

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

Jelena H. Pantel\*      Ruben J. Hermann†

23 July, 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  $N_{i,0} \sim \text{Pois}(10)$ , 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:

```
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](mailto:jelena.pantel@univ-fcomte.fr)

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

```

    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)
}

```

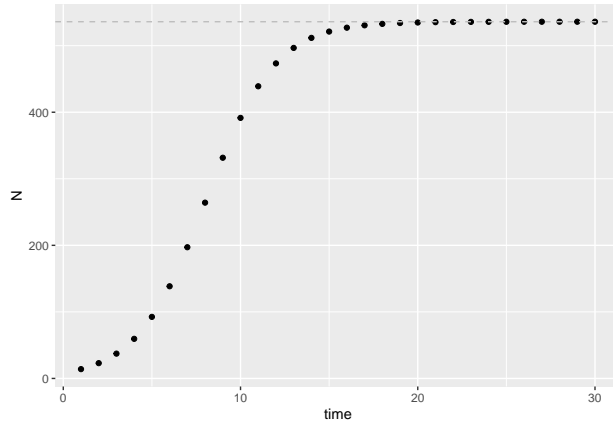


Figure 1: Population size  $N$  over time  $t$  for a discrete-time logistic growth model, with parameters  $r_i = 1.67$ ,  $N_{1,0} = 14$ , and  $\alpha_{11} = 0.00125$ .

## 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).

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

```

# 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

```

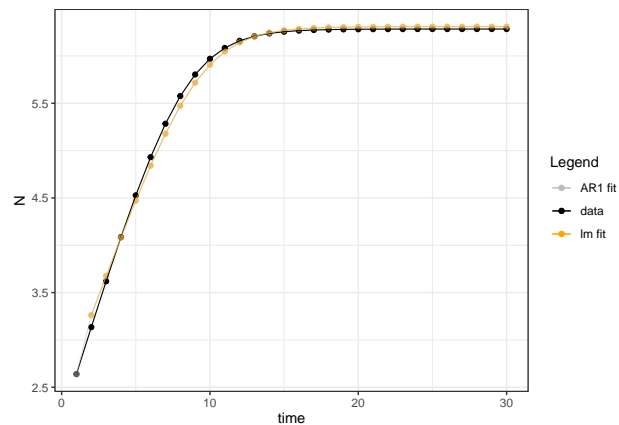


Figure 2: Population size over time (black line) with fitted values from a first-order autoregressive model (red dashed line).

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

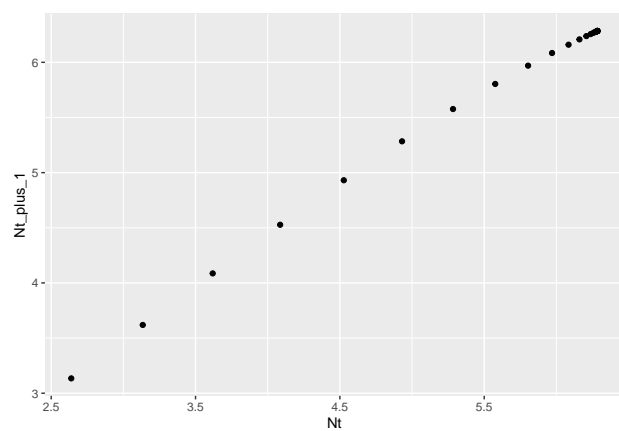


Figure 3: Population size (logarithm) at one time step  $N_{t+1}$  as a function of log-population size in the previous time step  $N_t$ .

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

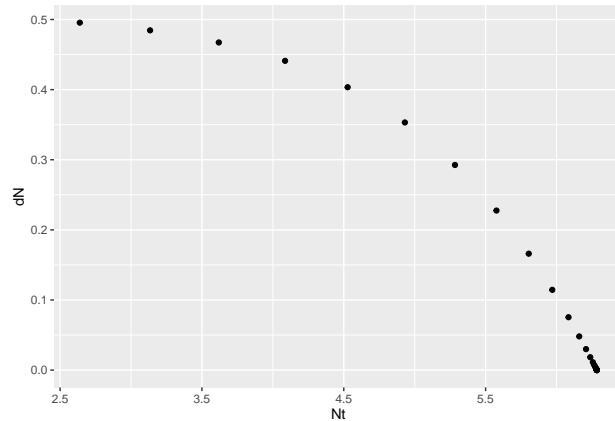


Figure 4: Change in population size from one time step to the next  $N_{t+1}$  as a function of  $N_t$

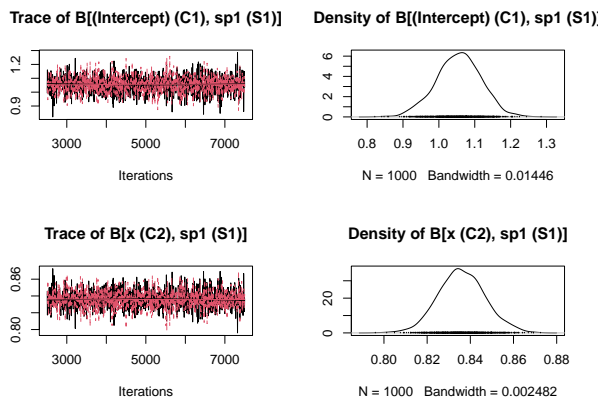
### 1.3 Bayesian linear statistical model: HMSC

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)
```



22

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

```
# AR1 coefficients (recall that the intercept is the term below multiplied 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")
```

