Using hierarchical joint models to study reproductive interactions in plant communities: Individual-level fitness data

Øystein H. Opedal & Stein Joar Hegland

30 April 2019

Contact: oystein.opedal@helsinki.fi

1. Introduction

Here, we demonstrate how to set up a HMSC analysis of fitness data (e.g. seed set) for 5 species cooccurring on 50 plots. Two traits (z_1 and z_2) are measured for each individual plant. We will use these data to estimate residual fitness correlations among species, i.e. how the fitness of each species in each plot affects the fitness of other species when growing in those plots.

Furthermore, we will use the trait and fitness data to jointly estimate phenotypic selection gradients for the two traits in each species using the method of Lande and Arnold 1983 (Evolution).

For technical details about the HMSC model, see Ovaskainen et al. 2017 (Ecology Letters) and Tikhonov et al. 2019 (bioRxiv).

2. Preparing data

Load the study design file

The file `SimulatedStudyDesign.csv` includes trait measurements for 5 species occurring on 50 plots, and an indicator of which species each row corresponds to. There is more than one individual of each species in some plots.

```
library(Hmsc)
library(corrplot)

alldat = read.csv("SimulatedStudyDesign.csv")
head(alldat)
```

```
##   plot       z1       z2 Species.1 Species.2 Species.3 Species.4 Species.5
## 1    1 19.72078 15.66592         1        NA        NA        NA        NA
## 2    1 14.75698 12.13327         1        NA        NA        NA        NA
## 3    1 21.17294 16.06965         1        NA        NA        NA        NA
## 4    2 21.41745 17.41753         1        NA        NA        NA        NA
## 5    2 19.24138 12.08059         1        NA        NA        NA        NA
## 6    2 18.92766 15.35306         1        NA        NA        NA        NA
```

Prepare the trait data

Trait data and other covariates are provided as a dataframe (`XData`), and a formula describing the regression equation (`XFormula`). The formula follows standard `R` syntax. HMSC does not currently allow missing values in XData, hence individuals with missing values must be excluded, or the missing values replaced with means or other gap-filling techniques.

```
XData = alldat[,2:3]
head(XData)
```

```
##         z1       z2
## 1 19.72078 15.66592
## 2 14.75698 12.13327
## 3 21.17294 16.06965
## 4 21.41745 17.41753
## 5 19.24138 12.08059
## 6 18.92766 15.35306
```

```
XFormula = ~ z1 + z2
```

Prepare the studyDesign dataframe

The study design is provided as a dataframe containing factors indicating the hierarchical structure of the data. Here, individuals belong to one of the 50 plots. Additional random effects can be included by adding columns to the studyDesign dataframe.

```
studyDesign = data.frame(plot = alldat$plot)
studyDesign$plot = as.factor(studyDesign$plot)
head(studyDesign)
```

```
##   plot
## 1    1
## 2    1
## 3    1
## 4    2
## 5    2
## 6    2
```

Generate fitness data

We simulate fitness data for each individual as a function of their traits z_1 and z_2. We let the fitness of Species 1 in each plot affect the fitness of Species 2 and 3 in that plot positively, and the fitness of Species 4 and 5 in that plot negatively.

Note that each line in the **Y** matrix correspond to one individual of one species, and all other species are set to NA.

```
Y = alldat[,4:8]

alpha = c(25, 7, 1, 5, 40)
beta_z1 = c(-1.8, 0.5, -1.2, 0.4, -1.1)
beta_z2 = c(1.4, -0.3, 0.7, 0.4, 0.2)
sp1eff = c(0, 1, 1, -1, -1)
set.seed(1)

Y[,1] = alpha[1] + beta_z1[1]*XData$z1 + beta_z2[1]*XData$z2 + rnorm(nrow(Y), 0, 1) +
        rnorm(50,0,2)[studyDesign$plot]

sp1plots = tapply(Y[,1], studyDesign$plot, mean, na.rm=T)
```

```r
for(s in 2:5){
Y[,s] = alpha[s] + beta_z1[s]*XData$z1 + beta_z2[s]*XData$z2 + rnorm(nrow(Y), 0, 1) +
        rnorm(50,0,.5)[studyDesign$plot] + sp1eff[s]*sp1plots[studyDesign$plot]
}

Y = Y*as.numeric(alldat[,4:8]>0)
apply(Y, 2, range, na.rm=T)
```

```
##      Species.1 Species.2 Species.3  Species.4  Species.5
## [1,]  3.650969  17.42882  5.640481  0.4459979  0.5880241
## [2,] 18.087696  32.42939 21.207792 15.7464919 15.6752873
```

```r
Y[c(1:5, 363:367),]
```

```
##     Species.1 Species.2 Species.3 Species.4 Species.5
## 1   11.944340        NA        NA        NA        NA
## 2   16.743559        NA        NA        NA        NA
## 3    9.686499        NA        NA        NA        NA
## 4   11.283327        NA        NA        NA        NA
## 5    6.462759        NA        NA        NA        NA
## 363        NA        NA        NA        NA  8.207977
## 364        NA        NA        NA        NA  7.524041
## 365        NA        NA        NA        NA  6.636924
## 366        NA        NA        NA        NA  7.404858
## 367        NA        NA        NA        NA  8.607421
```

