The beta-binomial family of probability mass functions is most commonly parameterized in terms of two parameters α and β,

$$\text{beta-binomial}(n \mid N, \alpha, \beta) = \binom{N}{n} \frac{B(n + \alpha, N - n + \beta)}{B(\alpha, \beta)}$$

where $B(x, y)$ is the beta function

$$B(x, y) = \frac{\Gamma(x)}{\Gamma(y)} \Gamma(x + y).$$

The means and variances of each member of this family are given by

$$\begin{aligned} \mathbb{M} &= N \frac{\alpha}{\alpha + \beta} \\ \mathbb{V} &= N \frac{\alpha}{\alpha + \beta} \left(1 - \frac{\alpha}{\alpha + \beta}\right) \left(1 - \frac{N - 1}{\alpha + \beta + 1}\right). \end{aligned}$$

This particular form arises when we convolve a binomial probability mass function with a beta density function for the probability parameter,

$$\int_0^1 \mathrm{d}p \, \text{binomial}(n \mid N, p) \, \text{beta}(p \mid \alpha, \beta) = \text{beta-binomial}(n \mid N, \alpha, \beta).$$

That said there are other parameterizations that can be more useful in practice.

For example consider the new parameters

$$\begin{aligned} p &= \frac{\alpha}{\alpha + \beta} \\ \phi &= \frac{1}{\alpha + \beta + 1}, \end{aligned}$$

or equivalently

$$\begin{aligned} \alpha &= \frac{1 - \phi}{\phi} \, p \\ \beta &= \frac{1 - \phi}{\phi} \, (1 - p). \end{aligned}$$

Note that both p and ϕ are constrained to the unit interval $[0, 1]$.

In terms of these parameters the means and variances are now given by

$$\begin{aligned} \mathbb{M} &= N \; p \\ \mathbb{V} &= N \; p \, (1 - p) \, (1 + (N - 1) \, \phi). \end{aligned}$$

When $\phi = 0$ this reduces to the mean and the variance of the binomial distribution,

$$\begin{aligned} \mathbb{M} &= N \, p \\ \mathbb{V} &= N \, p \, (1 - p). \end{aligned}$$

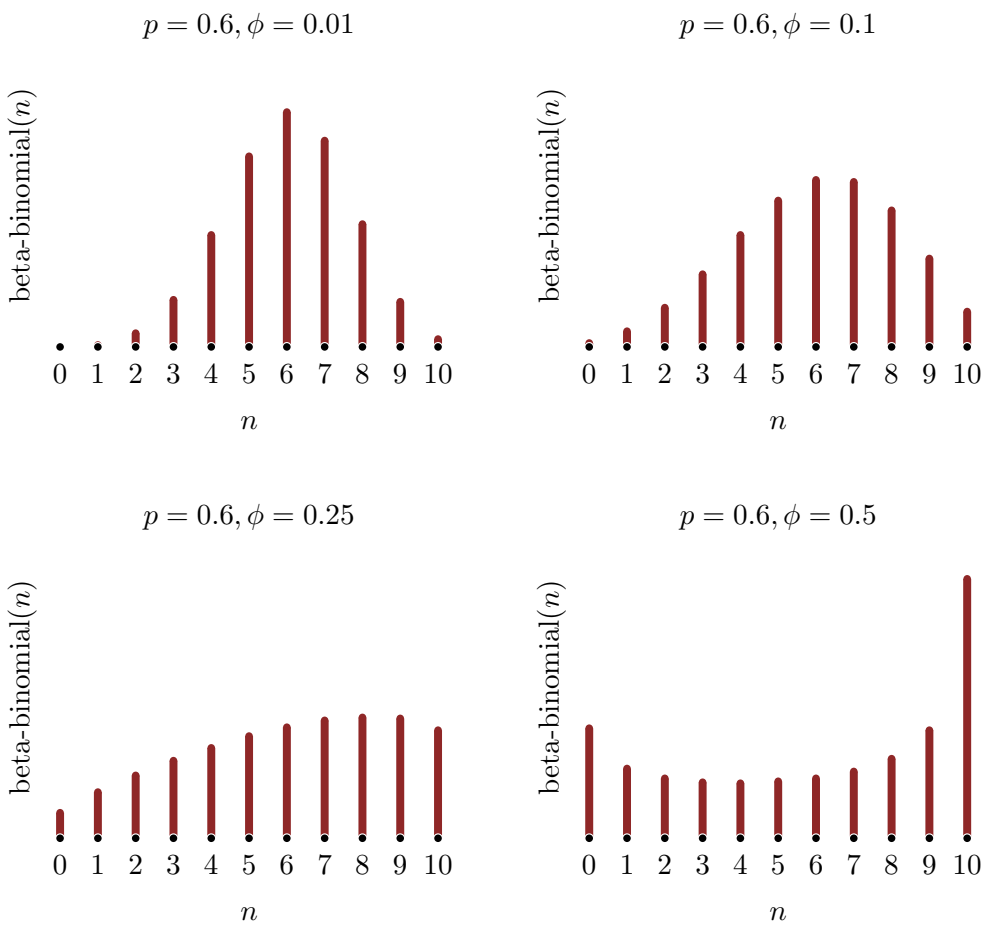
When $0 < \phi \leq 1$ the mean stays the same but the variance increases until it saturates at $N^2 \, p \, (1 - p)$.

