Appendix S1. Linear regression models and non-linear population dynamics

Jelena H. Pantel[*] Ruben J. Hermann[†]

11 September, 2024

1 One species, logistic growth

Population growth over time in a single species is first modelled using a Beverton-Holt (discrete-time, logistic) model (Beverton and Holt (1957)), using an intra-specific competition coefficient for density-dependent growth (Hart and Marshall (2013)) - thus $\alpha = 1/K$.

$$N_{i,t+1} = \frac{r_i N_{i,t}}{1 + \alpha_{ii} N_{i,t}}$$

Note that in this model, the system is at equilibrium when $N_{i,t+1} = N_{i,t}$, and therefore:

$$N^* = N^* \frac{r_i}{1 + \alpha_{ii} N^*}$$

$$1 = \frac{r_i}{1 + \alpha_{ii} N^*}$$

$$N^* = \frac{r_i - 1}{\alpha_{ii}}$$

1.1 Population dynamics simulation

In the metacommunity simulation in the main text, a species resides in a site with an initial population size $N_{i,0} \sim Pois(10)$, a growth rate r_i that depends on the local environmental value E_k and the species trait x_i, and a fixed intra-specific competition coefficient of $\alpha_{ii} = 0.00125$. We simulate population growth here:

```r
set.seed(42)
# Simulate initial species population growth
N1.0 <- rpois(1, 10)
r1.0 <- 1.67
alpha.11 <- 0.00125
# model function
disc_log <- function(r, N0, alpha) {
    Nt1 <- (r * N0)/(1 + alpha * N0)
```

[*]Laboratoire Chrono-environnement,UMR 6249 CNRS-UFC, 16 Route de Gray, 25030 Besançon cedex, France, jelena.pantel@univ-fcomte.fr

[†]University of Duisburg-Essen, Universitätsstraße 5, 45141 Essen, Germany, ruben.hermann@uni-due.de

```r
    return(Nt1)
}
# Simulation of model for t time steps
t <- 30
N <- rep(NA, t)
N[1] <- N1.0
for (i in 2:t) {
    N[i] <- disc_log(r = r1.0, N0 = N[i - 1], alpha = alpha.11)
}
```

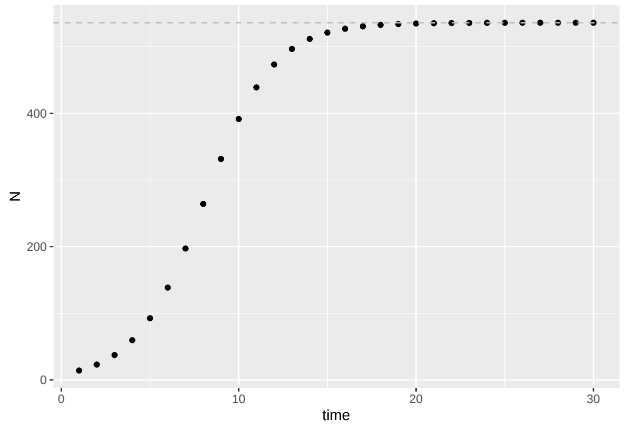


Figure 1: Population size N over time t for a discrete-time logistic growth model, with parameters $r_i = 1.67$, $N_{1,0} = 14$, and $\alpha_{11} = 0.00125$.

1.2 Linear statistical model

We fit the population time series data to a first-order auto-regressive model to predict N_{t+1} as a function of N_t, and compare that to a linear regression. We use ln-N after Ives (1995), as also discussed in Certain et al. (2018) and Olivença et al. (2021).

$$N_{t+1} = \beta_0 + \beta_1 N_t + \epsilon_t$$

```r
# Fit the model
m.1.ar <- arima(x = log(N), order = c(1, 0, 0), include.mean = T, method = "CSS")
m.1.lm <- lm(log(dat$N[2:t]) ~ log(dat$N[1:(t - 1)]))
# plotting the series along with the fitted values
m.1.ar.fit <- log(N) - residuals(m.1.ar)
m.1.lm.fit <- log(dat$N[2:t]) - m.1.lm$resid
dat$ar1.fit <- m.1.ar.fit
dat$lm.fit <- NA
dat$lm.fit[2:t] <- m.1.lm.fit
```

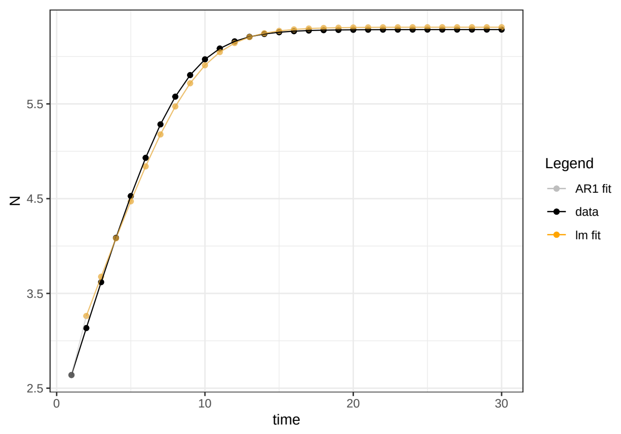


Figure 2: Population size over time (black line) with fitted values from a first-order autoregressive model (red dashed line).

18 The linear model is a good fit, and N_{t+1} and N_t are well-represented by a linear function:

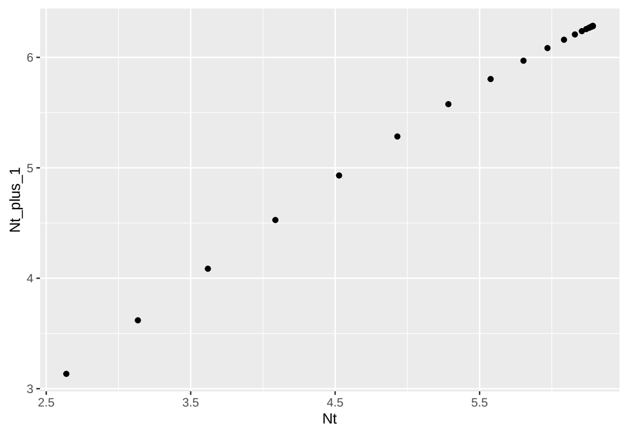


Figure 3: Population size (logarithm) at one time step N_{t+1} as a function of log-population size in the previous time step N_t.

19 We also examine density dependence by plotting $\Delta N = N_{t+1} - N_t$ vs. N_t:

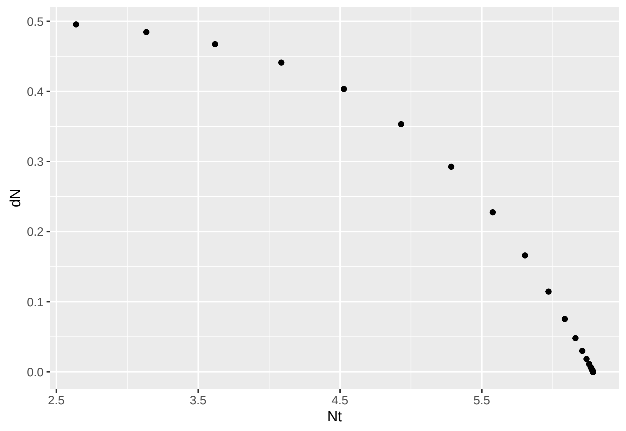


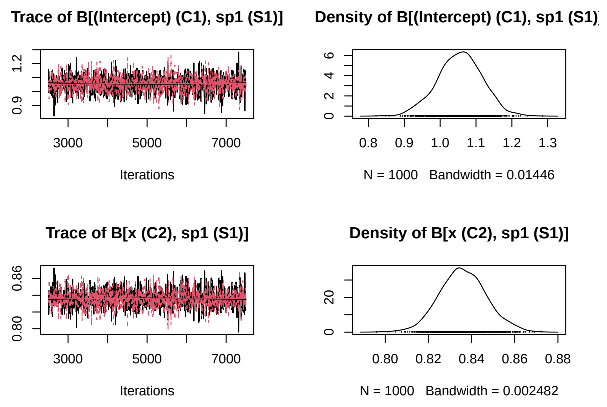
Figure 4: Change in population size from one time step to the next N_{t+1} as a function of N_{t+1}

1.3 Bayesian linear statistical model: HMSC

We can estimate the same model parameters using HMSC:

```
# prepare data in HMSC format
Y <- as.matrix(log(dat$N[2:t]))
XData <- data.frame(x = log(dat$N[1:(t - 1)]))
m.1.hmsc <- Hmsc(Y = Y, XData = XData, XFormula = ~x)
# Bayesian model parameters
nChains <- 2
thin <- 5
samples <- 1000
transient <- 500 * thin
verbose <- 500 * thin
# sample MCMC
m.1.sample <- sampleMcmc(m.1.hmsc, thin = thin, sample = samples, transient = transient,
    nChains = nChains, verbose = verbose)
#> setting updater$GammaEta=FALSE due to absence of random effects included to the model
#> Computing chain 1
#> Chain 1, iteration 2500 of 7500 (transient)
#> Chain 1, iteration 5000 of 7500 (sampling)
#> Chain 1, iteration 7500 of 7500 (sampling)
#> Computing chain 2
#> Chain 2, iteration 2500 of 7500 (transient)
#> Chain 2, iteration 5000 of 7500 (sampling)
#> Chain 2, iteration 7500 of 7500 (sampling)
```

```
m.post.hmsc <- convertToCodaObject(m.1.sample)
summary(m.post.hmsc$Beta)
#>
#> Iterations = 2505:7500
#> Thinning interval = 5
#> Number of chains = 2
#> Sample size per chain = 1000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#> 
#>                                     Mean      SD  Naive SE Time-series SE
#> B[(Intercept) (C1), sp1 (S1)] 1.0550 0.06240 0.0013952      0.0013954
#> B[x (C2), sp1 (S1)]           0.8361 0.01083 0.0002422      0.0002415
#> 
#> 2. Quantiles for each variable:
#> 
#>                                 2.5%    25%    50%    75%  97.5%
#> B[(Intercept) (C1), sp1 (S1)] 0.9288 1.0132 1.0564 1.0974 1.1693
#> B[x (C2), sp1 (S1)]           0.8154 0.8287 0.8358 0.8431 0.8582
plot(m.post.hmsc$Beta)
```



These estimates match well with those from the AR1 and linear model:

```
# AR1 coefficients (recall that the intercept is the term below multipled by 1
# - phi1)
m.1.ar$coef
#>       ar1 intercept
#> 0.8359839 6.4371388
m.1.ar$coef[2] * (1 - m.1.ar$coef[1])
#> intercept
#>  1.055794
# linear model
summary(m.1.lm)$coefficients[1:2, 1:2]
#>                         Estimate  Std. Error
#> (Intercept)            1.0557944 0.054425861
#> log(dat$N[1:(t - 1)]) 0.8359839 0.009441702
# Bayesian estimates
summary(m.post.hmsc$Beta)$statistics[1:2, 1:2]
#>                                    Mean         SD
#> B[(Intercept) (C1), sp1 (S1)] 1.0549596 0.06239605
#> B[x (C2), sp1 (S1)]           0.8360545 0.01082936
```

```
Gradient <- constructGradient(m.1.sample, focalVariable = "x", ngrid = 29)
predY <- predict(m.1.sample, Gradient = Gradient, expected = TRUE)
# preds <-computePredictedValues(m.1.sample)
plotGradient(m.1.sample, Gradient, pred = predY, showData = T, measure = "Y", main = "",
    xlab = "N_t", ylab = "predicted N_t+1")
```

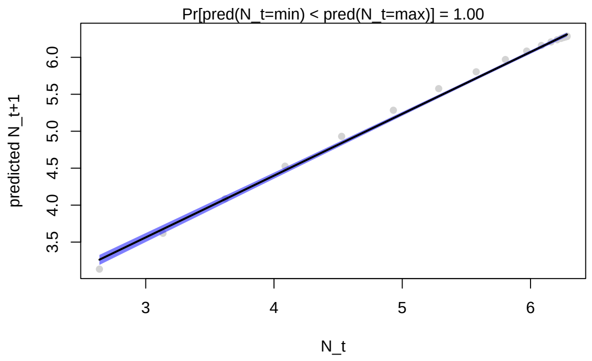
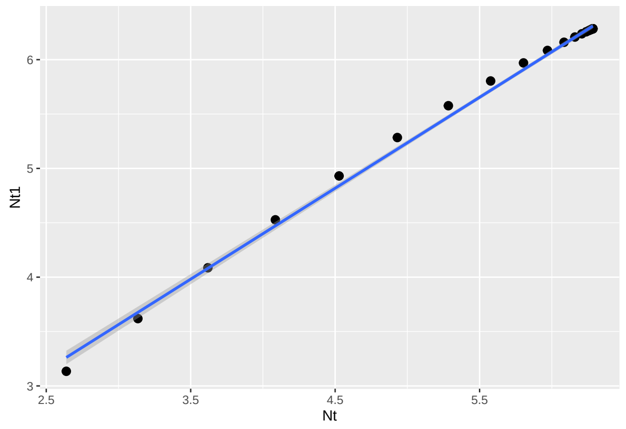


Figure 5: Observed (grey) and model-fit (blue) values for population size at time t (x-axis) and t+1 (y-axis).

```
lm_dat <- data.frame(cbind(log(dat$N[2:t]), log(dat$N[1:(t - 1)])))
colnames(lm_dat) <- c("Nt1", "Nt")
ggplot(lm_dat, aes(Nt, Nt1)) + stat_summary(fun.data = mean_cl_normal) + geom_smooth(method = "lm")
#> `geom_smooth()` using formula = 'y ~ x'
#> Warning: Removed 29 rows containing missing values (`geom_segment()`).
```



1.4 Conclusions

In this example, a first-order auto-regressive model works well, bypassing the need to estimate logistic growth parameters r_i and α_{ii}. The density-dependence dynamics ($\Delta N \sim f(N_t)$) show an overall declining trend over time. The Bayesian estimation implemented in HMSC gives good parameter estimates.

2 One species, logistic growth, environmental covariate

We now consider using a linear model to analyze population growth when the species growth rate is impacted by a single environmental covariate.

2.1 Growth depends on environment

First we add environment-dependent growth rate. The growth rate r_i becomes:

$$r_i = \hat{W}e^{-(E-x_{i,t})^2}$$

Here, $\hat{W}$ is the maximal population growth rate (set to 1.67 as above), E is the local environmental trait optimum value, and $x_{i,t}$ is species i trait value at time t. We see that if $E = x_{i,t}$ then the growth rate is at the value $r = 1.67$. Here, we begin with $E = x_{i,t} = 0.8$, then simulate the environment E value fluctuating randomly over time, and finally use a linear model to fit E as a covariate.

```r
# Simulate initial species population growth with environment fluctuations
N1.0 <- 10
r1.0 <- 1.67
alpha.11 <- 0.00125
E.0 <- 0.8
x1.0 <- 0.8
# model function
disc_log_E <- function(r, N0, alpha, E, x) {
    Nt1 <- ((r * exp(-(E - x)^2)) * N0)/(1 + alpha * N0)
    return(Nt1)
}
# Simulation of model for t time steps
t <- 40
N <- rep(NA, t)
N[1] <- N1.0
E <- rep(NA, t)
E[1] <- E.0
for (i in 2:t) {
    N[i] <- disc_log_E(r = r1.0, N0 = N[i - 1], alpha = alpha.11, E = E[i - 1], x = x1.0)
    E[i] <- E[i - 1] + rnorm(1, 0, 0.1)
}
```

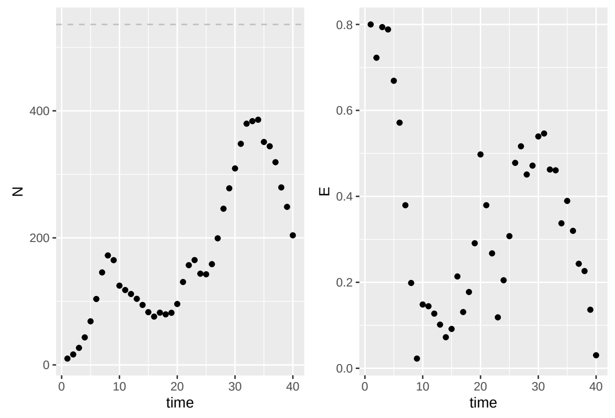


Figure 6: Population size N over time t for a discrete-time logistic growth model, with parameters $r_i = 1.67$, $N_{1,0} = 14$, and $\alpha_{11} = 0.00125$. Relationship between E and Nt is also shown.

