Metabarcoding model development in NIMBLE

A. J. Rominger

14 April 2017

The model

We know the total number of reads N_{reads}, the number of species S, the total number of individuals N, and the correlation structure from the phylogeny Σ_{phy}. We also know the vector of number of reads per species x_{read}. We do not know the rate of copy number evolution σ_{copy} nor the rate of primer affinity evolution σ_{primer}. We also most importantly don't know the actual vector of abundances x. We want to model the number of reads as multinomial conditioned on N_{reads} and S, and with a probability vector p given by

$$p \sim \text{dir}(x\nu)$$

where ν is a variable proportional to copy number. We will assume that $\log\nu$ evolved according to a Brownian motion process with rate σ_{copy} and mean $\mu_0 = 0$.

The variance-covariance matrix for a Brownian-motion process on a known phylogeny is given as

$$\Sigma_{i,j} = \sigma D_{MRCA(i,j)}$$

where $D_{MRCA(i,j)}$ is the distance from the root to the most recent common ancestor of tips i and j. $D_{MRCA(i,i)}$ is simply the depth of the root.

Thus our full model is

$$x_{reads} \sim \text{multinom}(N_{reads}, p_1, \ldots, p_S)$$

where

$$p \sim \text{dir}(x\nu)$$

and ν is distributed

$$\log(\nu) \sim \text{mvnorm}(0, \sigma_{copy} d_{MRCA(i,j)})$$

The priors for this hierrarchical model are

$$x/N \sim \text{dir}(N/S, \ldots, N/S)$$

$$\frac{1}{\sigma^2_{copy}} \sim \Gamma(0.001, 0.001)$$

NIMBLE model

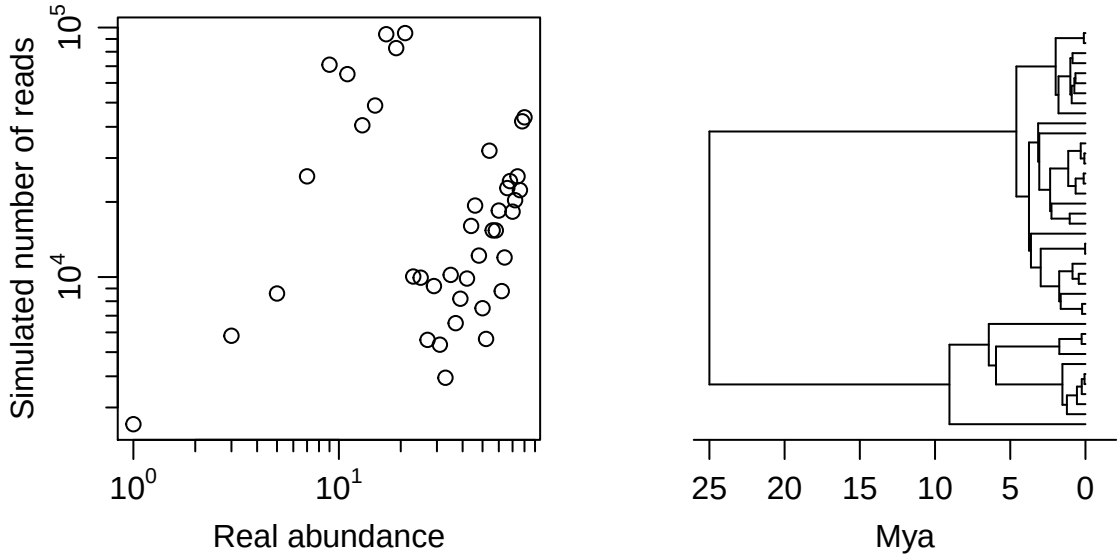
We first simulate some data that will be needed in the model specification:

- `S` number of species to simulate
- `tre` is the underlying phylogeny
- `N` is the number of actual individuals
- `Nreads` is the total number of reads
- `numberReads` is a vector of the number of reads assigned to each spp

```
# simulation functions
source('~/Dropbox/hawaiiDimensions/mol2ecol/simMetaBar.R')

# set parameters for simulation
S <- 40
n <- round(seq(1, 80, length.out = S))
N <- sum(n)
Nreads <- 1e+06
sigCopy <- 0.1

# simulate phylogeny and number of reads resulting form metabarcoding
set.seed(1)
sim <- simMetaBar(abund = n, sigCopy = sigCopy, nreads = Nreads)

# extract needed objects from output
numberReads <- sim$reads
tre <- sim$tre
```

