11 Generalized linear models

(ch 7,8 in BMUW). Link functions are used to map the covariate effects (the linear predictior) into the parameter of interest, usually representing the mean. The link function should be monotonic and differentiable. Common situations are:

Response variables in $\mathbb{R}$. Normal(μ, σ^2) regression model with linear link: $\mu = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$.

Response variables in $\mathbb{R}^+$. Try log-transformed variables, and use the normal regression model. (Effectively, this is log-normal model of original variables). Alternatively, try some other transformations to achieve approximately 'normally' distributed data for normal regression modeling. Also, try gamma, exponential, inverse gaussian, or Weibull distributions. E.g. gamma distribution could be parameterized as $\Gamma(\mu\alpha, \alpha)$ which has mean μ which could be modeled via log-link $\log(\mu) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$.

Response variables are binary. Use Bernoulli(p) distribution. If the response measures the number of successes out of n trials, then Binomial(n, p) distribution. A usual link function is logit: $\text{logit}(p) = \log(p/(1-p)) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$ which is used to map covariate effects (in $\mathbb{R}$) into p (in $(0, 1)$). Another choice could be probit-link:

$$\phi(p_i) = F^{-1}(p_i) = \beta X_i, \text{ where } F \text{ is the cumulative distribution of } \mathrm{N}(0, 1)$$

Response variables in $N = \{0, 1, 2, 3, ...\}$. Use Poisson distribution. Usual link function is $\log(\lambda) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$. If the data seem overdispersed (variance is larger than mean), then Negative Binomial distribution might be used.

Other types of response variables: if y are in a range (a, b), use rescaling to $(0, 1)$ and then Beta(α, β) distribution. Alternatively, transform to $\log((y-a)/(b-y))$ and use normal regression. If y are in $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$, then a model based on differences of Poisson latent variables.

In BUGS, some link functions are directly available: log, logit, probit, cloglog. With these, the effects of explanatory variables (in $\mathbb{R}$) can be mapped to a different space, e.g. the range $[0, 1]$.

In BUGS code, these available link functions are used e.g.:

```
logit(p[i]) <- inprod(beta[],X[i,])
```

Alternatively, we could write:

```
p[i] <- exp(eta[i])/(1+exp(eta[i]))
eta[i] <- inprod(beta[],X[i,])
```

where the expression for p_i is solved as a function of η_i, which in turn is given as the linear sum. In this way, any suitable link functions could be constructed whenever they are not readily among those provided in BUGS.

Sometimes, probit links in BUGS may cause numerical over/underflow problems. Then it may be better to use logit, or to truncate the tails at $(-\xi, \xi)$:

```
probit(p[i]) <- eta[i]*(1-step(abs(eta[i])-xi))
             -xi*step(-xi-eta[i]) + xi*step(eta[i]-xi)
```

Or in short:

```
probit(p[i]) <- max(min(eta[i],xi),-xi)
```

where `eta[i]` is the linear predictor.

11.1 Poisson regression: aircraft damage

Example 7.1 in BMUW, p. 245. The data consists of 30 observations: the number of damaged locations of the aircraft. There were three explanatory variables: type of aircraft (binary: 0 or 1), bombload in tons, months of crew experience. The generalized linear model is

$$\begin{aligned} &\text{damage}_i \sim \text{Poisson}(\lambda_i) \\ &\log(\lambda_i) = \beta_1 + \beta_2 \text{type}_i + \beta_3 \text{load}_i + \beta_4 \text{exper}_i \end{aligned}$$

For monitoring and reporting of results, we need posterior summaries of $\exp(\beta_j)$, so that the following is added in WinBUGS code:

```
for(i in 1:4){B[j] <- exp(beta[j])}
```

