SFS 2023 Short Course – Bayesian Applications in Environmental and Ecological Studies with R and Stan

Song S. Qian

6/3/2023

Introductory Examples

In this session, we will learn about the use of Bayesian inference with examples commonly encountered in ecological and environmental literature. These examples typically involve response variable distributions that are standard and covered in statistical textbooks. We will emphasize the flexibility of the Bayesian approach and explore how these models can be expanded to increase their realism compared to the data generation process.

The guiding principle for Bayesian applications in environmental and ecological statistics should adhere to the three criteria of an applied statistical model recommended by Cox (1995):

1. A probability distribution model for the response variable.
2. A parameter vector that defines the distribution model, where some parameters reflect features of the system under study.
3. The inclusion or representation of the data-generating process.

Of these criteria, the last one, the data-generating process, holds significant importance. Our aim is to develop models that are relevant to the problem at hand and reflect likely causal relationships rather than mere correlations. Furthermore, environmental and ecological data are often observational, meaning we collect the data without prior knowledge of the appropriate model or method for analysis. The statistics we learned in graduate school typically assume that we already know the correct model. We teach statistics chapter by chapter, assuming that we have the right model in mind. However, when working with data from environmental and ecological monitoring programs, we often start collecting data without a clear idea of the questions we want to answer. For instance, most of the data used in studying the effects of climate change were collected without specific hypotheses in mind. In my opinion, analyzing environmental/ecological data often begins with identifying the problem to be addressed. Since all models are inherently wrong (yet some are useful), our focus lies in determining how and when a flawed model can be useful and in what way.

I have found the following process to be effective in analyzing environmental/ecological data:

1. Exploratory analysis using many plots
2. Start with simple models that can be readily implemented in R.
3. Explore the model fit and identify its weaknesses using the concept of "posterior simulation." This involves using the fitted model to predict what we want to learn and compare it with what we already know.
4. Based on the identified weaknesses, reformulate the model to address those shortcomings.
5. Implement the model in Stan to further assess problems such as computational stability and identifiability (e.g., the snake fungal disease example with unknown error rates).
6. Repeat the posterior simulation to determine where and when the model is useful.

By following this iterative process, we can refine our models, understand their limitations, and identify their utility in analyzing environmental and ecological data.

Example 1 – The Effect of Gulf of Mexico Hypoxia on Benthic Communities

Section 4.2.2, Qian et al (2022).

```
benthic.data <- read.csv(paste(dataDIR, "BenthicData.csv", sep = "/"), header = T)
benthic.data$area2 <- ordered(benthic.data$area2, levels = levels(ordered(benthic.data$area2))[c(2,
    1, 3)])
benthic.data$Area <- ordered(benthic.data$Area, levels = levels(ordered(benthic.data$Area))[c(2,
    1, 3)])
head(benthic.data)
```

```
##   Area   area2 Station Replicates Richness Abundance log.abund. Depth
## 1    H hypoxic       2          1       13        68       1.84  6.10
## 2    H hypoxic       2          2       19       186       2.27  6.10
## 3    H hypoxic       2          3       15       132       2.12  6.10
## 4    H hypoxic      16          1       10        37       1.58 21.13
## 5    H hypoxic      16          2       15        88       1.95 21.13
## 6    H hypoxic      16          3       22       107       2.03 21.13
```

```
names(benthic.data)
```

```
## [1] "Area"       "area2"      "Station"    "Replicates" "Richness"
## [6] "Abundance"  "log.abund." "Depth"
```

```
## Station: core,
```

Initial analysis (ANOVA) was presented in Baustian et al (2009). It was published, but unsatisfactory nevertheless. Qian et al (2009) presented a hierarchical modeling alternative implemented in WinBUGS. The original study designed the sampling to mimic a randomized experiment. However, the treatment (benthic hypoxia) cannot be randomly assigned. Confounding factors cannot be ignored. Instead of a hypothesis testing problem (ANOVA), we address the problem as an estimation problem – estimating benthic community abundance and richness in three zones, hypoxic zone, inshore, and offshore "controls."

- MLE model – multilevel model with sampling cores nested in three zones.

```
benthic.data$AS <- with(benthic.data, factor(paste(Area, Station, sep = ":")))
benthicAb.Lme1 <- lmer(log(Abundance) ~ 1 + (1 | AS) + (1 | Area), data = benthic.data)
benthicRn.Lme1 <- lmer(log(Richness) ~ 1 + (1 | AS) + (1 | Area), data = benthic.data)
summary(benthicAb.Lme1)
```

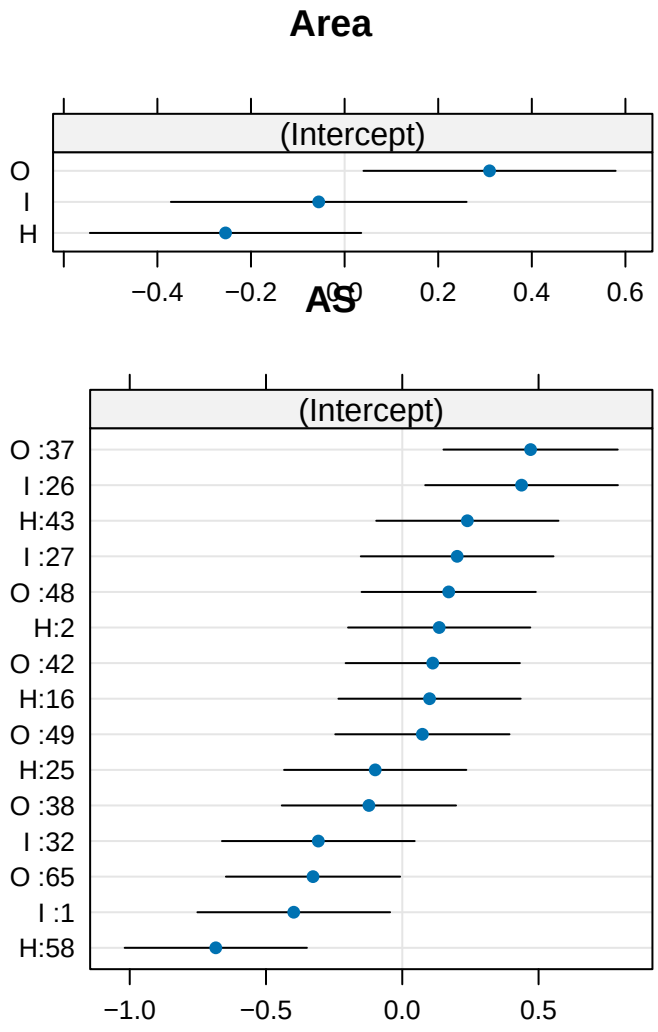
```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(Abundance) ~ 1 + (1 | AS) + (1 | Area)
##    Data: benthic.data
##
## REML criterion at convergence: 68.3
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.19360 -0.56238  0.06669  0.61965  1.45372
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  AS       (Intercept) 0.16598  0.4074
##  Area     (Intercept) 0.08157  0.2856
##  Residual             0.15250  0.3905
## Number of obs: 45, groups:  AS, 15; Area, 3
##
## Fixed effects:
```

```
##             Estimate Std. Error t value
## (Intercept)   4.7055     0.2047   22.99
```