2.2 Linear statistical model with environmental covariate

We now include environment E as a covariate in the linear model:

$$N_t = \beta_0 + \beta_1 N_{t-1} + \beta_2 E_{t-1} + \epsilon_t$$

```
# Fit the model
m.2.ar <- arima(x = log(N), order = c(1, 0, 0), include.mean = T, method = "CSS",
    xreg = E)
m.2.lm <- lm(log(dat$N[2:t]) ~ log(dat$N[1:(t - 1)]) + log(E[1:(t - 1)]))
# plotting the series along with the fitted values
m.2.ar.fit <- log(N) - residuals(m.2.ar)
m.2.lm.fit <- log(dat$N[2:t]) - m.2.lm$resid
dat$ar2.fit <- m.2.ar.fit
dat$lm2.fit <- NA
dat$lm2.fit[2:t] <- m.2.lm.fit
```

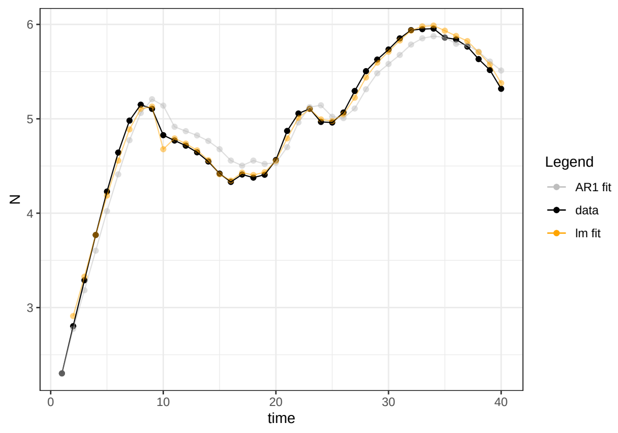


Figure 7: Population size over time (black line) with fitted values from a first-order autoregressive model (red dashed line).

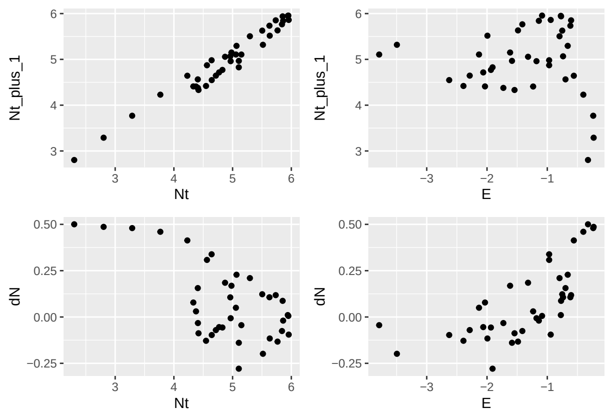


Figure 8: Population size (logarithm) at one time step N_{t+1} as a function of log-population size in the previous time step N_t.

The linear model is a good fit when including the environmental covariate. N_{t+1} and N_t can still be captured by a linear relationship. However we see that the relationship between N_{t+1} and E_t is non-linear. This tells us that the lm is good for predictions, but not for inference (for capturing well the relatiosnip between the predictor and response variable). The use of linear relationships in JSDMs is discussed in (Ingram et al. 2020), and in many applications (e.g. (Erickson and Smith 2023)) quadratic terms are used, which create bell-shaped response curves that may better match species with optimal niches (as opposed to linear, monotonically increasing relationships between population size and environmental predictors). We thus include a quadratic term for E_t to provide a better fit to the data.

```
df <- data.frame(cbind(log(dat$N[2:t]), log(dat$N[1:(t - 1)]), E[1:(t - 1)], E[1:(t -
    1)]^2))
colnames(df) <- c("Nt1", "Nt", "E", "Esq")
m.2.lm <- lm(Nt1 ~ Nt + E + Esq, data = df)
```

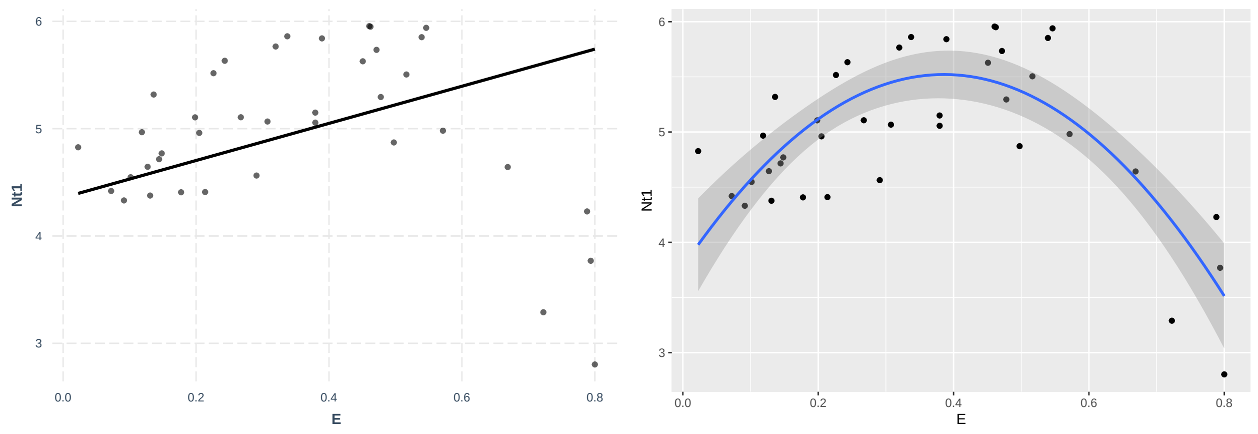


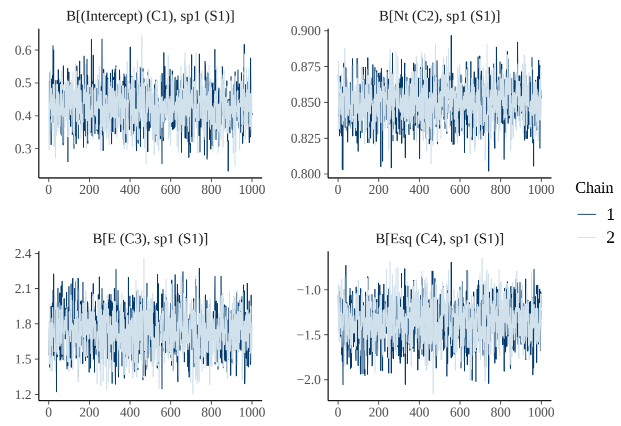
Figure 9: Population size (logarithm) at one time step N_{t+1} as a function of log-population size in the previous time step N_t.

2.3 Bayesian linear statistical model: HMSC

We can estimate the same model parameters using HMSC:

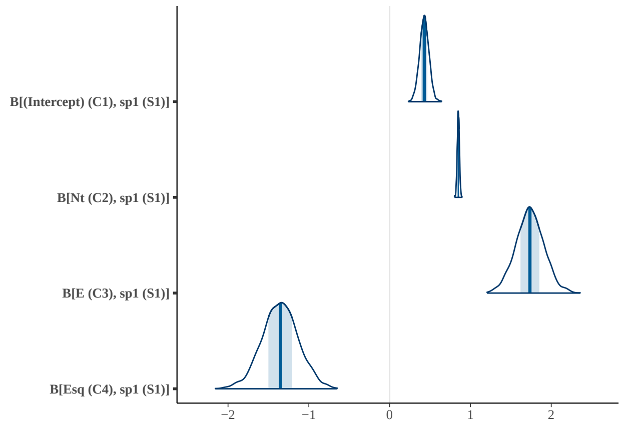
```
# prepare data in HMSC format
Y <- as.matrix(log(dat$N[2:t]))
XData <- df
m.2.hmsc <- Hmsc(Y = Y, XData = XData, XFormula = ~Nt + E + Esq)
# Bayesian model parameters
nChains <- 2
thin <- 5
samples <- 1000
transient <- 500 * thin
verbose <- 500 * thin
# sample MCMC
m.2.sample <- sampleMcmc(m.2.hmsc, thin = thin, sample = samples, transient = transient,
    nChains = nChains, verbose = verbose)
#> setting updater$GammaEta=FALSE due to absence of random effects included to the model
#> Computing chain 1
#> Chain 1, iteration 2500 of 7500 (transient)
#> Chain 1, iteration 5000 of 7500 (sampling)
#> Chain 1, iteration 7500 of 7500 (sampling)
#> Computing chain 2
#> Chain 2, iteration 2500 of 7500 (transient)
#> Chain 2, iteration 5000 of 7500 (sampling)
#> Chain 2, iteration 7500 of 7500 (sampling)
```

```
m2.post.hmsc <- convertToCodaObject(m.2.sample)
summary(m2.post.hmsc$Beta)
#>
#> Iterations = 2505:7500
#> Thinning interval = 5
#> Number of chains = 2
#> Sample size per chain = 1000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>                                   Mean      SD  Naive SE Time-series SE
#> B[(Intercept) (C1), sp1 (S1)]  0.4287 0.05840 0.0013059      0.0013051
#> B[Nt (C2), sp1 (S1)]           0.8500 0.01357 0.0003034      0.0003136
#> B[E (C3), sp1 (S1)]            1.7365 0.17407 0.0038922      0.0039946
#> B[Esq (C4), sp1 (S1)]         -1.3544 0.22541 0.0050403      0.0051805
#>
#> 2. Quantiles for each variable:
#>
#>                                   2.5%     25%     50%    75%   97.5%
#> B[(Intercept) (C1), sp1 (S1)]  0.3125  0.3902  0.4285  0.467  0.5435
#> B[Nt (C2), sp1 (S1)]           0.8235  0.8413  0.8500  0.859  0.8768
#> B[E (C3), sp1 (S1)]            1.3967  1.6194  1.7361  1.853  2.0816
#> B[Esq (C4), sp1 (S1)]         -1.8157 -1.5001 -1.3518 -1.207 -0.9151
bayesplot::mcmc_trace(m2.post.hmsc$Beta)
```



50

```
bayesplot::mcmc_areas(m2.post.hmsc$Beta, area_method = c("equal height"))
```



These estimates match well with those from the AR1 and linear model:

```
# AR1 coefficients (recall that the intercept is the term below multipled by 1
# - phi1)
m.2.ar$coef
#>        ar1  intercept          E
#>  0.8471120  5.4224546 -0.1096173
m.2.ar$coef[2] * (1 - m.2.ar$coef[1])
#> intercept
#>  0.829028
# linear model
summary(m.2.lm)$coefficients[1:4, 1:2]
#>               Estimate Std. Error
#> (Intercept)  0.4286437 0.04917405
#> Nt           0.8502417 0.01151860
#> E            1.7296394 0.14595151
#> Esq         -1.3462240 0.18902070
# Bayesian estimates
summary(m2.post.hmsc$Beta)$statistics[1:4, 1:2]
#>                                    Mean         SD
#> B[(Intercept) (C1), sp1 (S1)]  0.4287409 0.05840051
#> B[Nt (C2), sp1 (S1)]           0.8500472 0.01356690
#> B[E (C3), sp1 (S1)]            1.7365191 0.17406609
#> B[Esq (C4), sp1 (S1)]         -1.3543664 0.22541119
```

We recall that the interpretation of the coefficients in an arimaX (arima with covariates) model is difficult. They do not give the impact on N_t per unit increase in X as in a regression. So we do not interpret the causation implied by the coefficient in the arimaX model. In the regression model, we can see that E has a positive impact on N_t.

```
Gradient <- constructGradient(m.2.sample, focalVariable = "E", non.focalVariables = list(Nt = list(2),
    Esq = list(2)), ngrid = 39)
# Esq is manually constructed as gradient-produced E^2
Gradient$XDataNew$Esq <- Gradient$XDataNew$E^2
predY <- predict(m.2.sample, XData = Gradient$XDataNew, expected = TRUE)
plotGradient(m.2.sample, Gradient, pred = predY, showData = T, measure = "Y", main = "",
    xlab = "E_t", ylab = "predicted N_t+1")
```

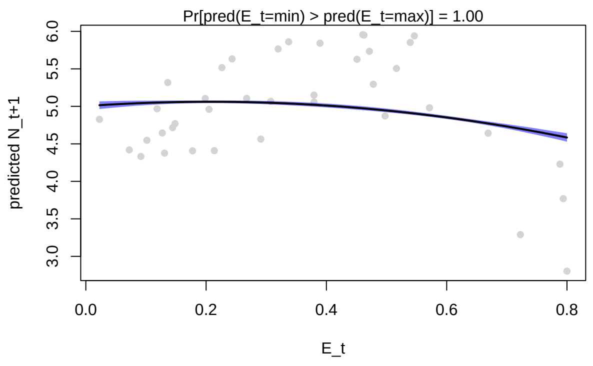


Figure 10: Observed (grey) and model-fit (blue) values for population size at time t (x-axis) and t+1 (y-axis).

2.4 Conclusions

In this example, the linear regression again works well to describe the impact of E_t for N_t when using the quadratic formulation. The arimaX model works well for fitting and subsequent prediction, but less well for inference about the impacts of E. From the quadratic regression terms for E, we correctly see that the population size is maximal at the species trait value and decreases away from that value. We will continue to use log-transformed abundance and now introduce quadratic terms for the environmental parameter.

3 Two species, logistic growth, competition

Here we investigate how a second species impacts the inference we can make from linear models.

3.1 Interspecific competition

The growth equation for each species now becomes:

$$N_{i,t+1} = \frac{r_i N_{i,t}}{1 + \alpha_{ii} N_{i,t} + \alpha_{ij} N_{j,t}}$$

We use a distinct growth rate for the 2nd species and introduce the interspecific interaction coefficient α_{ij}.

```
# Simulate initial species population growth with a second species present
N1.0 <- 10
N2.0 <- 10
r1.0 <- 1.67
r2.0 <- 1.7
alpha.11 <- 0.00125
alpha.22 <- 0.00125
alpha.12 <- 0.008
alpha.21 <- 0.008725
# model function
disc_LV_comp <- function(r1, r2, N1.0, N2.0, alpha.11, alpha.22, alpha.12, alpha.21) {
    Nt1 <- (r1 * N1.0)/(1 + alpha.11 * N1.0 + alpha.12 * N2.0)
    Nt2 <- (r2 * N2.0)/(1 + alpha.22 * N2.0 + alpha.21 * N1.0)
    return(c(Nt1, Nt2))
}
# Simulation of model for t time steps
t <- 40
N <- array(NA, dim = c(t, 2))
N <- as.data.frame(N)
colnames(N) <- c("N1", "N2")
N$N1[1] <- N1.0
N$N2[1] <- N2.0
for (i in 2:t) {
    res <- disc_LV_comp(r1 = r1.0, r2 = r2.0, N1.0 = N[i - 1, 1], N2.0 = N[i - 1,
        2], alpha.11 = alpha.11, alpha.22 = alpha.22, alpha.12 = alpha.12, alpha.21 = alpha.21)
    N$N1[i] <- res[1]
    N$N2[i] <- res[2]
}
```

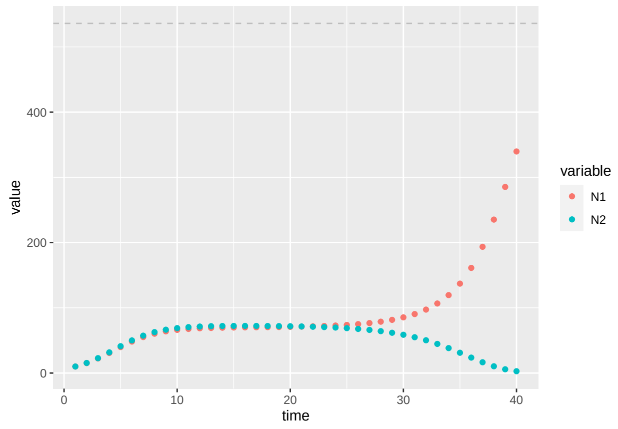


Figure 11: Population size N over time t for a discrete-time logistic growth model, with parameters $r_1 = 1.67$, $r_2 = 1.7$, $N_{i,0} = 10$, $\alpha_{ii} = 0.00125$, $\alpha_{12} = 0.008$, and $\alpha_{21} = 0.008725$.