Convert fitness values to relative fitness

To estimate selection gradients, we divide the fitness values by the mean fitness for each species.

```r
Y = apply(Y, 2, function(x) x/mean(x, na.rm=T))
```

3. Setup the HMSC model

Define a HMSC random level for plots

A plot-level random effect will allow us to assess residual correlations among species' fitness values in a given plot. Multiple random levels can be set up. For spatially explicit data, the argument `units` is replaced by providing a matrix of x and y coordinates with the argument `sData`.

```r
rL1 = HmscRandomLevel(units = unique(studyDesign$plot))
```

Setup the HMSC model

At this stage we also set the error distribution of the analyses, here "normal" for Gaussian errors.

```r
m = Hmsc(Y=as.matrix(Y), XData = XData,  XFormula = XFormula,
         dist = "normal", studyDesign = studyDesign, ranLevels = list(plot=rL1))
```

4. Sample the posterior distribution

As with all MCMC-based Bayesian analyses, the posterior needs to be sampled until convergence. A good strategy is to start with a small number of iterations, and then increase the number of iterations until the results converge. We run 2 replicate MCMC chains to assess whether these converge.

```
samples = 1000
thin = 5
adaptNf = .4*(samples*thin)
transient = .5*(samples*thin)
nChains = 2

start = Sys.time()
m = sampleMcmc(m, samples = samples, thin = thin, transient = transient,
             verbose = F, adaptNf = rep(adaptNf, m$nr), nChains = nChains)
```

```
## [1] "Computing chain 1"
## [1] "Computing chain 2"
```

```
Sys.time()-start
```

```
## Time difference of 15.36469 mins
```

5. Assessing model convergence

To assess chain convergence, we can look at posterior trace plots, effective sample sizes, and potential scale reduction factors. In this case, all of these indicate convergence of the sampling, because there is no trend in the posterior trace plots, the two chains overlap, the effective sample sizes are close to the number of samples, and the potential scale reduction factors are close to 1.

Extract posterior and convert to `Coda` object

```
post = convertToCodaObject(m)
plot(post$Beta[,1:3])
```

Effective sample size for beta parameters

```
summary(effectiveSize(post$Beta))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1737    2000    2000    2000    2035    2201
```

