From SIAR to MixSIAR

Andrew Parnell
andrew.parnell@mu.ie

Maynooth University
National University of Ireland Maynooth

Learning outcomes

- Random effects models
- Understand how MixSIAR (and simmr) extends MixSIR and SIAR
- Understand the differences in likelihoods and priors

Revision: simmr model in JAGS

- Let's go back to the model defined earlier:

$$y_{ij} \sim N\left(\frac{\sum_{k=1}^K p_k q_{jk}(\mu_{s,jk} + \mu_{c,jk})}{\sum_{k=1}^K p_k q_{jk}}, \frac{\sum_{k=1}^K p_k^2 q_{jk}^2 (\sigma_{s,jk}^2 + \sigma_{c,jk}^2)}{(\sum_{k=1}^K p_k q_{jk})^2} + \sigma_j^2 \right)$$

- We also have prior distributions (usually uniform) on σ^2 and a Dirichlet prior on p
- How is this model still a simplification of reality?

Expanding the SIAR model further

Some (of the many) possible extensions:

1. We are assuming that all consumers have identical dietary proportions
2. We are assuming that residuals, sources and TEFs are uncorrelated across isotopes
3. We are assuming that concentration dependence is known
4. We cannot add in any extra covariates (height, weight, etc, etc)

Mixed effects models in linear regression

- Often data are available in *groups*, for example wolves might belong to different packs
- We want to capture the different levels of variation, both *within* groups, and *between* groups
- Example: suppose y_{ij} is a measurement for individual i in group j, $i = 1, \ldots, N_j$, $j = 1, \ldots, M$
- We might use a model such as:

$$y_{ij} \sim N(\mu + b_j, \sigma^2), \; b_j \sim N(0, \sigma_b^2)$$

- Now b_j is called a *random effect* and measures the change in the mean for each group
- σ measures the standard deviation *within* a group whilst σ_b measures the standard deviation *between* groups

Fitting a random effects model in JAGS

```
modelstring ='
model {
  for(i in 1:N) { y[i] ~ dnorm(mu+b[group[i]],sigma^-2) }
  for(j in 1:M) { b[j] ~ dnorm(0,sigma_b^-2) }
  mu ~ dnorm(0,100^-2)
  sigma ~ dunif(0,100)
  sigma_b ~ dunif(0,10)
}'
data=list(y=c(3.03, 2.68, 2.04, 3.23, 2.82, 2.46, 3.06, 3.05,
              3.02, 2.95, 3.12, 2.81, 2.69, 2.65, 2.49, 2.73,
              2.21, 3.92, 4.7, 3.53, 3.89, 3.86, 4.48),
          group=c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2,
                  2, 2, 2, 3, 3, 3, 3, 3),
          N=23,M=3)
model=jags.model(textConnection(modelstring), data=data)
output=coda.samples(model=model,
                    variable.names=c("mu","sigma","sigma_b"),
                    n.iter=1000)
```

Output from the random effects model

```r
t(round(apply(output[[1]],2,quantile,
               probs=c(0.025,0.5,0.975)),2))
```

```
##          2.5%  50% 97.5%
## mu      -0.15 3.24  7.13
## sigma    0.35 0.46  0.67
## sigma_b  0.43 1.48  8.70
```

- You need a reasonable number of groups to estimate σ_b well. When the number of groups is very small you might run into problems
- You could also include `b` in the `variable.names` if you want the deviations from the overall mean
- Convergence is often better if you *hierarchically centre* the model, which means set $b_j \sim N(\mu, \sigma_b^2)$ in the previous slides' JAGS code (if you look in the code for these slides that is what I ran)

Defining mixed effects models in the SIMM case

- In the SIMM case we might have information that some consumers might share the same dietary proportions
- We might be interested in how the dietary proportions vary between groups and within groups, just as in the simple example
- An issue is: how do we achieve this in the SIMM case?
- There are two ways to deal with probability distributions for proportions; use a direct probability distribution (e.g. the Dirichlet) or *transform* to another set of parameters

More on random effects in proportions

- A slightly more flexible prior distribution for proportions is obtained by transforming the proportions instead
- We already met this with logistic regression where we can use:

$$f = logit(p) = \log\left(\frac{p}{1-p}\right) \text{ or equivalently } p = \frac{\exp(f)}{\exp(f)+1}$$

