

# The statistical model behind simmr (and SIAR)

Andrew Parnell  
andrew.parnell@mu.ie



## Learning outcomes:

- ▶ Understand the statistical model behind simmr/SIAR
- ▶ Know how to run a model in simmr/SIAR and check that it works
- ▶ Be able to follow the technical details of the 2010 SIAR Plos ONE paper

## Our simple SIMM

- ▶ In the last class we had a simple SIMM defined via:

$$y_i \sim N\left(\sum_{k=1}^2 p_k s_k, \sigma^2\right)$$

with  $s_k \sim N(\mu_{s_k}, \sigma_{s_k}^2)$ ,  $p_1 \sim U(0, 1)$  and  $\sigma \sim U(0, 100)$

- ▶ Here  $y_i$  is the isotope value,  $s$  are the source values,  $p$  are the dietary proportions, and  $\sigma$  is the residual standard deviation
- ▶ The goal is to estimate the  $p$  and its uncertainty. The other parameters can be considered nuisance parameters

## Expanding the simple SIMM

- ▶ This SIMM is currently too simplistic. We need to expand it by:
  - ▶ increasing the number of food sources
  - ▶ including trophic enrichment factors (TEFs)
  - ▶ including concentration dependence
  - ▶ allowing for multiple isotopes
  - ▶ allowing for richer source sampling by consumers
- ▶ If we include all of these factors we end up with the `simmr`/SIAR model
- ▶ We will take them in turn and add them into our JAGS code

## Reminder: the SIAR geese data

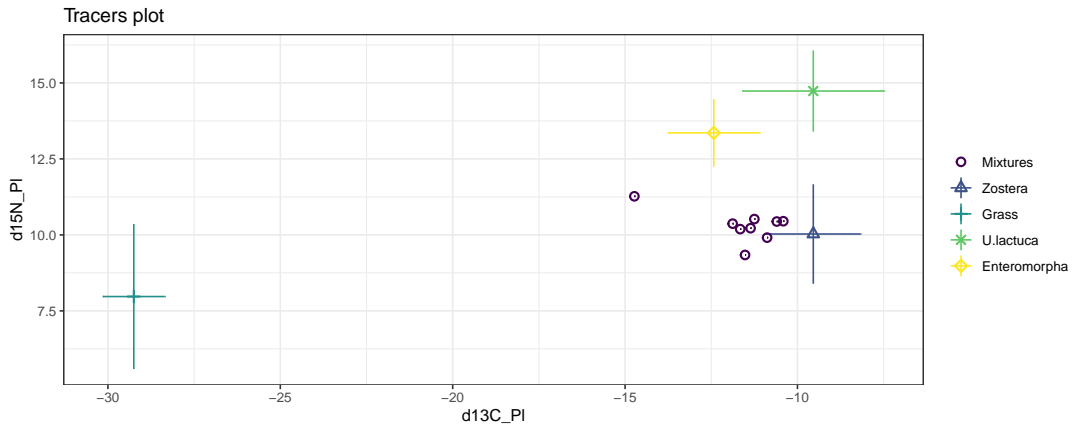
```
data(geese_data_day1)
str(geese_data_day1)
```

```
## List of 8
## $ mixtures          : num [1:9, 1:2] -11.4 -11.9 -10.6 -11.2 -11.7 ..
## ..- attr(*, "dimnames")=List of 2
## .. ..$ : NULL
## .. ..$ : chr [1:2] "d13C_P1" "d15N_P1"
## $ tracer_names      : chr [1:2, 1] "d13C_P1" "d15N_P1"
## $ source_names      : chr [1:4] "Zostera" "Grass" "U.lactuca" "Enteromorpha"
## $ source_means      : num [1:4, 1:2] -11.17 -30.88 -11.17 -14.06 6.49
## ..- attr(*, "dimnames")=List of 2
## .. ..$ : NULL
## .. ..$ : chr [1:2] "meand13CP1" "meand15NP1"
## $ source_sds        : num [1:4, 1:2] 1.215 0.641 1.959 1.172 1.459 ..
## ..- attr(*, "dimnames")=List of 2
## .. ..$ : NULL
```