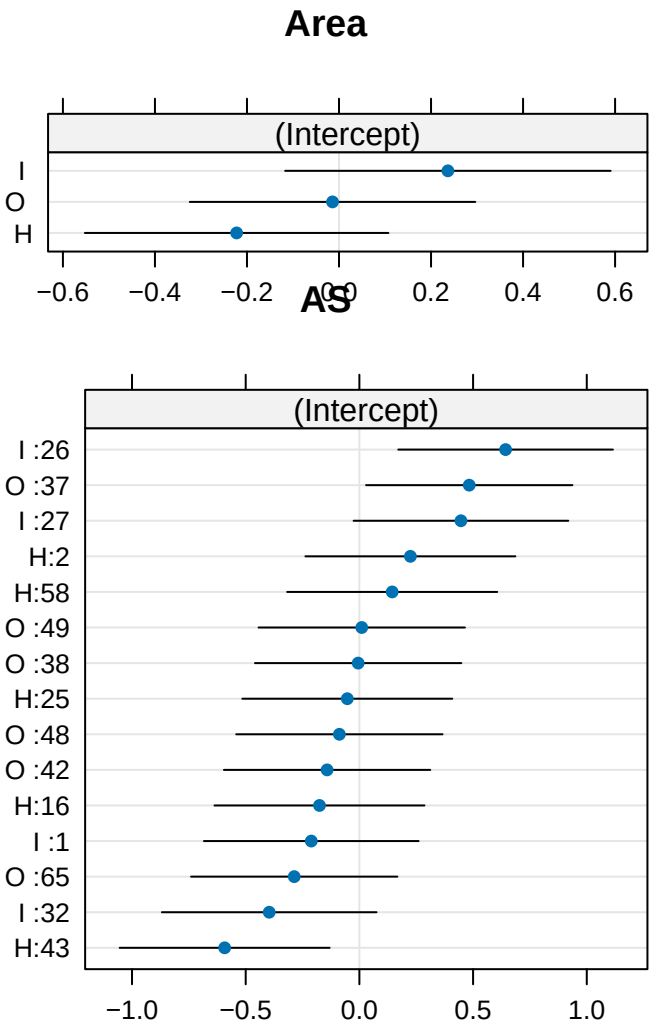
```
summary(benthicRn.Lme1)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(Richness) ~ 1 + (1 | AS) + (1 | Area)
##    Data: benthic.data
##
## REML criterion at convergence: 19.2
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.79130 -0.44849  0.02172  0.52833  2.12511
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  AS       (Intercept) 0.12665  0.3559
##  Area     (Intercept) 0.10414  0.3227
##  Residual             0.03539  0.1881
## Number of obs: 45, groups:  AS, 15; Area, 3
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   2.8456     0.2101   13.54
```

```
dotLmeAb1 <- dotplot(ranef(benthicAb.Lme1, condVar = T))
dotLmeRn1 <- dotplot(ranef(benthicRn.Lme1, condVar = T))
print(dotLmeRn1[[1]], pos = c(0, 0, 1, 0.75), more = T)
print(dotLmeRn1[[2]], pos = c(0, 0.65, 1, 1), more = F)
```




```
print(dotLmeAb1[[1]], pos = c(0, 0, 1, 0.75), more = T)
print(dotLmeAb1[[2]], pos = c(0, 0.65, 1, 1), more = F)
```



Within zone variance is too high for the MLE method to properly estimate the among zone variance.

- Bayesian model Using a Bayesian hierarchical model, separating cores within each zone:

$$y_{ijk} \sim N(\mu_{jk}, \sigma^2_{yk})$$

where, ijk represents the ith observation from the jth core, in zone k, and

$$\mu_{jk} \sim N(\theta_k, \sigma^2_z)$$

and finally,

$$\theta_k \sim N(\mu_{hyp}, \sigma^2_{hyp})$$

```
## Station: core, Area: zone
stan1_gom <- "
data{
  int K;  //total sample size
  int J;  //number of sediment cores
  int I;  //number of zones
  real y[K]; //observed response
  int core[K]; //core index
  int zone[K]; //zone index
  int core_zone[J]; //zone
}
```

```
parameters{
  real mu[J];
  real theta[I];
  real mu_hyp;
  real<lower=0> sigma_y[I];
  real<lower=0> sigma_i;
  real<lower=0> sigma_hyp;
}
model{
  sigma_hyp ~ normal(0,1); // for fast comuting
  for (i in 1:I){
    theta[i] ~ normal(mu_hyp, sigma_hyp);
  }
  for (j in 1:J){
    mu[j] ~ normal(theta[core_zone[j]], sigma_i);
  }
  for (k in 1:K){
    y[k] ~ normal(mu[core[k]], sigma_y[zone[k]]);
  }
}
generated quantities{
  real delta1;
  real delta2;
  real delta3;
  delta1 = theta[2]-theta[1];
  delta2 = theta[2]-theta[3];
  delta3 = theta[1]-theta[3];
}
"
stan.fit <- stan_model(model_code = stan1_gom)
```

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## using C compiler: ‘gcc (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0’
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/usr/lib/R/site-library/Rcpp/include/"  -I"/usr/lib/R/sit
## In file included from /usr/lib/R/site-library/RcppEigen/include/Eigen/Core:88,
##                  from /usr/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp
##                  from <command-line>:
## /usr/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown type na
##   628 | namespace Eigen {
##       | ^~~~~~~~
## /usr/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: error: expected ‘=’,
##   628 | namespace Eigen {
##       |                 ^
## In file included from /usr/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp
##                  from <command-line>:
## /usr/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or di
##    96 | #include <complex>
##       |          ^~~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:191: foo.o] Error 1
```

We used a `generated quantities` code block to directly calculate the differences between two zones.

Now organizing input data and run the model

```
stan.in3 <- function(data = benthic.data, y.col=5,  ## richness
                     chains=nchains){ ## no slope
    n <- dim(data)[1]
    y <- log(data[,y.col])
    core <- as.numeric(ordered(data$Station))
    n.core <- max(core)
    zone <- as.numeric(ordered(data$Area))
    n.zone <- max(zone)
    oo <- order(core)
    ind <- cumsum(table(core[oo]))
    Core.zone <- zone[oo][ind] ## each core belongs to which zone
    stan.dat <- list(K=n, J=n.core, I=n.zone, y=y, core=core,
                     zone=zone, core_zone=Core.zone)
    inits <- list()
    for (i in 1:chains)
        inits[[i]] <- list( mu = rnorm(n.core), theta=rnorm(n.zone),
                            mu_hyp=rnorm(1), sigma_i=runif(1),
                            sigma_y=runif(n.zone), sigma_hyp=runif(1))
    parameters <- c("mu","theta","mu_hyp", "sigma_y", "sigma_i",
                    "sigma_hyp",  "delta1", "delta2","delta3")
    return(list(para=parameters, data=stan.dat, inits=inits,
                n.chains=chains))
}

input.to.stan <- stan.in3()  ## long-runs -- results without sigma_hyp prior saved
fit2keep <- sampling(stan.fit, data = input.to.stan$data,
                     init=input.to.stan$inits,
                     pars = input.to.stan$para,
                     iter=niters, thin=nthin,
                     chains=input.to.stan$n.chains,
                     control=list(adapt_delta=0.99, max_treedepth=20))
```

```
## Warning: There were 108 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

```
print(fit2keep)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
```

```
## mu[1]       2.37    0.00 0.09  2.21  2.32  2.37  2.43  2.56  1143 1.01
## mu[2]       2.72    0.00 0.13  2.46  2.64  2.72  2.81  2.98  2045 1.00
## mu[3]       2.69    0.00 0.14  2.43  2.61  2.70  2.78  2.97  1112 1.01
## mu[4]       2.50    0.00 0.13  2.23  2.41  2.50  2.58  2.76  2586 1.00
## mu[5]       3.24    0.00 0.09  3.05  3.19  3.25  3.31  3.41  2486 1.00
## mu[6]       3.00    0.00 0.08  2.83  2.95  3.00  3.05  3.17  2551 1.00
## mu[7]       2.47    0.00 0.09  2.31  2.41  2.47  2.52  2.65  1299 1.00
## mu[8]       3.61    0.00 0.13  3.33  3.53  3.62  3.70  3.85  2264 1.00
## mu[9]       3.05    0.01 0.13  2.79  2.96  3.04  3.13  3.32   573 1.02
## mu[10]      3.27    0.00 0.13  3.00  3.19  3.27  3.35  3.51  2144 1.00
## mu[11]      2.82    0.00 0.14  2.54  2.73  2.82  2.90  3.09  1211 1.01
## mu[12]      3.32    0.00 0.13  3.07  3.24  3.33  3.41  3.57  1986 1.01
## mu[13]      3.23    0.00 0.13  2.97  3.15  3.23  3.31  3.47  2547 1.00
## mu[14]      1.94    0.00 0.14  1.67  1.84  1.93  2.02  2.24  1691 1.00
## mu[15]      2.84    0.00 0.13  2.58  2.75  2.85  2.93  3.10  2383 1.00
## theta[1]    2.79    0.01 0.20  2.41  2.66  2.79  2.92  3.16  1539 1.01
## theta[2]    2.61    0.01 0.20  2.22  2.48  2.61  2.73  3.04   391 1.02
## theta[3]    3.14    0.01 0.19  2.72  3.02  3.15  3.27  3.49   672 1.01
## mu_hyp      2.85    0.01 0.37  2.09  2.67  2.85  3.03  3.64  1757 1.01
## sigma_y[1]  0.14    0.00 0.05  0.08  0.11  0.13  0.16  0.26  1856 1.00
## sigma_y[2]  0.24    0.00 0.06  0.15  0.19  0.23  0.27  0.39  2300 1.00
## sigma_y[3]  0.23    0.00 0.05  0.16  0.20  0.23  0.26  0.35  2322 1.00
## sigma_i     0.41    0.00 0.11  0.25  0.33  0.40  0.47  0.66   982 1.01
## sigma_hyp   0.51    0.01 0.38  0.03  0.25  0.41  0.67  1.52   809 1.01
## delta1     -0.18    0.01 0.25 -0.69 -0.34 -0.17 -0.01  0.28  2237 1.00
## delta2     -0.52    0.01 0.29 -1.07 -0.72 -0.55 -0.34  0.02   418 1.02
## delta3     -0.34    0.01 0.27 -0.88 -0.52 -0.34 -0.15  0.12   730 1.01
## lp__       53.10    0.10 4.21 43.22 50.56 53.43 56.15 60.06  1626 1.01
##
## Samples were drawn using NUTS(diag_e) at Tue May 30 11:53:30 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
rich_stan <- rvsims(as.matrix(as.data.frame(extract(fit2keep))))

input.to.stan <- stan.in3(y.col = 6)  ## abundance
fit2keep <- sampling(stan.fit, data = input.to.stan$data,
                     init=input.to.stan$inits,
                     pars = input.to.stan$para,
                     iter=niters, thin=nthin,
                     chains=input.to.stan$n.chains,
                     control=list(adapt_delta=0.99, max_treedepth=20))
```

```
## Warning: There were 28 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
print(fit2keep)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
```

