# Random effect multinomial model

*A. J. Rominger*

*20 April 2017*

## The model

Given the number of sequencing reads assigned to each of $S$ species, we want to estimate the number of individuals for each species that went into the sequencing run, call that vector $x$. We know the total number of reads $N_{reads}$, the number of species $S$, the total number of individuals $N$ that were sequenced, and the vector of number of reads per species $x_{read}$. We assume that stochastic evolutionary processes have led to some species having a greater propensity to be sequenced. These stochastic processes we summarize in a random effect $nu$. We further assume that within orders these random effects are constant, thus each species within the same order gets the same random effect.

Thus our model for the number of reads is:

$$x_{reads} \sim \text{multinom}\left(N_{reads}, \frac{x\nu}{\sum_i x_i \nu_i}\right)$$

Thus the vector of probabilities for the multinomial distribution is proportional to the unknown abundance times the unknown random effect $\nu$. Given that $\nu$ is constant within orders, varies across orders, and must be positive, we model it as

$$\log(\nu_{order=j}) \sim \text{norm}(0, \sigma_\nu^2)$$

That is, for order $j$ the log of the random effect is distributed normally with mean 0 and variance $\sigma_\nu^2$.

## The punchline

When I build and run this model using NIMBLE, I don't get good estimates for $x$ and $\sigma_\nu$, and I'm not sure why. I'm not sure if:

1) the model is not identifiable, or
2) I'm simulating too few orders (currently 7), or
3) I've misspecified the model somehow

Any help figure that out would be really great. Below you'll find my NIMBLE code.

## The model in NIMBLE code

First we set up the simulation

```
# number of orders
nOrd <- 7

# number of species
nspp <- nOrd * 10

# vector identifying which species are in which orders
ordID <- rep(1:nOrd, each = nspp/nOrd)
```

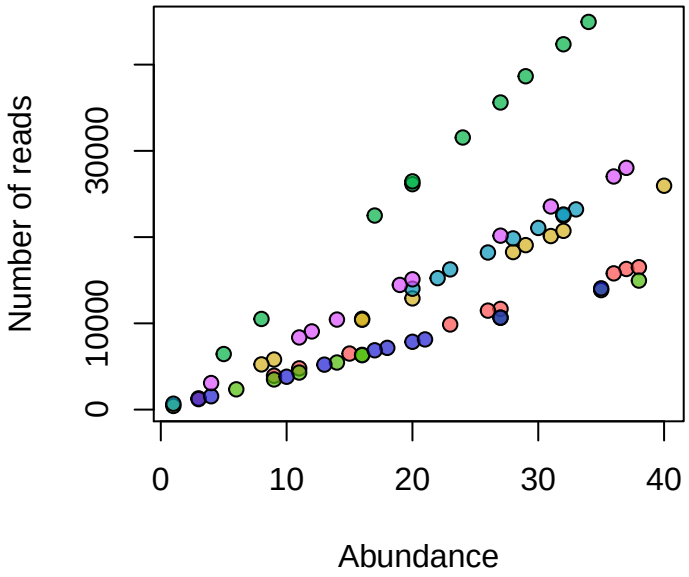
```
# simulate random effect
ordSD <- 0.5
set.seed(1)
ordEffect <- exp(rnorm(nOrd, mean = 0, sd = ordSD))

# generate true abundances for each species
set.seed(1)
x <- sample(1:40, nspp, rep = TRUE)

# simulate number of reads for each species
totReads <- 10^6
set.seed(1)
xreads <- rmulti(1, totReads, x*ordEffect[ordID])
```

A quick plot of what those simulated data look like (colors correspond to order identity)

```
palette(hsv(h = seq(0, 0.8, length.out = max(ordID)),
            s = 1-0.3*seq(-1, 1, length.out = max(ordID))^2,
            v = 0.7 + 0.3*seq(-1, 1, length.out = max(ordID))^2,
            alpha = 0.7))
plot(x, xreads, bg = ordID, pch = 21,
     xlab = 'Abundance', ylab = 'Number of reads')
```




