Binomial and negative binomial distribution

Parametrisation

The Binomial distribution is

$$\text{Prob}(y) = \binom{n}{y} p^y (1-p)^{n-y}$$

for responses $y = 0, 1, 2, \ldots, n$, where

n: number of trials.

p: probability of success in each trial.

The negative binomial distribution is

$$\text{Prob}(n) = \binom{n-1}{y-1} p^y (1-p)^{n-y}$$

for given $y = 1, 2, \ldots$ and response $n - y = 0, 1, 2, \ldots$.

Link-function

The mean and variance of y are given in the binomial case as

$$\mu = np \qquad \text{and} \qquad \sigma^2 = np(1-p)$$

and the probability p is linked to the linear predictor by

$$p(\eta) = \frac{\exp(\eta)}{1 + \exp(\eta)}$$

Hyperparameters

None.

Hyperparameter spesification and default values

doc The Binomial likelihood

hyper

survival FALSE

discrete TRUE

link default logit loga cauchit probit cloglog loglog log sslogit logitoffset quantile pquantile robit sn

pdf binomial

Specification

- family = `binomial`
- Required arguments: y and n (keyword `Ntrials`)
- Optional argument: `variant=0` for binomial (default), and `variant=1` for the negative binomial.

Expert version

There is also an "expert" version were you are supposed to know what you are doing. Here, we allow y and n to be non-integers (whatever that means), however, the condition $0 \leq y \leq n$ apply. The normalizing constant is computed as above using the integer part of y and n. This is similar to using `floor(y)` and `floor(n)` in `R`. The marginal likelihood estimate will in this case make less sense.

- family = `xbinomial`
- Required arguments: y and n (keyword `Ntrials`)
- Optional argument: `scale=q`, which scales the probability with $0 < q \leq 1$ into p', where

  $$p' = qp(\eta).$$

  By default, $q = 1$. Note that "fitted values" will still be be $p(\eta)$.

doc The Binomial likelihood (expert version)

hyper

survival FALSE

discrete TRUE

link default logit loga cauchit probit cloglog loglog log sslogit logitoffset quantile pquantile robit sn

pdf binomial

status experimental

Examples

In the following example we estimate the parameters in a simulated example with binomial responses.

```
## binomial
n=100
a = 1
b = 1
z = rnorm(n)
eta = a + b*z
formula <- y ~ 1 + z
prob = exp(eta)/(1 + exp(eta))

Ntrials = sample(1:10, size=n, replace=TRUE)
y = rbinom(n, size = Ntrials, prob = prob)
data = data.frame(y, z, Ntrials)
r = inla(formula, family = "binomial", data = data, Ntrials=Ntrials)
summary(r)

## negative binomial
y = sample(1:3, size=n, replace=TRUE)
Ntrials = y + rnbinom(n, size = y, prob = prob)
r = inla(formula,
         family = "binomial",
         control.family = list(variant = 1),
```

```
         Ntrials = Ntrials,
         data = data.frame(y, x, Ntrials))
summary(r)
```

In the following example we estimate the parameters in a simulated example with binomial responses using the `scale`-argument as well. This require the use of the expert-version "xbinomial".

```
n <- 10000
x <- rnorm(n, sd = 1)
q <- runif(n)
eta <- 0.88 + 0.77*x
p <- q * 1.0/(1+exp(-eta))
ntrials <- sample(1:25,  size=n, replace=TRUE)
y <- rbinom(n = n, size=ntrials, prob = p)
r <- inla(y ~ 1 + x,
         family = "xbinomial",
         Ntrials = ntrials,
         scale = q,
         data = data.frame(y, x, q, ntrials))
summary(r)
```

Notes

- If the response is a `factor` it must be converted to $\{0, 1\}$ before calling `inla()`, as this conversion is not done automatic (as for example in `glm()`).
- This version of the negative binomial mimics the binomial distribution, and the "data" kind of enter in the `Ntrials` argument (as y is pre-determinded) which both can appear, and should appear, strange. There is also an alternative implementation, `family="nbinomial"`, which mimics the Poisson distribution.