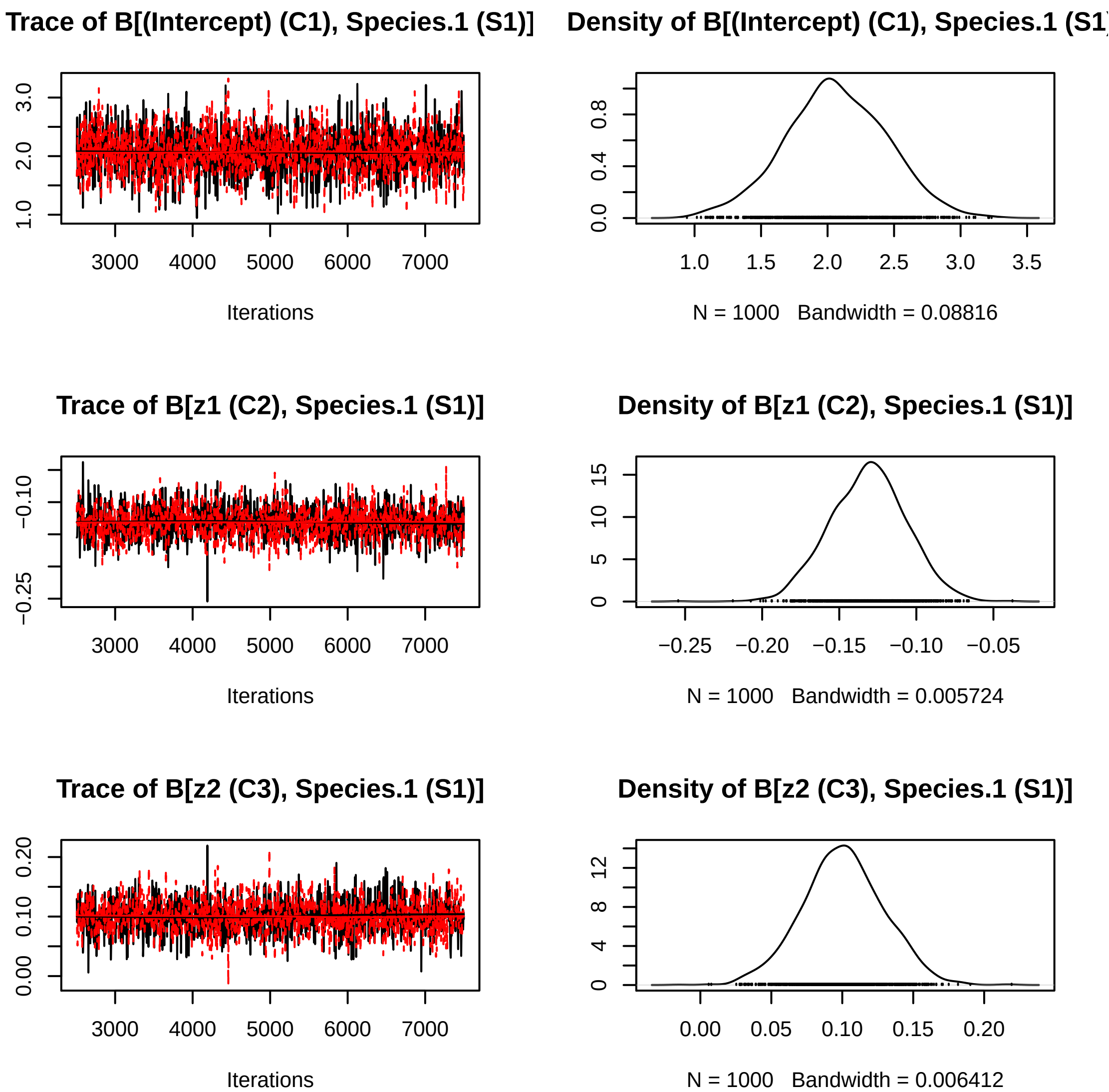

Figure 1: Posterior trace plot of the first three beta parameters

Potential scale reduction factors for beta parameters

```
summary(gelman.diag(post$Beta)$psrf)
```

```
##    Point est.       Upper C.I.
##  Min.   :0.9994   Min.   :0.9997
##  1st Qu.:0.9999   1st Qu.:1.0003
##  Median :1.0007   Median :1.0040
##  Mean   :1.0013   Mean   :1.0076
##  3rd Qu.:1.0017   3rd Qu.:1.0112
##  Max.   :1.0059   Max.   :1.0282
```

5. Evaluate model fit

HMSC comes with tools for computing measures of model fit for each species

```
predY = computePredictedValues(m)
MF = evaluateModelFit(m, predY)
MF
```

```
## $RMSE
## [1] 0.15777165 0.08281468 0.15576911 0.21396909 0.22597466
##
## $R2
## [1] 0.7000790 0.8197435 0.7550054 0.8296127 0.7938286
```

6. Assessing parameter estimates

Compute and plot variance partitioning

HMSC also comes with tools for performing variance component analyses and plotting the results. Because we have many `NA`'s in the $\mathbf{Y}$ matrix, we use the `na.ignore = TRUE` option to compute the variances and covariances of the traits only for those sites where the species is present.

```
group = c(1, 1, 2)
groupnames = m$covNames[-1]
VP = computeVariancePartitioning(m, group = group, groupnames = groupnames, na.ignore = TRUE)
par(mar=c(4,4,2,12), xpd=T)
plotVariancePartitioning(m, VP = VP, args.legend=list(x=9.3, y=1, bty="n"))
```

