Module 5: The statistical model behind SIAR

Andrew Parnell, School of Mathematics and Statistics,
University College Dublin

Learning outcomes:

- Understand the statistical model behind SIAR
- Know how to run a model in 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 $s_k \sim N(\mu_{s_k}, \sigma^2_{s_k})$, $p_1 \sim U(0,1)$ and $\sigma \sim U(0,100)$
- Here y_i is the isotope value, s are the source values, p are the dietary proportions, and σ 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 SIAR model
- We will take them in turn and add them into our JAGS code

Reminder: the SIAR geese data

```
data(geese1demo,sourcesdemo)
head(geese1demo,3)
```

```
##      d15NPl d13CPl
## [1,]  10.22 -11.36
## [2,]  10.37 -11.88
## [3,]  10.44 -10.60
```

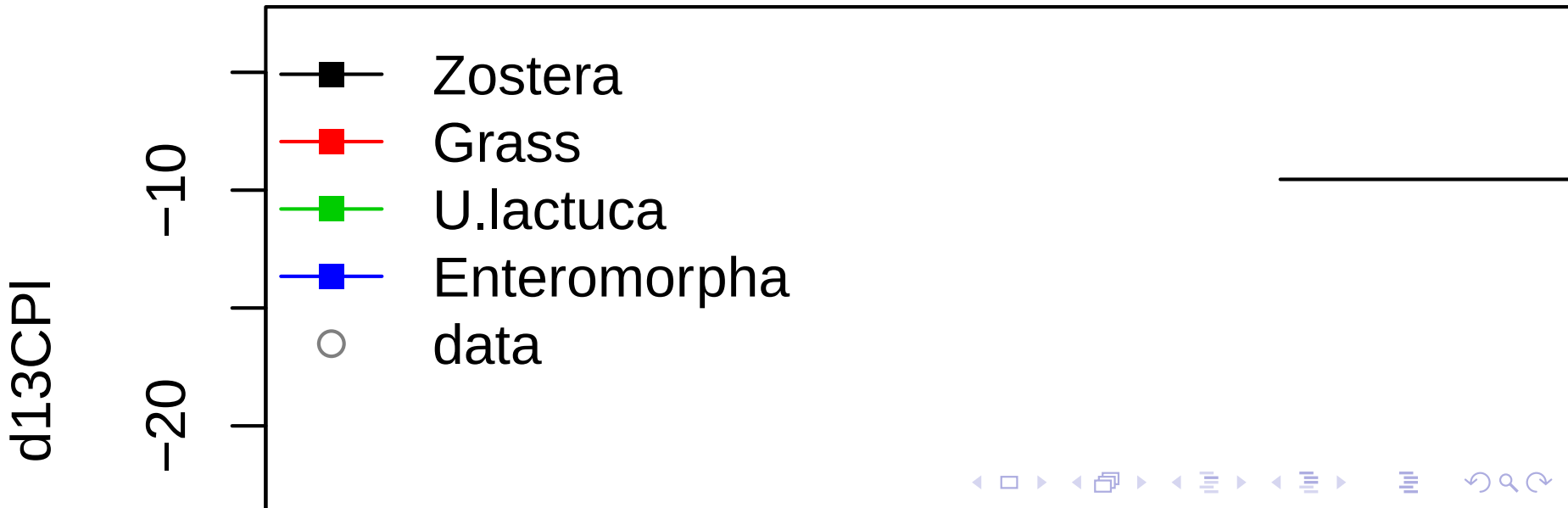
```
sourcesdemo
```

```
##          Sources  Meand15N    SDd15N  Meand13C    SDd13C
## 1       Zostera  6.488984 1.4594632 -11.17023 1.2149562
## 2         Grass  4.432160 2.2680709 -30.87984 0.6413182
## 3    U.lactuca 11.192613 1.1124385 -11.17090 1.9593306
## 4 Enteromorpha  9.816280 0.8271039 -14.05701 1.1724677
```

Plotting the data

A plot in isotope space:

```
data(correctionsdemo,concdepdemo)
out = siarmcmcdirichletv4(geese1demo,sourcesdemo,correctio
out$TITLE = 'Geese data'
siarplotdata(out)
```



Including multiple sources

- Adding in multiple sources to the likelihood is straightforward:

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

- 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 ps 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 $\alpha_1, \ldots, \alpha_K$. The larger the α value the larger prior weight that dietary proportion will be given
- Setting all the α 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

```
modelstring ='
model {
  for(i in 1:N) { y[i] ~ dnorm(inprod(p,s),1/pow(sigma,2))
  p ~ ddirch(alpha)
  for(k in 1:K) { s[k] ~ dnorm(s_mean[k],s_prec[k]) }
  sigma ~ dunif(0,100)
}
'
sources = sourcesdemo[,4:5]
data=list(y=consumers,s_mean=sources[,1],s_prec=1/sources[,
          N=length(consumers),K=nrow(sources),
          alpha=rep(1,nrow(sources)))
model=jags.model(textConnection(modelstring), data=data)
output=coda.samples(model=model,variable.names=c("p"),n.ite
```

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:

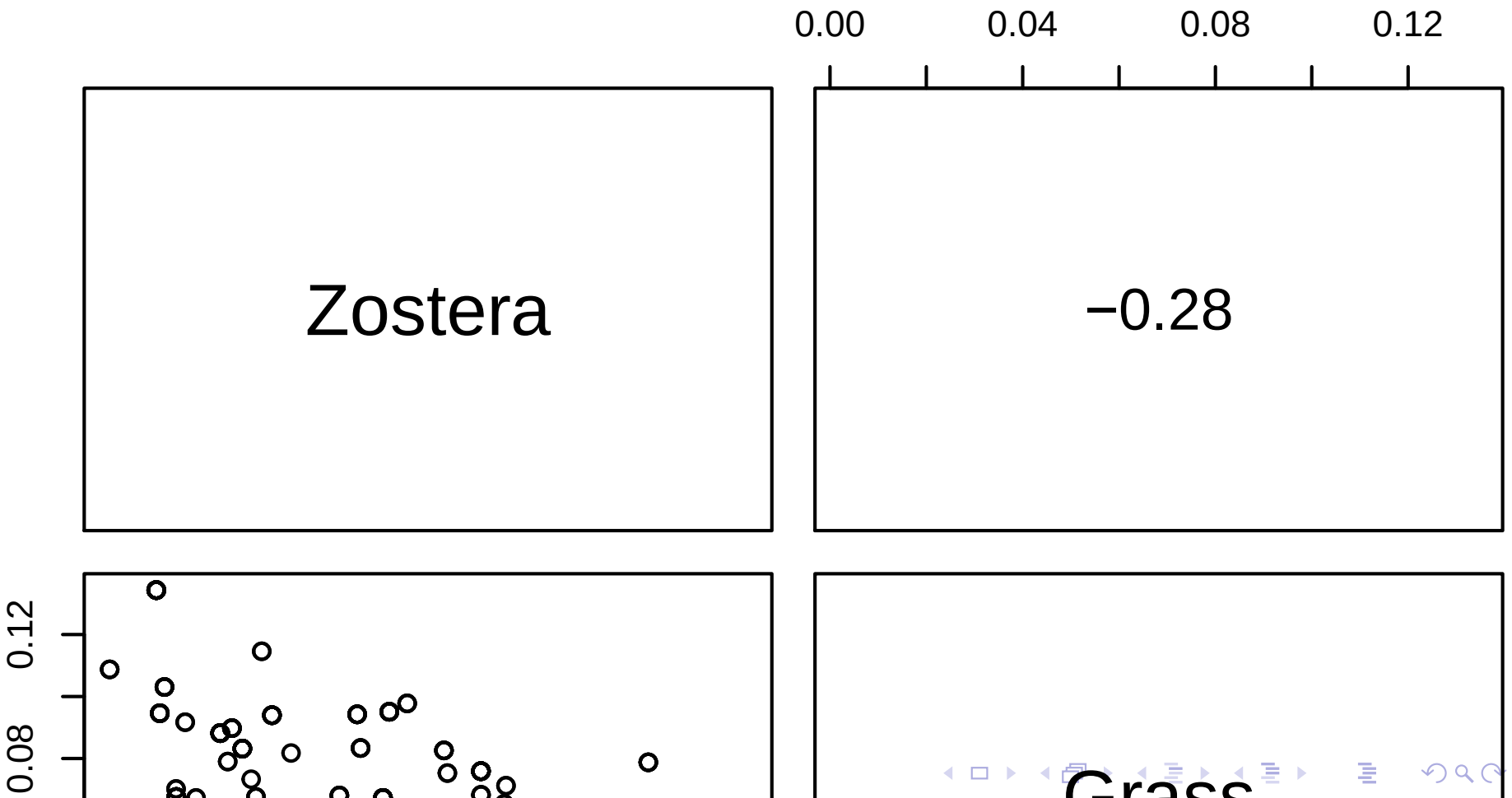
```
head(as.matrix(output[[1]]),2)
```

```
##           p[1]       p[2]      p[3]      p[4]
## [1,] 0.2945057 0.01663469 0.3375181 0.3513415
## [2,] 0.2945057 0.01663469 0.3375181 0.3513415
```