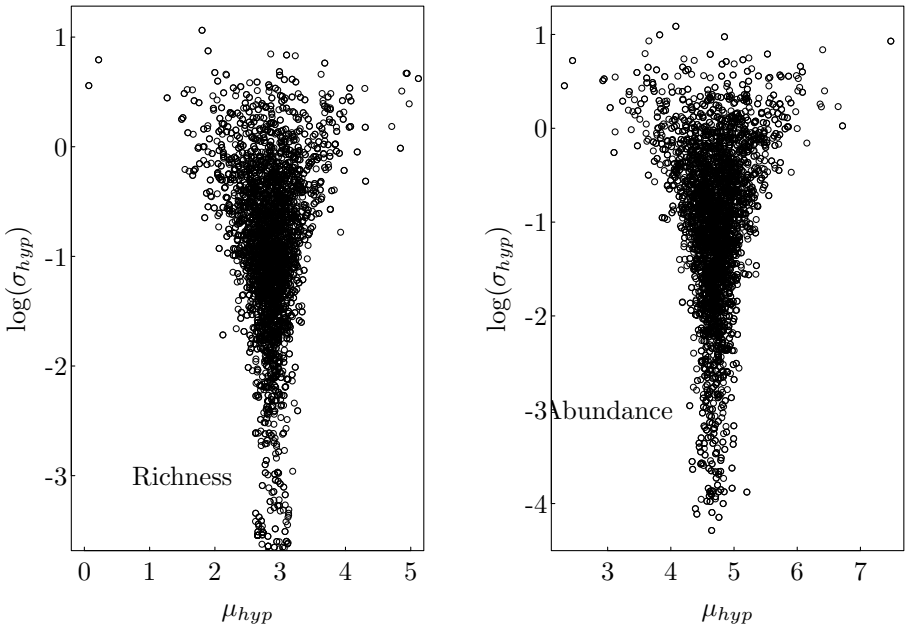
```
##             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## mu[1]       4.77    0.01 0.26  4.28  4.60  4.77  4.93  5.29  2371    1
## mu[2]       4.67    0.00 0.25  4.18  4.51  4.67  4.84  5.14  2463    1
## mu[3]       4.33    0.00 0.24  3.85  4.17  4.33  4.49  4.79  2584    1
## mu[4]       4.43    0.00 0.24  3.97  4.26  4.42  4.59  4.91  2472    1
## mu[5]       5.48    0.01 0.31  4.78  5.29  5.52  5.69  6.03  2192    1
## mu[6]       5.31    0.01 0.28  4.73  5.14  5.33  5.50  5.81  2188    1
## mu[7]       4.61    0.01 0.28  4.08  4.43  4.60  4.79  5.19  2316    1
## mu[8]       5.15    0.00 0.22  4.66  5.01  5.15  5.30  5.58  2477    1
## mu[9]       4.69    0.00 0.20  4.30  4.55  4.69  4.81  5.09  2348    1
## mu[10]      4.55    0.00 0.20  4.16  4.42  4.56  4.68  4.94  2352    1
## mu[11]      3.98    0.01 0.28  3.45  3.79  3.97  4.15  4.56  2415    1
## mu[12]      4.60    0.00 0.20  4.21  4.47  4.60  4.73  4.99  2226    1
## mu[13]      4.70    0.00 0.20  4.31  4.57  4.70  4.83  5.08  2543    1
## mu[14]      4.60    0.00 0.24  4.12  4.44  4.60  4.76  5.06  2405    1
## mu[15]      4.42    0.00 0.20  4.02  4.29  4.41  4.55  4.80  2503    1
## theta[1]    4.95    0.01 0.26  4.46  4.77  4.95  5.11  5.46  2265    1
## theta[2]    4.48    0.00 0.23  4.02  4.33  4.49  4.63  4.93  2165    1
## theta[3]    4.69    0.00 0.19  4.31  4.58  4.69  4.80  5.07  2422    1
## mu_hyp      4.71    0.01 0.39  3.88  4.52  4.70  4.89  5.52  2484    1
## sigma_y[1]  0.52    0.00 0.17  0.29  0.39  0.48  0.60  0.94  2383    1
## sigma_y[2]  0.50    0.00 0.13  0.31  0.41  0.47  0.56  0.82  2414    1
## sigma_y[3]  0.38    0.00 0.08  0.26  0.33  0.37  0.43  0.58  2433    1
## sigma_i     0.42    0.00 0.14  0.17  0.33  0.41  0.50  0.74  2233    1
## sigma_hyp   0.50    0.01 0.39  0.04  0.22  0.41  0.67  1.50  2176    1
## delta1     -0.47    0.01 0.36 -1.18 -0.71 -0.46 -0.19  0.13  2130    1
## delta2     -0.21    0.01 0.28 -0.80 -0.38 -0.20 -0.03  0.28  2287    1
## delta3      0.25    0.01 0.30 -0.29  0.04  0.24  0.45  0.85  2219    1
## lp__       20.34    0.10 4.64 10.45 17.63 20.72 23.34 28.17  2077    1
##
## Samples were drawn using NUTS(diag_e) at Tue May 30 11:53:36 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
abun_stan <- rvsims(as.matrix(as.data.frame(extract(fit2keep))))

## Extract output
## zone means
tempA <- abun_stan[1:15]
names(tempA) <- paste("$\\mu_{", 1:15, "}$", sep="")
tempR <- rich_stan[1:15]
names(tempR) <- paste("$\\mu_{", 1:15, "}$", sep="")
## zone mean differences
delta0R <- rich_stan[25:27]
delta0A <- abun_stan[25:27]
names(delta0R) <- c("H-I","H-O","I-O")
names(delta0A) <- c("H-I","H-O","I-O")
```

We noticed warning messages about divergence transition. It is a sign of computational difficulties in sampling the posterior distribution.

```
par(mfrow=c(1,2),mar=c(3,3,1,1),
    mgp=c(1.25,0.125,0), las=1,tck=0.01)
plot(rich_stan$mu_hyp, log(rich_stan$sigma_hyp), cex=0.5,
```

```
     ylim=c(-3.5,log(3)), xlim=c(-0,5),
     xlab="$\\mu_{hyp}$", ylab="$\\log(\\sigma_{hyp})$")
text(1.5,-3, "Richness")
plot(abun_stan$mu_hyp, log(abun_stan$sigma_hyp), cex=0.5,
##     ylim=c(-3.5,log(150)), xlim=c(-100,100),
     xlab="$\\mu_{hyp}$", ylab="$\\log(\\sigma_{hyp})$")
text(3,-3, "Abundance")
```



An issue we encountered in this analysis is the presence of Neal's funnel, indicating a lack of information to adequately quantify both μ_{hyp} and σ^2_{hyp} simultaneously. This often results in highly correlated θ_k values. To address this, we require an informative prior for one of the two parameters. In this case, a strong prior was used for σ^2_{hyp}, specifically a half-normal distribution N(0,1). When using a noninformative prior, the computational performance of the program is significantly slower. To mitigate computational difficulties, we can reparameterize θ_k. Instead of directly modeling it as a normal random variable with parameters μ_{hyp} and σ^2_{hyp}, we introduce a new parameter $z_k \sim N(0,1)$ and model θ_k as a transformed variable: $\theta_k = \mu_{hyp} + \sigma_{hyp} z_k$. Consequently, we no longer directly sample θ_k, which effectively reduces the occurrence of divergent transitions. However, the underlying model remains the same, and the presence of Neal's funnel persists.