Let's quickly check the simulation to see that it's reasonable

```
library(socorro)
par(mfrow = c(1, 2), mar = c(3, 3, 0, 0) + 0.5, mgp = c(2, 0.75, 0))
plot(sim$abund, numberReads, xlab = 'Real abundance', ylab = 'Simulated number of reads',
     log = 'xy', axes = FALSE, frame.plot = TRUE)
logAxis(1, expLab = TRUE); logAxis(2, expLab = TRUE)
plot(tre, show.tip.label = FALSE)
axis(1, at = pretty(par('usr')[1:2]), labels = rev(pretty(par('usr')[1:2])))
mtext('Mya', side = 1, line = 2)
```



To specify our model in NIMBLE we can define and use custom multinomial-Dirichlet functions:

```
library(nimble)

ddirchmulti <- nimbleFunction(
    run = function(x = double(1), alpha = double(1), size = double(0),
                   log = integer(0, default = 0)) {
        returnType(double(0))
```

```
        alpha0 <- sum(alpha)

        # new log prob that ignores 0's instead of throwing NaN/Inf
        lgammaSum <- numeric(length = length(x), value = 0, init = TRUE)
        for(i in 1:length(lgammaSum)) {
            if(x[i] > 0) {
                lgammaSum[i] <- log(x[i]) + lgamma(alpha[i]) +
                    lgamma(x[i]) -
                    lgamma(alpha[i] + x[i])
            }
        }
        logProb <- log(size) +
            lgamma(alpha0) + lgamma(size) - lgamma(alpha0 + size) -
            sum(lgammaSum)

        if(log) return(logProb)
        else return(exp(logProb))
    }
)

rdirchmulti <- nimbleFunction(
    run = function(n = double(0, default = 1), alpha = double(1), size = double(0)) {
        returnType(double(1))

        # modified from MCMCpack to allow alpha_k = 0
        x <- numeric(length = length(alpha), value = 0, init = TRUE)
        for(i in 1:length(x)) x[i] <- rgamma(1, shape = alpha[i], rate = 1)
        p <- x/sum(x)

        return(rmulti(1, size = size, prob = p))
    }
)
```

The arguments are:

- `x` vector of values (e.g. number of reads for each species x_{reads})
- `alpha` vector of parameters of the Dirichlet distribution
- `size` number of trials (e.g. total number of reads N_{reads})
- `n` number of observations (only `n = 1` supported)

We can now specify the model:

```
mod <- nimbleCode({
    # p is proportion of total abundance for each spp
    # x[1:S] <- N*p[1:S]

    # define phylogenetically-correlated copy number (nu) and primer affinity (lambda)
    # Note: tauCopy and tauPrimer are inverse variances, and 'Prec' stands for
    # 'Preceission' matrix (the inverse of the var-covar matrix)
    # Note: var-cov matrix D is defined in constants, as is mu0 (which is a vector of 0)
    PrecCopy[1:S, 1:S] <- tauCopy * Dinv[1:S, 1:S]
    logNu[1:S] ~ dmnorm(mu0[1:S], PrecCopy[1:S, 1:S])
    nu[1:S] <- exp(logNu[1:S])

    # define dirichlet-multinom params and the distribution of x_{reads}
```

```
    alpha[1:S] <- x[1:S] * nu[1:S]
    xreads[1:S] ~ ddirchmulti(alpha[1:S], Nreads)

    # priors
    x[1:S] ~ dmulti(p0[1:S], N) # p0 defined in constants
    tauCopy ~ dgamma(0.001, 0.001)
})
```

And initialize the model

```
# model constants, data and inits
S <- length(tre$tip.label)
modConstants <- list(S = S, N = N, Nreads = Nreads,
                     Dinv = solve(vcv(tre)),  # vcv gives the vcov mat of a phylo for sig = 1
                     mu0 = rep(0, S), p0 = rep(1/S, S))
modData <- list(xreads = numberReads)
modInits <- list(
    # priors
    # p = rep(1/modConstants$S, modConstants$S),
    tauCopy = 1,
    x = {
        foo <- rep(round(N / S), S)
        foo[0:(N - sum(foo))] <- foo[0:(N - sum(foo))] + 1
        foo
    },

    # deterministic relationships
    PrecCopy = modConstants$Dinv,

    # hyper distributions
    logNu = rep(0, modConstants$S),

    # deterministic relationships arrising from hyperdistributions
    nu = rep(1, modConstants$S)
)

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