- 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 = as.matrix(output[[1]])
colnames(out_2) = sourcesdemo[,1]
pairs(out_2, lower.panel = panel.smooth, upper.panel = pane
```



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

```
modelstring ='
model {
  for(i in 1:N) {
    y[i] ~ dnorm(inprod(p*q,s+c)/inprod(p,q),1/pow(sigma,2)
  }
  p ~ ddirch(alpha)
  for(k in 1:K) {
    s[k] ~ dnorm(s_mean[k],s_prec[k])
    c[k] ~ dnorm(c_mean[k],c_prec[k])
  }
  sigma ~ dunif(0,100)
}
'
data(concdepdemo)
data(correctionsdemo)
data=list(y=consumers,s_mean=sources[,1],s_prec=1/sources[,
          c_mean=correctionsdemo[,4],c_prec=1/correctionsde
          q=concdepdemo[,4],N=length(consumers),K=nrow(sou
```

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 $j = 1, \ldots, J$
- We now have source values s_{jk}, TEF values c_{jk}, concentration dependencies q_{jk}, and each isotope has its own residual standard deviation σ_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_prec[k])
  }
}
```

- 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 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

```
modelstring ='
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])
    }
  }
  p ~ ddirch(alpha)
  for(j in 1:J) {
    var_y[j] <- inprod(pow(p*q[,j],2),1/s_prec[,j]+1/c_prec
      + pow(sigma[j],2)
  }
  for(j in 1:J) { sigma[j] ~ dunif(0,100) }
}
'
```

Full SIAR model: R code

```
sources = as.matrix(sourcesdemo[,2:5])
tefs = as.matrix(correctionsdemo[,2:5])
cd = as.matrix(concdepdemo[,c(2,4)])
data=list(y=geese1demo,s_mean=sources[,c(1,3)],s_prec=1/sou
          c_mean=tefs[,c(1,3)],c_prec=1/tefs[,c(2,4)]^2,
          q=cd,N=nrow(geese1demo),
          J=ncol(geese1demo),alpha=rep(1,nrow(sources)))
model=jags.model(textConnection(modelstring), data=data)
output=coda.samples(model=model,variable.names=c("p",'sigma
```

Summary of posterior dietary proportions

```
out_2 = as.matrix(output[[1]])
colnames(out_2) = c(as.character(sourcesdemo[,1]),'SD1','SD
t(round(apply(out_2,2,quantile,probs=c(0.025,0.5,0.975)),2)
```

```
##                2.5%  50% 97.5%
## Zostera        0.41 0.60  0.81
## Grass          0.03 0.07  0.12
## U.lactuca      0.01 0.11  0.33
## Enteromorpha 0.01 0.18  0.46
## SD1            0.02 0.38  1.48
## SD2            0.06 0.88  2.50
```

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

Running SIAR

- The SIAR R package runs 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 α values to put in proper prior information
- It includes convergence checking
- Most of this covered in the practical this afternoon

Running SIAR 2

- Running SIAR as simple as giving it the relevant parts:

```
out = siarmcmcdirichletv4(geese1demo,sourcesdemo,correctio
```

```
out_2 = out$output
colnames(out_2) = c(as.character(sourcesdemo[,1]),'SD1','SI
t(round(apply(out_2,2,quantile,probs=c(0.025,0.5,0.975)),2)
```