3.2 Linear statistical model with covariate for both species

We fit each species' population time series to an arimaX and lm with population size of the others species as a covariate.

$$N_{1,t} = \beta_0 + \beta_1 N_{1,t-1} + \beta_2 N_{2,t-1} + \epsilon_t$$

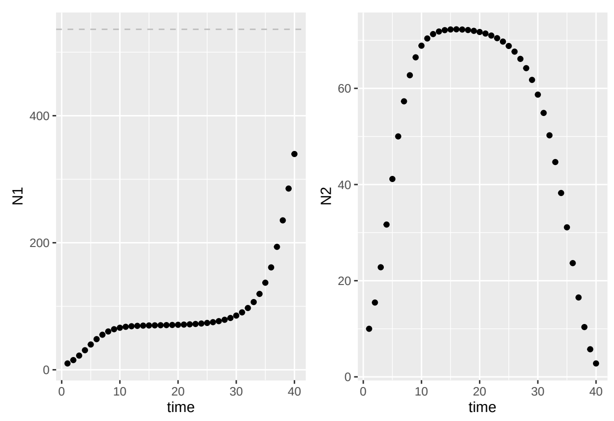


Figure 12: Population size N over time t for a discrete-time logistic growth model, with competition.

```r
## Fit the model for Species 1
m.3.ar.n1 <- arima(x = log(N$N1), order = c(1, 0, 0), include.mean = T, method = "CSS",
    xreg = N$N2)
m.3.lm.n1 <- lm(log(dat$N1[2:t]) ~ log(dat$N1[1:(t - 1)]) + log(dat$N2[1:(t - 1)]))
# plotting the series along with the fitted values
m.3.ar.fit.n1 <- log(N$N1) - residuals(m.3.ar.n1)
m.3.lm.fit.n1 <- log(dat$N1[2:t]) - m.3.lm.n1$resid
dat$ar3.fit.n1 <- m.3.ar.fit.n1
dat$lm3.fit.n1 <- NA
dat$lm3.fit.n1[2:t] <- m.3.lm.fit.n1
## Species 2
m.3.ar.n2 <- arima(x = log(N$N2), order = c(1, 0, 0), include.mean = T, method = "CSS",
    xreg = N$N1)
m.3.lm.n2 <- lm(log(dat$N2[2:t]) ~ log(dat$N2[1:(t - 1)]) + log(dat$N1[1:(t - 1)]))
# plotting the series along with the fitted values
m.3.ar.fit.n2 <- log(N$N2) - residuals(m.3.ar.n2)
m.3.lm.fit.n2 <- log(dat$N2[2:t]) - m.3.lm.n2$resid
dat$ar3.fit.n2 <- m.3.ar.fit.n2
dat$lm3.fit.n2 <- NA
dat$lm3.fit.n2[2:t] <- m.3.lm.fit.n2
```

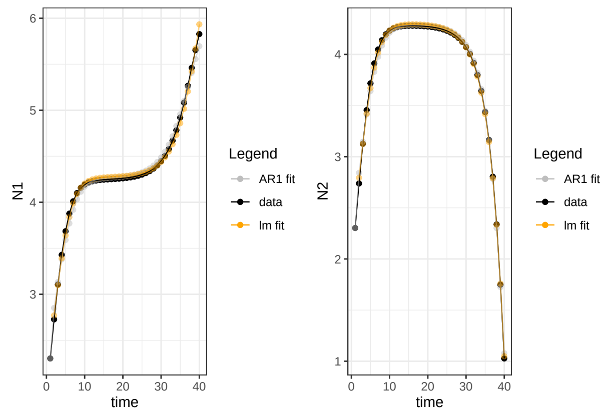


Figure 13: Population size over time (black line) with fitted values from a first-order autoregressive model (gray line) and from a linear model (orange line).

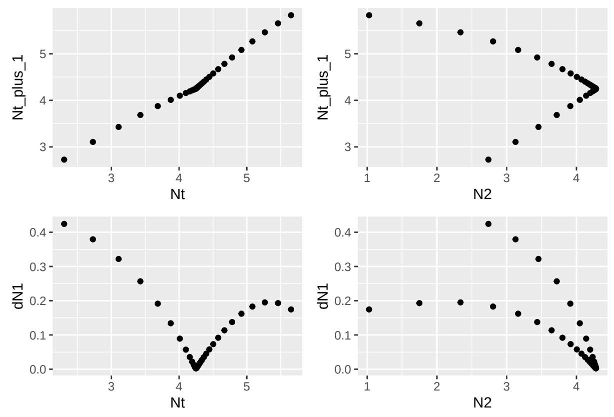
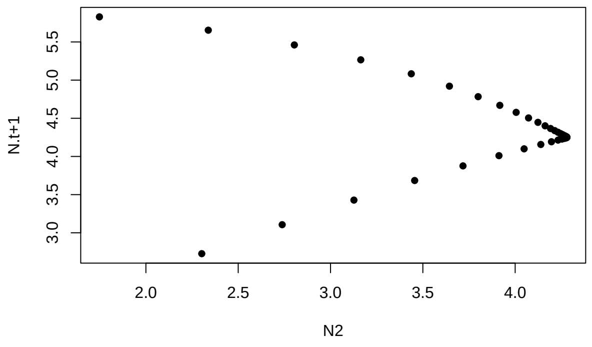


Figure 14: Population size (logarithm) at one time step N_{t+1} as a function of log-population size of Species 1 and Species 2 in the previous time step N_t. Also shown is the change in population size of Species 1 as a function of N_t for both species.

71 We can see that the population size of Species 2 does not have a linear relationship with that of Species 1.

```
df <- data.frame(cbind(log(dat$N1[2:t]), log(dat$N1[1:(t - 1)]), log(dat$N2[1:(t -
    1)])))
colnames(df) <- c("Nt1", "N1", "N2")
m.3.lm <- lm(Nt1 ~ I(1/N1) + I(1/N2), data = df)

cor(df$Nt1, predict(m.3.lm))
#> [1] 0.9845708
summary(m.3.lm)
#>
#> Call:
#> lm(formula = Nt1 ~ I(1/N1) + I(1/N2), data = df)
#>
#> Residuals:
#>       Min        1Q    Median        3Q       Max
#> -0.30952 -0.04467 -0.03442  0.03430  0.22447
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   6.4056     0.1060   60.45  < 2e-16 ***
#> I(1/N1)     -12.2863     0.3774  -32.55  < 2e-16 ***
#> I(1/N2)       3.3309     0.2448   13.61 9.24e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.1063 on 36 degrees of freedom
#> Multiple R-squared:  0.9694, Adjusted R-squared:  0.9677
#> F-statistic: 569.8 on 2 and 36 DF,  p-value: < 2.2e-16
xx <- seq(0, 100, length = 1000)
prediction <- data.frame(N2 = xx, N1 = 100)
plot(df$N2, df$Nt1, xlab = "N2", ylab = "N.t+1", pch = 16)
lines(prediction$N2, predict(m.3.lm, prediction), lty = 2, col = "red", lwd = 3)
```



```
#> Using data df from global environment. This could cause incorrect results
#> if df has been altered since the model was fit. You can manually provide
#> the data to the "data =" argument.
```

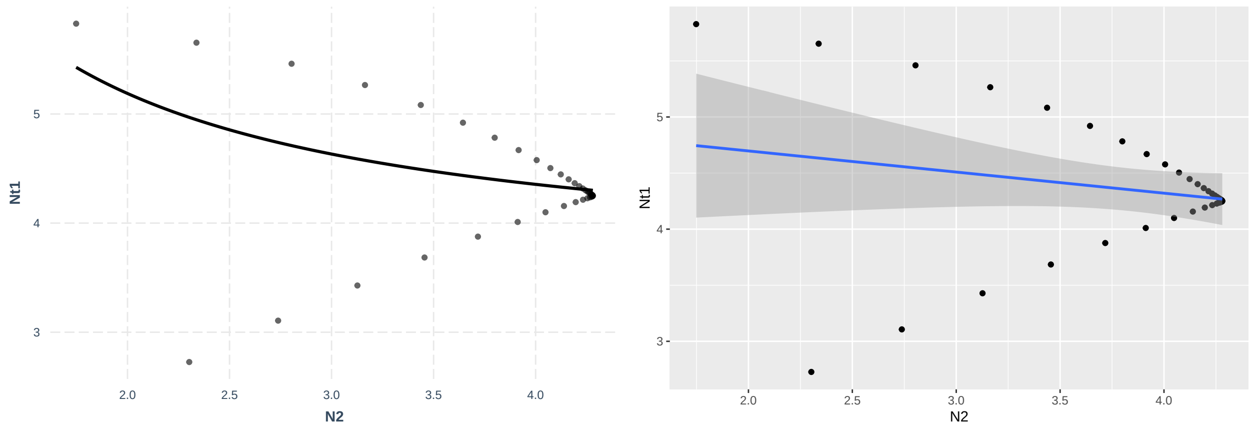


Figure 15: Population size (logarithm) at one time step N_{t+1} as a function of log-population size of the second species in the previous time step N_t. Fitted (line) and observed (points) values are shown.

We can already see that the linear model does not provide a useful fit to the data points (neither does taking the inverse function help). The regression parameters can predict well the population size of the other species, but they are not informative for inference. This has been observed previously (Certain et al. (2018)), and non-linear least squares models (among others) are more often used to estimate parameters for dynamics resulting from these kinds of species interaction models (Kloppers and Greeff (2013); Mühlbauer et al. (2020); Olivença et al. (2021)). The inability of the linear models to reproduce the interaction coefficients is more clearly demonstrated in Certain et al. (2018). However, they do indicate that the slope of the linear model can reflect the direction of effect for the interacting species.

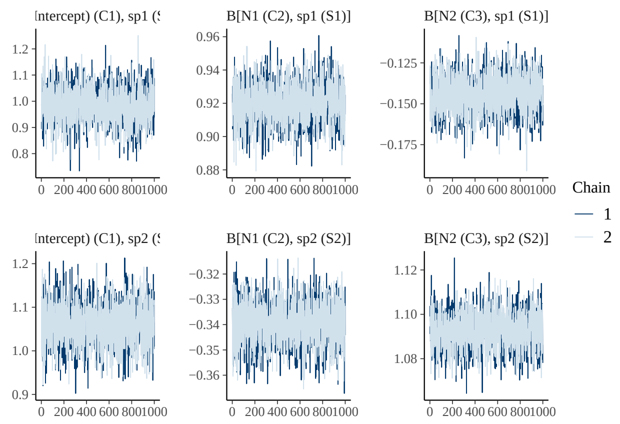
Instead of further exploring techniques to fit to time-series derived from non-linear competition processes, we look more closely at how HMSC manages to fit data emerging from competition dynamics and make inference about species interactions. Instead of estimating full interaction coefficients, or assuming sparse interactions, they instead assume that a reduced number of linear combinations of species abundances that are most relevant to determining future growth rates for species in the community can bypass the 'curse of dimensionality' problem. This is well presented in Ovaskainen et al. (2017). We apply that approach here:

3.3 Bayesian linear statistical model: HMSC

We can fit the data to a linear model using HMSC, with latent variables to capture the species associations. These are implemented by including a random effect at the time level. For this analysis, we do not expect the impact of Species 1 and 2 that results from competition to vary over time - competition will always have negative impacts for species abundances. Our goal is to estimate the overall impact of species covariance across all of the time points. We do not estimate these impacts as a function of the lag in time points. In other words, we assume that each time point has random variance and that species may covary in their response at each time point. We do not consider that species covariance is a function of distance between sampled time points (see Ovaskainen and Abrego (2020), equation 5.9). We instead assume that site loadings η are independent among the sampled time points.

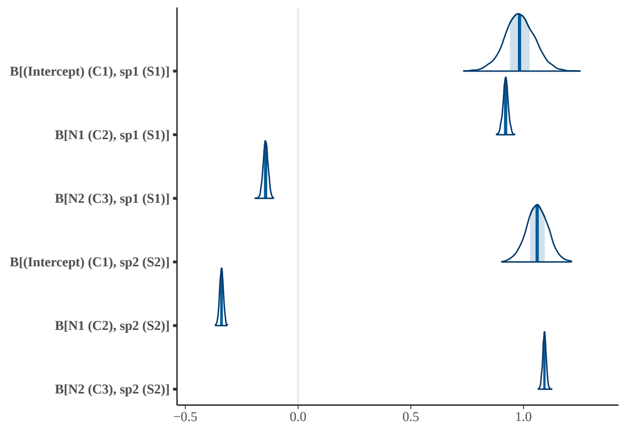
```
# prepare data in HMSC format
Y <- as.matrix(cbind(log(dat$N1[2:t]), log(dat$N2[2:t])))
XData <- df[, 2:3]
studyDesign = data.frame(sample = as.factor(1:(t - 1)))
studyDesign$sample <- as.factor(studyDesign$sample)
rL.sample = HmscRandomLevel(units = studyDesign$sample)
m.3.hmsc = Hmsc(Y = Y, XData = XData, studyDesign = studyDesign, ranLevels = list(sample = rL.sample))
# Bayesian model parameters
nChains <- 2
thin <- 5
samples <- 1000
transient <- 500 * thin
verbose <- 500 * thin
# sample MCMC
m.3.sample <- sampleMcmc(m.3.hmsc, thin = thin, sample = samples, transient = transient,
    nChains = nChains, verbose = verbose)
#> Computing chain 1
#> Chain 1, iteration 2500 of 7500 (transient)
#> Chain 1, iteration 5000 of 7500 (sampling)
#> Chain 1, iteration 7500 of 7500 (sampling)
#> Computing chain 2
#> Chain 2, iteration 2500 of 7500 (transient)
#> Chain 2, iteration 5000 of 7500 (sampling)
#> Chain 2, iteration 7500 of 7500 (sampling)
```

```
m3.post.hmsc <- convertToCodaObject(m.3.sample)
summary(m3.post.hmsc$Beta)
#>
#> Iterations = 2505:7500
#> Thinning interval = 5
#> Number of chains = 2
#> Sample size per chain = 1000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>                                   Mean       SD  Naive SE Time-series SE
#> B[(Intercept) (C1), sp1 (S1)]  0.9829 0.067231 0.0015033      0.0015031
#> B[N1 (C2), sp1 (S1)]           0.9202 0.011671 0.0002610      0.0002610
#> B[N2 (C3), sp1 (S1)]          -0.1442 0.011042 0.0002469      0.0002516
#> B[(Intercept) (C1), sp2 (S2)]  1.0606 0.048507 0.0010846      0.0010845
#> B[N1 (C2), sp2 (S2)]          -0.3398 0.008349 0.0001867      0.0001867
#> B[N2 (C3), sp2 (S2)]           1.0926 0.007894 0.0001765      0.0001827
#>
#> 2. Quantiles for each variable:
#>
#>                                   2.5%     25%     50%     75%   97.5%
#> B[(Intercept) (C1), sp1 (S1)]  0.8470  0.9394  0.9820  1.0269  1.1174
#> B[N1 (C2), sp1 (S1)]           0.8973  0.9129  0.9205  0.9278  0.9432
#> B[N2 (C3), sp1 (S1)]          -0.1653 -0.1514 -0.1442 -0.1369 -0.1226
#> B[(Intercept) (C1), sp2 (S2)]  0.9638  1.0285  1.0603  1.0938  1.1557
#> B[N1 (C2), sp2 (S2)]          -0.3563 -0.3455 -0.3397 -0.3339 -0.3235
#> B[N2 (C3), sp2 (S2)]           1.0772  1.0874  1.0926  1.0977  1.1080
bayesplot::mcmc_trace(m3.post.hmsc$Beta)
```



100

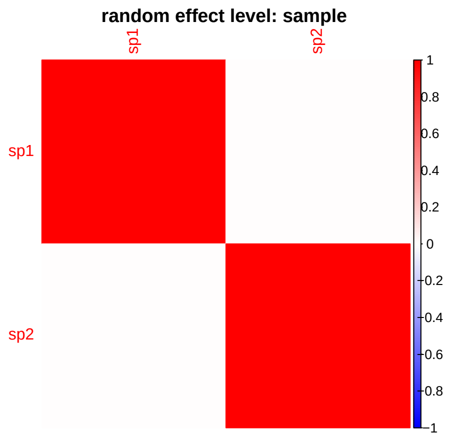
```
bayesplot::mcmc_areas(m3.post.hmsc$Beta, area_method = c("equal height"))
```



101

102 Now we look at the estimates for the species associations:

```r
OmegaCor = computeAssociations(m.3.sample)
supportLevel = 0.95
toPlot = ((OmegaCor[[1]]$support > supportLevel) + (OmegaCor[[1]]$support < (1 -
    supportLevel)) > 0) * OmegaCor[[1]]$mean
corrplot::corrplot(toPlot, method = "color", col = colorRampPalette(c("blue", "white",
    "red"))(200), title = paste("random effect level:", m.3.sample$rLNames[1]), mar = c(0,
    0, 1, 0))
```