```
## defining model...
```

```
## Registering the following user-provided distributions: ddirchmulti .
## NIMBLE has registered ddirchmulti as a distribution based on its use in BUGS code. Note that if you r
## Adding Dinv,mu0,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.
```

Now we can compile and run the model

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

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

```
## compilation finished.
```

```
modConf <- configureMCMC(mod)

# set the thinning param
thin <- 10
modConf$setThin(thin)
```

```
## thin = 10: tauCopy, x
```

```
# add block sampling for p (=relative spp abund)
# modConf$addSampler(target = sprintf('x[1:%s]', modConstants$S),
#                    type = 'RW_block')

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

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

```
# set MCMC params
mcmcN <- 1e+05
burn <- 8000
niter <- (mcmcN + burn) * modConf$thin
CmodMCMC$run(niter)
```

```
## |-------------|-------------|-------------|-------------|
## |
```

```
## Warning in CmodMCMC$run(niter): value out of range in 'lgamma'
```

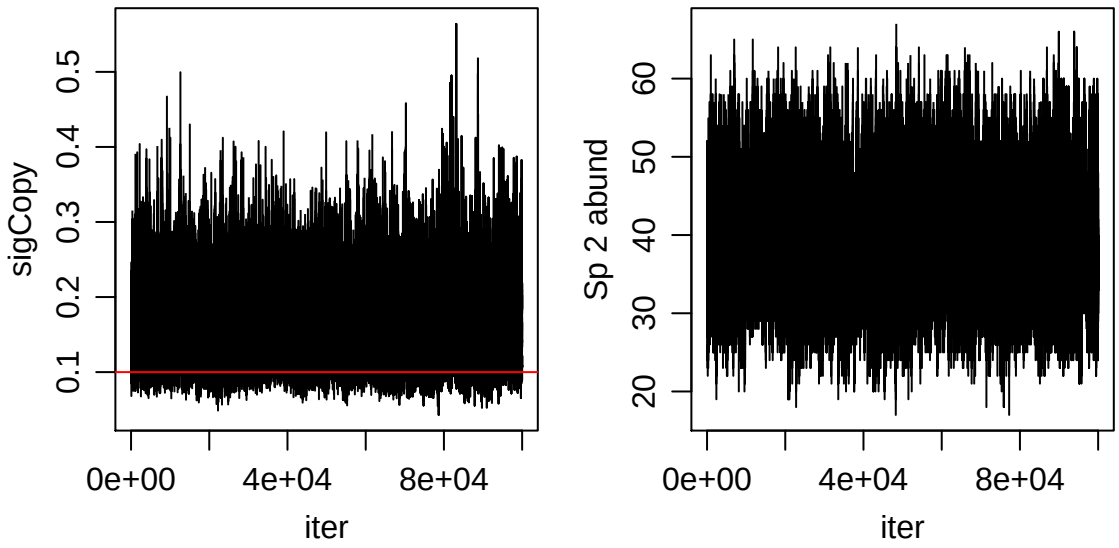
```
## ------------------------------------------------------|
```

```
## NULL
```