The data set is:

```
list(
damage=c(0, 1, 0, 0, 0, 0, 1, 0, 0, 2, 1, 1, 1, 1, 2, 3, 1, 1, 1, 2,
0, 1, 1, 2, 5, 1, 1, 5, 5, 7),
type=c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1),
load=c(4,  4,  4,  5,  5,  5,  6,  6,  6,  7,  7,  7,  8,  8,  8,  7,
7,  7, 10, 10, 10, 12, 12, 12,  8,  8,  8, 14, 14, 14),
exper=c(91.5,  84.0,  76.5,  69.0,  61.5,  80.0,  72.5,  65.0,  57.5,  50.0,
103.0,  95.5,  88.0,  80.5,  73.0, 116.1, 100.6,  85.0,  69.4,  53.9, 112.3,
96.7, 81.1, 65.6, 50.0, 120.0, 104.4,  88.9,  73.7,  57.8))
```

Set diffuse normal priors `beta[j]` $\sim$ `dnorm(0,0.001)`. Run the BUGS model and compute results.

```
model{
   for (i in 1:30){
       damage[i] ~ dpois( lambda[i] )
       log(lambda[i]) <- beta[1]+beta[2]*type[i]+beta[3]*load[i]+beta[4]*exper[i]
   }
       #
       # prior
       for (j in 1:4){
           beta[j]~dnorm( 0.0, 0.001 )
           B[j] <- exp( beta[j] )
   }}
```

See table 7.5, p. 246 in BMUW. Based on posterior mean of `B[2]`, aircraft of type 1 is expected to get about twice as damaged as type 0. Based on posterior mean of `B[3]`, every ton of bomb load increases the expected damages by 20%. Based on posterior mean of `B[4]`, every additional month of experience reduces the expected damage by 1%.

Expected values for scenarios

Consider different scenarios: worst case, best case, typical. To assess each scenario, we can compute posterior estimates of damage under each. For example, for aircraft type 1, monitor the following quantity with selected values for bomb load and experience:

```
type1.scenario <- exp(beta[1]+beta[2]+beta[3]*bl + beta[4]*ex)
bl <- ranked(load[],1)   # to get minimum
ex <- ranked(exper[],30) # to get maximum
```

This is the best case scenario with minimum bomb load and maximum experience.

Finally, DIC can be used to select those covariates that give the best model.

11.1.1 Logic of unbiased estimators: Poisson example

In Poisson models, parameter λ is the conditional mean of the variable: $E(X \mid \lambda) = \lambda$. Typically, point estimates based on data alone can be calculated, and the mean of observations $\bar{X}$ can be used to estimate λ. In frequentist theory, this is also the unbiased estimator for λ. Unbiased estimators are usually preferred over other possible estimators in frequentist statistics. Yet, the Poisson model provides an interesting example about the seemingly logical preference of unbiased estimators in comparison to the bayesian approach.

In bayesian inference, there is no concept of 'unbiased estimators' in the same way as it is defined in frequentist statistical theory. An 'estimator' is a frequentist concept: for some parameter θ we suggest some *function of data*, $f(X)$, as its estimator, so that we write $\hat{\theta} = f(X)$. This estimator has a sampling distribution, given the true (but unknown) parameter θ. Different estimators have different properties. An estimator is called unbiased if the expectation of the sampling distribution is:

$$E(f(X) \mid \theta) = \int f(X)\pi(X \mid \theta)\mathbf{d}X = \theta.$$

This is a statement about the conditional expectation of $f(X)$, given the true (albeit unknown) parameter value. In bayesian inference we quantify our uncertainty about θ conditionally given the data X, i.e. obtain posterior distribution $\pi(\theta \mid X)$. The posterior density itself contains the total state of knowledge we have, and we can summarize this posterior by credible intervals, means, modes and medians, if we like. With Poisson distribution, assuming data n, the unbiased estimator $f(n)$ has to satisfy the following:

$$E(f(n) \mid \lambda) = \sum_{n=0}^{\infty} \exp(-\lambda)\frac{\lambda^n}{n!} f(n) = \lambda,$$

In other words:

$$\sum_{n=0}^{\infty} \frac{\lambda^n}{n!} f(n) = \lambda \exp(\lambda) =: g(\lambda).$$