Define the model and initialize it with constants, data and inits.

```
modCode <- nimbleCode({
    ## reads model
    xreads[1:S] ~ dmulti(alpha[1:S], totReads)

    # priors
    x[1:S] ~ dmulti(p0[1:S], N) # p0 defined in constants
    tauCopy ~ dgamma(0.01, 0.0001)

    # order-level random effect
    for(j in 1:nOrd) {
        logNu[j] ~ dnorm(0, tauCopy)
        nu[j] <- exp(logNu[j])
```

```
    }

    # define multinom param
    for(i in 1:S) {
        alpha[i] <- x[i] * nu[ordID[i]]
    }
})

# model constants, data and inits
modConstants <- list(S = nspp, N = sum(x), totReads = totReads, ordID = ordID, nOrd = nOrd,
                     p0 = rep(1/nspp, nspp))

modData <- list(xreads = xreads)

modInits <- list(
    # priors
    tauCopy = 1,
    x = as.numeric(table(cut(1:modConstants$N, nspp))), # uniform partition

    # hyperdistribution
    logNu = rep(0, modConstants$nOrd),
    nu = rep(1, modConstants$nOrd),

    # deterministic relationships
    alpha = rep(1/nspp, nspp)
)

# build model
mod <- nimbleModel(code = modCode,
                   constants = modConstants, data = modData, inits = modInits)
```

```
## defining model...
## Adding p0 as data for building model.
## building model...
## setting data and initial values...
## running calculate on model (any error reports that follow may simply
## reflect missing values in model variables) ...
##
## checking model sizes and dimensions...
##
## model building finished.
```

Compile the model and configure it for MCMC:

```
Cmod <- compileNimble(mod)
```

```
## compiling... this may take a minute. Use 'showCompilerOutput = TRUE' to see C++ compiler details.
## compilation finished.
```

```
modConf <- configureMCMC(mod)
```

One quick question: `configureMCMC` automatically adds monitors for `tauCopy` and `x`, but why does it not for `logNu`? Because `logNu` is a hyper-distribution?

```
modConf$getMonitors()
```

```
## thin = 1: tauCopy, x
```

```
modConf$addMonitors('logNu')
```

```
## thin = 1: tauCopy, x, logNu
```

Now I build and compile the MCMC (note, the below code is not actually run by this rmarkdown document, but run in the background and its output is loaded silently before continuing).

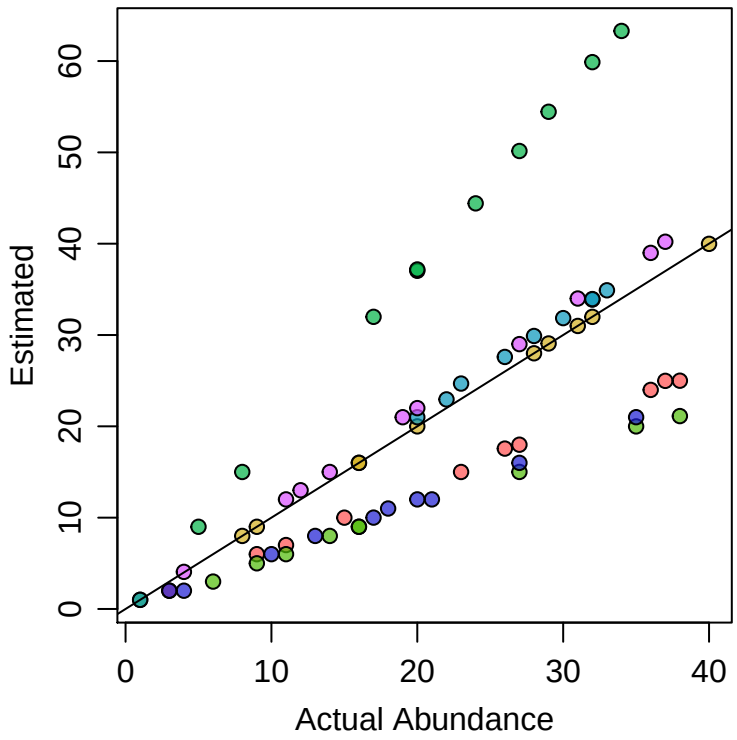
```
# compile MCMC
modMCMC <- buildMCMC(modConf)
CmodMCMC <- compileNimble(modMCMC, project = mod)

# set MCMC iterations
mcmcN <- 5e+03
burn <- 4e+02
niter <- (mcmcN + burn) * modConf$thin
CmodMCMC$run(niter)

# the posterior sample
samp <- as.matrix(CmodMCMC$mvSamples)[-(1:burn), ]
```