In fact when $\phi = 0$ the beta-binomial probability family reduces to the binomial family exactly. Consequently we can interpret the beta-binomial family as an over-dispersed generalization of the binomial family, and in this parameterization ϕ allows us to control the precise amount of over-dispersion. A prior model on ϕ that suppresses values close to 1 directly controls the amount of possible over-dispersion.

This alternative parameterization is straightforward to implement in `Stan`,

```
functions {
  // 0 <= p   <= 1
  // 0 <= phi <= 1
  real beta_binomial_alt_lpdf(int n, int N, real p, real phi) {
    real c = (1 - phi) / phi;
    return beta_binomial_lpdf(n | N, c * p, c * (1 - p));
  }

  int beta_binomial_alt_rng(int n, int N, real p, real phi) {
    real c = (1 - phi) / phi;
    return beta_binomial_rng(N, c * p, c * (1 - p));
  }
}
```



The only potential problem with this implementation is that if the data are consistent with a pure binomial distribution then inferences will concentrate around $\phi = 0$ where the $(1 - \phi)/\phi)$ term can cause numerical issues. If you encounter this then you might consider the alternative parameter

$$\psi = \text{logit}\, \phi.$$

In this case

$$\alpha = \exp(-\psi)\, p$$
$$\beta = \exp(-\psi)\, (1 - p).$$

and

```
functions {
  // 0 <= p   <= 1
```

```
  // -inf <=psi < +inf
  real beta_binomial_alt_lpdf(int n, int N, real p, real psi) {
    real c = exp(-psi);
    return beta_binomial_lpdf(n | N, c * p, c * (1 - p));
  }

  int beta_binomial_alt_rng(int n, int N, real p, real psi) {
    real c = exp(-psi);
    return beta_binomial_rng(N, c * p, c * (1 - p));
  }
}
```

The challenge with this implementation is designing a prior model for ψ that concentrates around $-\infty$. I recommend designing an appropriate beta prior model for ϕ and then transforming it to a corresponding prior model for ϕ. In particular if the beta(a, b) probability density function has the right shape then you could use

```
functions {
  // a > 0
  // b > 0
  real logit_beta_lpdf(int x, real a, real b) {
    return a * log_inv_logit(x) + b * log_inv_logit(-x) - lbeta(a, b);
  }
}

...

parameters {
  ...
  real psi;
  ...
}

...

model {
  ...
  psi ~ logit_beta(a, b);
  ...
}
```