This equation is the same as Taylor[1] expansion around 0 for function $g(\lambda)$. Hence, the equation is satisfied when $f(n)$ is $g^{(n)}(0)$, that is:

$$\begin{array}{lcll} f(0) = g(\lambda) \mid_{\{\lambda=0\}} & = & 0\exp(0) & = & 0 \\ f(1) = g'(\lambda) \mid_{\{\lambda=0\}} & = & \exp(0) + 0\exp(0) & = & 1 \\ f(2) = g''(\lambda) \mid_{\{\lambda=0\}} & = & \exp(0) + \exp(0) + 0\exp(0) & = & 2 \\ f(3) = g'''(\lambda) \mid_{\{\lambda=0\}} & = & \ldots & & \\ \vdots & & & & \end{array}$$

Unbiased estimator for λ is thus $f(n) = n$. But what if the parameter to be estimated would be some function of λ, say λ^2? As above, the solution is found from

$$\sum_{n=0}^{\infty} \frac{\lambda^n}{n!} f(n) = \lambda^2 \exp(\lambda) =: g(\lambda),$$

which corresponds to Taylor series expansion

$$\begin{array}{lcll} f(0) = g(\lambda) \mid_{\{\lambda=0\}} & = & 0^2\exp(0) & = & 0 \\ f(1) = g'(\lambda) \mid_{\{\lambda=0\}} & = & 2\lambda\exp(\lambda) + \lambda^2\exp(\lambda) \mid_{\{\lambda=0\}} & = & 0 \\ f(2) = g''(\lambda) \mid_{\{\lambda=0\}} & = & 2\exp(\lambda) + 2\lambda\exp(\lambda) + 2\lambda\exp(\lambda) + \lambda^2\exp(\lambda) \mid_{\{\lambda=0\}} & = & 2 \\ f(3) = g'''(\lambda) \mid_{\{\lambda=0\}} & & & = & 6 \\ \vdots & & & & \\ f(n) & & & = & n(n-1) \\ \vdots & & & & \end{array}$$

Unbiased estimator for λ^2 is thus

$$f(n) = \begin{cases} 0 & \text{if } n = 0 \text{ or } 1 \\ n(n-1) & \text{if } n > 1 \end{cases}$$

Which seems peculiar if we happen to get $n = 1$. The estimate would then be zero, but if λ was truly zero, it would have been impossible to obtain $n = 1$. Furthermore, unbiased estimator for $1/\lambda$ does not exist, and the unbiased estimator for $\exp(-\lambda)$ is

$$f(n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n > 0 \end{cases}.$$

In Bayesian inference, we can always summarize the posterior distribution of any $h(\theta)$ by e.g. posterior mean. Sometimes, it is even possible to use improper uniform density over entire real axis (but we need to check that the posterior density is a proper density, i.e. integrates to one). In this example, we could use this prior: $\pi(\lambda) = 1$ over entire positive real axis. Now we obtain:

$$E(h(\lambda) \mid n) = \int_0^{\infty} \exp(-\lambda) \frac{\lambda^n}{n!} h(\lambda) \mathbf{d}\lambda.$$

And the posterior mean for parameter $1/\lambda$ exists, and it is $1/n$. (Inside the integral, we have $\Gamma(\lambda \mid n, 1)$ density multiplied by $1/n$, and the density integrates to one). In BUGS, we usually simply monitor the parameter transformation of interest.

[1] Taylor expansion for function $g(\lambda)$ at point a is $g(\lambda) = \sum_{n=0}^{\infty} \frac{g^{(n)}(a)}{n!}(\lambda - a)^n$, where $g^{(n)}$ is the nth derivative.

11.2 Contingency tables and Poisson log-linear model

Consider the 2×2 contingency table with counts $N_{i,j} \sim \text{Poisson}(\lambda_{i,j})$. with CR constraints this could be implemented:

```
for(i in 1:2){for(j in 1:2){
x[i,j] ~ dpois(lambda[i,j])
log(lambda[i,j]) <- lambda0 + a[i] + b[j] + g[i,j] }}
a[1] <- 0
b[1] <- 0
for(i in 1:2){g[i,1]<-0}
for(j in 2:2){g[1,j]<-0}
for(i in 1:2){a[i] ~ dnorm(0,0.001); b[i] ~ dnorm(0,0.001)
for(j in 1:2){g[i,j] ~ dnorm(0,0.001)}}
lambda0 ~ dnorm(0,0.001)
```

For example, use the data:

	Heart disease		
	yes	no	
Serum cholesterol < 260	51	992	1043
Serum cholesterol > 260	41	245	286
	92	1237	1329

```
list(x=structure(.Data=c(51,992,41,245),.Dim=c(2,2)))
```

The variables are mutually independent if the interaction term is zero. This can be studied by computing its posterior distribution and checking if zero is included in the range of the distribution. In this example it seems that its 95% CI is [-1.64,-0.7662], significantly away from zero. Also, computing DIC for models with and without interaction, gives a lower DIC for the model with interaction.