The computational challenges encountered during this analysis prompted us to investigate the problem further. We realized that the available data do not provide sufficient information to simultaneously quantify θ_k, μ_{hyp}, and σ^2_{hyp}, primarily due to the substantial within-zone variation among the cores. The multilevel model indicated that including zone as a random effect does not effectively explain the overall variance. As a result, we proposed two alternative models that avoid directly parameterizing the among-zone variation.

In the first alternative, we removed the zone as a hierarchical factor and treated the core means as exchangeable:

$$\mu_{jk} \sim N(\mu_{hyp}, \sigma^2_{hyp})$$

We estimated each zone mean as the average of the means of the cores within that zone. To accommodate relatively constrained priors for variance parameters (e.g., half-normal N(0,0.5)), the response variable was standardized.

```
### First alternative (One-way ANOVA)
stan2_gom <- "
data{
  int K;  //total sample size
```

```
  int J;  //number of sediment cores
  real y[K]; //observed response
  int core[K]; //core index
}
parameters{
  real mu[J];
  real mu_hyp;
  real<lower=0> sigma_y;
  real<lower=0> sigma_hyp;
}
model{
  sigma_hyp ~ normal(0,0.5);
  sigma_y ~ normal(0,0.5);
  for (j in 1:J){
    mu[j] ~ normal(mu_hyp, sigma_hyp);
  }
  for (k in 1:K){
    y[k] ~ normal(mu[core[k]], sigma_y);
  }
}
"
stan.fit2 <- stan_model(model_code = stan2_gom)
```

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## using C compiler: 'gcc (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0'
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/usr/lib/R/site-library/Rcpp/include/"  -I"/usr/lib/R/sit
## In file included from /usr/lib/R/site-library/RcppEigen/include/Eigen/Core:88,
##                  from /usr/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp
##                  from <command-line>:
## /usr/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown type na
##   628 | namespace Eigen {
##       | ^~~~~~~~~
## /usr/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: error: expected '=',
##   628 | namespace Eigen {
##       |                 ^
## In file included from /usr/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp
##                  from <command-line>:
## /usr/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or di
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:191: foo.o] Error 1
```

```
stan.in4 <- function(data = benthic.data, y.col = 5, chains = nchains) {
    ## no slope
    n <- dim(data)[1]
    y <- log(data[, y.col])
    y_ave <- mean(y)
    y_sd <- sd(y)

    y <- (y - y_ave)/y_sd
```

```
    core <- as.numeric(ordered(data$Station))
    n.core <- max(core)

    stan.dat <- list(K = n, J = n.core, y = y, core = core)
    inits <- list()
    for (i in 1:chains) inits[[i]] <- list(mu = rnorm(n.core), mu_hyp = rnorm(1),
        sigma_y = runif(1), sigma_hyp = runif(1))
    parameters <- c("mu", "mu_hyp", "sigma_y", "sigma_hyp")
    return(list(para = parameters, data = stan.dat, inits = inits, n.chains = chains,
        y_cen = y_ave, y_spd = y_sd))
}

input.to.stan <- stan.in4()
fit2keep <- sampling(stan.fit2, data = input.to.stan$data, init = input.to.stan$inits,
    pars = input.to.stan$para, iter = niters, thin = nthin, chains = input.to.stan$n.chains)
print(fit2keep)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##            mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## mu[1]     -0.99    0.00 0.24 -1.47 -1.15 -1.00 -0.83 -0.51  2611    1
## mu[2]     -0.24    0.00 0.23 -0.70 -0.40 -0.25 -0.08  0.19  2401    1
## mu[3]     -0.32    0.00 0.23 -0.78 -0.47 -0.32 -0.17  0.13  2496    1
## mu[4]     -0.75    0.00 0.23 -1.22 -0.90 -0.75 -0.60 -0.29  2589    1
## mu[5]      0.77    0.00 0.23  0.31  0.63  0.78  0.92  1.22  2430    1
## mu[6]      0.27    0.00 0.23 -0.21  0.12  0.27  0.42  0.69  2489    1
## mu[7]     -0.81    0.00 0.23 -1.25 -0.96 -0.82 -0.67 -0.36  2731    1
## mu[8]      1.54    0.00 0.23  1.08  1.39  1.55  1.70  1.98  2507    1
## mu[9]      0.29    0.00 0.22 -0.15  0.15  0.28  0.43  0.74  2192    1
## mu[10]     0.79    0.00 0.23  0.34  0.64  0.79  0.94  1.22  2251    1
## mu[11]    -0.04    0.00 0.23 -0.48 -0.19 -0.04  0.12  0.41  2459    1
## mu[12]     0.90    0.00 0.23  0.43  0.75  0.91  1.06  1.35  2605    1
## mu[13]     0.71    0.00 0.23  0.26  0.56  0.71  0.85  1.15  2380    1
## mu[14]    -1.98    0.00 0.23 -2.43 -2.13 -1.98 -1.83 -1.49  2519    1
## mu[15]    -0.15    0.00 0.23 -0.62 -0.29 -0.15  0.01  0.29  2486    1
## mu_hyp     0.00    0.00 0.25 -0.48 -0.16  0.00  0.15  0.49  2527    1
## sigma_y    0.40    0.00 0.06  0.31  0.36  0.40  0.44  0.53  2312    1
## sigma_hyp  0.89    0.00 0.16  0.63  0.78  0.88  0.99  1.24  2387    1
## lp__       8.97    0.07 3.63  0.58  6.74  9.40 11.66 14.74  2368    1
##
## Samples were drawn using NUTS(diag_e) at Tue May 30 11:54:53 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
rich_stan2 <- rvsims(as.matrix(as.data.frame(extract(fit2keep))))

## processing output
core <- as.numeric(ordered(benthic.data$Station))
n.core <- max(core)
zone <- as.numeric(ordered(benthic.data$Area))
n.zone <- max(zone)
```

```r
oo <- order(core)
ind <- cumsum(table(core[oo]))
Core.zone <- zone[oo][ind]  ## each core belongs to which zone
input_sd <- input.to.stan$y_spd
input_mu <- input.to.stan$y_cen
## return to the original scale
core1.musR <- input_mu + input_sd * rich_stan2[1:15]
zone11.Rmu <- mean(core1.musR[Core.zone == 1])
zone12.Rmu <- mean(core1.musR[Core.zone == 2])
zone13.Rmu <- mean(core1.musR[Core.zone == 3])

deltaR11 = zone12.Rmu - zone11.Rmu
deltaR12 = zone12.Rmu - zone13.Rmu
deltaR13 = zone11.Rmu - zone13.Rmu

delta1R <- c(deltaR11, deltaR12, deltaR13)
names(delta1R) <- c("H-I", "H-O", "I-O")

## repeat for Abundance:
input.to.stan <- stan.in4(y.col = 6)
fit2keep <- sampling(stan.fit2, data = input.to.stan$data, init = input.to.stan$inits,
    pars = input.to.stan$para, iter = niters, thin = nthin, chains = input.to.stan$n.chains)
```

```
## Warning: There were 12 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```r
print(fit2keep)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##             mean se_mean   sd   2.5%    25%   50%   75% 97.5% n_eff Rhat
## mu[1]      -0.03    0.01 0.34  -0.69  -0.26 -0.02  0.20  0.64  2464    1
## mu[2]       0.12    0.01 0.34  -0.54  -0.11  0.12  0.34  0.76  2544    1
## mu[3]      -0.54    0.01 0.35  -1.23  -0.78 -0.54 -0.31  0.17  2513    1
## mu[4]      -0.35    0.01 0.34  -1.04  -0.57 -0.34 -0.12  0.32  2295    1
## mu[5]       1.39    0.01 0.40   0.54   1.13  1.40  1.66  2.13  1514    1
## mu[6]       1.06    0.01 0.37   0.29   0.81  1.07  1.32  1.71  2133    1
## mu[7]      -0.35    0.01 0.35  -1.02  -0.58 -0.34 -0.12  0.32  2391    1
## mu[8]       0.80    0.01 0.35   0.11   0.57  0.80  1.03  1.49  2088    1
## mu[9]       0.00    0.01 0.34  -0.66  -0.22  0.00  0.22  0.66  2436    1
## mu[10]     -0.23    0.01 0.35  -0.89  -0.45 -0.22  0.00  0.47  2441    1
## mu[11]     -1.22    0.01 0.39  -1.96  -1.48 -1.24 -0.99 -0.44  1330    1
## mu[12]     -0.15    0.01 0.34  -0.82  -0.37 -0.14  0.09  0.49  2204    1
## mu[13]      0.02    0.01 0.33  -0.63  -0.20  0.02  0.23  0.66  2435    1
## mu[14]     -0.01    0.01 0.34  -0.66  -0.24 -0.01  0.21  0.66  2433    1
## mu[15]     -0.48    0.01 0.35  -1.15  -0.70 -0.47 -0.25  0.21  2230    1
## mu_hyp      0.00    0.00 0.22  -0.43  -0.15 -0.01  0.15  0.44  2560    1
## sigma_y     0.67    0.00 0.09   0.52   0.60  0.66  0.72  0.88  1354    1
## sigma_hyp   0.73    0.00 0.17   0.42   0.62  0.72  0.84  1.12  1314    1
## lp__      -10.19    0.07 3.37 -17.74 -12.23 -9.81 -7.83 -4.57  2539    1
```