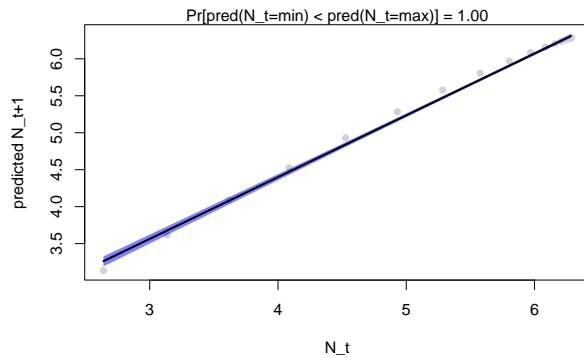
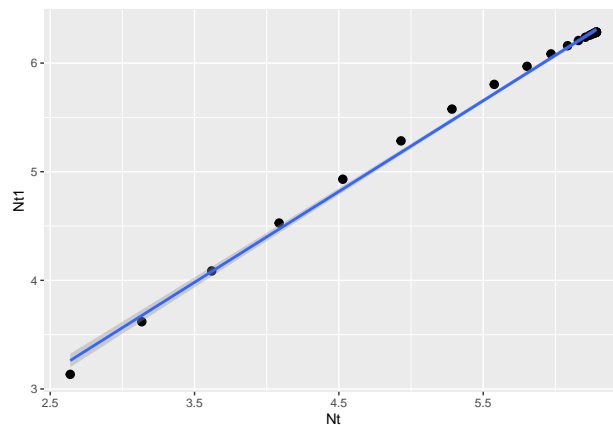


Figure 5: Observed (grey) and model-fit (blue) values for population size at time  $t$  (x-axis) and  $t+1$  (y-axis).

```
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  $\alpha_{ii}$ . 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.

```
# 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)
}
```

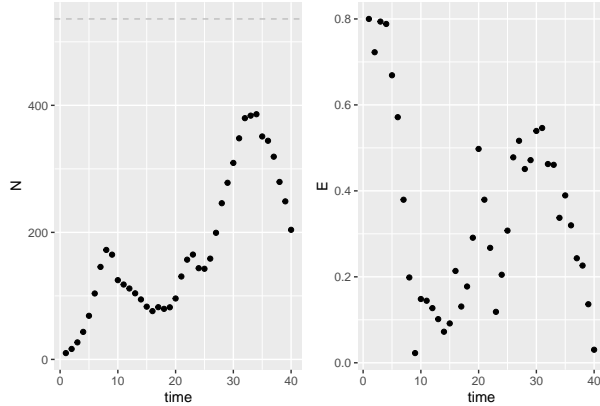


Figure 6: Population size  $N$  over time  $t$  for a discrete-time logistic growth model, with parameters  $r_i = 1.67$ ,  $N_{1,0} = 14$ , and  $\alpha_{11} = 0.00125$ . Relationship between  $E$  and  $N_t$  is also shown.

## 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

```

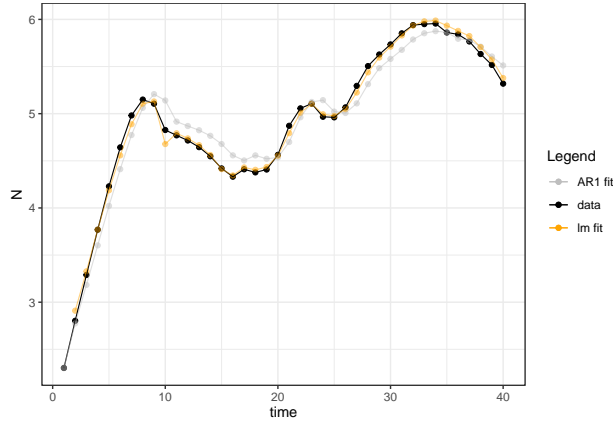


Figure 7: Population size over time (black line) with fitted values from a first-order autoregressive model (red dashed line).

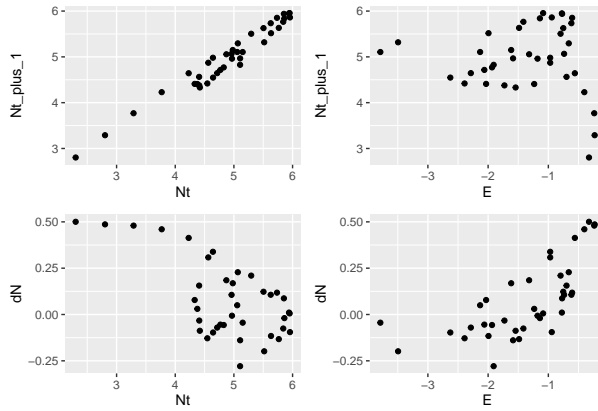


Figure 8: Population size (logarithm) at one time step  $N_{t+1}$  as a function of log-population size in the previous time step  $N_t$ .