11.3 Binomial regression: senility symptoms

The response is a binary variable, from $n = 54$ individuals. Symptoms of senility: yes/no (1/0). Hence, the model can be done by either Bernoulli distribution or Binomial (with $n = 1$). The explanatory variable is WAIS score, which is a discrete quantitative variable.

```
for(i in 1:n){
senility[i] ~ dbin(p[i],1)   # or dbern(p[i])
logit(p[i]) <- beta0 + beta1*wais[i]
}
beta0 ~ dnorm(0,0.001)
beta1 ~ dnorm(0,0.001)
or <- exp(beta1)  # to monitor odds ratio
```

The value of wais which corresponds to disease probability of 0.5 can be examined by `w.half.prob <- - beta0/beta1` which is solved from $0 = \beta_0 + \beta_1 X$. The data are

```
list(
senility[i]=c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
wais=c(9, 13,  6,  8, 10,  4, 14,  8, 11,  7,  9,  7,  5, 14, 13,
16, 10, 12, 11, 14, 15, 18,  7, 16,  9, 9, 11, 13, 15, 13, 10, 11,
6, 17, 14, 19,  9, 11, 14, 10, 16, 10, 16, 14, 13, 13, 9, 15, 10,
11, 12, 4, 14, 20))
```

Run the BUGS model and compute results. Plot the disease probability as a function of wais score in the range 0-20. Try also probit link function. In that case, numerical overflow can happen and the truncation trick could be used:

```
probit(p[i]) <- eta[i]*(1-step(abs(eta[i])-xi))
                -xi*step(-xi-eta[i])
                +xi*step(eta[i]-xi)
eta[i] <- beta0 + beta1*wais[i]
```

in which e.g. `xi=5` for truncation point. A shorter code for the same would be `probit(p[i]) <- max(min(eta[i],xi),-xi)`.