```
##
## Samples were drawn using NUTS(diag_e) at Tue May 30 11:54:56 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
abun_stan2 <- rvsims(as.matrix(as.data.frame(extract(fit2keep))))

## processing output
core <- as.numeric(ordered(benthic.data$Station))
n.core <- max(core)
zone <- as.numeric(ordered(benthic.data$Area))
n.zone <- max(zone)
oo <- order(core)
ind <- cumsum(table(core[oo]))
Core.zone <- zone[oo][ind]  ## each core belongs to which zone
input_sd <- input.to.stan$y_spd
input_mu <- input.to.stan$y_cen
## return to the original scale
core1.musA <- input_mu + input_sd * abun_stan2[1:15]
zone11.Amu <- mean(core1.musA[Core.zone == 1])
zone12.Amu <- mean(core1.musA[Core.zone == 2])
zone13.Amu <- mean(core1.musA[Core.zone == 3])

deltaA11 = zone12.Amu - zone11.Amu
deltaA12 = zone12.Rmu - zone13.Amu
deltaA13 = zone11.Rmu - zone13.Amu

delta1A <- c(deltaA11, deltaA12, deltaA13)
names(delta1A) <- c("H-I", "H-O", "I-O")
```

The second alternative is to mimic the multilevel model:

$$\begin{aligned} y_i &\sim N(\mu_i, \sigma_y^2) \\ \mu_i &= \mu_0 + \alpha_{k[i]} + \beta_{j[i]} \\ \alpha_k &\sim N(0, \sigma_z^2) \\ \beta_j &\sim N(0, \sigma_c^2) \end{aligned}$$

where $k[i]$ and $j[i]$ represent that the ith observation is in kth zone and jth core.

```
stan3_gom <- "
data{
  int K;  //total sample size
  int J;  //number of sediment cores
  int I;  //number of zones
  real y[K]; //observed response
  int core[K]; //core index
  int zone[K]; //zone index
}
parameters{
  real muK[J];
  real muZ[I];
  real mu0;
  real<lower=0> sigmaY;
  real<lower=0> sigmaK;
  real<lower=0> sigmaZ;
```

```
}
model{
  sigmaK ~ normal(0,0.5);
  sigmaZ ~ normal(0,0.5);
  sigmaY ~ normal(0,0.5);
  for (i in 1:I){
    muZ[i] ~ normal(0, sigmaZ);
  }
  for (j in 1:J){
    muK[j] ~ normal(0, sigmaK);
  }
  for (k in 1:K){
    y[k] ~ normal(mu0+muK[core[k]]+muZ[zone[k]], sigmaY);
  }
}
generated quantities{
  real delta1;
  real delta2;
  real delta3;
  delta1 = muZ[2]-muZ[1];
  delta2 = muZ[2]-muZ[3];
  delta3 = muZ[1]-muZ[3];
}
"

stan.fit3 <- stan_model(model_code = stan3_gom)
```

```
## Trying to compile a simple C file
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## using C compiler: 'gcc (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0'
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/usr/lib/R/site-library/Rcpp/include/"  -I"/usr/lib/R/site
## In file included from /usr/lib/R/site-library/RcppEigen/include/Eigen/Core:88,
##                  from /usr/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp
##                  from <command-line>:
## /usr/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown type nam
##   628 | namespace Eigen {
##       | ^~~~~~~~~
## /usr/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: error: expected '=',
##   628 | namespace Eigen {
##       |                 ^
## In file included from /usr/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp
##                  from <command-line>:
## /usr/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or dir
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:191: foo.o] Error 1
```

```
stan.in5 <- function(data = benthic.data, y.col = 5, chains = nchains) {
    n <- dim(data)[1]
    y <- log(data[, y.col])
    y_ave <- mean(y)
```

```
    y_sd <- sd(y)
    y <- (y - y_ave)/y_sd
    core <- as.numeric(ordered(data$Station))
    n.core <- max(core)
    zone <- as.numeric(ordered(data$Area))
    n.zone <- max(zone)

    stan.dat <- list(K = n, J = n.core, I = n.zone, y = y, core = core, zone = zone)
    inits <- list()
    for (i in 1:chains) inits[[i]] <- list(muK = rnorm(n.core), muZ = rnorm(n.zone),
        mu0 = rnorm(1), sigmaY = runif(1), sigmaK = runif(1), sigmaZ = runif(1))
    parameters <- c("mu0", "muK", "muZ", "sigmaY", "sigmaK", "sigmaZ", "delta1",
        "delta2", "delta3")
    return(list(para = parameters, data = stan.dat, inits = inits, n.chains = chains,
        y_cen = y_ave, y_spd = y_sd))
}

input.to.stan <- stan.in5()
fit2keep <- sampling(stan.fit3, data = input.to.stan$data, init = input.to.stan$inits,
    pars = input.to.stan$para, iter = niters, thin = nthin, chains = input.to.stan$n.chains,
    control = list(adapt_delta = 0.99, max_treedepth = 15))
print(fit2keep)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## mu0     -0.04    0.01 0.43 -0.90 -0.29 -0.04  0.22  0.81  2436    1
## muK[1]  -0.84    0.01 0.38 -1.60 -1.09 -0.84 -0.59 -0.10  2239    1
## muK[2]   0.18    0.01 0.39 -0.61 -0.07  0.19  0.44  0.96  2167    1
## muK[3]   0.11    0.01 0.39 -0.67 -0.14  0.12  0.38  0.87  2194    1
## muK[4]  -0.30    0.01 0.39 -1.06 -0.56 -0.29 -0.02  0.45  2360    1
## muK[5]   0.87    0.01 0.37  0.15  0.63  0.85  1.12  1.65  2406    1
## muK[6]   0.39    0.01 0.38 -0.30  0.13  0.39  0.64  1.14  2441    1
## muK[7]  -0.65    0.01 0.37 -1.38 -0.90 -0.65 -0.41  0.11  2487    1
## muK[8]   1.07    0.01 0.39  0.35  0.80  1.05  1.34  1.87  2098    1
## muK[9]  -0.14    0.01 0.38 -0.86 -0.39 -0.14  0.11  0.63  2037    1
## muK[10]  0.35    0.01 0.38 -0.36  0.10  0.33  0.58  1.16  2100    1
## muK[11]  0.39    0.01 0.38 -0.39  0.14  0.40  0.65  1.13  2461    1
## muK[12]  0.46    0.01 0.37 -0.24  0.21  0.44  0.69  1.23  2025    1
## muK[13]  0.26    0.01 0.38 -0.46  0.01  0.24  0.52  1.02  1979    1
## muK[14] -1.50    0.01 0.41 -2.30 -1.77 -1.48 -1.21 -0.72  2026    1
## muK[15] -0.56    0.01 0.39 -1.30 -0.81 -0.58 -0.30  0.21  2017    1
## muZ[1]  -0.09    0.01 0.42 -1.01 -0.32 -0.08  0.15  0.72  2380    1
## muZ[2]  -0.43    0.01 0.46 -1.42 -0.72 -0.39 -0.11  0.35  2250    1
## muZ[3]   0.51    0.01 0.45 -0.28  0.21  0.48  0.77  1.45  2207    1
## sigmaY   0.40    0.00 0.05  0.31  0.36  0.40  0.44  0.52  2625    1
## sigmaK   0.75    0.00 0.16  0.49  0.64  0.73  0.85  1.10  2210    1
## sigmaZ   0.55    0.01 0.27  0.08  0.36  0.52  0.71  1.15  1838    1
## delta1  -0.34    0.01 0.41 -1.17 -0.62 -0.32 -0.04  0.41  2185    1
## delta2  -0.94    0.01 0.51 -1.89 -1.31 -0.96 -0.60  0.02  1765    1
## delta3  -0.60    0.01 0.44 -1.47 -0.90 -0.60 -0.28  0.15  1924    1
## lp__    11.67    0.08 3.94  2.85  9.32 12.11 14.51 18.21  2284    1
```