40 The linear model is a good fit when including the environmental covariate.  $N_{t+1}$  and  $N_t$  can still be captured  
 41 by a linear relationship. However we see that the relationship between  $N_{t+1}$  and  $E_t$  is non-linear. This tells  
 42 us that the lm is good for predictions, but not for inference (for capturing well the relationship between  
 43 the predictor and response variable). The use of linear relationships in JSDMs is discussed in (Ingram et  
 44 al. 2020), and in many applications (e.g. (Erickson and Smith 2023)) quadratic terms are used, which



45 create bell-shaped response curves that may better match species with optimal niches (as opposed to linear,  
 46 monotonically increasing relationships between population size and environmental predictors). We thus  
 47 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)
```

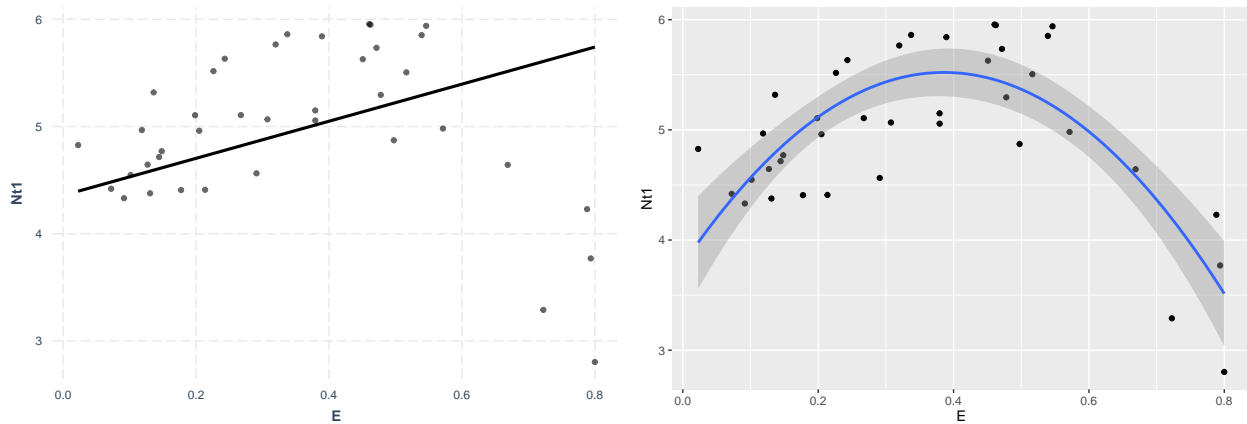


Figure 9: Population size (logarithm) at one time step  $N_{t+1}$  as a function of log-population size in the previous time step  $N_t$ .

## 48 2.3 Bayesian linear statistical model: HMSC

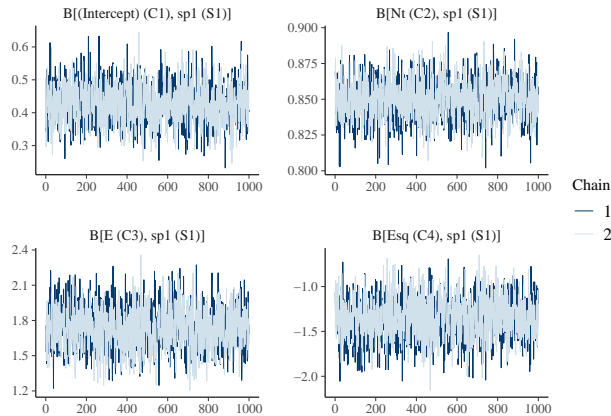
49 We can estimate the same model parameters using HMSC:

```
# 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)

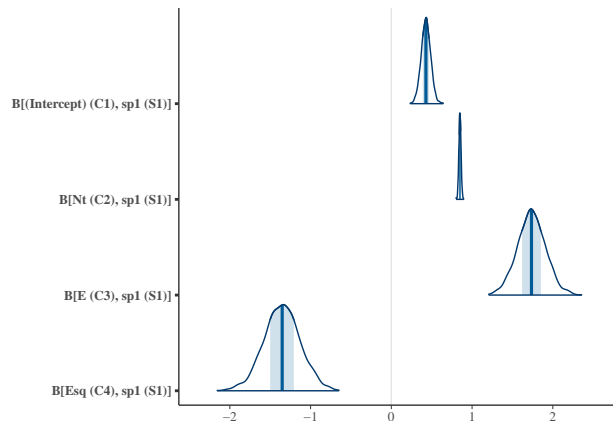
```



```

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

```



51

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

```
# AR1 coefficients (recall that the intercept is the term below multiplied 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
```

53 We recall that the interpretation of the coefficients in an arimaX (arima with covariates) model is difficult.  
 54 They do not give the impact on  $N_t$  per unit increase in X as in a regression. So we do not interpret the  
 55 causation implied by the coefficient in the arimaX model. In the regression model, we can see that  $E$  has a  
 56 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")
```

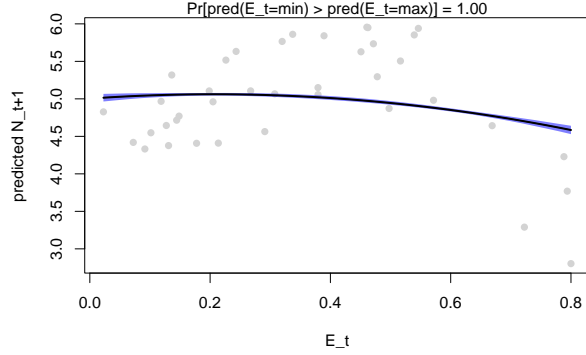


Figure 10: Observed (grey) and model-fit (blue) values for population size at time  $t$  (x-axis) and  $t+1$  (y-axis).