11.4 Bernoulli response with Bernoulli covariate with missing values

The following data represents combined pathogenic (*Yersinia E, Yersinia P, Listeria M*) status of 26 pigs from 15 farms, based on faecal samples. The same pigs were sampled again at slaughter. Here, the intestinal sample results are shown from this follow-up sample. It is of interest to study if there is tendency for positive animals to remain positives (or to become negative) at the second sampling. Likewise, for negative animals to become positives (or to remain negative) at the second sampling. This could be described by transition probabilities $P_{i,j}$ for transitions $i \rightarrow j$, with $i, j \in \{0, 1\}$. Some of the samples are missing, both for the 1st and 2nd sampling. Hence, we do not know all the starting states or the final states. The final states become automatically predicted from the model, but for the initial states we need to define a model. This might be based on the overall prevalence q_k at the same farm k, which can be estimated from the observed samples. Furthermore, we could study the effect of the farm type on q_k so that $q_k \in \{Q_1, Q_2, Q_3\}$ based on the three farm types: organic, small conventional, large conventional. (=Categorical explanatory factors). Posterior distribution is thus computed for parameters P, Q. No CR or STZ constraints are needed in this simple parametrization.

```
 model{
for(i in 1:15){
for(j in 1:26){
slaughter_total_intestinal[i,j] ~  dbern(p[i,j])
p[i,j] <- equals(faecal_total[i,j],0)*P[1,2] + equals(faecal_total[i,j],1)*P[2,2]
faecal_total[i,j] ~ dbern(q[i])
# q to predict missing initial state based on observed states from i.
}
```

```
q[i] <- equals(organic[i],1)*Q[1] + equals(sma[i],1)*Q[2] + equals(big[i],1)*Q[3]
}
for(k in 1:3){Q[k] ~ dunif(0,1) }
P[1,1] <- 1-P[1,2];  P[2,1]<-P[2,2]
P[1,2] ~ dunif(0,1); P[2,2] ~ dunif(0,1)
}
list(organic=c(1,0,0,1,0,0,1,0,0,1,0,0,1,0,0),sma=c(0,1,0,0,1,0,0,1,0,0,1,0,0,1,0),
big=c(0,0,1,0,0,1,0,0,1,0,0,1,0,0,1),
faecal_total=structure(.Data=c(
1,1,1,0,1,1,0,0,0,1,0,0,0,1,1,0,1,1,1,1,1,0,1,0,0, NA,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, NA,
1,0,0,0,0,1,1,0,1,1,1,1,0,0,0,0,1,1,1,0,0,0,0,1,1, NA,
0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,1,0,0,0,1,0, NA,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, NA,
0,0,0,0,0,1,1,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0, NA,
0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, NA,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, NA,
0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0, NA, NA, NA, NA, NA,
0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1, NA, NA, NA, NA,
0,0,0,0,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0, NA,
1,0,0,0,0,0,1,0,0,0,1,0,1,0,0,0,0,1,0,0,1,0,0,1, NA, NA,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, NA,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, NA, NA, NA, NA, NA),.Dim=c(15,26)),
slaughter_total_intestinal=structure(.Data=c(
0,0,0,1,1,1,1,0,0,0,0,1,NA,0,1,1,0,1,0,0,0,0,0,0,1, NA,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, NA,
NA,1,NA,0,NA,1,NA,1,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,1,1, NA,
0,1,0,0,NA,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0, NA,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, NA,
1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,NA,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,NA,
0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,NA,
0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,NA, NA, NA, NA, NA,
0,0,1,1,0,0,1,0,1,0,0,1,0,0,0,0,0,1,0,0,0,1, NA, NA, NA, NA,
0,0,0,0,0,0,1,0,0,0,1,1,0,0,0,0,0,0,0,1,1,1,0,0,1, NA,
0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0, NA, NA,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, NA,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0, NA, NA, NA, NA, NA),.Dim=c(15,26)))
```

11.5 Extensions

The easiest extensions can be based on simple transformations. For example, to get log-normal density, we can use it directly in BUGS:

```
y[i] ~ dlnorm(mu[i],tau)
```

or indirectly:

```
yy[i] <- log(y[i])
yy[i] ~ dnorm(mu[i],tau)
```

Also, the χ^2_μ distribution could be modeled directly in BUGS, or by using $\Gamma(\mu/2, 1/2)$.

The inverse-gamma density (not directly available) could be implemented:

```
inv.y[i] <- 1/y[i]
inv.y[i] ~ dgamma(a[i],tau)
a[i] <- mu[i]*tau
log(mu[i]) <- ...
```

based on $1/Y$ with $Y \sim \Gamma(\mu\tau, \tau)$.

When modeling some distribution indirectly by using other available distributions, be careful in checking that the parametrization is correct! In general, this concerns any specification of distributions, directly or indirectly. For example, in the literature there are some different conventions or alternative ways to write parameters for some distributions, e.g. Weibull or Negative Binomial. Mistakes are easy to make, even with normal distribution: with $\mathrm{N}(0, 2)$, in different literature we might have either $\sigma^2 = 2$, or $\sigma = 2$ or $1/\sigma^2 = \tau = 2$. Also, remember that in different parameterizations, the posterior will be the same only if the priors are equivalent under both parameterizations. For example, with logistic normal prior $\mathrm{logit}(p) \sim \mathrm{N}(0, 1.69^2)$, we get roughly uniform prior for p.