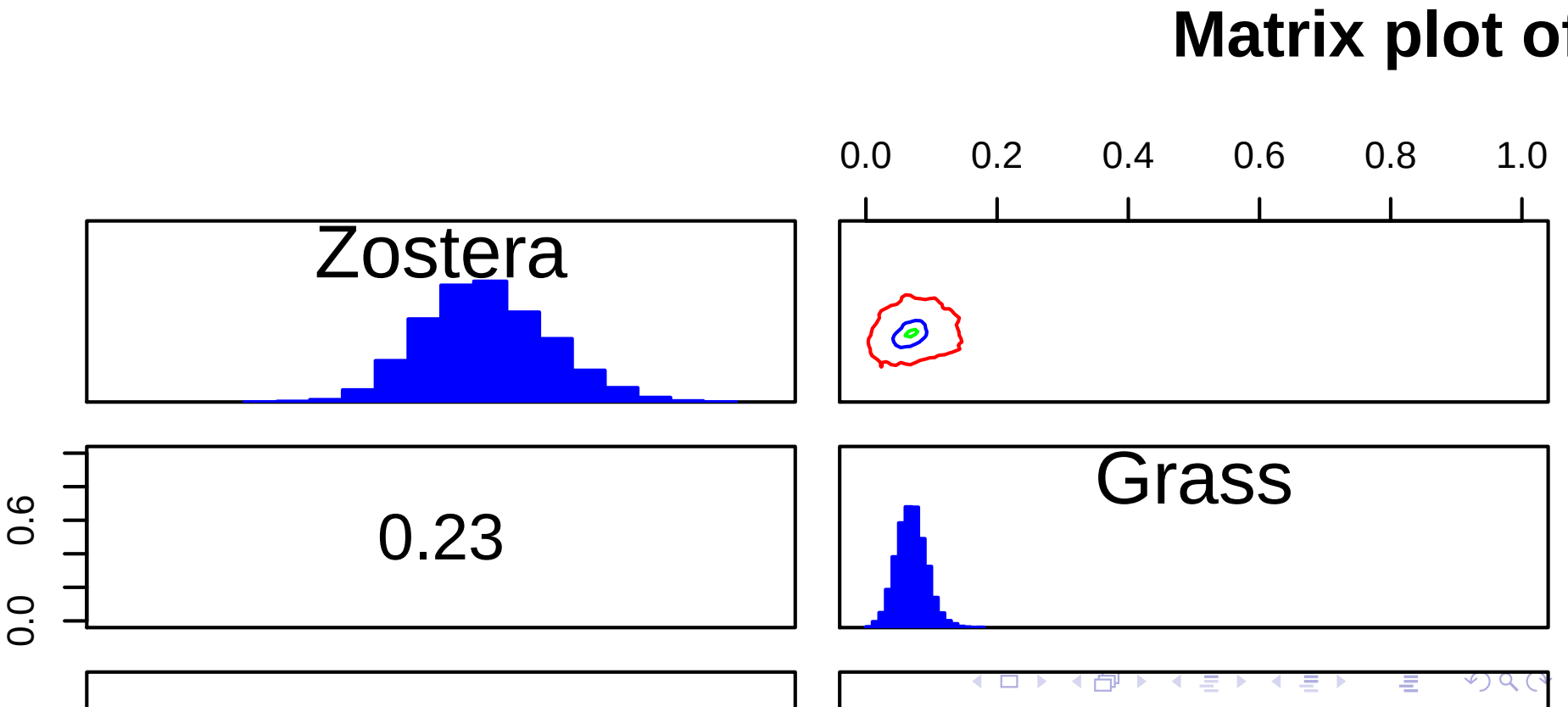
```
##                2.5%  50% 97.5%
## Zostera        0.40 0.56  0.77
## Grass          0.03 0.07  0.12
## U.lactuca      0.01 0.12  0.35
## Enteromorpha 0.01 0.22  0.49
## SD1            0.02 0.37  1.48
## SD2            0.07 0.90  2.42
```

- Most people get stuck using SIAR just getting their data into the right format

Matrix plot of dietary proportions

```
siarmatrixplot(out)
```

```
## Matrix plot of groups proportions.
## Producing plot.....
##
```



Summary

- The SIAR model is just a complicated regression-type model
- 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
- More details on running SIAR in the practical this afternoon