103

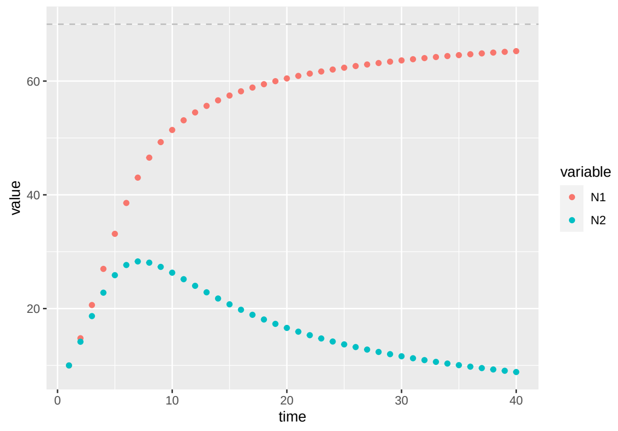
104 The model fixed effect estimates match well with those from the AR1 and linear model.

```r
# AR1 coefficients (recall that the intercept is the term below multipled by 1
# - phi1)
m.3.ar.n1$coef
#>         ar1   intercept        N$N2
#>  0.79696422  5.69840619 -0.01881979
m.3.ar.n1$coef[2] * (1 - m.3.ar.n1$coef[1])
#> intercept
#>   1.15698
# linear model
summary(m.3.lm.n1)$coefficients[1:3, 1:2]
#>                          Estimate  Std. Error
#> (Intercept)             0.9839341 0.053084123
#> log(dat$N1[1:(t - 1)])  0.9201864 0.009157214
#> log(dat$N2[1:(t - 1)]) -0.1443693 0.008777527
# Bayesian estimates
summary(m3.post.hmsc$Beta)$statistics[1:3, 1:2]
#>                                     Mean         SD
#> B[(Intercept) (C1), sp1 (S1)]  0.9829117 0.06723119
#> B[N1 (C2), sp1 (S1)]           0.9202360 0.01167056
#> B[N2 (C3), sp1 (S1)]          -0.1441626 0.01104150
```

105 I try again with a next example where the only difference between species is the α_{ij}.

```
# Simulate initial species population growth with a second species present
N1.0 <- 10
N2.0 <- 10
r1.0 <- 1.7
r2.0 <- 1.7
alpha.11 <- 0.01
alpha.22 <- 0.01
alpha.12 <- 0.005
alpha.21 <- 0.01
# model function
disc_LV_comp <- function(r1, r2, N1.0, N2.0, alpha.11, alpha.22, alpha.12, alpha.21) {
    Nt1 <- (r1 * N1.0)/(1 + alpha.11 * N1.0 + alpha.12 * N2.0)
    Nt2 <- (r2 * N2.0)/(1 + alpha.22 * N2.0 + alpha.21 * N1.0)
    return(c(Nt1, Nt2))
}
# Simulation of model for t time steps
t <- 40
N <- array(NA, dim = c(t, 2))
N <- as.data.frame(N)
colnames(N) <- c("N1", "N2")
N$N1[1] <- N1.0
N$N2[1] <- N2.0
for (i in 2:t) {
    res <- disc_LV_comp(r1 = r1.0, r2 = r2.0, N1.0 = N[i - 1, 1], N2.0 = N[i - 1,
        2], alpha.11 = alpha.11, alpha.22 = alpha.22, alpha.12 = alpha.12, alpha.21 = alpha.21)
    N$N1[i] <- res[1]
    N$N2[i] <- res[2]
}
```

```
# Plot simulation: ggplot
N$time <- 1:t
dat <- melt(N, id.vars = "time")
ggplot2::ggplot(dat, aes(time, value, col = variable)) + geom_point() + geom_hline(yintercept = ((r1.0
    1)/alpha.11), linetype = "dashed", color = "gray")
```

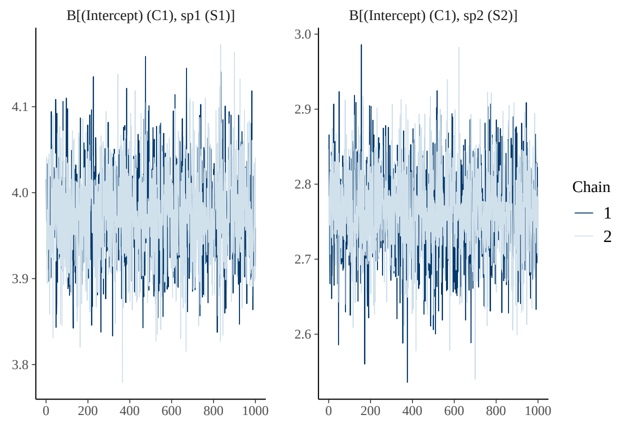


106

```
# Plot simulation: ggplot
dat <- as.data.frame(cbind(N$N1, N$N2))
colnames(dat) <- c("N1", "N2")
dat$time <- 1:t
df <- data.frame(cbind(log(dat$N1[2:t]), log(dat$N2[2:t])))
colnames(df) <- c("Nt1", "Nt2")
# prepare data in HMSC format
Y <- as.matrix(cbind(df$Nt1, df$Nt2))
XData <- data.frame(cbind(log(dat$N1[1:(t - 1)]), log(dat$N2[1:(t - 1)])))
studyDesign = data.frame(sample = as.factor(1:(t - 1)))
studyDesign$sample <- as.factor(studyDesign$sample)
rL.sample = HmscRandomLevel(units = studyDesign$sample)
m.4.hmsc = Hmsc(Y = Y, XData = XData, XFormula = ~1, studyDesign = studyDesign, ranLevels = list(sample
# Bayesian model parameters
nChains <- 2
thin <- 5
samples <- 1000
transient <- 500 * thin
verbose <- 500 * thin
# sample MCMC
m.4.sample <- sampleMcmc(m.4.hmsc, thin = thin, sample = samples, transient = transient,
    nChains = nChains, verbose = verbose)
#> Computing chain 1
#> Chain 1, iteration 2500 of 7500 (transient)
#> Chain 1, iteration 5000 of 7500 (sampling)
#> Chain 1, iteration 7500 of 7500 (sampling)
#> Computing chain 2
#> Chain 2, iteration 2500 of 7500 (transient)
#> Chain 2, iteration 5000 of 7500 (sampling)
#> Chain 2, iteration 7500 of 7500 (sampling)
```

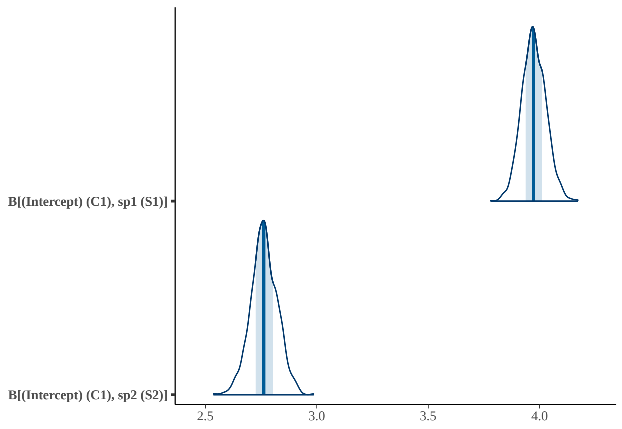
```
m4.post.hmsc <- convertToCodaObject(m.4.sample)
summary(m4.post.hmsc$Beta)
#>
#> Iterations = 2505:7500
#> Thinning interval = 5
#> Number of chains = 2
#> Sample size per chain = 1000
#>
```

```
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>                                  Mean      SD Naive SE Time-series SE
#> B[(Intercept) (C1), sp1 (S1)] 3.974 0.05406 0.001209       0.001173
#> B[(Intercept) (C1), sp2 (S2)] 2.764 0.06002 0.001342       0.001388
#>
#> 2. Quantiles for each variable:
#>
#>                                  2.5%   25%   50%   75% 97.5%
#> B[(Intercept) (C1), sp1 (S1)] 3.872 3.937 3.973 4.011 4.085
#> B[(Intercept) (C1), sp2 (S2)] 2.642 2.725 2.762 2.804 2.885
bayesplot::mcmc_trace(m4.post.hmsc$Beta)
```



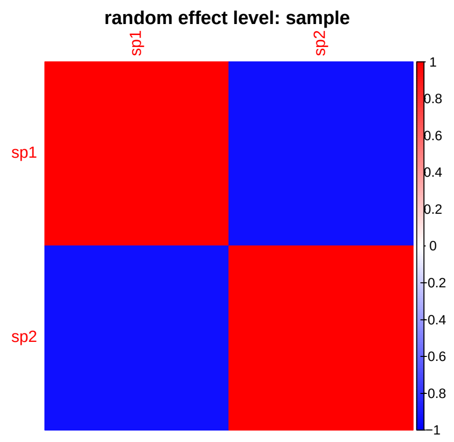
107

```
bayesplot::mcmc_areas(m4.post.hmsc$Beta, area_method = c("equal height"))
```



108

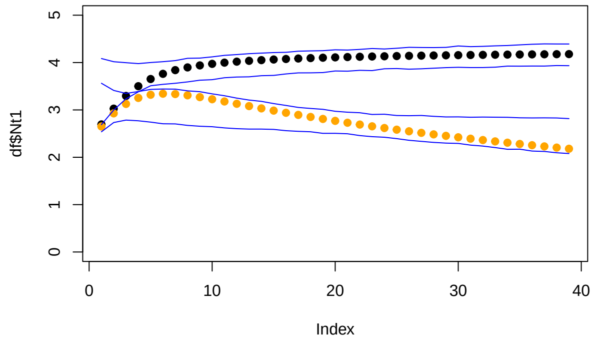
```
OmegaCor = computeAssociations(m.4.sample)
supportLevel = 0.95
toPlot = ((OmegaCor[[1]]$support > supportLevel) + (OmegaCor[[1]]$support < (1 -
    supportLevel)) > 0) * OmegaCor[[1]]$mean
corrplot::corrplot(toPlot, method = "color", col = colorRampPalette(c("blue", "white",
    "red"))(200), title = paste("random effect level:", m.4.sample$rLNames[1]), mar = c(0,
    0, 1, 0))
```



The HMSC model correctly shows the negative association between the species (shown here as correlations - JSDM CHapter 7 tells us the intra-specific correlation is always 1).

```r
# explanatory power
preds = computePredictedValues(m.4.sample)
# evaluateModelFit(hM = m.4.sample, predY = preds) predictive power /
# cross-validation partition = createPartition(m.4.sample, nfolds = 2) preds =
# computePredictedValues(m.4.sample, partition = partition, nParallel =
# nChains) evaluateModelFit(hM = m.4.sample, predY = preds) xx preds =
# computePredictedValues(m.4.sample, partition=partition, partition.sp=c(1,2),
# mcmcStep=10, nParallel = nChains) evaluateModelFit(hM=m.4.sample,
# predY=preds) xx etaPost=getPostEstimate(m.4.sample, 'Eta')
# lambdaPost=getPostEstimate(m.4.sample, 'Lambda') biPlot(m.4.sample, etaPost =
# etaPost, lambdaPost = lambdaPost, factors = c(1,2),'X1')
```

```r
df$n1_25 <- NA
df$n1_975 <- NA
df$n2_25 <- NA
df$n2_975 <- NA
for (i in 1:39) {
    red <- preds[i, , ]
    a <- apply(red, 1, quantile, probs = c(0.025, 0.975))
    df$n1_25[i] <- a[1, 1]
    df$n1_975[i] <- a[2, 1]
    df$n2_25[i] <- a[1, 2]
    df$n2_975[i] <- a[2, 2]
}
plot(df$Nt1, pch = 19, col = "black", ylim = c(0, 5))
points(df$Nt2, pch = 19, col = "orange")
lines(df$n1_25, col = "blue")
lines(df$n1_975, col = "blue")
lines(df$n2_25, col = "blue")
lines(df$n2_975, col = "blue")
```



The residual associations should not be interpreted as interaction coefficients, but rather used for prediction. We can see here they are effective for this (shown above is the observed and model-predicted 95% CI).

4 Species growth, competition, with environmental change

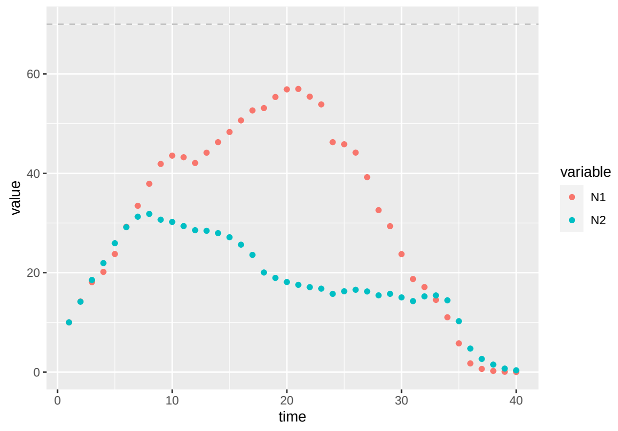
The goal of the process was to use HMSC's latent variable approach to study the results of non-linear processes in ecology using a linear model. We can infer the direction and magnitude of the species interaction. We now consider an environmental covariate that changes over time and impacts each species growth. We will use only HMSC from this point forward, as the species interactions represent a departure from what a linear or AR model can manage.

In this example, the local environmental optimum trait value $E = 0.8$, and Species 1 $x_1 = 0.6$, while $x_1 = 0.8$.

```r
# Simulate initial species population growth with environment fluctuations
N1.0 <- 10
N2.0 <- 10
r1.0 <- 1.7
r2.0 <- 1.7
alpha.11 <- 0.01
alpha.22 <- 0.01
alpha.12 <- 0.005
alpha.21 <- 0.01
E.0 <- 0.8
x1.0 <- 0.6
x2.0 <- 0.8

# model function
disc_LV_E <- function(r1, r2, N1.0, N2.0, alpha.11, alpha.22, alpha.12, alpha.21,
    E, x1, x2) {
    Nt1 <- ((r1 * exp(-(E - x1)^2)) * N1.0)/(1 + alpha.11 * N1.0 + alpha.12 * N2.0)
    Nt2 <- ((r2 * exp(-(E - x2)^2)) * N2.0)/(1 + alpha.22 * N2.0 + alpha.21 * N1.0)
    return(c(Nt1, Nt2))
}
# Simulation of model for t time steps
t <- 40
N <- array(NA, dim = c(t, 2))
N <- as.data.frame(N)
colnames(N) <- c("N1", "N2")
N$N1[1] <- N1.0
```

```
N$N2[1] <- N2.0
E <- rep(NA, t)
E[1] <- E.0
for (i in 2:t) {
    res <- disc_LV_E(r1 = r1.0, r2 = r2.0, N1.0 = N[i - 1, 1], N2.0 = N[i - 1, 2],
        alpha.11 = alpha.11, alpha.22 = alpha.22, alpha.12 = alpha.12, alpha.21 = alpha.21,
        E = E[i - 1], x1 = x1.0, x2 = x2.0)
    N$N1[i] <- res[1]
    N$N2[i] <- res[2]
    E[i] <- E[i - 1] + rnorm(1, 0, 0.1)
}
```

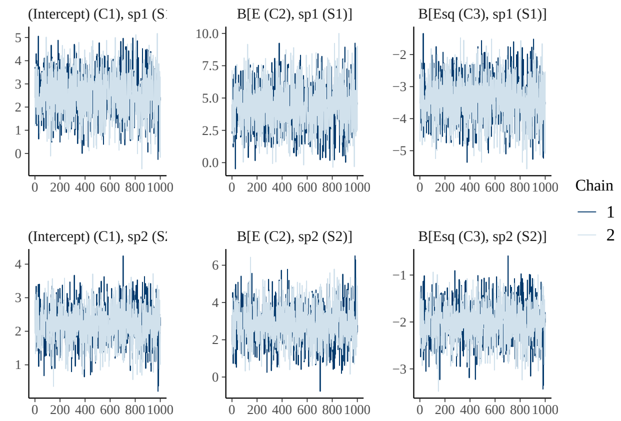
```
# Plot simulation: ggplot
N$time <- 1:t
dat <- melt(N, id.vars = "time")
ggplot2::ggplot(dat, aes(time, value, col = variable)) + geom_point() + geom_hline(yintercept = ((r1.0
    1)/alpha.11), linetype = "dashed", color = "gray")
```



```
# Plot simulation: ggplot
dat <- as.data.frame(cbind(log(N$N1), log(N$N2)))
colnames(dat) <- c("N1", "N2")
dat$time <- 1:t
df <- data.frame(cbind(dat$N1[2:t], dat$N2[2:t]))
colnames(df) <- c("Nt1", "Nt2")
# prepare data in HMSC format
Y <- as.matrix(cbind(df$Nt1, df$Nt2))
XData <- data.frame(cbind(dat$N1[1:(t - 1)], dat$N2[1:(t - 1)]), E[1:(t - 1)], E[1:(t -
    1)]^2)
colnames(XData) <- c("n1", "n2", "E", "Esq")
studyDesign = data.frame(sample = as.factor(1:(t - 1)))
rL = HmscRandomLevel(units = studyDesign$sample)
m.5.hmsc = Hmsc(Y = Y, XData = XData, XFormula = ~E + Esq, studyDesign = studyDesign,
    ranLevels = list(sample = rL))
# Bayesian model parameters
nChains <- 2
thin <- 5
samples <- 1000
transient <- 500 * thin
```