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

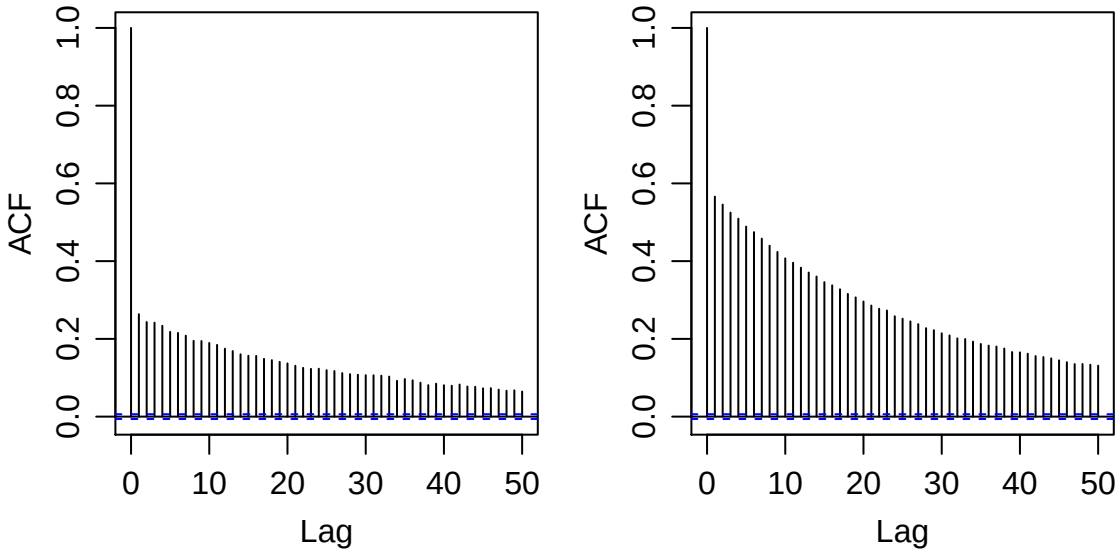
All sampled parameters are saved in `samp`. We can evaluate stationarity of our MCMC with some diagnostics. First just trace plots:

```
par(mfrow = c(1, 2), mar = c(3, 3, 0, 0) + 0.5, mgp = c(2, 0.75, 0))
plot(1/samp[, 'tauCopy'], type = 'l', xlab = 'iter', ylab = 'sigCopy')
abline(h = sigCopy, col = 'red')
plot(samp[, 'x[2]'], type = 'l', xlab = 'iter', ylab = 'Sp 2 abund')
abline(h = sim$abund[2], col = 'red')
```



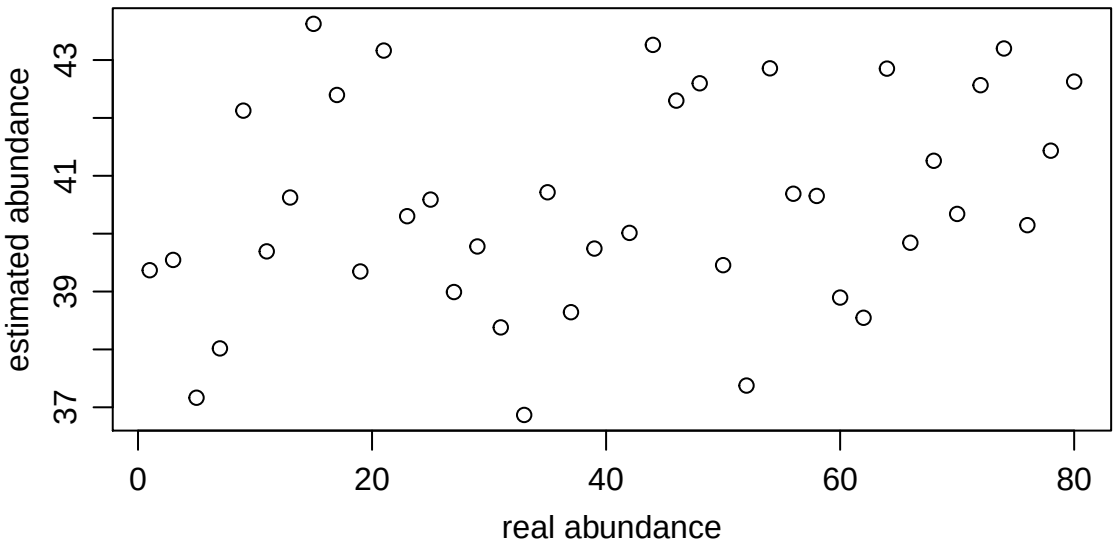
Now auto-correlation plots

```
par(mfrow = c(1, 2), mar = c(3, 3, 0, 0) + 0.5, mgp = c(2, 0.75, 0))
acf(1/samp[, 'tauCopy'])
acf(samp[, 'x[2]'])
```



Now a plot of real versus estimated abundance

```
par(mar = c(3, 3, 0, 0) + 0.5, mgp = c(2, 0.75, 0))
plot(sim$abund, colMeans(samp[, (1:S)+1]),
     xlab = 'real abundance', ylab = 'estimated abundance')
```


estimated abundance
37
39
41
43
0
20
40
60
80
real abundance