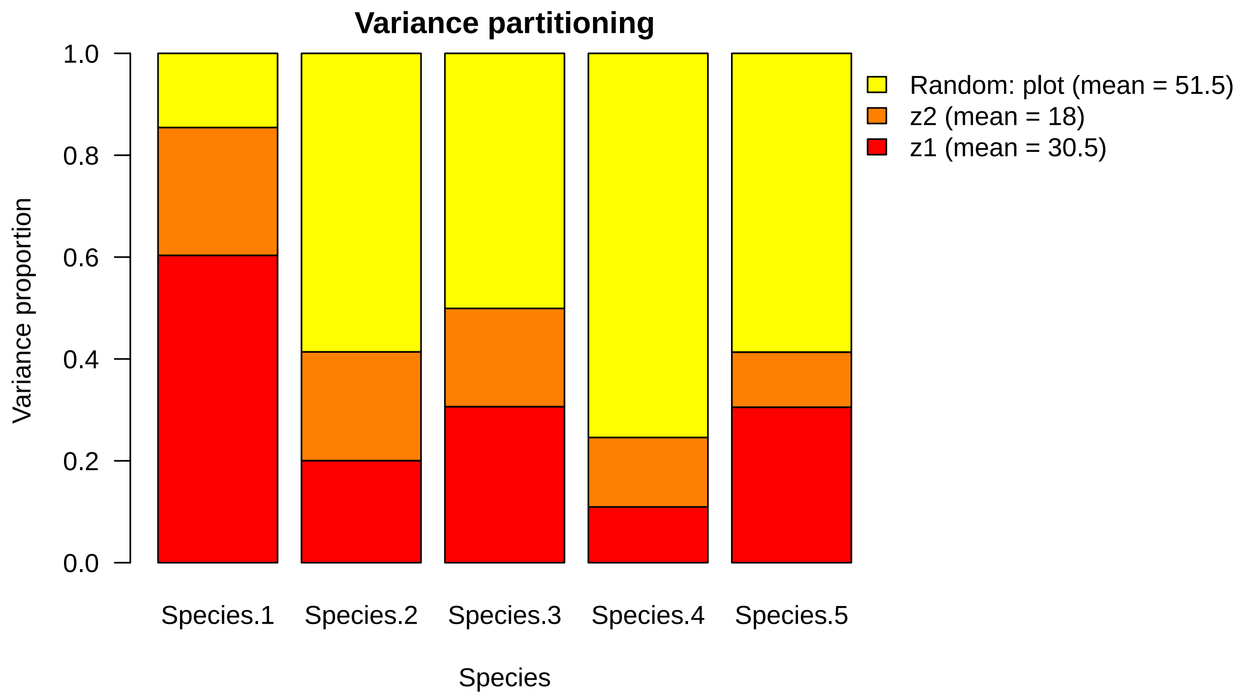

Figure 2: Variance partitioning for each species

Extract and plot beta parameters

The regression coefficients for the fixed part of the HMSC model can be extracted with the `getPostEstimate` function and visualised with the `plotBeta` function.

```
mbeta = getPostEstimate(m, "Beta")
mbeta
```

```
## $mean
##            [,1]        [,2]        [,3]        [,4]          [,5]
## [1,]  2.0653893  0.92672275  1.31907897 -0.94484482  3.0367872302
## [2,] -0.1311632  0.01288881 -0.05905772  0.03695991 -0.1131187361
## [3,]  0.1006101 -0.01164001  0.02591665  0.05586867  0.0003737696
##
## $support
##        [,1]  [,2]   [,3]  [,4]   [,5]
## [1,] 1.0000 0.866 0.8540 0.119 0.9925
## [2,] 0.0000 0.599 0.1525 0.825 0.0525
## [3,] 0.9995 0.353 0.7295 0.912 0.5025
##
## $supportNeg
##       [,1]  [,2]   [,3]  [,4]   [,5]
## [1,] 0e+00 0.134 0.1460 0.881 0.0075
## [2,] 1e+00 0.401 0.8475 0.175 0.9475
## [3,] 5e-04 0.647 0.2705 0.088 0.4975
```

```
par(mar = c(0,0,0,0))
plotBeta(m, post = mbeta, "Mean", covOrder = "Vector", covVector = c(2:3), supportLevel=0)
```