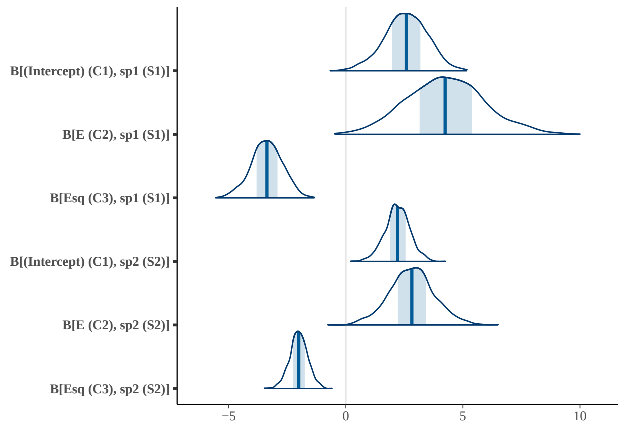
```
verbose <- 500 * thin
# sample MCMC
m.5.sample <- sampleMcmc(m.5.hmsc, thin = thin, sample = samples, transient = transient,
    nChains = nChains, verbose = verbose)
#> Computing chain 1
#> Chain 1, iteration 2500 of 7500 (transient)
#> Chain 1, iteration 5000 of 7500 (sampling)
#> Chain 1, iteration 7500 of 7500 (sampling)
#> Computing chain 2
#> Chain 2, iteration 2500 of 7500 (transient)
#> Chain 2, iteration 5000 of 7500 (sampling)
#> Chain 2, iteration 7500 of 7500 (sampling)
```

```
m5.post.hmsc <- convertToCodaObject(m.5.sample)
summary(m5.post.hmsc$Beta)
#>
#> Iterations = 2505:7500
#> Thinning interval = 5
#> Number of chains = 2
#> Sample size per chain = 1000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>                                   Mean     SD Naive SE Time-series SE
#> B[(Intercept) (C1), sp1 (S1)]  2.572 0.9131 0.020417        0.02342
#> B[E (C2), sp1 (S1)]            4.274 1.6630 0.037186        0.04303
#> B[Esq (C3), sp1 (S1)]         -3.374 0.6726 0.015039        0.01740
#> B[(Intercept) (C1), sp2 (S2)]  2.204 0.5331 0.011920        0.01379
#> B[E (C2), sp2 (S2)]            2.832 0.9647 0.021571        0.02548
#> B[Esq (C3), sp2 (S2)]         -2.010 0.3893 0.008705        0.01037
#>
#> 2. Quantiles for each variable:
#>
#>                                  2.5%    25%    50%    75%  97.5%
#> B[(Intercept) (C1), sp1 (S1)]  0.6559  1.966  2.584  3.186  4.300
#> B[E (C2), sp1 (S1)]            1.1272  3.151  4.240  5.384  7.713
#> B[Esq (C3), sp1 (S1)]         -4.7501 -3.810 -3.367 -2.915 -2.088
#> B[(Intercept) (C1), sp2 (S2)]  1.1184  1.876  2.208  2.555  3.278
#> B[E (C2), sp2 (S2)]            0.8746  2.216  2.823  3.419  4.800
#> B[Esq (C3), sp2 (S2)]         -2.8112 -2.254 -2.009 -1.756 -1.235
bayesplot::mcmc_trace(m5.post.hmsc$Beta)
```



123

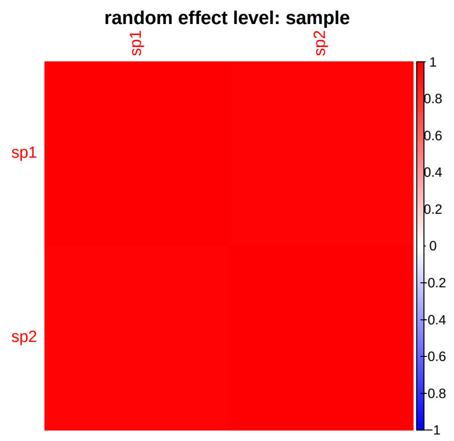
```
bayesplot::mcmc_areas(m5.post.hmsc$Beta, area_method = c("equal height"))
```



124

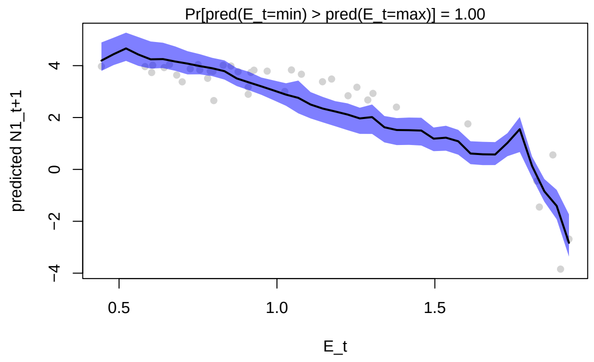
```
# Bayesian estimates
summary(m5.post.hmsc$Beta)$statistics[1:4, 1:2]
#>                                    Mean        SD
#> B[(Intercept) (C1), sp1 (S1)]  2.571888 0.9130903
#> B[E (C2), sp1 (S1)]            4.274430 1.6630073
#> B[Esq (C3), sp1 (S1)]         -3.373885 0.6725570
#> B[(Intercept) (C1), sp2 (S2)]  2.203589 0.5330636
```

```
OmegaCor = computeAssociations(m.5.sample)
supportLevel = 0.7
toPlot = ((OmegaCor[[1]]$support > supportLevel) + (OmegaCor[[1]]$support < (1 -
    supportLevel)) > 0) * OmegaCor[[1]]$mean
corrplot::corrplot(toPlot, method = "color", col = colorRampPalette(c("blue", "white",
    "red"))(200), title = paste("random effect level:", m.5.sample$rLNames[1]), mar = c(0,
    0, 1, 0))
```



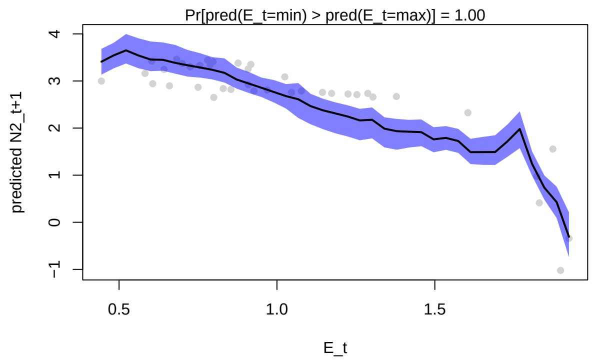
125

```
Gradient <- constructGradient(m.5.sample, focalVariable = "E", non.focalVariables = list(Esq = list(2))
    ngrid = 39)
predY <- predict(m.5.sample, XData = Gradient$XDataNew, expected = TRUE)
a <- plotGradient(m.5.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 1, main = "", xlab = "E_t", ylab = "predicted N1_t+1")
```



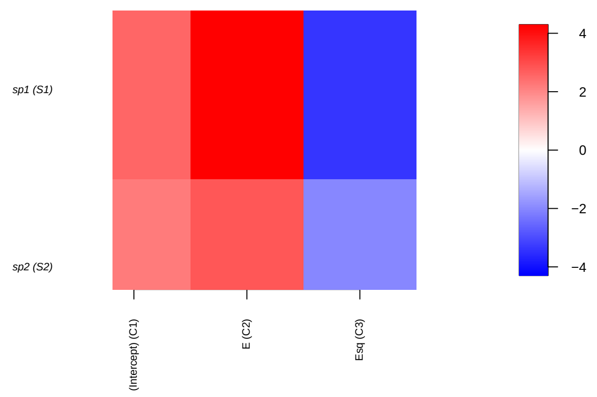
126

```
b <- plotGradient(m.5.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 2, main = "", xlab = "E_t", ylab = "predicted N2_t+1")
```



127

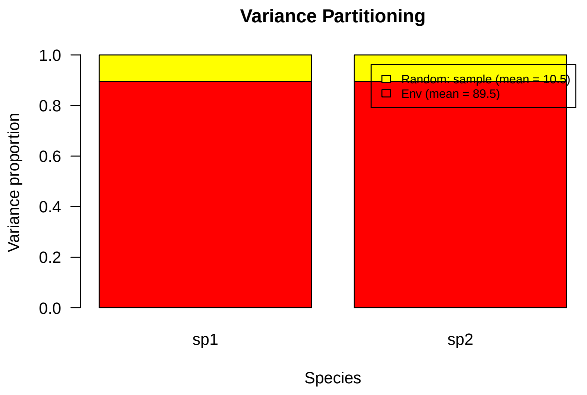
```r
postBeta = getPostEstimate(m.5.sample, parName = "Beta")
plotBeta(m.5.sample, post = postBeta, param = "Mean", supportLevel = 0.7)
```



We can see that the Environment has an overall negative impact for both species as E drifts randomly towards more positive values (where Species 2 is closer because it has a higher trait value). We also see that the residual species association effects are strongly positive.

We use variation partition to estimate the relative importance of environment and species associations for species abundances.

```r
VP <- computeVariancePartitioning(m.5.sample, group = c(1, 1, 1), groupnames = "Env")
plotVariancePartitioning(m.5.sample, VP, args.legend = list(cex = 0.75, bg = "transparent"))
```

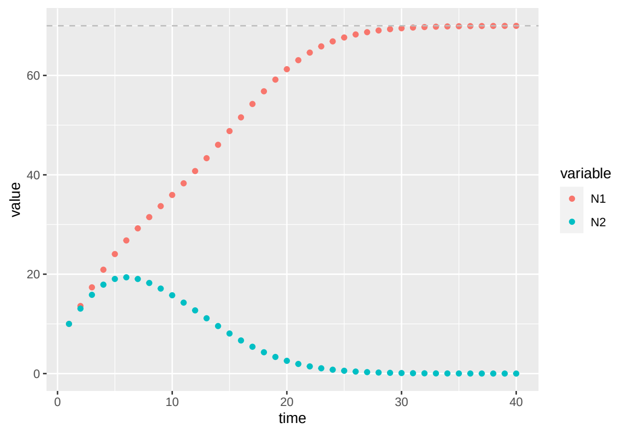


We evaluate this by greatly increasing the effects of interspecific competition (while decreasing the strength of environmental variation).

```r
# Simulate initial species population growth with environment fluctuations
N1.0 <- 10
N2.0 <- 10
r1.0 <- 1.7
r2.0 <- 1.7
alpha.11 <- 0.01
alpha.22 <- 0.01
alpha.12 <- 0.015
alpha.21 <- 0.02
```

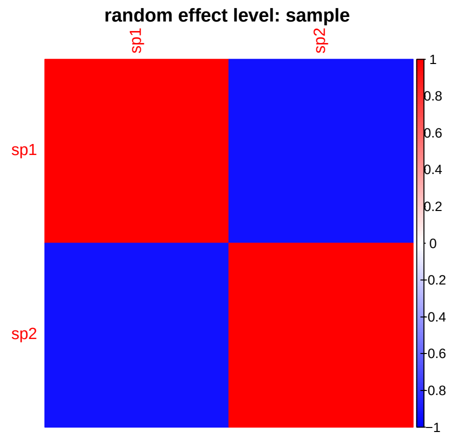
```r
E.0 <- 0.8
x1.0 <- 0.8
x2.0 <- 0.8

# model function
disc_LV_E <- function(r1, r2, N1.0, N2.0, alpha.11, alpha.22, alpha.12, alpha.21,
    E, x1, x2) {
    Nt1 <- ((r1 * exp(-(E - x1)^2)) * N1.0)/(1 + alpha.11 * N1.0 + alpha.12 * N2.0)
    Nt2 <- ((r2 * exp(-(E - x2)^2)) * N2.0)/(1 + alpha.22 * N2.0 + alpha.21 * N1.0)
    return(c(Nt1, Nt2))
}
# Simulation of model for t time steps
t <- 40
N <- array(NA, dim = c(t, 2))
N <- as.data.frame(N)
colnames(N) <- c("N1", "N2")
N$N1[1] <- N1.0
N$N2[1] <- N2.0
E <- rep(NA, t)
E[1] <- E.0
for (i in 2:t) {
    res <- disc_LV_E(r1 = r1.0, r2 = r2.0, N1.0 = N[i - 1, 1], N2.0 = N[i - 1, 2],
        alpha.11 = alpha.11, alpha.22 = alpha.22, alpha.12 = alpha.12, alpha.21 = alpha.21,
        E = E[i - 1], x1 = x1.0, x2 = x2.0)
    N$N1[i] <- res[1]
    N$N2[i] <- res[2]
    E[i] <- E[i - 1] + rnorm(1, 0, 0.001)
}
```

```r
# Plot simulation: ggplot
N$time <- 1:t
dat <- melt(N, id.vars = "time")
ggplot2::ggplot(dat, aes(time, value, col = variable)) + geom_point() + geom_hline(yintercept = ((r1.0
    1)/alpha.11), linetype = "dashed", color = "gray")
```



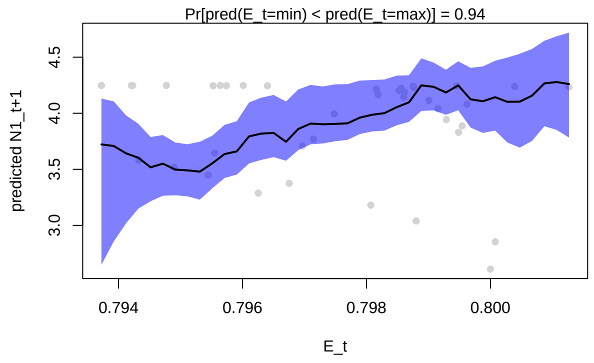
```
#> Computing chain 1
#> Chain 1, iteration 2500 of 10500 (sampling)
#> Chain 1, iteration 5000 of 10500 (sampling)
#> Chain 1, iteration 7500 of 10500 (sampling)
#> Chain 1, iteration 10000 of 10500 (sampling)
#> Computing chain 2
#> Chain 2, iteration 2500 of 10500 (sampling)
#> Chain 2, iteration 5000 of 10500 (sampling)
#> Chain 2, iteration 7500 of 10500 (sampling)
#> Chain 2, iteration 10000 of 10500 (sampling)
```

```
OmegaCor = computeAssociations(m.6.sample)
supportLevel = 0.7
toPlot = ((OmegaCor[[1]]$support > supportLevel) + (OmegaCor[[1]]$support < (1 -
    supportLevel)) > 0) * OmegaCor[[1]]$mean
corrplot::corrplot(toPlot, method = "color", col = colorRampPalette(c("blue", "white",
    "red"))(200), title = paste("random effect level:", m.6.sample$rLNames[1]), mar = c(0,
    0, 1, 0))
```

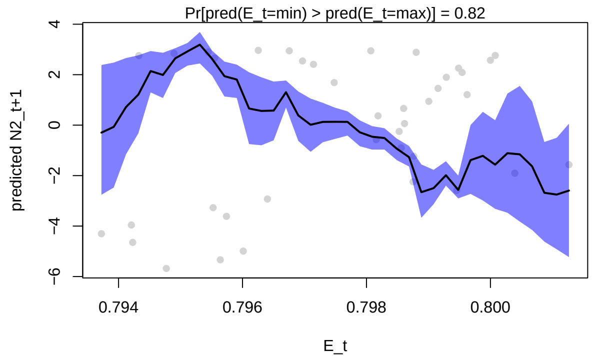


```
OmegaCor
#> [[1]]
#> [[1]]$mean
#>             sp1        sp2
#> sp1  1.0000000 -0.9201566
#> sp2 -0.9201566  1.0000000
#>
#> [[1]]$support
#>     sp1 sp2
#> sp1   1   0
#> sp2   0   1
```