```
##
## Samples were drawn using NUTS(diag_e) at Tue May 30 11:55:53 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
rich_stan3 <- rvsims(as.matrix(as.data.frame(extract(fit2keep))))
input_sd <- input.to.stan$y_spd
input_mu <- input.to.stan$y_cen
zone2.musR <- input_sd * rich_stan3[17:19]
core2.musR <- input_mu + input_sd * rich_stan3[2:16]

zone21.Rmu <- mean(core2.musR[Core.zone == 1]) + zone2.musR[1]
zone22.Rmu <- mean(core2.musR[Core.zone == 2]) + zone2.musR[2]
zone23.Rmu <- mean(core2.musR[Core.zone == 3]) + zone2.musR[3]

deltaR21 = zone22.Rmu - zone21.Rmu
deltaR22 = zone22.Rmu - zone23.Rmu
deltaR23 = zone21.Rmu - zone23.Rmu

delta2R <- c(deltaR21, deltaR22, deltaR23)
names(delta2R) <- c("H-I", "H-O", "I-O")

input.to.stan <- stan.in5(y.col = 6)
fit2keep <- sampling(stan.fit3, data = input.to.stan$data, init = input.to.stan$inits,
    pars = input.to.stan$para, iter = niters, thin = nthin, chains = input.to.stan$n.chains,
    control = list(adapt_delta = 0.99, max_treedepth = 15))
```

```
## Warning: There were 9 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
print(fit2keep)
```

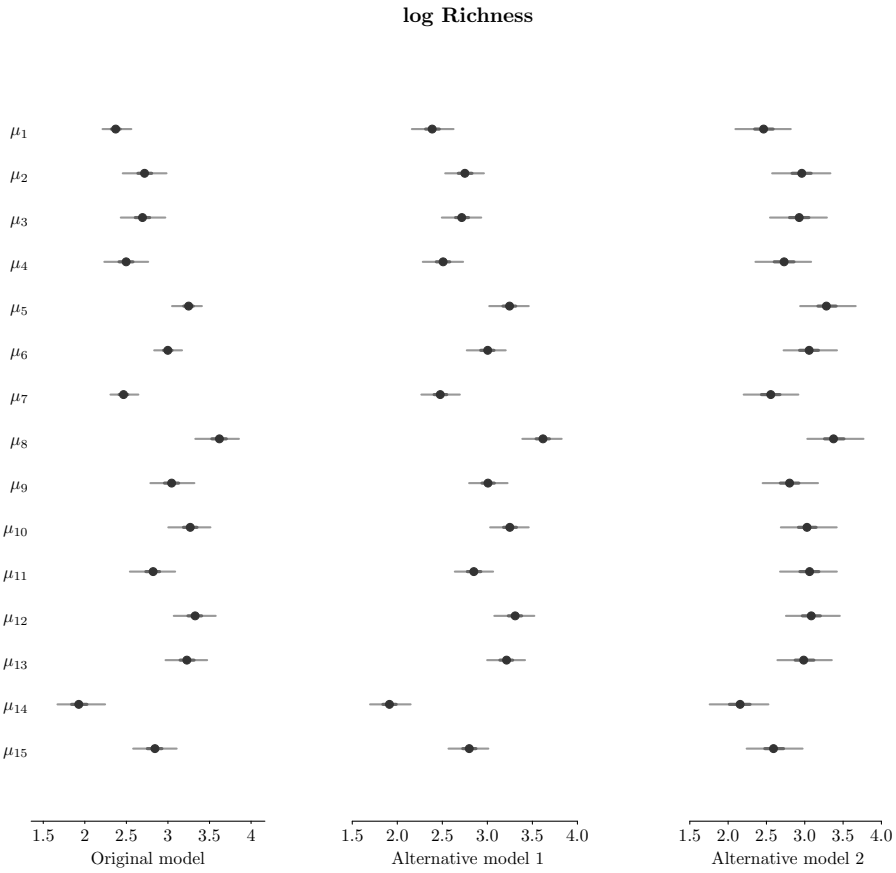
```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##          mean se_mean   sd   2.5%    25%   50%   75% 97.5% n_eff Rhat
## mu0      0.00    0.01 0.35  -0.67  -0.20  0.01  0.21  0.68  2100    1
## muK[1]  -0.25    0.01 0.42  -1.08  -0.52 -0.24  0.03  0.56  2520    1
## muK[2]   0.30    0.01 0.41  -0.52   0.04  0.30  0.57  1.12  2678    1
## muK[3]  -0.32    0.01 0.42  -1.16  -0.60 -0.30 -0.02  0.48  2354    1
## muK[4]  -0.13    0.01 0.40  -0.96  -0.38 -0.13  0.14  0.64  2370    1
## muK[5]   1.09    0.01 0.48   0.17   0.74  1.08  1.43  2.02  2243    1
## muK[6]   0.77    0.01 0.46  -0.12   0.47  0.76  1.08  1.65  2388    1
## muK[7]  -0.55    0.01 0.43  -1.46  -0.83 -0.54 -0.26  0.23  2451    1
## muK[8]   0.76    0.01 0.41  -0.03   0.50  0.75  1.03  1.57  2283    1
## muK[9]   0.00    0.01 0.39  -0.77  -0.25  0.00  0.25  0.77  2737    1
## muK[10] -0.22    0.01 0.39  -0.99  -0.48 -0.21  0.04  0.55  2366    1
## muK[11] -0.98    0.01 0.46  -1.91  -1.28 -0.98 -0.67 -0.12  2337    1
## muK[12] -0.13    0.01 0.39  -0.92  -0.38 -0.12  0.12  0.64  2496    1
## muK[13]  0.02    0.01 0.38  -0.71  -0.24  0.02  0.26  0.76  2373    1
## muK[14]  0.17    0.01 0.41  -0.65  -0.11  0.16  0.44  0.98  2259    1
```

```
## muK[15] -0.44    0.01 0.39  -1.21  -0.69 -0.43 -0.18  0.32  2396    1
## muZ[1]   0.32    0.01 0.39  -0.31   0.05  0.26  0.54  1.25  2176    1
## muZ[2]  -0.28    0.01 0.38  -1.15  -0.51 -0.21 -0.02  0.34  2295    1
## muZ[3]  -0.01    0.01 0.35  -0.75  -0.18  0.00  0.17  0.71  2330    1
## sigmaY   0.67    0.00 0.09   0.52   0.61  0.66  0.72  0.87  2519    1
## sigmaK   0.67    0.00 0.18   0.35   0.54  0.66  0.78  1.05  2176    1
## sigmaZ   0.43    0.01 0.26   0.04   0.24  0.40  0.57  1.03  2206    1
## delta1  -0.60    0.01 0.48  -1.63  -0.93 -0.57 -0.21  0.13  2298    1
## delta2  -0.28    0.01 0.37  -1.06  -0.52 -0.24 -0.01  0.38  2326    1
## delta3   0.33    0.01 0.39  -0.31   0.04  0.28  0.56  1.24  2253    1
## lp__    -8.18    0.09 3.87 -16.39 -10.66 -7.93 -5.46 -1.61  1834    1
##
## Samples were drawn using NUTS(diag_e) at Tue May 30 11:55:59 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