## 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  $\alpha_{ij}$ .

```
# 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]
}

```

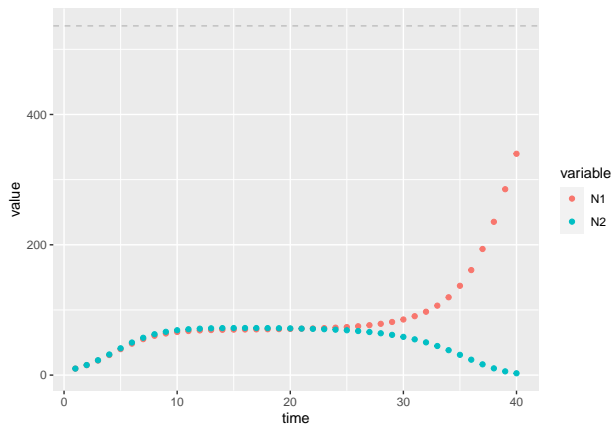


Figure 11: Population size  $N$  over time  $t$  for a discrete-time logistic growth model, with parameters  $r_1 = 1.67$ ,  $r_2 = 1.7$ ,  $N_{i,0} = 10$ ,  $\alpha_{ii} = 0.00125$ ,  $\alpha_{12} = 0.008$ , and  $\alpha_{21} = 0.008725$ .

### 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.

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

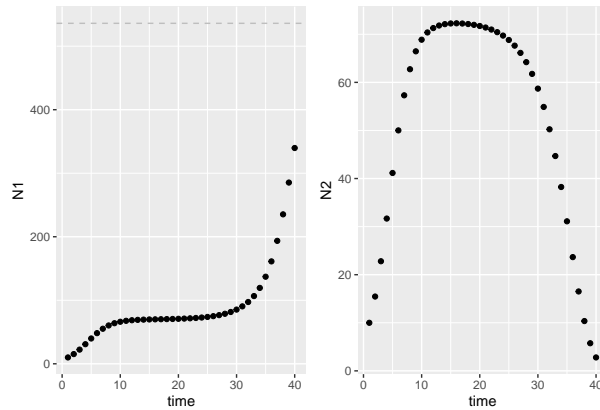


Figure 12: Population size  $N$  over time  $t$  for a discrete-time logistic growth model, with competition.

```
## 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
```

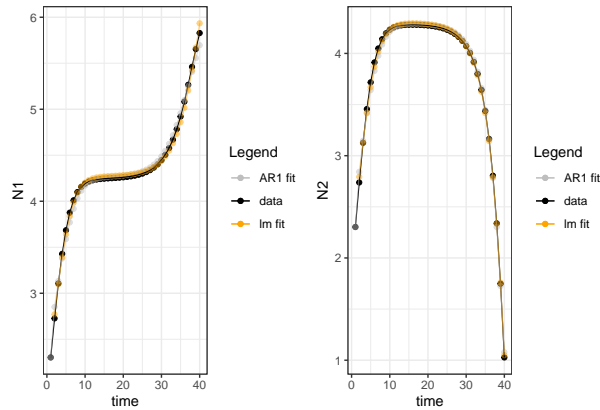


Figure 13: Population size over time (black line) with fitted values from a first-order autoregressive model (gray line) and from a linear model (orange line).

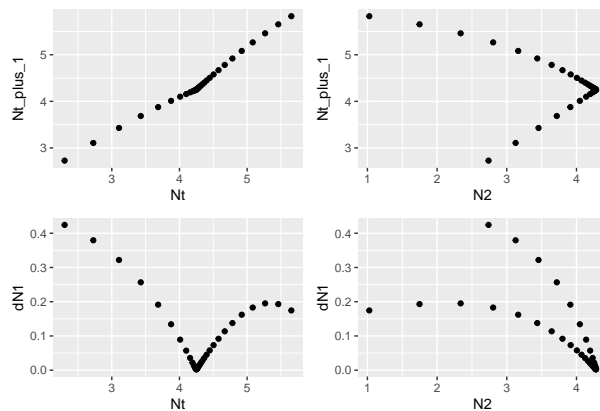


Figure 14: Population size (logarithm) at one time step  $N_{t+1}$  as a function of log-population size of Species 1 and Species 2 in the previous time step  $N_t$ . Also shown is the change in population size of Species 1 as a function of  $N_t$  for both species.

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

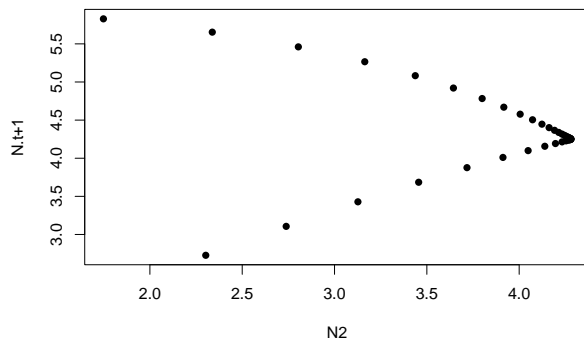
```
df <- data.frame(cbind(log(dat$N1[2:t]), log(dat$N1[1:(t - 1)]), log(dat$N2[1:(t - 1)])))
colnames(df) <- c("Nt1", "N1", "N2")
m.3.lm <- lm(Nt1 ~ I(1/N1) + I(1/N2), data = df)

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)