When a distribution cannot be constructed via other available distributions in BUGS, specification of arbitrary distributions for the data (the likelihood function) can be done using zeros trick. This is based on observing that

$$P(y \mid \theta) = \prod_{i=1}^{n} e^{L_i} = \prod_{i=1}^{n} \frac{e^{-(-L_i)}(-L_i)^0}{0!} = \prod_{i=1}^{n} P_{\text{Poisson}}(0 \mid -L_i)$$

where $L_i = \log(P(y_i \mid \theta))$ is the log-likelihood function. Hence, the model likelihood can be expressed via Poisson likelihoods for pseudo-observations of zeros, with poisson parameter $-L_i$. To ensure the parameter is positive, we can add a constant C. This does not effect the results since it is equivalent of multiplying the resulting unnormalized posterior density by constant $\exp(-nC)$. Therefore:

```
C <- 10000
for(i in 1:n){
zeros[i] <- 0
zeros[i] ~ dpois(zeros.mean[i])
zeros.mean[i] <- -L[i]+C
L[i] <- ...  # here the log-likelihood for your model
}
```

Computational tips: using grouped data can be much more faster to compute than individual data. Therefore, instead of writing a loop over all individuals, we could write a loop over all distinct observed values, and specify the corresponding frequencies (counts), n_k for each $k = 1, \ldots, K$. The full likelihood would be of the form: $\prod_{k=1}^{K} (\text{like}_k)^{n_k}$. Hence, the log-likelihood of the k:th distinct value is $L_k n_k$.

```
C <- 10000
for(k in 1:K){   # loop over K distinct values
zeros[k] <- 0
zeros[k] ~ dpois(zeros.mean[k])
zeros.mean[k] <- -L[k]*n[k]+C
L[k] <- ...  # here the log-likelihood for your model
}
```

Alternatively, the zeros trick can be replaced by ones trick, based on the following:

$$P(y \mid \theta) = \prod_{i=1}^{n} e^{L_i} = \prod_{i=1}^{n} (e^{L_i})^1 (1 - e^{L_i})^0 = \prod_{i=1}^{n} P_{\text{Bernoulli}}(1 \mid e^{L_i})$$

To ensure that the bernoulli probability $\exp(L_i)$ is always lower than one, we can multiply each likelihood term with $\exp(-C)$. Again, this does not affect the results because the unnormalized posterior density is multiplied by constant.

```
C <-10000
for(i in 1:n){
ones[i] <- 1
ones[i] ~ dbern(p[i])
p[i] <- exp(L[i]-C)
L[i] <- ... # here the log-likelihood for your model
}
```

Poisson zeros approach may be recommended to avoid possible numerical overflow problems.

Specifying a new prior distribution. To get a prior that is not directly available in WinBUGS, we can use the zeros trick at the prior level:

```
zero <- 0
theta ~ dflat()
L <- ...  # here expression for -log(desired prior density for theta)
zero ~ dpois(L)
```

Here `dflat()` is the improper flat density over $(-\infty, \infty)$, i.e. over the entire parameter space of $\theta \in \mathbb{R}$. However, it has been warned in the manual that this method produces high auto-correlation, poor convergence and high MC error, so it is computationally slow and long runs are necessary.

11.5.1 Zero inflated models

Some positive count data may contain too many values at zero so that standard distributions are not adequate to model such behavior. The zero inflated model is of the form:

$$\pi(y) = \alpha I_{\{y=0\}} + (1-\alpha)\pi(y \mid \theta)$$

which gives probability α for the zero value, and with probability $1-\alpha$ the values follow the conditional distribution $\pi(y \mid \theta)$. In BUGS, this could be implemented using zeros-ones tricks by using:

```
L[i] <- log(alpha[i]*equals(y[i],0)+(1-alpha[i])*fd[i])
fd[i] <- exp(-lambda[i]+y[i]*log(lambda[i])-loggam(y[i]+1) )
```

which gives the zero inflated Poisson model. Here, `fd` could be replaced by some other probability density function. Alternatively, we could specify the model as

$$Y \mid U \sim \text{Poisson}(\lambda(1-U)) \ \ , U \sim \text{Bernoulli}(\alpha)$$

because a Poisson model with zero parameter reduces to point value of zero with probability one.

```
for(i in 1:n){
y[i] ~ dpois(mu[i])
mu[i] <- (1-u[i])*lambda[i]
u[i] ~ dbern(alpha)
alpha ~ dunif(0,1)
log(lambda[i]) <- ... the linear predictor
}
```

We might also specify a link function for α to allow covariate effects.