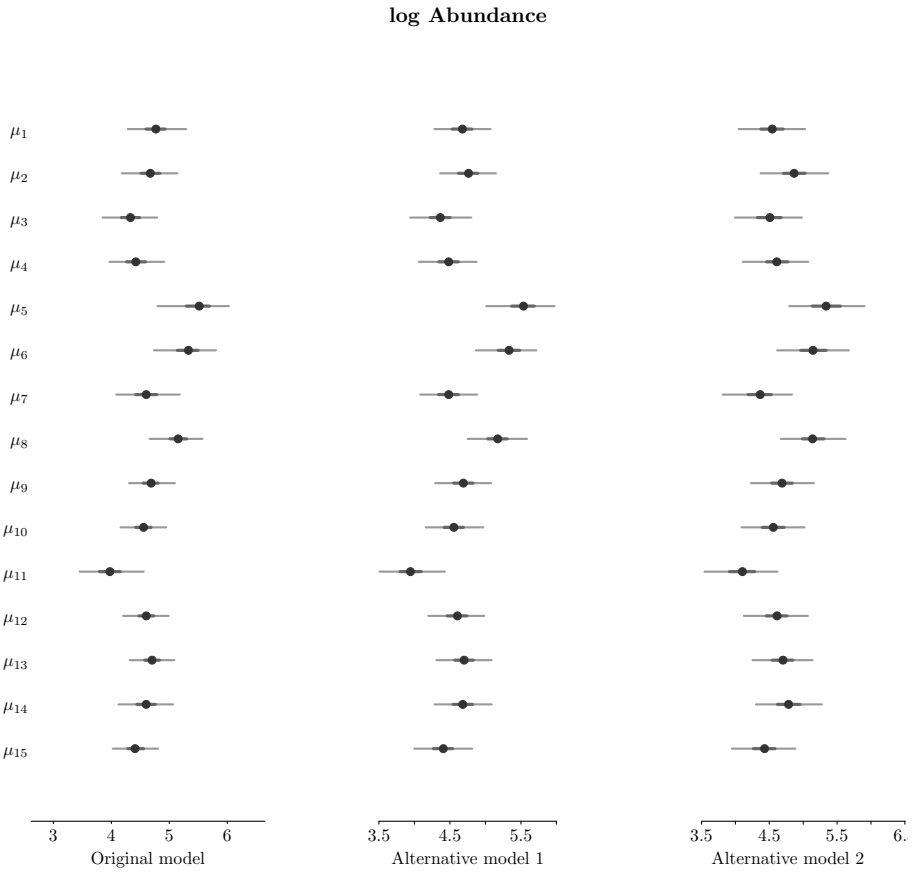
```
abun_stan3 <- rvsims(as.matrix(as.data.frame(extract(fit2keep))))

input_sd <- input.to.stan$y_spd
input_mu <- input.to.stan$y_cen
zone2.musA <- input_sd * abun_stan3[17:19]
core2.musA <- input_mu + input_sd * abun_stan3[2:16]

zone21.Amu <- mean(core2.musA[Core.zone == 1]) + zone2.musA[1]
zone22.Amu <- mean(core2.musA[Core.zone == 2]) + zone2.musA[2]
zone23.Amu <- mean(core2.musA[Core.zone == 3]) + zone2.musA[3]

deltaA21 = zone22.Amu - zone21.Amu
deltaA22 = zone22.Amu - zone23.Amu
deltaA23 = zone21.Amu - zone23.Amu

delta2A <- c(deltaA21, deltaA22, deltaA23)
names(delta2A) <- c("H-I", "H-O", "I-O")
```

Now we make some comparisons of the three alternative models

```
## 1. Estimated core means
par(mfrow = c(1, 3), mar = c(3, 1, 2, 0.1), mgp = c(1.25, 0.125, 0), las = 1, tck = -0.01)
mlplot(tempR, xlab = "Original model", top.axis = F)
mlplot(core1.musR, xlab = "Alternative model 1", main = "log Richness", axes = F)
axis(1)
mlplot(core2.musR, xlab = "Alternative model 2", axes = F)
axis(1)
```

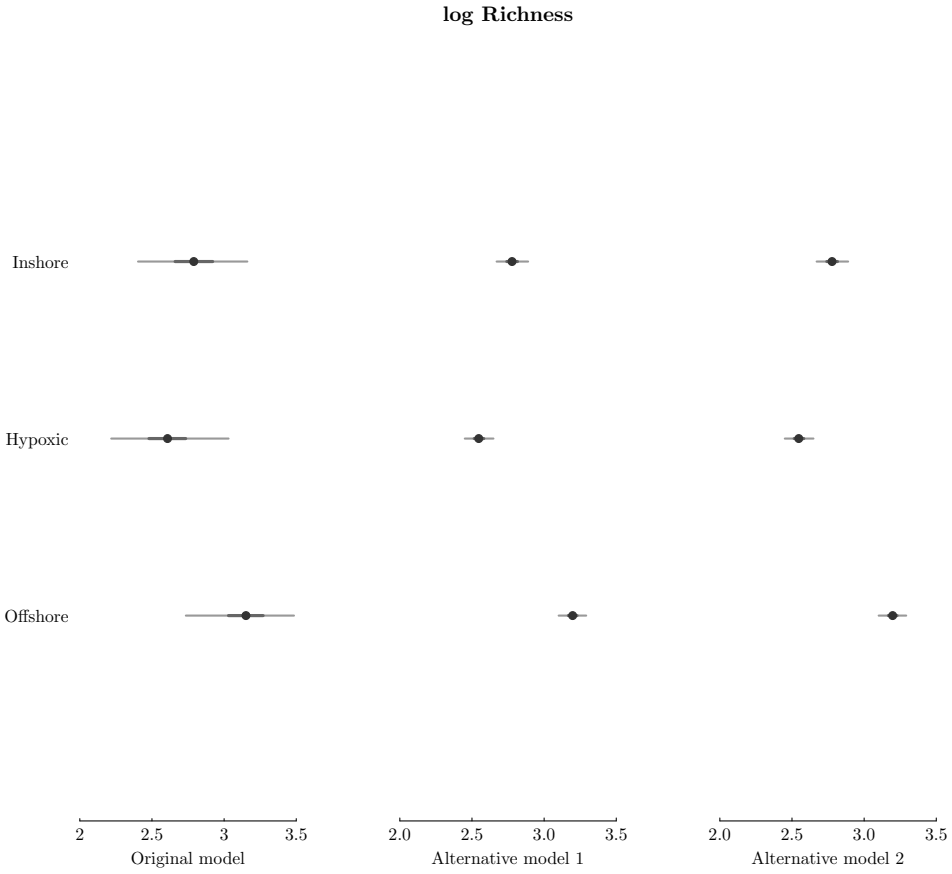



```
par(mfrow = c(1, 3), mar = c(3, 1, 2, 0.1), mgp = c(1.25, 0.125, 0), las = 1, tck = -0.01)
mlplot(tempA, xlab = "Original model", top.axis = F)
mlplot(core1.musA, xlab = "Alternative model 1", main = "log Abundance", axes = F)
axis(1)
mlplot(core2.musA, xlab = "Alternative model 2", axes = F)
axis(1)
```

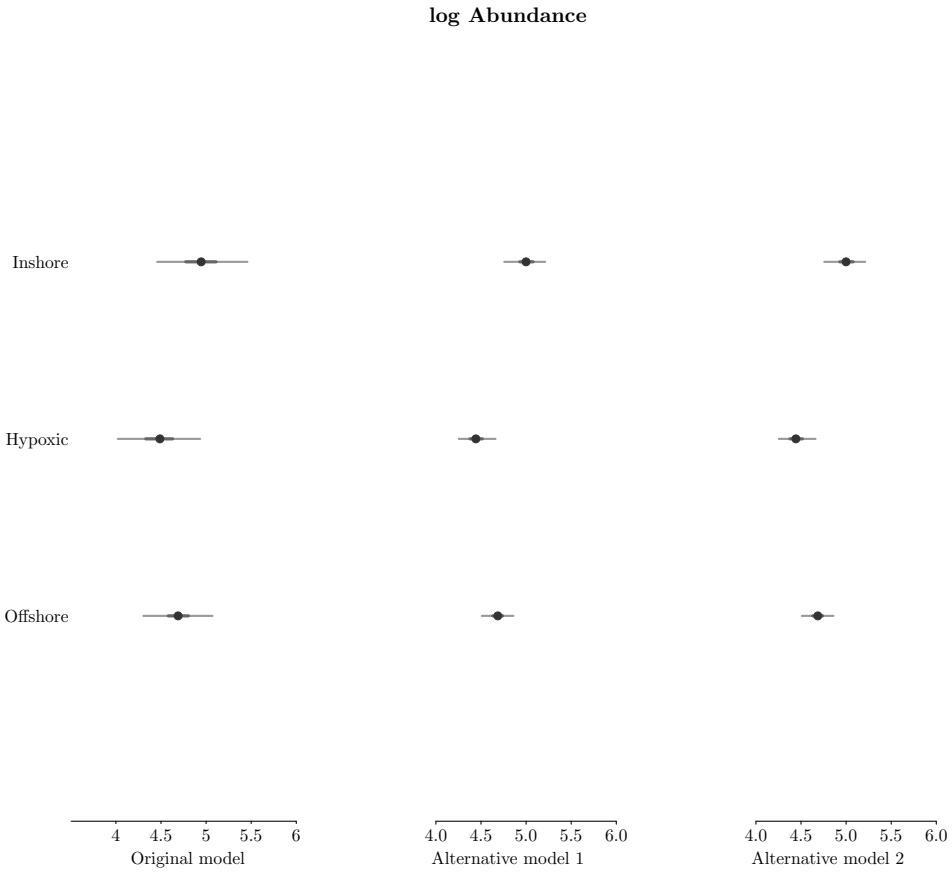



```
## 2. Estimated zone means extract zone means and mean differences
zone0R.thetas <- rich_stan[16:18]
names(zone0R.thetas) <- c("Inshore", "Hypoxic", "Offshore")
zone0A.thetas <- abun_stan[16:18]
names(zone0A.thetas) <- c("Inshore", "Hypoxic", "Offshore")

zone1R.thetas <- c(zone11.Rmu, zone12.Rmu, zone13.Rmu)
names(zone1R.thetas) <- c("Inshore", "Hypoxic", "Offshore")

zone1A.thetas <- c(zone11.Amu, zone12.Amu, zone13.Amu)
names(zone1A.thetas) <- c("Inshore", "Hypoxic", "Offshore")

zone2R.thetas <- c(zone21.Rmu, zone22.Rmu, zone23.Rmu)
names(zone2R.thetas) <- c("Inshore", "Hypoxic", "Offshore")

zone2A.thetas <- c(zone21.Amu, zone22.Amu, zone23.Amu)
names(zone2A.thetas) <- c("Inshore", "Hypoxic", "Offshore")

par(mfrow = c(1, 3), mar = c(3, 1, 2, 0.1), mgp = c(1.25, 0.125, 0), las = 1, tck = 0.01)
mlplot(zone0R.thetas, xlab = "Original model", top.axis = F, xlim = c(2, 3.5))
abline(v = 0)
mlplot(zone1R.thetas, xlab = "Alternative model 1", main = "log Richness", axes = F,
    xlim = c(2, 3.5))
axis(1)
abline(v = 0)
mlplot(zone1R.thetas, xlab = "Alternative model 2", axes = F, xlim = c(2, 3.5))
```