# Plotting the data

A plot in isotope space:

```
plot(simmr_in)
```



## Including multiple sources

- ▶ Adding in multiple sources to the likelihood means having more terms in the sum:

$$y_i \sim N \left( \sum_{k=1}^K p_k s_k, \sigma^2 \right)$$

- ▶ In the above we have  $K$  sources and hence  $K$  dietary proportions
- ▶ We also now need  $K$  source prior distributions
- ▶ The tricky part about adding in multiple proportions is the prior distribution

## Priors for constrained dietary proportions

- ▶ We must have  $\sum_{k=1}^K p_k = 1$  so any prior distribution we place on the  $p$ s must satisfy this restriction
- ▶ (You will often hear values restricted in sum referred to as a *simplex*)
- ▶ Luckily there is a distribution known as the *Dirichlet* which is suitable for restricted sum parameters
- ▶ The Dirichlet has one parameter for each proportion  $\alpha_1, \dots, \alpha_K$ . The larger the  $\alpha$  value the larger prior weight that dietary proportion will be given
- ▶ Setting all the  $\alpha$  values to 1 is equivalent to the simplex uniform distribution, i.e. a prior assumption that all sources are consumed equally



## JAGS SIMM with a Dirichlet prior

```
model_code = '
model {
  for(i in 1:N) { y[i] ~ dnorm(inprod(p,s),sigma^-2) }
  p ~ ddirch(alpha)
  for(k in 1:K) { s[k] ~ dnorm(s_mean[k],s_sd[k]^-2) }
  sigma ~ dunif(0,100)
}
'

data=with(geese_data_day1,
          list(y=mixtures[,1],
               s_mean=source_means[,1],
               s_sd=source_sds[,1],
               N = nrow(mixtures),K=nrow(source_means),
               alpha=rep(1,nrow(source_means))))

set.seed(123)
model_run = jags(data = data,
                  parameters.to.save = c("p", "sigma"),
                  model.file = textConnection(model_code))
```

This is now running with all 4 sources

## Results

- ▶ We can explore/plot results with `summary(output)`, `plot(output)`, and also run multiple chains, form predictive distributions, check convergence, etc
- ▶ One important thing to note is that the fitting method (MCMC) produces a joint posterior distribution of the dietary proportions. This means that each set of samples will sum to 1:

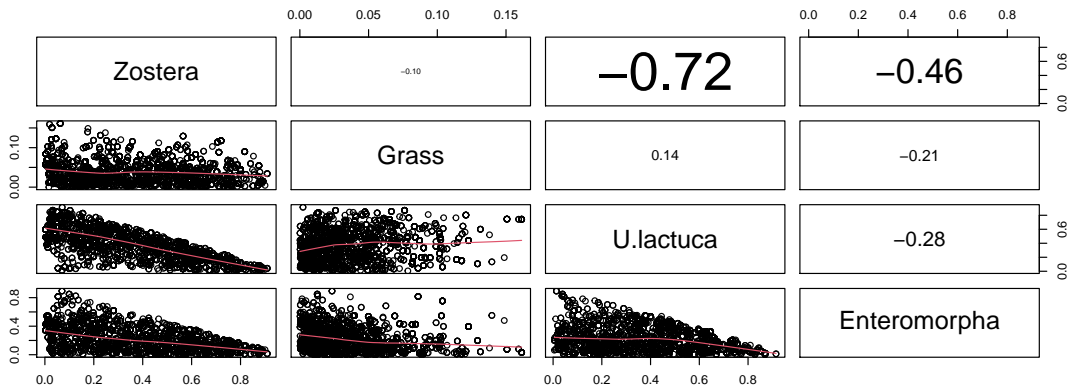
```
head(model_run$BUGSoutput$sims.matrix,4)
```

| ## |      | deviance | p[1]      | p[2]        | p[3]      | p[4]       | sigma    |
|----|------|----------|-----------|-------------|-----------|------------|----------|
| ## | [1,] | 38.24013 | 0.2931735 | 0.037865208 | 0.6128807 | 0.05608053 | 2.666854 |
| ## | [2,] | 29.94426 | 0.2925945 | 0.049061414 | 0.4754383 | 0.18290585 | 1.237191 |
| ## | [3,] | 29.19475 | 0.3280195 | 0.051608762 | 0.4583445 | 0.16202724 | 1.385694 |
| ## | [4,] | 29.35735 | 0.5883123 | 0.004139584 | 0.1634236 | 0.24412454 | 1.037509 |