```



72

```

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

```

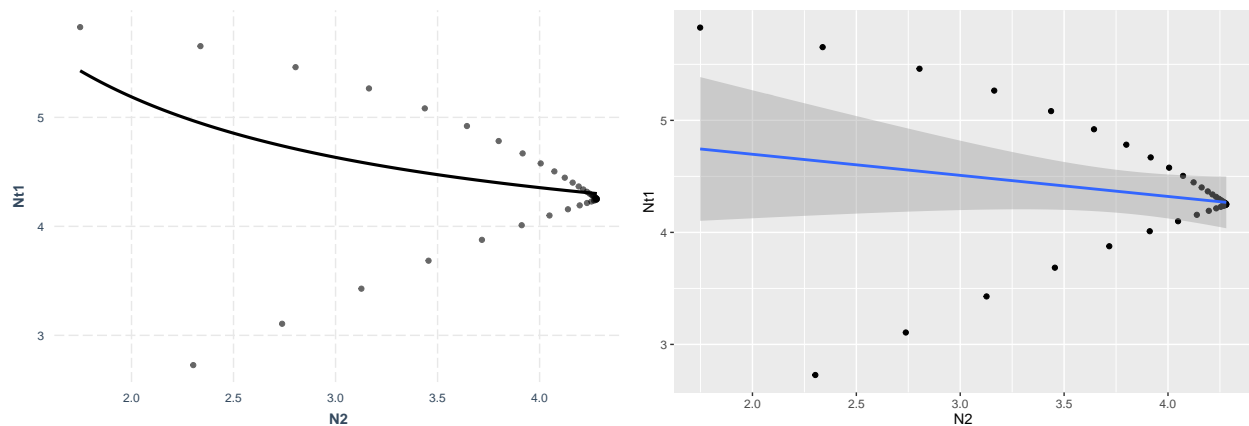


Figure 15: Population size (logarithm) at one time step  $N_{t+1}$  as a function of log-population size of the second species in the previous time step  $N_t$ . Fitted (line) and observed (points) values are shown.



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:

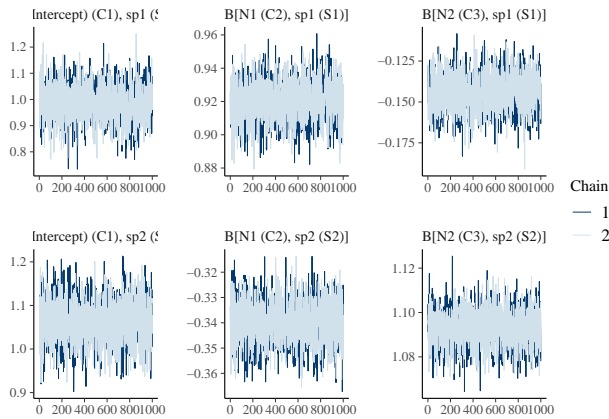
### 3.3 Bayesian linear statistical model: HMSC

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.

```
# 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)))
rL = HmscRandomLevel(units = studyDesign$sample)
m.3.hmsc = Hmsc(Y = Y, XData = XData, studyDesign = studyDesign, ranLevels = list(sample = rL))
# 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)

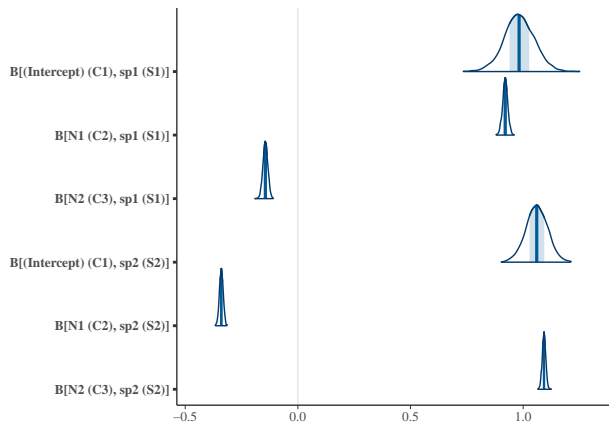
m3.post.hmsc <- convertToCodaObject(m.3.sample)
summary(m3.post.hmsc$Beta)
#>
#> Iterations = 2505:7500
#> Thinning interval = 5
#> Number of chains = 2
#> Sample size per chain = 1000
#>
```

```
#> 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)
```



93

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



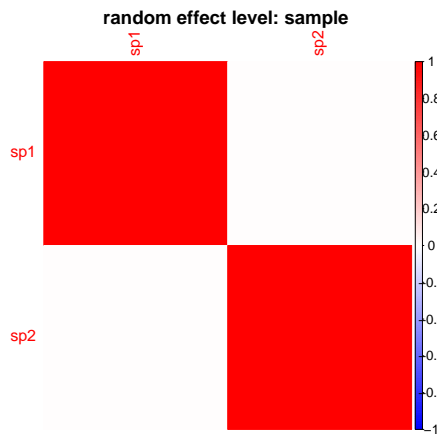
94

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

```

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))

```



The model fixed effect estimates match well with those from the AR1 and linear model. However, the model up to this point includes  $N_{t1}$  and  $N_{t2}$  as fixed predictors. I change that here, as it better matches what can be done in large multi-species models, and the residual associations will better match the effects species have on one another.

```

# AR1 coefficients (recall that the intercept is the term below multiplied 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

```

I try again with a next example where the only difference between species is the  $\alpha_{ij}$ .

```

# 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]
}

```

```

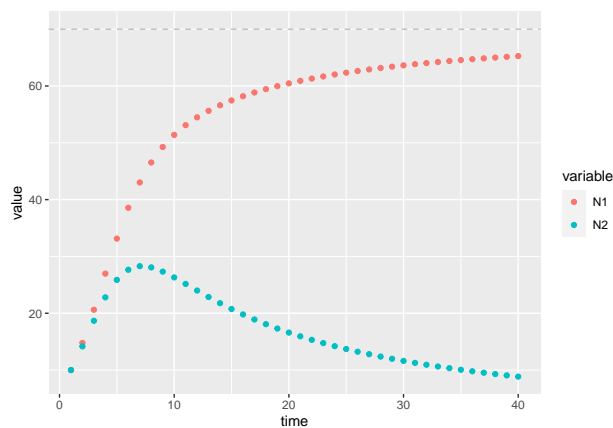
# Plot simulation: ggplot

```

```

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")