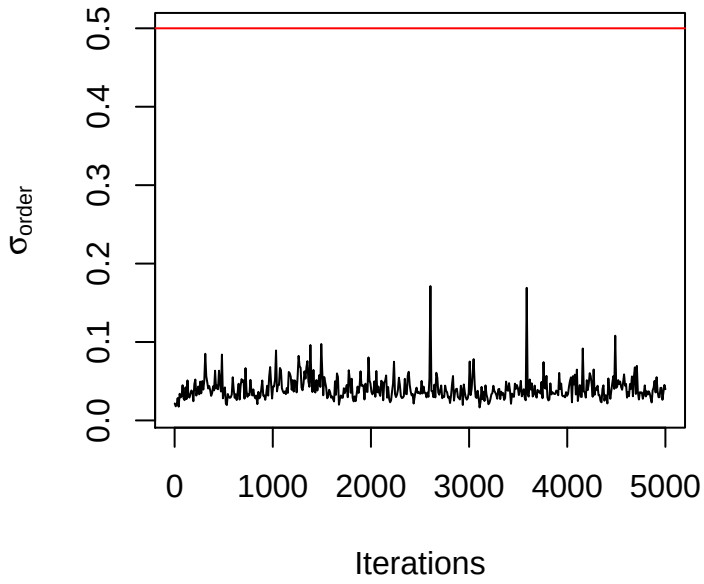
Now plot the true versus estimated abundances: it looks like a scaled version of the plot of abundance versus number of reads.

```
par(mar = c(3, 3, 0, 0) + 0.5, mgp = c(2, 0.75, 0))
plot(x, colMeans(samp[, grep('x', colnames(samp))]), bg = ordID, pch = 21,
     xlab = 'Actual Abundance', ylab = 'Estimated')
abline(0, 1)
```




We can see that the parameter $\sigma_{order}$ (red line) is underestimated (posterior trace: black line; note, trace is thinned to reduce file size)

```
plot(round(seq(1, nrow(samp), length.out = nrow(samp)/10)),
     samp[round(seq(1, nrow(samp), length.out = nrow(samp)/10)), 'tauCopy']^-0.5,
     type = 'l', ylim = range(ordSD, samp[, 'tauCopy']^-0.5),
     xlab = 'Iterations', ylab = expression(sigma[order]))
abline(h = ordSD, col = 'red')
```




This leads to the estimates of random effects being too small as well, so the estimate for `x` must do the work that `nu` should be doing—thus explaining why the plot of true versus estimated abundances looks like a scaled version of the plot of abundance versus reads.

```
plot(ordEffect, colMeans(samp[, grep('Nu', colnames(samp))]), bg = unique(ordID), pch = 21,
     xlab = 'Actual Order Effect', ylab = 'Estimated Order Effect',
     xlim = range(ordEffect, samp[, grep('Nu', colnames(samp))]),
     ylim = range(ordEffect, samp[, grep('Nu', colnames(samp))]))
abline(0, 1)
```