- ▶ The key implication of this is that, aside from exploring the *marginal* posterior distributions (with means, sds, etc) we can explore the *joint* uncertainty of the dietary proportions

## A joint plot of the posterior dietary proportions

```
out_2 = model_run$BUGSoutput$sims.list$p  
colnames(out_2) = geese_data_day1$source_names  
pairs(out_2, lower.panel = panel.smooth,  
       upper.panel = panel.cor)
```



## Trophic enrichment factors and concentration dependence

- ▶ Trophic enrichment factors ( $c$ ) and concentration dependence ( $q$ ) represent adjustments to the source values to account for various measurement effects
- ▶ We can include them by expanding the likelihood:

$$y_i \sim N \left( \frac{\sum_{k=1}^K p_k q_k (s_k + c_k)}{\sum_{k=1}^K p_k q_k}, \sigma^2 \right)$$

- ▶ The extra part on the denominator is needed so that the dietary proportions still sum to 1
- ▶ The prior for  $c_k$  comes from external data and are given normal distributions like the source values
- ▶ In SIAR the concentration dependencies must be less than 1 (given as proportions) and are treated as fixed. You could use a strong Dirichlet prior on these instead

## Including TEFs and CD - JAGS model

```
model_code = '  
model {  
  for(i in 1:N) {  
    y[i] ~ dnorm(inprod(p*q,s+c)/inprod(p,q),sigma^-2)  
  }  
  p ~ ddirch(alpha)  
  for(k in 1:K) {  
    s[k] ~ dnorm(s_mean[k],s_sd[k]^-2)  
    c[k] ~ dnorm(c_mean[k],c_sd[k]^-2)  
  }  
  sigma ~ dunif(0,100)  
}  
,  
data=with(geese_data_day1,  
  list(y=mixtures[,1], s_mean=source_means[,1],  
        s_sd=source_sds[,1], c_mean = correction_means[,1],  
        c_sd = correction_sds[,1], q = concentration_means[,1],  
        N = nrow(mixtures),K=nrow(source_means),  
        alpha=rep(1,length(source_names))))  
model_run = jags(data = data
```

## Notes on the TEF and CD model

- ▶ If you run this, you'll find that convergence isn't quite as neat and it starts to get a bit slower
- ▶ Although it's a nuisance parameter, saving  $\sigma$  is often a good idea because a large value indicates a poorly fitting model (usually also seen in the iso-space plot)
- ▶ The model will also create posterior distributions for  $s$  and  $c$ , though these are usually pretty similar to the prior, as there isn't much information about their values in the data

## Adding extra isotopes

- ▶ If we have extra isotopes we can just list the likelihood twice, once for each value of the isotope. Only the dietary proportions are 'shared' between the isotopes
- ▶ Now write  $y_{ij}$  as the consumer values for observation  $i$  on *isotope*  $j$ , where  $j = 1, \dots, J$
- ▶ We now have source values  $s_{jk}$ , TEF values  $c_{jk}$ , concentration dependencies  $q_{jk}$ , and each isotope has its own residual standard deviation  $\sigma_j$
- ▶ The likelihood is now:

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

## Richer source sampling

- ▶ The model we've been fitting up to now assumes that all individuals sample the same source value  $s_k$  for each source and isotope. This is unrealistic
- ▶ A better model has each individual sampling a different source value from the source prior distribution, i.e. we now have  $s_{ik}$  (or  $s_{ikj}$  with multiple isotopes)
- ▶ The JAGS code becomes:

```
for(k in 1:K) {  
  for(i in 1:N) {  
    s[i,k] ~ dnorm(s_mean[k], s_sd[k]^-2)  
  }  
}
```

- ▶ We can do the same with the concentration dependence values
- ▶ In fact with a bit of clever maths we can remove (*marginalise over*) the  $s_{ik}$  values to get a simpler model with fewer parameters.