```



102

```

# 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")

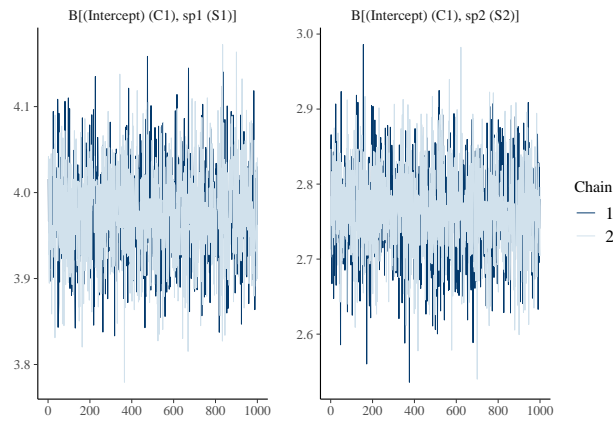
```

```

# 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)))
rL = 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)

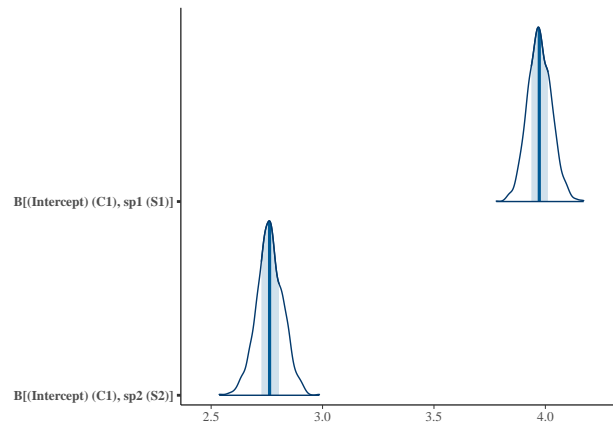
m4.post.hmsc <- convertToCodaObject(m.4.sample)
summary(m4.post.hmsc$Beta)
#>
#> Iterations = 2505:7500
#> Thinning interval = 5
#> Number of chains = 2
#> Sample size per chain = 1000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>
#>               Mean      SD Naive SE Time-series SE
#> B[(Intercept) (C1), sp1 (S1)] 3.974 0.05406 0.001209      0.001173
#> B[(Intercept) (C1), sp2 (S2)] 2.764 0.06002 0.001342      0.001388
#>
#> 2. Quantiles for each variable:
#>
#>
#>               2.5%   25%   50%   75% 97.5%
#> B[(Intercept) (C1), sp1 (S1)] 3.872 3.937 3.973 4.011 4.085
#> B[(Intercept) (C1), sp2 (S2)] 2.642 2.725 2.762 2.804 2.885
bayesplot::mcmc_trace(m4.post.hmsc$Beta)

```



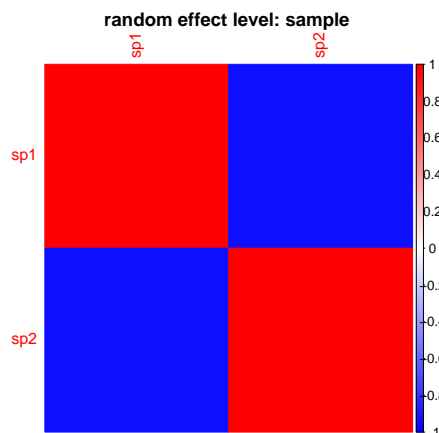
103

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



104

```
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))
```

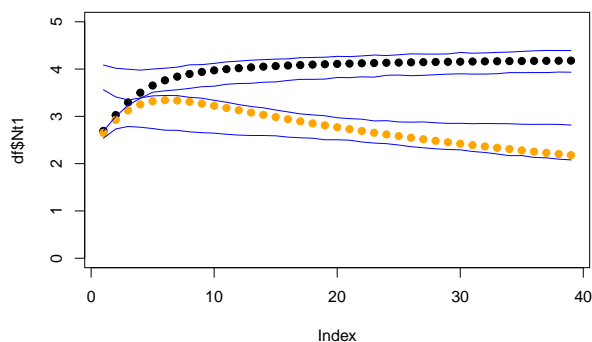


105

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

```
# 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')
```

```
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")
```



108

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

## 4 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  $x_2 = 0.8$ .

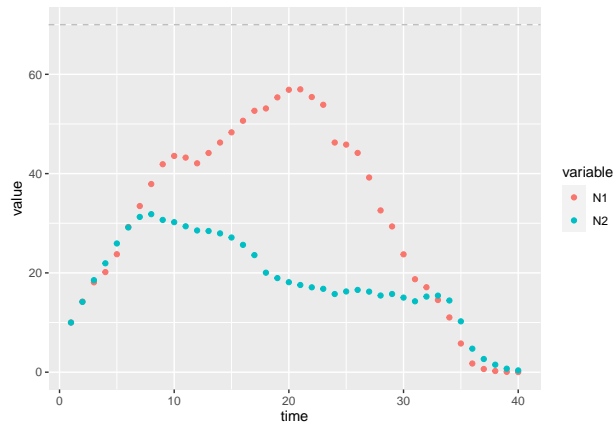
```
# Simulate initial species population growth with environment fluctuations
N1.0 <- 10
N2.0 <- 10
r1.0 <- 1.7
r2.0 <- 1.7
alpha.11 <- 0.01
alpha.22 <- 0.01
alpha.12 <- 0.005
alpha.21 <- 0.01
E.0 <- 0.8
x1.0 <- 0.6
x2.0 <- 0.8

# 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")
```