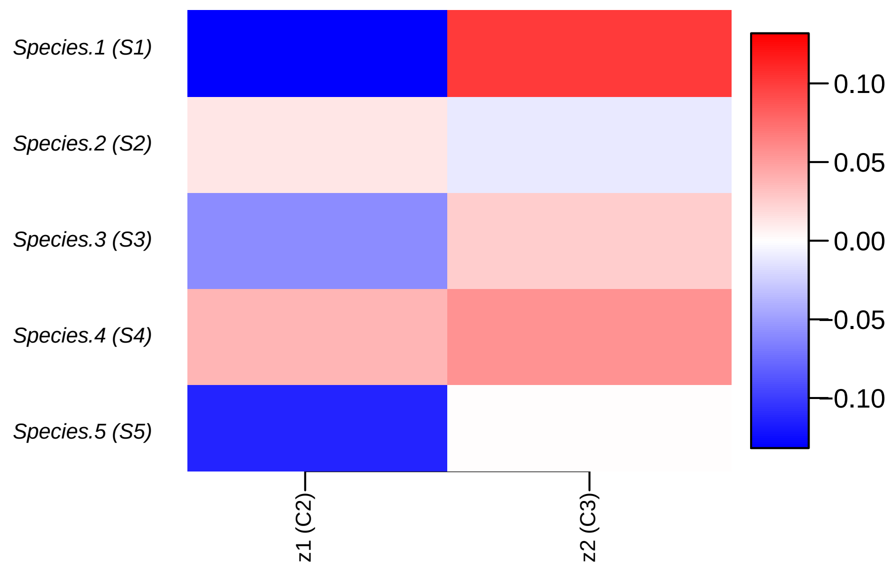

Figure 3: Heatmap of beta parameters (raw selection gradients)

Extract selection gradients

To compare the strength of selection across traits and species, selection gradients are typically standardized either by trait means or standard deviations. Mean-standardization yields a measure of proportional change in fitness per proportional change in the trait (i.e. an elasticity), while SD-standardization yields a measure of proportional change in fitness per standard deviation change in the trait (sometimes called selection intensity or i).

Table 1: Means, standard deviations, and selection gradients for each species

		Species 1	Species 2	Species 3	Species 4	Species 5
z1	Mean	19.26	20.01	16.12	22.58	17.6
	SD	2.04	1.06	1.21	1.31	1.17
	Beta	-0.13	0.01	-0.06	0.04	-0.11
	Beta_mean	-2.53	0.26	-0.95	0.83	-1.99
	Beta_var	-0.27	0.01	-0.07	0.05	-0.13
z2	Mean	14.47	14.9	22.62	19.87	14.57
	SD	1.71	1.65	1.4	1.26	1.47
	Beta	0.1	-0.01	0.03	0.06	0
	Beta_mean	1.46	-0.17	0.59	1.11	0.01
	Beta_var	0.17	-0.02	0.04	0.07	0

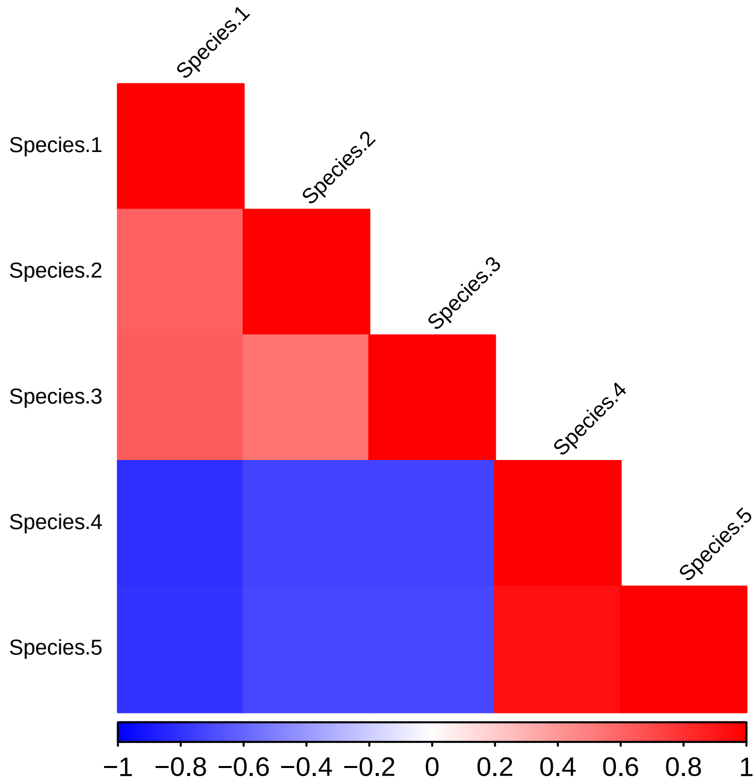

Figure 4: Residual fitness correlations

Compute and plot species associations

Residual correlations at each random level (here plot) can be extracted with the `computeAssociations`function. The estimated associations match those assumed when simulating the data.

```
OmegaCor = computeAssociations(m)
supportLevel = 0.75

toPlot = ((OmegaCor[[1]]$support>supportLevel) +
            (OmegaCor[[1]]$support<(1-supportLevel))>0)*OmegaCor[[1]]$mean

corrplot(toPlot, type="lower", tl.cex=.7,tl.col="black",tl.srt=45,method = "color",
         col=colorRampPalette(c("blue","white", "red"))(200), mar=c(0,0,0,1))
```