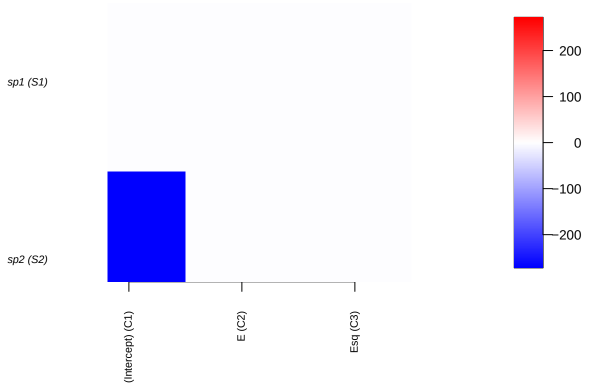
```
Gradient <- constructGradient(m.6.sample, focalVariable = "E", non.focalVariables = list(Esq = list(2))
    ngrid = 39)
predY <- predict(m.6.sample, XData = Gradient$XDataNew, expected = TRUE)
a <- plotGradient(m.6.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 1, main = "", xlab = "E_t", ylab = "predicted N1_t+1")
```



```r
b <- plotGradient(m.6.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 2, main = "", xlab = "E_t", ylab = "predicted N2_t+1")
```

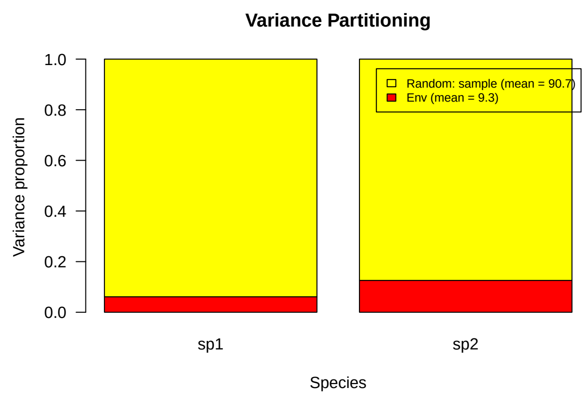


```r
postBeta = getPostEstimate(m.6.sample, parName = "Beta")
plotBeta(m.6.sample, post = postBeta, param = "Mean", supportLevel = 0.7)
```



With the reduced impact of the environment, we now see the strong negative associations between the two competing species. We also correctly see no impact of environment for species abundances.

```
VP <- computeVariancePartitioning(m.6.sample, group = c(1, 1, 1), groupnames = "Env")
plotVariancePartitioning(m.6.sample, VP, args.legend = list(cex = 0.75, bg = "transparent"))
```



5 Species growth, competition, environmental change, and trait evolution

We now consider that trait evolution can occur, with an impact on the species growth rate / fitness in the context of the environment.

5.1 Trait evolution

Previously the population growth rate was fixed as $r_i = \hat{W}e^{-(E-x_{i,t})^2}$. However, we now introduce the quantitative genetic model of evolutionary rescue (Gomulkiewicz and Holt 1995).

The growth equation for each species now becomes:

$$N_{i,t+1} = \frac{\hat{W}e^{\frac{-[(\frac{w+(1-h^2)P}{P+w})(E-x_{i,t})]^2}{2(P+w)}} N_{i,t}}{1+\alpha_{ii}N_{i,t}+\alpha_{ij}N_{j,t}}$$

where $\hat{W}$ is calculated as $\hat{W} = W_{max}\sqrt{(\frac{w}{P+w})}$, W_{max} is the species' maximum per-capita growth rate, w is the width of the Gaussian fitness function (which determines the strength of selection, as increasing values indicate a weaker reduction in fitness with distance from optimum trait value), P is the width of the distribution of the phenotype x, and h^2 is the heritability of the trait x. For the simulation we use $W_{max} = 2$, $P = 1$, and $w = 2$.

The change in the average trait value each time step is given by:

$$d_{i,t+1} = kd_{i,t}$$

where $k = \frac{w+(1-h^2)P}{w+P}$ and $d_{i,t} = E_t - x_{i,t}$.

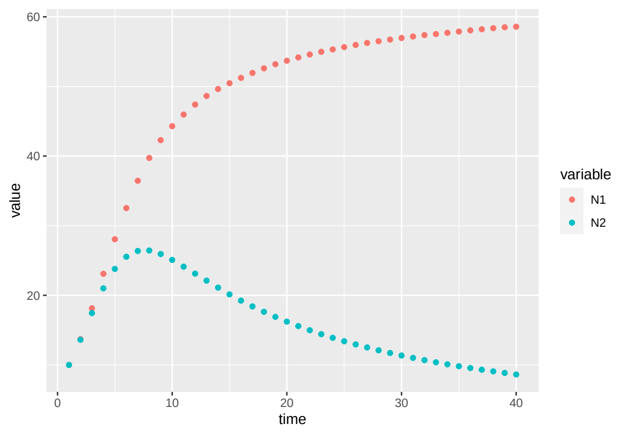
5.2 Population dynamics simulation

We use one of the same examples as above, with weak strength of interspecific interactions. We use initial trait values that favor the weaker competitor, and we simulate random increasing shifts in the local environment.

```r
# Case 1: Weaker interactions, stronger environment Simulate initial species
# population growth with environment fluctuations
N1.0 <- 10
N2.0 <- 10
alpha.11 <- 0.01
alpha.22 <- 0.01
alpha.12 <- 0.005
alpha.21 <- 0.01
E.0 <- 0.8
x1.0 <- 0.1
x2.0 <- 0.8
P <- 1
w <- 2
Wmax <- 2
h2 <- 1
k <- (w + (1 - h2) * P)/(P + w)

# model function
disc_LV_evol <- function(N1.0, N2.0, alpha.11, alpha.22, alpha.12, alpha.21, E, x1.0,
    x2.0, P, w, Wmax, h2) {
    What <- Wmax * sqrt(w/(P + w))
    r1 <- What * exp((-(((w + (1 - h2) * P)/(P + w)) * (E - x1.0))^2)/(2 * (P + w)))
    Nt1 <- (r1 * N1.0)/(1 + alpha.11 * N1.0 + alpha.12 * N2.0)
    r2 <- What * exp((-(((w + (1 - h2) * P)/(P + w)) * (E - x2.0))^2)/(2 * (P + w)))
    Nt2 <- (r2 * N2.0)/(1 + alpha.22 * N2.0 + alpha.21 * N1.0)
    return(c(Nt1, Nt2, r1, r2))
}
# Simulation of model for t time steps
t <- 40
N <- array(NA, dim = c(t, 2))
N <- as.data.frame(N)
colnames(N) <- c("N1", "N2")
x <- array(NA, dim = c(t, 2))
x <- as.data.frame(x)
colnames(x) <- c("x1", "x2")
r <- array(NA, dim = c(t, 2))
r <- as.data.frame(r)
colnames(r) <- c("r1", "r2")
N$N1[1] <- N1.0
N$N2[1] <- N2.0
x$x1[1] <- x1.0
x$x2[1] <- x2.0
What <- Wmax * sqrt(w/(P + w))
r$r1[1] <- What * (exp((-(((w + (1 - h2) * P)/(P + w)) * (E.0 - x1.0))^2)/(2 * (P +
    w))))
r$r2[1] <- What * (exp((-(((w + (1 - h2) * P)/(P + w)) * (E.0 - x2.0))^2)/(2 * (P +
    w))))
E <- rep(NA, t)
E[1] <- E.0
```

```
for (i in 2:t) {
    res <- disc_LV_evol(N1.0 = N[i - 1, 1], N2.0 = N[i - 1, 2], alpha.11 = alpha.11,
        alpha.22 = alpha.22, alpha.12 = alpha.12, alpha.21 = alpha.21, E = E[i -
            1], x1 = x[i - 1, 1], x2 = x[i - 1, 2], P = P, w = w, Wmax = Wmax, h2 = h2)
    N$N1[i] <- res[1]
    N$N2[i] <- res[2]
    r$r1[i] <- res[3]
    r$r2[i] <- res[4]
    # trait change
    d1 <- E[i - 1] - x[i - 1, 1]
    d2 <- E[i - 1] - x[i - 1, 2]
    d1.1 <- k * d1
    d2.1 <- k * d2
    x$x1[i] <- E[i - 1] - d1.1
    x$x2[i] <- E[i - 1] - d2.1
    # environmental change
    E[i] <- E[i - 1] + abs(rnorm(1, 0, 0.05))
}
```

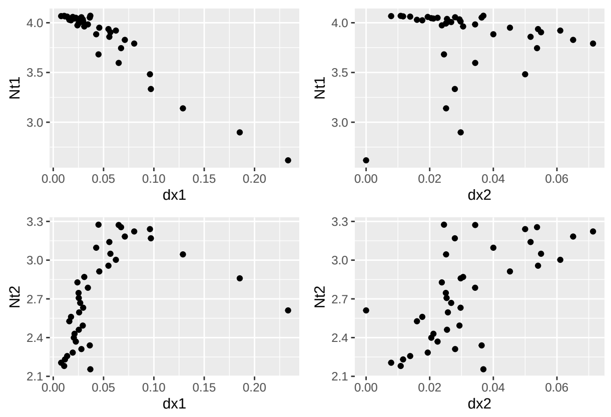
```
# Plot simulation: ggplot
N$time <- 1:t
dat <- melt(N, id.vars = "time")
ggplot2::ggplot(dat, aes(time, value, col = variable)) + geom_point()
```



The overall research goal is to determine the effect of trait evolution for species abundances. We hypothesize that $|x_{t+1} - x_{i,t}|$ is an important driver of species abundances, alongside the environment and species interactions. We plan to fit a statistical model to estimate the impact of E, E^2, and $|x_{t+1} - x_{i,t}|$ as fixed effects, and time points as a random effect (to estimate species-to-species associations). We first plot the effects of interest as scatter plots with species abundances.

```
dat2 <- as.data.frame(cbind(log(N$N1), log(N$N2), x$x1, x$x2))
colnames(dat2) <- c("N1", "N2", "x1", "x2")
dat2$time <- 1:t
df <- data.frame(cbind(dat2$N1[2:t], dat2$N2[2:t], dat2$x1[2:t], dat2$x2[2:t]))
colnames(df) <- c("Nt1", "Nt2", "xt1", "xt2")
df$dx1 <- abs(dat2$x1[2:t] - dat2$x1[1:(t - 1)])
df$dx2 <- abs(dat2$x2[2:t] - dat2$x2[1:(t - 1)])
# Plot
p1 <- ggplot2::ggplot(df, aes(dx1, Nt1)) + geom_point()
```

```r
p2 <- ggplot2::ggplot(df, aes(dx2, Nt1)) + geom_point()
p3 <- ggplot2::ggplot(df, aes(dx1, Nt2)) + geom_point()
p4 <- ggplot2::ggplot(df, aes(dx2, Nt2)) + geom_point()
p1 + p2 + p3 + p4
```



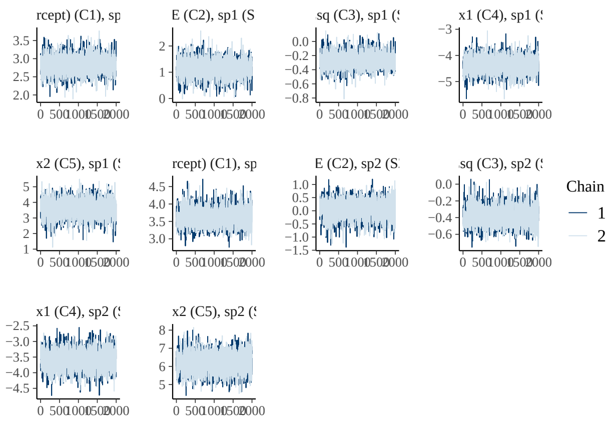
We expect that change in traits in Species 1 will have an effect on Species abundances, and we do not expect a strong impact of change in traits in Species 2.

We now use the species abundances as the response variable in the model, with E, E^2, and $|x_{t+1} - x_{i,t}|$ as fixed effects, and time points as a random effect (to estimate species-to-species associations). We use the absolute value of the change in trait value from one time point to the next (the total amount of trait change, regardless of direction) as a fixed effect in the HMSC.

```r
# Plot simulation: ggplot
dat <- as.data.frame(cbind(log(N$N1), log(N$N2), x$x1, x$x2))
colnames(dat) <- c("N1", "N2", "x1", "x2")
dat$time <- 1:t
df <- data.frame(cbind(dat$N1[2:t], dat$N2[2:t], dat$x1[2:t], dat$x2[2:t]))
colnames(df) <- c("Nt1", "Nt2", "xt1", "xt2")
df$dx1 <- abs(dat$x1[2:t] - dat$x1[1:(t - 1)])
df$dx2 <- abs(dat$x2[2:t] - dat$x2[1:(t - 1)])
# prepare data in HMSC format
Y <- as.matrix(cbind(df$Nt1, df$Nt2))
XData <- data.frame(cbind(E[1:(t - 1)], E[1:(t - 1)]^2, abs(dat$x1[2:t] - dat$x1[1:(t -
    1)]), abs(dat$x2[2:t] - dat$x2[1:(t - 1)])))
colnames(XData) <- c("E", "Esq", "dx1", "dx2")
# studyDesign = data.frame(sample = as.factor(1:(t-1))) rL =
# HmscRandomLevel(units = studyDesign$sample)
time1 <- 1:t
time1 <- as.data.frame(time1)
time1 <- subset(time1, time1 > 1)
rownames(XData) <- row.names(time1)
studyDesign = data.frame(sample = as.factor(1:(t - 1)), time = row.names(XData))
studyDesign$time <- as.factor(studyDesign$time)
studyDesign$sample <- as.factor(studyDesign$sample)
rL.time = HmscRandomLevel(sData = time1)
rL.sample = HmscRandomLevel(units = studyDesign$sample)
m.7.hmsc = Hmsc(Y = Y, XData = XData, XFormula = ~E + Esq + dx1 + dx2, studyDesign = studyDesign,
    ranLevels = list(sample = rL.sample, time = rL.time))
# Bayesian model parameters
nChains <- 2
```

