)	temp	batch1	batch2
-6.08	0.01	1.69	1.27
batch3	batch4	batch5	batch6
1.52	1.03	1.10	1.01
batch7	batch8	batch9 (phi)_	(Intercept)
0.52	0.47	0.37	5.43
(phi)_pressure			
0.03			

The plots below illustrate simulated values of gasoline yield. While the first model accounts for variation in batch conditions its predictions looks somewhat uniform rather than resembling the peaked and right-skewed behavior of the true data. The second model does a somewhat better job at capturing the shape of the distribution, however its location is off as it is centered around 0.50 rather than 0.20.

```
library(ggplot2)
bayesplot_grid (http://www.rdocumentation.org/packages/bayesplot/topics/bayesplot_grid)(
  pp_check (../reference/pp_check.stanreg.html)(gas_fit1), pp_check (../reference/pp_check.stanreg.html)(gas_fit2),
  xlim = c(0,1),
  ylim = c(0,5),
  titles = c("gas_fit1", "gas_fit2"),
  grid_args = list(ncol = 2)
)
```



```
gas_loo1 <- loo (../reference/loo.stanreg.html)(gas_fit1)
gas_loo2 <- loo (../reference/loo.stanreg.html)(gas_fit2)
compare_models (../reference/loo.stanreg.html)(gas_loo1, gas_loo2)
```

Model comparison:
(negative 'elpd_diff' favors 1st model, positive favors 2nd)

elpd_diff	se
33.3	3.6

Evaluating the expected log predictive distribution using `loo` reveals that the second of the two models is preferred.

References

Ferrari, SLP and Cribari-Neto, F (2004) “Beta Regression for Modeling Rates and Proportions”. *Journal of Applied Statistics*. Vol. 31, No. 07, p799-815.