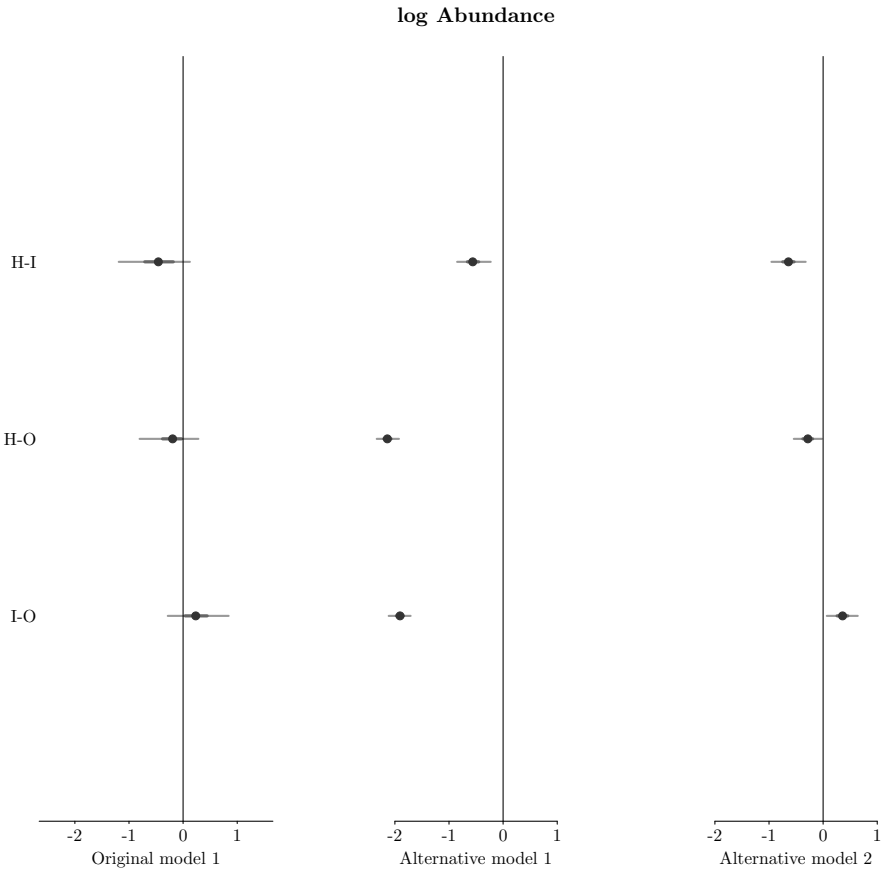
```
axis(1)
abline(v = 0)
```



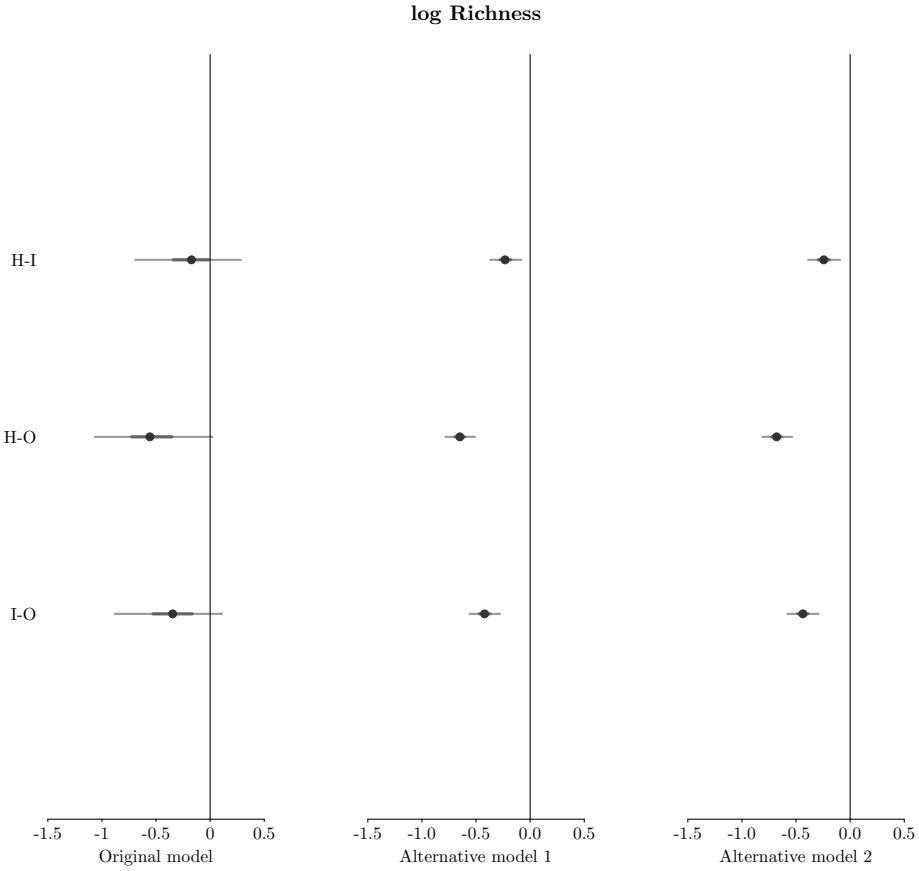

```
par(mfrow = c(1, 3), mar = c(3, 1, 2, 0.1), mgp = c(1.25, 0.125, 0), las = 1, tck = -0.01)
mlplot(zone0A.thetas, xlab = "Original model", top.axis = F, xlim = c(3.6, 6))
mlplot(zone1A.thetas, xlab = "Alternative model 1", main = "log Abundance", axes = F,
    xlim = c(3.6, 6))
axis(1)
mlplot(zone1A.thetas, xlab = "Alternative model 2", axes = F, xlim = c(3.6, 6))
axis(1)
```




```
## between zone differences
par(mfrow = c(1, 3), mar = c(3, 1, 2, 0.1), mgp = c(1.25, 0.125, 0), las = 1, tck = -0.01)
mlplot(delta0A, xlab = "Original model 1", top.axis = F, xlim = c(-2.5, 1.5))
abline(v = 0)
mlplot(delta1A, xlab = "Alternative model 1", main = "log Abundance", axes = F, xlim = c(-2.5,
    1.5))
axis(1)
abline(v = 0)
mlplot(delta2A, xlab = "Alternative model 2", axes = F, xlim = c(-2.5, 1.5))
axis(1)
abline(v = 0)
```




```
par(mfrow = c(1, 3), mar = c(3, 1, 2, 0.1), mgp = c(1.25, 0.125, 0), las = 1, tck = -0.01)
mlplot(delta0R, xlab = "Original model", top.axis = F, xlim = c(-1.5, 0.5))
abline(v = 0)
mlplot(delta1R, xlab = "Alternative model 1", main = "log Richness", axes = F, xlim = c(-1.5,
    0.5))
abline(v = 0)
axis(1)
mlplot(delta2R, xlab = "Alternative model 2", axes = F, xlim = c(-1.5, 0.5))
axis(1)
abline(v = 0)
```


log Richness
H-I
H-O
I-O
-1.5
-1
-0.5
0
0.5
Original model
-1.5
-1.0
-0.5
0.0
0.5
Alternative model 1
-1.5
-1.0
-0.5
0.0
0.5
Alternative model 2