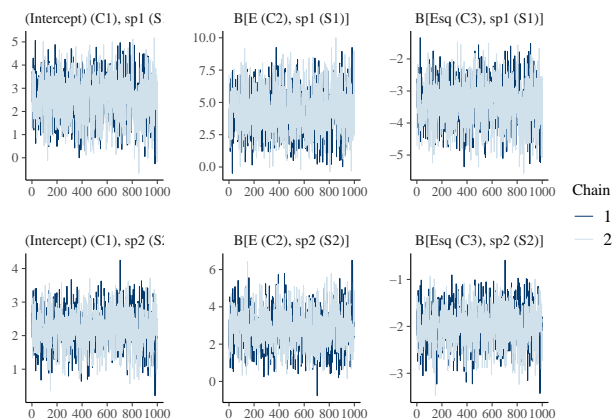


118

```
# 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)

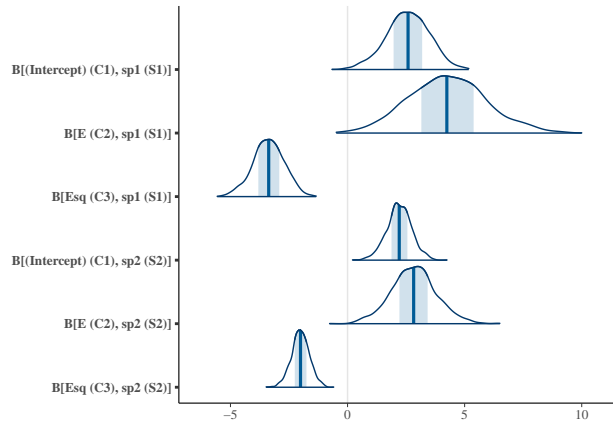
m5.post.hmsc <- convertToCodaObject(m.5.sample)
summary(m5.post.hmsc$Beta)
#>
#> Iterations = 2505:7500
#> Thinning interval = 5
#> Number of chains = 2
```

```
#> 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)
```



119

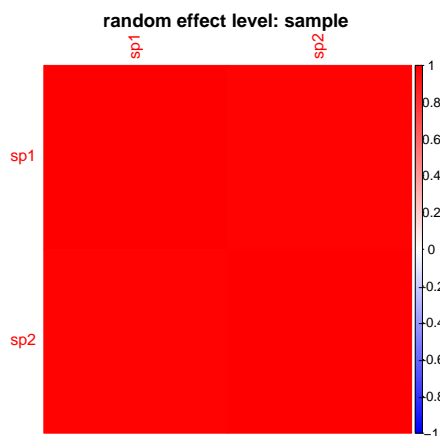
```
bayesplot::mcmc_areas(m5.post.hmsc$Beta, area_method = c("equal height"))
```



120

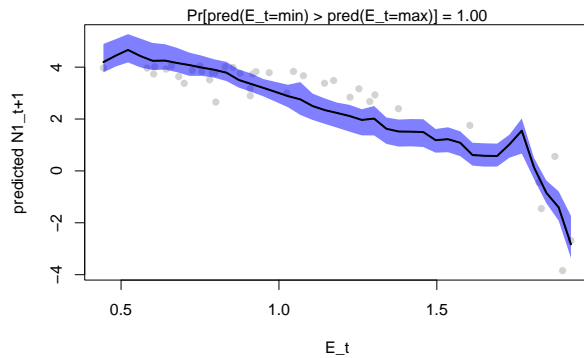
```
# Bayesian estimates
summary(m5.post.hmsc$Beta)$statistics[1:4, 1:2]
#>
#>           Mean      SD
#> B[(Intercept) (C1), sp1 (S1)] 2.571888 0.9130903
#> B[E (C2), sp1 (S1)]          4.274430 1.6630073
#> B[Esq (C3), sp1 (S1)]        -3.373885 0.6725570
#> B[(Intercept) (C1), sp2 (S2)] 2.203589 0.5330636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

```
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))
```



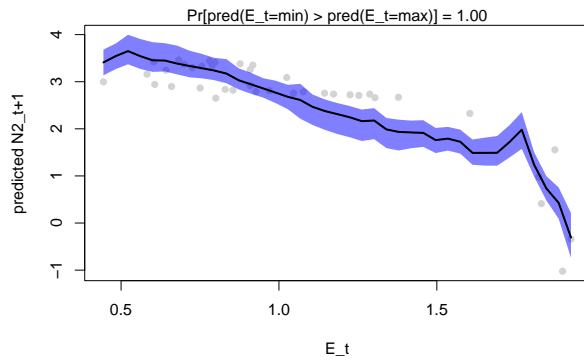
121

```
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")
```



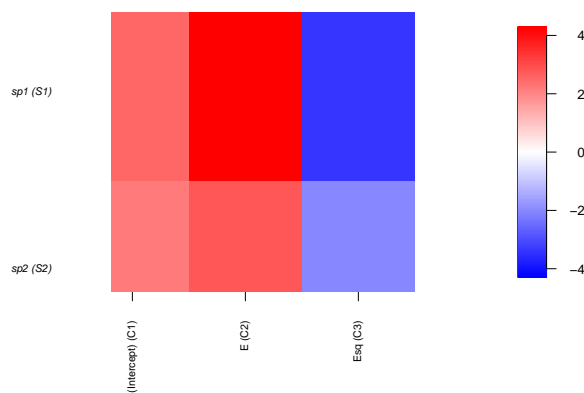
122

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



123

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



124

125 We can see that the Environment has an overall neg-  
 126 ative impact for both species as  $E$  drifts randomly towards more positive values (where Species 2 is closer  
 127 because it has a higher trait value). We also see that the residual species association effects are strongly  
 128 positive. We evaluate this by greatly increasing the effects if interspecific competition (while decreasing the  
 strength of environmental variation).