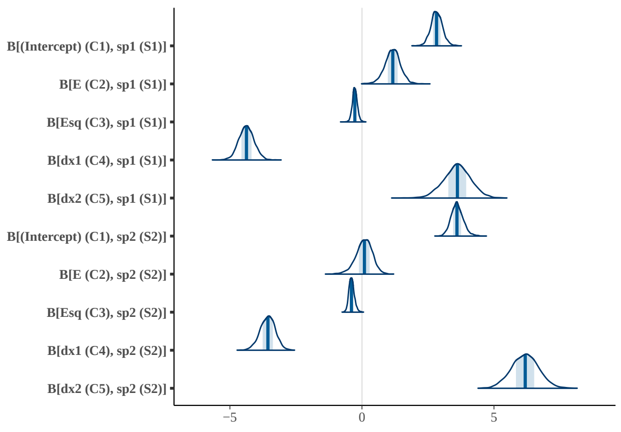
```
thin <- 5
samples <- 2000
transient <- 1000 * thin
verbose <- 500 * thin
# sample MCMC
m.7.sample <- sampleMcmc(m.7.hmsc, thin = thin, sample = samples, transient = transient,
    nChains = nChains, verbose = verbose)
#> Computing chain 1
#> Chain 1, iteration 2500 of 15000 (transient)
#> Chain 1, iteration 5000 of 15000 (transient)
#> Chain 1, iteration 7500 of 15000 (sampling)
#> Chain 1, iteration 10000 of 15000 (sampling)
#> Chain 1, iteration 12500 of 15000 (sampling)
#> Chain 1, iteration 15000 of 15000 (sampling)
#> Computing chain 2
#> Chain 2, iteration 2500 of 15000 (transient)
#> Chain 2, iteration 5000 of 15000 (transient)
#> Chain 2, iteration 7500 of 15000 (sampling)
#> Chain 2, iteration 10000 of 15000 (sampling)
#> Chain 2, iteration 12500 of 15000 (sampling)
#> Chain 2, iteration 15000 of 15000 (sampling)
```

```
m7.post.hmsc <- convertToCodaObject(m.7.sample)
summary(m7.post.hmsc$Beta)
#>
#> Iterations = 5005:15000
#> Thinning interval = 5
#> Number of chains = 2
#> Sample size per chain = 2000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>                                     Mean      SD Naive SE Time-series SE
#> B[(Intercept) (C1), sp1 (S1)]  2.82952 0.22295 0.003525       0.003531
#> B[E (C2), sp1 (S1)]            1.16050 0.29920 0.004731       0.004736
#> B[Esq (C3), sp1 (S1)]         -0.26855 0.09746 0.001541       0.001570
#> B[dx1 (C4), sp1 (S1)]         -4.37250 0.29079 0.004598       0.004622
#> B[dx2 (C5), sp1 (S1)]          3.60491 0.51929 0.008211       0.008212
#> B[(Intercept) (C1), sp2 (S2)]  3.60931 0.23452 0.003708       0.003625
#> B[E (C2), sp2 (S2)]            0.07662 0.31780 0.005025       0.004926
#> B[Esq (C3), sp2 (S2)]         -0.39100 0.10392 0.001643       0.001615
#> B[dx1 (C4), sp2 (S2)]         -3.57065 0.29059 0.004595       0.004594
#> B[dx2 (C5), sp2 (S2)]          6.18095 0.51606 0.008160       0.008301
#>
#> 2. Quantiles for each variable:
#>
#>                                   2.5%     25%      50%     75%    97.5%
#> B[(Intercept) (C1), sp1 (S1)]  2.4052  2.6894  2.82388  2.9750  3.27969
#> B[E (C2), sp1 (S1)]            0.5476  0.9777  1.16739  1.3505  1.73079
#> B[Esq (C3), sp1 (S1)]         -0.4549 -0.3291 -0.27087 -0.2084 -0.07167
#> B[dx1 (C4), sp1 (S1)]         -4.9336 -4.5647 -4.37357 -4.1820 -3.79713
#> B[dx2 (C5), sp1 (S1)]          2.5735  3.2699  3.61349  3.9462  4.60563
#> B[(Intercept) (C1), sp2 (S2)]  3.1757  3.4533  3.59716  3.7522  4.11500
#> B[E (C2), sp2 (S2)]           -0.6187 -0.1135  0.09333  0.2910  0.65173
#> B[Esq (C3), sp2 (S2)]         -0.5761 -0.4621 -0.39769 -0.3298 -0.16769
#> B[dx1 (C4), sp2 (S2)]         -4.1619 -3.7611 -3.56081 -3.3731 -3.02471
#> B[dx2 (C5), sp2 (S2)]          5.1784  5.8304  6.18246  6.5213  7.20052
bayesplot::mcmc_trace(m7.post.hmsc$Beta)
```



185

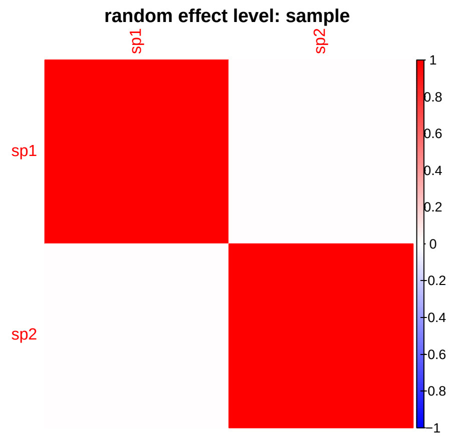
```
bayesplot::mcmc_areas(m7.post.hmsc$Beta, area_method = c("equal height"))
```



186

```
# Bayesian estimates
cbind(summary(m7.post.hmsc$Beta)$statistics[1:10, 1], summary(m7.post.hmsc$Beta)$quantiles[,
    c(1, 5)])
#>                                                 2.5%       97.5%
#> B[(Intercept) (C1), sp1 (S1)]  2.82951513  2.4052104  3.27969165
#> B[E (C2), sp1 (S1)]            1.16050438  0.5475719  1.73078575
#> B[Esq (C3), sp1 (S1)]         -0.26854625 -0.4549479 -0.07166688
#> B[dx1 (C4), sp1 (S1)]         -4.37250242 -4.9336430 -3.79713481
#> B[dx2 (C5), sp1 (S1)]          3.60491487  2.5735215  4.60563115
#> B[(Intercept) (C1), sp2 (S2)]  3.60931462  3.1756553  4.11499581
#> B[E (C2), sp2 (S2)]            0.07662271 -0.6187171  0.65172628
#> B[Esq (C3), sp2 (S2)]         -0.39099731 -0.5760972 -0.16768845
#> B[dx1 (C4), sp2 (S2)]         -3.57065242 -4.1618599 -3.02470978
#> B[dx2 (C5), sp2 (S2)]          6.18095292  5.1783528  7.20051840
```

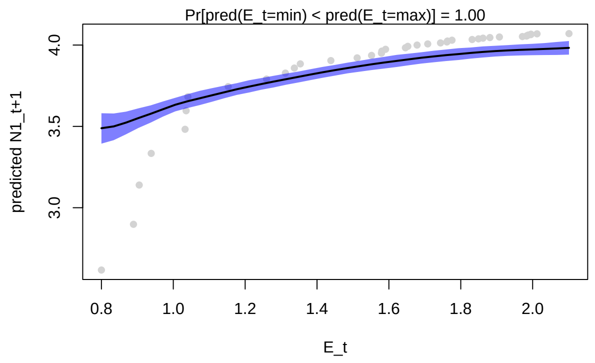
```
OmegaCor = computeAssociations(m.7.sample)
supportLevel = 0.95
toPlot = ((OmegaCor[[1]]$support > supportLevel) + (OmegaCor[[1]]$support < (1 -
    supportLevel)) > 0) * OmegaCor[[1]]$mean
corrplot::corrplot(toPlot, method = "color", col = colorRampPalette(c("blue", "white",
    "red"))(200), title = paste("random effect level:", m.7.sample$rLNames[1]), mar = c(0,
    0, 1, 0))
```



187

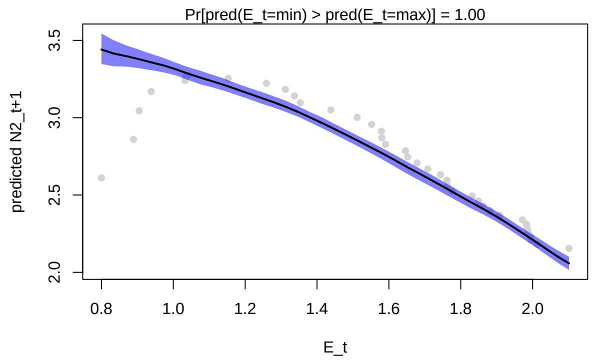
```
OmegaCor
#> [[1]]
#> [[1]]$mean
#>             sp1        sp2
#> sp1 1.00000000 0.07393031
#> sp2 0.07393031 1.00000000
#>
#> [[1]]$support
#>         sp1     sp2
#> sp1 1.00000 0.54775
#> sp2 0.54775 1.00000
#>
#>
#> [[2]]
#> [[2]]$mean
#>             sp1        sp2
#> sp1 1.00000000 0.02072506
#> sp2 0.02072506 1.00000000
#>
#> [[2]]$support
#>         sp1     sp2
#> sp1 1.00000 0.50975
#> sp2 0.50975 1.00000
```

```
Gradient <- constructGradient(m.7.sample, focalVariable = "E", non.focalVariables = list(Esq = list(2),
    dx1 = list(1), dx2 = list(1)), ngrid = 39)
# Esq is manually constructed as gradient-produced E^2
Gradient$XDataNew$Esq <- Gradient$XDataNew$E^2
predY <- predict(m.7.sample, XData = Gradient$XDataNew, expected = TRUE)
a <- plotGradient(m.7.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 1, main = "", xlab = "E_t", ylab = "predicted N1_t+1")
```



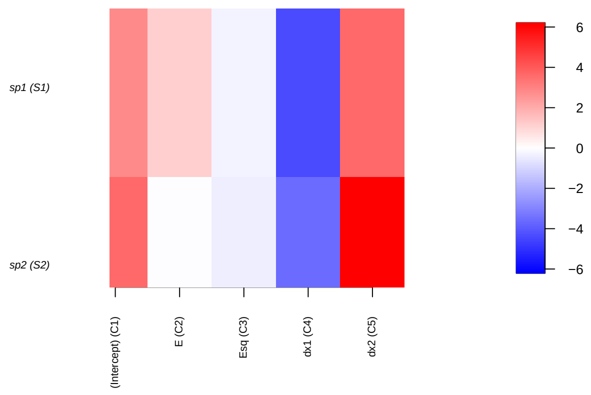
188

```
b <- plotGradient(m.7.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 2, main = "", xlab = "E_t", ylab = "predicted N2_t+1")
```



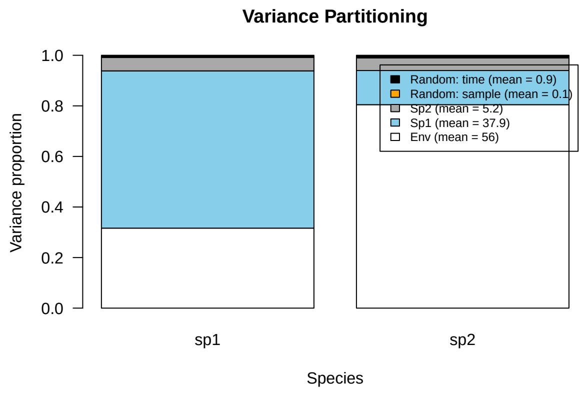
189

```
postBeta = getPostEstimate(m.7.sample, parName = "Beta")
plotBeta(m.7.sample, post = postBeta, param = "Mean", supportLevel = 0.95)
```



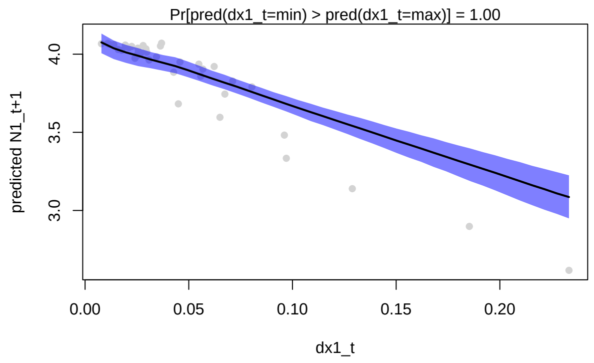
190

```r
VP <- computeVariancePartitioning(m.7.sample, group = c(1, 1, 1, 2, 3), groupnames = c("Env",
    "Sp1", "Sp2"))
plotVariancePartitioning(m.7.sample, VP, cols = c("white", "skyblue", "darkgrey",
    "orange", "black"), args.legend = list(cex = 0.75, bg = "transparent"))
```

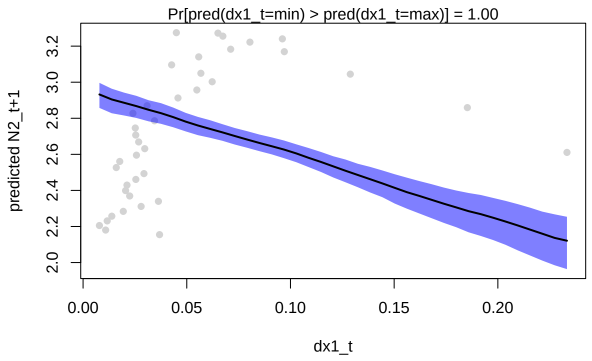


We can also look at a gradient plot for the effect of evolution in Species 1 for abundance in Species 1 and Species 2.

```r
Gradient <- constructGradient(m.7.sample, focalVariable = "dx1", non.focalVariables = list(E = list(1),
    Esq = list(2), dx2 = list(1)), ngrid = 39)
Gradient$XDataNew$Esq <- Gradient$XDataNew$E^2
predY <- predict(m.7.sample, XData = Gradient$XDataNew, expected = TRUE)
a <- plotGradient(m.7.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 1, main = "", xlab = "dx1_t", ylab = "predicted N1_t+1")
```

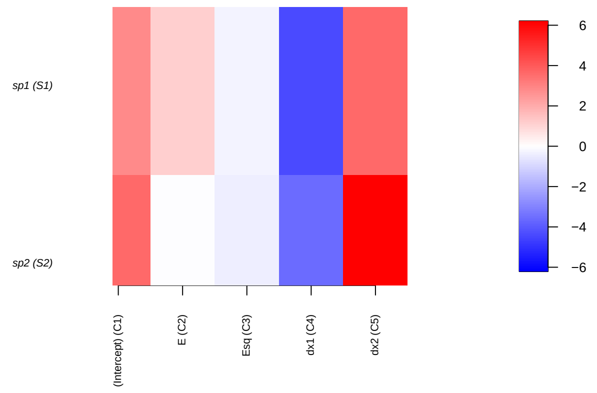


```r
b <- plotGradient(m.7.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 2, main = "", xlab = "dx1_t", ylab = "predicted N2_t+1")
```



194

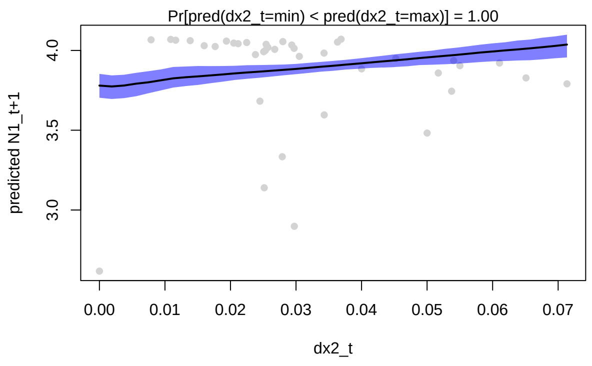
```r
postBeta = getPostEstimate(m.7.sample, parName = "Beta")
plotBeta(m.7.sample, post = postBeta, param = "Mean", supportLevel = 0.95)
```



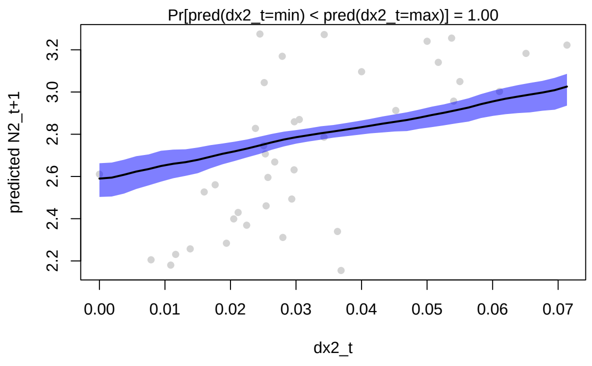
195

196 We repeat this plot for the effect of evolution in Species 2 for abundance in Species 1 and Species 2.

```r
Gradient <- constructGradient(m.7.sample, focalVariable = "dx2", non.focalVariables = list(E = list(1),
    Esq = list(2), dx1 = list(1)), ngrid = 39)
Gradient$XDataNew$Esq <- Gradient$XDataNew$E^2
predY <- predict(m.7.sample, XData = Gradient$XDataNew, expected = TRUE)
a <- plotGradient(m.7.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 1, main = "", xlab = "dx2_t", ylab = "predicted N1_t+1")
```

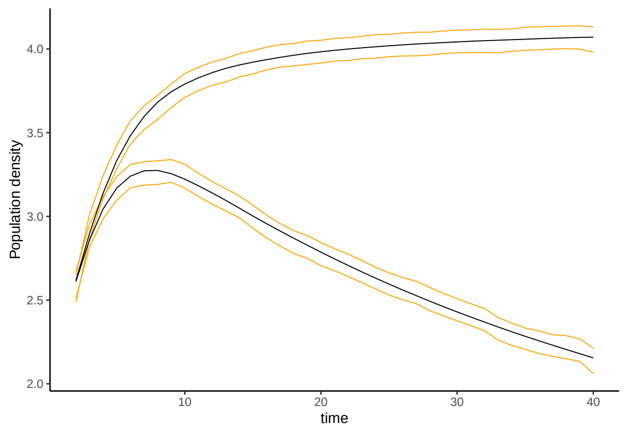


```r
b <- plotGradient(m.7.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 2, main = "", xlab = "dx2_t", ylab = "predicted N2_t+1")
```



The analysis indicates that we can successfully use the HMSC analysis to partition drivers of species abundances that include evolving traits. We see that change in Species 1 trait is a strong driver of abundances in Species1 and 2, and that trait change in Species 2 has a weaker impact. This aligns well with the observed simulation results.