## The full simmr/SIAR model

- ▶ Using the trick mentioned on the last slide, we end up with a full model which looks like this:

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

- ▶ This model has a more complicated likelihood, but removes the extra  $s$  and  $c$  parameters

## Full SIAR model: JAGS code

```
model_code = '  
model {  
  for (i in 1:N) {  
    for (j in 1:J) {  
      y[i,j] ~ dnorm(inprod(p*q[,j], s_mean[,j]+c_mean[,j]) /  
        inprod(p,q[,j]), var_y[j]^-1)  
    }  
  }  
  p ~ ddirch(alpha)  
  for(j in 1:J) {  
    var_y[j] <- inprod(pow(p*q[,j],2),s_sd[,j]^2+c_sd[,j]^2)/pow(inprod(p,c  
      + pow(sigma[j],2)  
  }  
  for(j in 1:J) { sigma[j] ~ dunif(0,100) }  
}  
'
```

## Full simmr/SIAR model: R code

```
data=with(geese_data_day1,  
  list(y=mixtures,s_mean=source_means,  
    s_sd=source_sds,  
    c_mean=correction_means,c_sd=correction_sds,  
    q=concentration_means,N=nrow(mixtures),  
    J=ncol(mixtures),alpha=rep(1,length(source_names))))  
model_run = jags(data = data,  
  parameters.to.save = c("p", "sigma"),  
  model.file = textConnection(model_code),  
  DIC = FALSE)
```

## Summary of posterior dietary proportions

```
out_2 = model_run$BUGSoutput$sims.matrix  
colnames(out_2) = c(geese_data_day1$source_names, 'SD1', 'SD2')  
t(round(apply(out_2, 2, quantile, probs=c(0.025, 0.5, 0.975)), 2))
```

| ##              | 2.5% | 50%  | 97.5% |
|-----------------|------|------|-------|
| ## Zostera      | 0.43 | 0.61 | 0.80  |
| ## Grass        | 0.03 | 0.07 | 0.12  |
| ## U.lactuca    | 0.01 | 0.14 | 0.35  |
| ## Enteromorpha | 0.01 | 0.15 | 0.46  |
| ## SD1          | 0.05 | 0.86 | 2.45  |
| ## SD2          | 0.02 | 0.36 | 1.40  |

Some of these proportions are quite imprecise: perhaps see better with matrix plot?

## Running SIAR/simmr

- ▶ The SIAR/simmr R packages run exactly this model with a few extra tweaks
- ▶ It contains a slightly optimised algorithm as JAGS sometimes gets a bit stuck on harder data sets. It's also much faster than JAGS for complicated problems
- ▶ It allows for direct plotting of the data in isotope space and  $p$ -space (i.e. dietary proportion space - pairs plots)
- ▶ It allows for changing the  $\alpha$  values to put in proper prior information
- ▶ It includes convergence checking
- ▶ Most of this covered in the practical this afternoon

## simmr version

- ▶ `simmr` is a much more elegantly written version of SIAR with neater plots and many more features
- ▶ Four steps to run a `simmr` model
  1. Call `simmr_load` to load in the data
  2. Call `plot` to see the iso-space plot
  3. Call `simmr_mcmc` to run the model
  4. Call `plot` or `summary` to access the output
- ▶ `simmr` has further features to combine sources and to compare dietary proportions

## simmr code

```
# Load
data("geese_data_day1")
simmr_in = with(geese_data_day1,
               simmr_load(mixtures = mixtures,
                          source_names = source_names,
                          source_means = source_means,
                          source_sds = source_sds,
                          correction_means = correction_means,
                          correction_sds = correction_sds,
                          concentration_means = concentration_means))

# Iso-space plot
plot(simmr_in)

# MCMC run
simmr_out = simmr_mcmc(simmr_in)

# Box-plots
plot(simmr_out, type = 'boxplot')
```

## Summary

- ▶ The `simmr` and `SIAR` models are just complicated versions of linear regression
- ▶ The response is multivariate and the prior distributions on some of the parameters have to be constrained to sum to 1
- ▶ It used to be the case that `JAGS` was slow and couldn't run `SIMM`-type models. This is no longer true. You can fit much richer models in `JAGS` (and now `MixSIAR`) than with `SIAR/simmr`
- ▶ More details on running `simmr` in the practical next