```

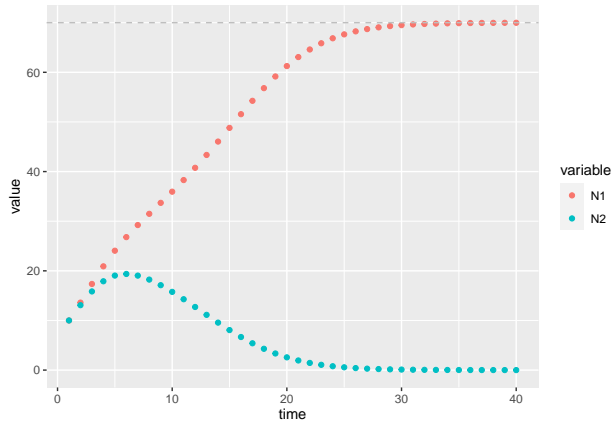
# Simulate initial species population growth with environment fluctuations
N1.0 <- 10
N2.0 <- 10
r1.0 <- 1.7
r2.0 <- 1.7
alpha.11 <- 0.01
alpha.22 <- 0.01
alpha.12 <- 0.015
alpha.21 <- 0.02
E.0 <- 0.8
x1.0 <- 0.8
x2.0 <- 0.8

# 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)
}

# 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")

```



129

```

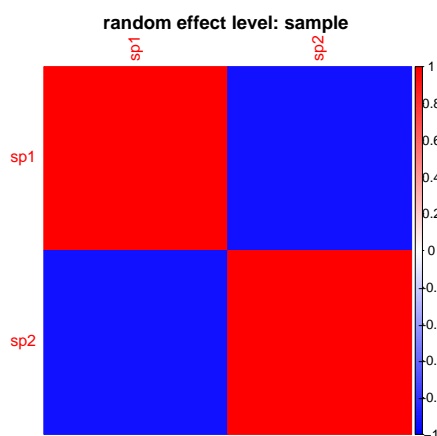
130 #> Computing chain 1
131 #> Chain 1, iteration 2500 of 10500 (sampling)
132 #> Chain 1, iteration 5000 of 10500 (sampling)
133 #> Chain 1, iteration 7500 of 10500 (sampling)
134 #> Chain 1, iteration 10000 of 10500 (sampling)
135 #> Computing chain 2
136 #> Chain 2, iteration 2500 of 10500 (sampling)
137 #> Chain 2, iteration 5000 of 10500 (sampling)
138 #> Chain 2, iteration 7500 of 10500 (sampling)
139 #> 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))

```



140

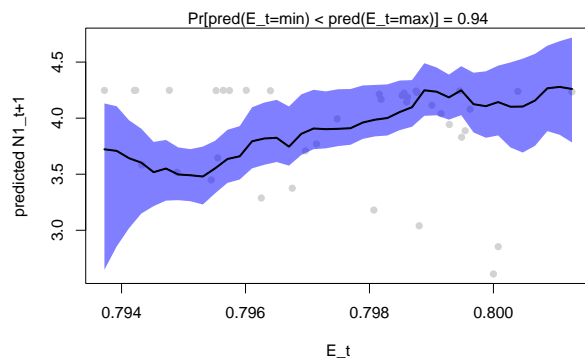
```

OmegaCor
#> [[1]]
#> [[1]]$mean
#>          sp1          sp2
#> sp1  1.000000 -0.9201566

```

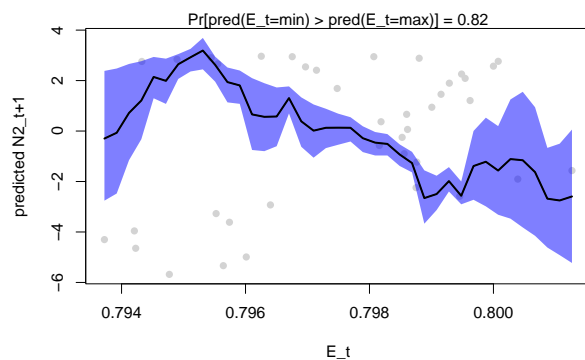
```
#> sp2 -0.9201566  1.0000000
#>
#> [[1]]$support
#>      sp1 sp2
#> sp1    1  0
#> sp2    0  1
```

```
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")
```



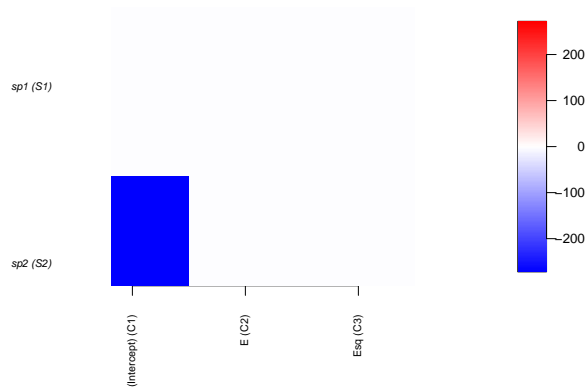
141

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



142

```
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.

## 5 Two species, logistic growth, competition XX

## References

- 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.
- 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.
- Oliveira, 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., 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.