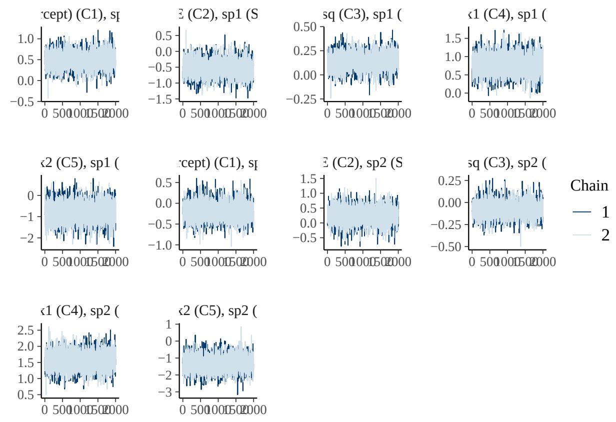
Though some of the main effect coefficients aren't well reflected in the data, we recall the model has strong predictive power when the elements are combined, with high predictive power to the observed data.



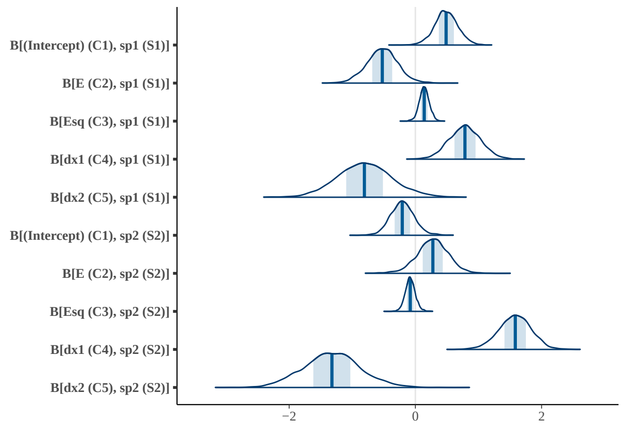
However, we can see the coefficients are less informative for inference. It is specific to this set of simulation conditions that the degree of evolution occurs at the beginning of the simulation, when the species is more mal-adapted. To better consider how trait evolution drives changes in population size, we use this change as the response variable Y in the predictive model.

```
# Plot simulation: ggplot
dat <- as.data.frame(cbind(log(N$N1), log(N$N2), x$x1, x$x2))
colnames(dat) <- c("N1", "N2", "x1", "x2")
dat$time <- 1:t
df <- data.frame(cbind(dat$N1[2:t], dat$N2[2:t], dat$x1[2:t], dat$x2[2:t]))
colnames(df) <- c("Nt1", "Nt2", "xt1", "xt2")
df$dN1 <- abs(dat$N1[2:t] - dat$N1[1:(t - 1)])
df$dN2 <- abs(dat$N2[2:t] - dat$N2[1:(t - 1)])
df$dx1 <- abs(dat$x1[2:t] - dat$x1[1:(t - 1)])
df$dx2 <- abs(dat$x2[2:t] - dat$x2[1:(t - 1)])
# prepare data in HMSC format
Y <- as.matrix(cbind(df$dN1, df$dN2))
XData <- data.frame(cbind(E[1:(t - 1)], E[1:(t - 1)]^2, abs(dat$x1[2:t] - dat$x1[1:(t -
    1)]), abs(dat$x2[2:t] - dat$x2[1:(t - 1)])))
colnames(XData) <- c("E", "Esq", "dx1", "dx2")
# studyDesign = data.frame(sample = as.factor(1:(t-1))) rL =
# HmscRandomLevel(units = studyDesign$sample)

time1 <- 1:t
time1 <- as.data.frame(time1)
time1 <- subset(time1, time1 > 1)
rownames(XData) <- row.names(time1)
studyDesign = data.frame(sample = as.factor(1:(t - 1)), time = row.names(XData))
studyDesign$time <- as.factor(studyDesign$time)
studyDesign$sample <- as.factor(studyDesign$sample)
rL.time = HmscRandomLevel(sData = time1)
rL.sample = HmscRandomLevel(units = studyDesign$sample)


m.8.hmsc = Hmsc(Y = Y, XData = XData, XFormula = ~E + Esq + dx1 + dx2, studyDesign = studyDesign,
    ranLevels = list(sample = rL.sample, time = rL.time))
# Bayesian model parameters
nChains <- 2
thin <- 5
samples <- 2000
transient <- 1000 * thin
verbose <- 500 * thin
# sample MCMC
m.8.sample <- sampleMcmc(m.8.hmsc, thin = thin, sample = samples, transient = transient,
    nChains = nChains, verbose = verbose)
#> Computing chain 1
#> Chain 1, iteration 2500 of 15000 (transient)
#> Chain 1, iteration 5000 of 15000 (transient)
#> Chain 1, iteration 7500 of 15000 (sampling)
#> Chain 1, iteration 10000 of 15000 (sampling)
#> Chain 1, iteration 12500 of 15000 (sampling)
#> Chain 1, iteration 15000 of 15000 (sampling)
#> Computing chain 2
#> Chain 2, iteration 2500 of 15000 (transient)
#> Chain 2, iteration 5000 of 15000 (transient)
#> Chain 2, iteration 7500 of 15000 (sampling)
#> Chain 2, iteration 10000 of 15000 (sampling)
#> Chain 2, iteration 12500 of 15000 (sampling)
#> Chain 2, iteration 15000 of 15000 (sampling)
```

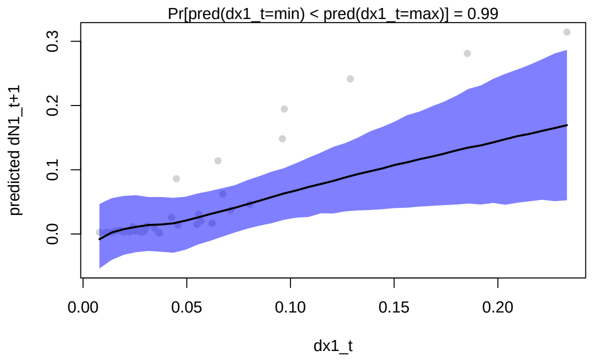
```
m8.post.hmsc <- convertToCodaObject(m.8.sample)
summary(m8.post.hmsc$Beta)
#>
#> Iterations = 5005:15000
#> Thinning interval = 5
#> Number of chains = 2
#> Sample size per chain = 2000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>                                    Mean      SD Naive SE Time-series SE
#> B[(Intercept) (C1), sp1 (S1)]  0.4896 0.18588 0.002939       0.002889
#> B[E (C2), sp1 (S1)]           -0.5249 0.24718 0.003908       0.003751
#> B[Esq (C3), sp1 (S1)]          0.1410 0.08003 0.001265       0.001212
#> B[dx1 (C4), sp1 (S1)]          0.7845 0.25194 0.003984       0.003781
#> B[dx2 (C5), sp1 (S1)]         -0.8001 0.43852 0.006934       0.007087
#> B[(Intercept) (C1), sp2 (S2)] -0.2032 0.19172 0.003031       0.002988
#> B[E (C2), sp2 (S2)]            0.2723 0.25382 0.004013       0.003895
#> B[Esq (C3), sp2 (S2)]         -0.0797 0.08196 0.001296       0.001261
#> B[dx1 (C4), sp2 (S2)]          1.5786 0.25859 0.004089       0.004089
#> B[dx2 (C5), sp2 (S2)]         -1.3206 0.45322 0.007166       0.007166
#>
#> 2. Quantiles for each variable:
#>
#>                                    2.5%      25%      50%      75%    97.5%
#> B[(Intercept) (C1), sp1 (S1)]  0.12136  0.37168  0.48712  0.60991  0.86014
#> B[E (C2), sp1 (S1)]           -1.00924 -0.68311 -0.52400 -0.36735 -0.03012
#> B[Esq (C3), sp1 (S1)]         -0.01934  0.09013  0.14021  0.19261  0.29733
#> B[dx1 (C4), sp1 (S1)]          0.29357  0.61966  0.78537  0.95353  1.27985
#> B[dx2 (C5), sp1 (S1)]         -1.64711 -1.09531 -0.80766 -0.51393  0.09131
#> B[(Intercept) (C1), sp2 (S2)] -0.56263 -0.32862 -0.20757 -0.08404  0.18675
#> B[E (C2), sp2 (S2)]           -0.24856  0.11632  0.27705  0.43391  0.75956
#> B[Esq (C3), sp2 (S2)]         -0.23797 -0.13289 -0.08153 -0.02915  0.08842
#> B[dx1 (C4), sp2 (S2)]          1.06226  1.41189  1.58305  1.75076  2.06433
#> B[dx2 (C5), sp2 (S2)]         -2.19648 -1.61649 -1.32174 -1.03020 -0.41482
bayesplot::mcmc_trace(m8.post.hmsc$Beta)
```



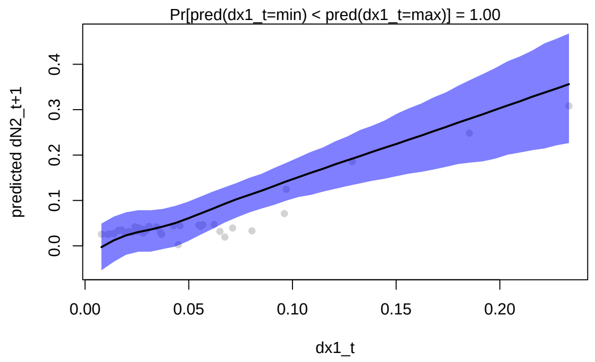
```
bayesplot::mcmc_areas(m8.post.hmsc$Beta, area_method = c("equal height"))
```



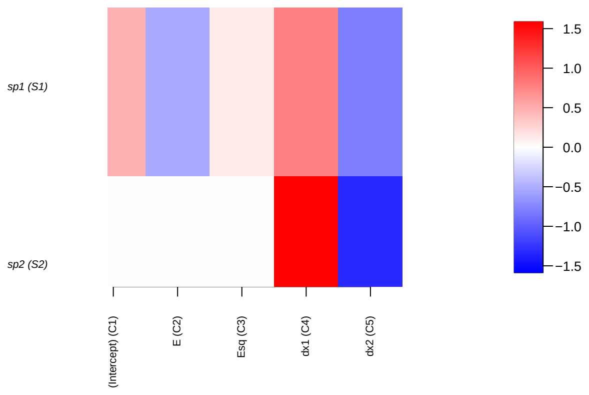
```
Gradient <- constructGradient(m.8.sample, focalVariable = "dx1", non.focalVariables = list(E = list(1),
    Esq = list(2), dx2 = list(1)), ngrid = 39)
Gradient$XDataNew$Esq <- Gradient$XDataNew$E^2
predY <- predict(m.8.sample, XData = Gradient$XDataNew, expected = TRUE)
a <- plotGradient(m.8.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 1, main = "", xlab = "dx1_t", ylab = "predicted dN1_t+1")
```



```
b <- plotGradient(m.8.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 2, main = "", xlab = "dx1_t", ylab = "predicted dN2_t+1")
```

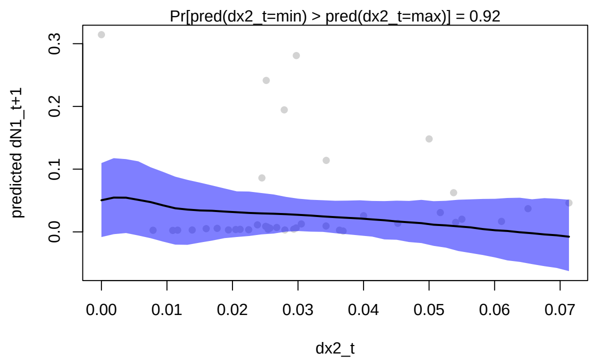


```
postBeta = getPostEstimate(m.8.sample, parName = "Beta")
plotBeta(m.8.sample, post = postBeta, param = "Mean", supportLevel = 0.95)
```

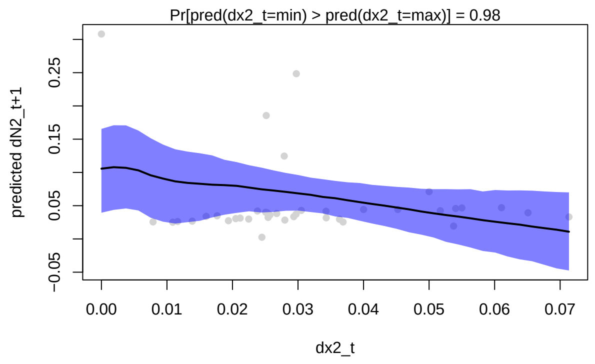


We repeat this plot for the effect of evolution in Species 2 for abundance in Species 1 and Species 2.

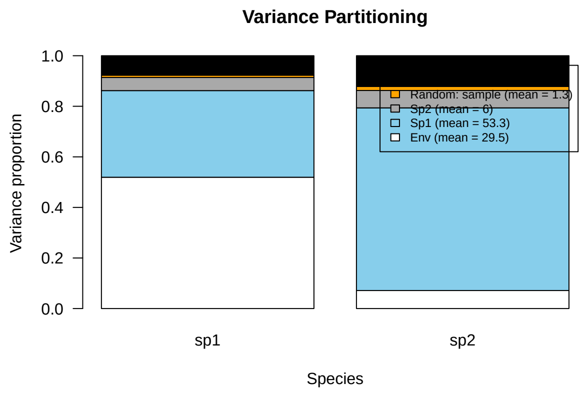
```
Gradient <- constructGradient(m.8.sample, focalVariable = "dx2", non.focalVariables = list(E = list(1),
    Esq = list(2), dx1 = list(1)), ngrid = 39)
Gradient$XDataNew$Esq <- Gradient$XDataNew$E^2
predY <- predict(m.8.sample, XData = Gradient$XDataNew, expected = TRUE)
a <- plotGradient(m.8.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 1, main = "", xlab = "dx2_t", ylab = "predicted dN1_t+1")
```



```
b <- plotGradient(m.8.sample, Gradient, pred = predY, showData = T, measure = "Y",
    index = 2, main = "", xlab = "dx2_t", ylab = "predicted dN2_t+1")
```



```
VP <- computeVariancePartitioning(m.8.sample, group = c(1, 1, 1, 2, 3), groupnames = c("Env",
    "Sp1", "Sp2"))
plotVariancePartitioning(m.8.sample, VP, cols = c("white", "skyblue", "darkgrey",
    "orange", "black"), args.legend = list(cex = 0.75, bg = "transparent"))
```



Here we can see that trait evolution in Species 1 had a strong positive relationship with the *change* in population size for both species. Trait evolution in Species 2 had much less of an effect.

```r
knitr::knit_exit()
```

Beverton, R. J., and S. J. Holt. 1957. On the dynamics of exploited fish populations (Vol. 11). Springer Science & Business Media.

Certain, G., F. Barraquand, and A. Gårdmark. 2018. How do MAR(1) models cope with hidden nonlinearities in ecological dynamics? Methods in Ecology and Evolution 9:1975–1995.

Erickson, K. D., and A. B. Smith. 2023. Modeling the rarest of the rare: A comparison between multi-species distribution models, ensembles of small models, and single-species models at extremely low sample sizes. Ecography 2023:e06500.

Gomulkiewicz, R., and R. D. Holt. 1995. When does evolution by natural selection prevent extinction? Evolution 49:201–207.

Hart, S. P., and D. J. Marshall. 2013. Environmental stress, facilitation, competition, and coexistence. Ecology 94:2719–2731.

Ingram, M., D. Vukcevic, and N. Golding. 2020. Multi-output gaussian processes for species distribution modelling. Methods in Ecology and Evolution 11:1587–1598.

Ives, A. R. 1995. Predicting the response of populations to environmental change. Ecology 76:926–941.

Kloppers, P. H., and J. C. Greeff. 2013. Lotka–volterra model parameter estimation using experiential data. Applied Mathematics and Computation 224:817–825.

Mühlbauer, L. K., M. Schulze, W. S. Harpole, and A. T. Clark. 2020. gauseR: Simple methods for fitting lotka-volterra models describing gause's "struggle for existence". Ecology and Evolution 10:13275–13283.

Olivença, D. V., J. D. Davis, and E. O. Voit. 2021. Comparison between lotka-volterra and multivariate autoregressive models of ecological interaction systems. bioRxiv.

Ovaskainen, O., and N. Abrego. 2020. Joint species distribution modelling: With applications in r. Ecology biodiversity and conservation. Cambridge University Press, United Kingdom.

Ovaskainen, O., G. Tikhonov, D. Dunson, V. Grøtan, S. Engen, B.-E. Sæther, and N. Abrego. 2017. How are species interactions structured in species-rich communities? A new method for analysing time-series data. Proceedings of the Royal Society B: Biological Sciences 284:20170768.