- When we have multiple proportions a generalisation of this is the *centralised log ratio* (CLR) or *softmax* transformation:

$$[p_1, \ldots, p_K] = \left[\frac{\exp(f_1)}{\sum_j \exp(f_j)}, \ldots, \frac{\exp(f_K)}{\sum_k \exp(f_k)}\right]$$

The CLR transformation

- In logistic regression we can put a prior distribution on f (i.e. $logit(p)$) e.g. $f \sim N(\alpha + \beta x, \sigma^2)$ which allows us to relate the probability p to a covariate x. We can use the normal distribution because f is unrestricted
- In CLR regression, we put a prior on the f_k so that each one relates to the covariate with different coefficient values
- The CLR transformation guarantees that all the dietary proportions will sum to 1

Random effects for individuals

- We don't necessarily need a grouping structure (e.g. pack, sex, etc) to be able to include random effects in a SIMM
- In a SIMM we might reasonably assume that every consumer is eating something slightly different and want to quantify the overall mean diet as well as the variability between consumers
- We can do this by modelling each consumer's dietary proportion p_{ik} with a normally distributed prior on the CLR transform of p

A ‘simple’ CLR example

```
modelstring ='
model {
  for (j in 1:J) {
    for (i in 1:N) {
      y[i,j] ~ dnorm(inprod(p[i,]*q[,j], s_mean[,j]+c_mean[,j])
      var_y[i,j] <- inprod(pow(p[i,]*q[,j],2),s_sd[,j]^2+c_sd[,j
    }
  }
  for(i in 1:N) {
    p[i,1:K] <- expf[i,]/sum(expf[i,])
    for(k in 1:K) {
      expf[i,k] <- exp(f[i,k])
      f[i,k] ~ dnorm(mu_f[k],sigma_f[k]^-2)
    }
  }
  for(k in 1:K) {
    mu_f[k] ~ dnorm(0,1)
    sigma_f[k] ~ dgamma(2,1)
  }
  for(j in 1:J) { sigma[j] ~ dunif(0,10) }
}
```

CLR model: R code

```
data=list(y=con,s_mean=sources[,c(1,3)],
          s_sd=sources[,c(2,4)],
          c_mean=tefs[,c(1,3)],c_sd=tefs[,c(2,4)],
          q=cd,N=nrow(con),K=nrow(sources),
          J=ncol(con))
model=jags.model(textConnection(modelstring), data=data)
output=coda.samples(model=model,
                    variable.names=c('p','sigma',
                                     'mu_f','sigma_f'),
                    n.iter=10000)
out_summ = summary(output)
```

Output

```r
head(out_summ$statistics,12)
```

```
##                Mean        SD    Naive SE Time-series SE
## mu_f[1]  1.5809159 0.7083237 0.007083237    0.050299319
## mu_f[2] -0.8182919 0.7144027 0.007144027    0.052816573
## mu_f[3] -0.5432100 0.8863447 0.008863447    0.046865867
## mu_f[4] -0.1140957 0.9124691 0.009124691    0.041839187
## p[1,1]   0.6707826 0.1942892 0.001942892    0.004241812
## p[2,1]   0.6354725 0.2012488 0.002012488    0.004661179
## p[3,1]   0.6953520 0.1942468 0.001942468    0.005728227
## p[4,1]   0.6590746 0.2021487 0.002021487    0.004529225
## p[5,1]   0.6560662 0.1994627 0.001994627    0.005045782
## p[6,1]   0.6994554 0.1930907 0.001930907    0.005038948
## p[7,1]   0.7057615 0.1853004 0.001853004    0.004945508
## p[8,1]   0.4396148 0.2249038 0.002249038    0.007938298
```

Notes about the CLR model

- This is a great starter script for your own work. If you can understand this code and adapt it to your data you can get some really powerful results
- We can now put covariates in the model: we just have to expand μ_f in the previous JAGS code
- We now have individual dietary proportion estimates (p_{ik}) and overall dietary proportion estimates (via CLR transform of μ_k), and also estimates of the variability (from σ_f)
- Things start to get complicated with prior distributions at this stage. Be very careful and always examine prior sensitivity by re-running the model with slightly different prior distributions. Look at the effect on p (the effect on μ_f and σ_f is less important)

Summary

- We have looked at the differences between SIAR and MixSIAR
- We studied the CLR as an alternative to the Dirichlet
- We showed how to include random effects in a SIMM