Appendix S2. Evolutionary drivers of species and community dynamics

Jelena H. Pantel* Ruben J. Hermann†

07 September, 2024

1 Background

In Appendix S1.5, we evaluated the ability of the statistical model to estimate the impact of trait evolution for species abundances. Trait evolution's impact for species abundances can be modeled as a predictor in different ways. Here we derive the numerical form of the trait x_i that we hypothesize has the most impact on species abundances.

2 Model for population growth, competition, environmental change, and evolution

The growth equation for each species is:

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

We simulate population growth under particular parameter values, to generate population dynamics where species interactions, environment, and trait evolution all drive population dynamics.

```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 <- 0.6
x2.0 <- 0.8
P <- 1
w <- 10
```

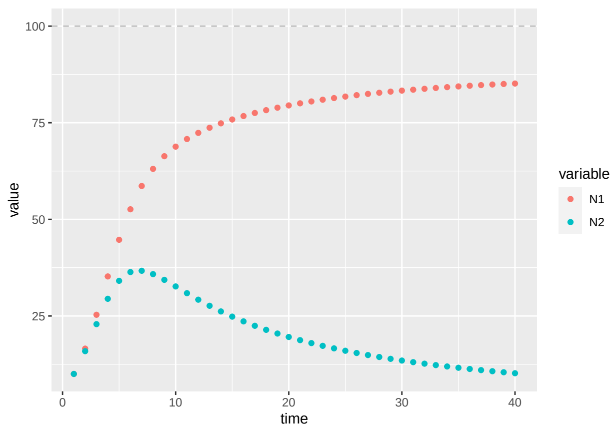
*Laboratoire Chrono-environnement,UMR 6249 CNRS-UFC, 16 Route de Gray, 25030 Besançon cedex, France, jelena.pantel@univ-fcomte.fr

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

```r
Wmax <- 2
h2 <- 0.1
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
r$r1[1] <- exp((-(((w + (1 - h2) * P)/(P + w)) * (E.0 - x1.0))^2)/(2 * (P + w)))
r$r2[1] <- 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] + rnorm(1, 0, 0.01)
}
```

```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 = (1/alpha
    linetype = "dashed", color = "gray")
```

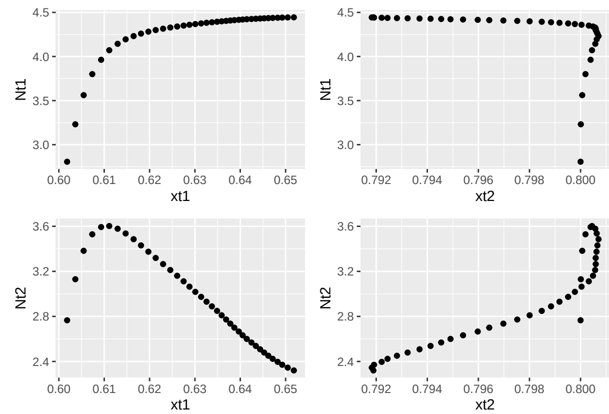


3 Hypotheses for evolution as a driver of species abundances

The growth equation is not linear, yet we fit abundance data to a heirarchical model with linear predictors because in most natural systems, a single theoretical growth model can't adequately capture the complexity in a given system. Linear models don't capture mechanistic relationships, but instead correlative. They are useful for inferring direction and magnitude of effect sizes. In Appendix S1, we explore how non-linear dynamics are captured when fit to a linear model. Here, we explore different forms to encode evolution in the trait x that determines the organism's fitness in this system.

3.1 H1. Raw trait value drives species abundances

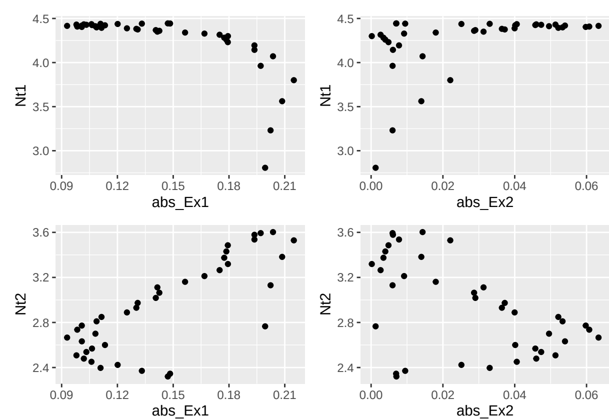
```r
# Plot simulation: ggplot
dat <- as.data.frame(cbind(log(N$N1), log(N$N2), x$x1, x$x2))
colnames(dat) <- c("N1", "N2", "x1", "x2")
dat$time <- 1:t
df <- data.frame(cbind(dat$N1[2:t], dat$N2[2:t], dat$x1[2:t], dat$x2[2:t]))
colnames(df) <- c("Nt1", "Nt2", "xt1", "xt2")
# Plot
p1 <- ggplot2::ggplot(df, aes(xt1, Nt1)) + geom_point()
p2 <- ggplot2::ggplot(df, aes(xt2, Nt1)) + geom_point()
p3 <- ggplot2::ggplot(df, aes(xt1, Nt2)) + geom_point()
p4 <- ggplot2::ggplot(df, aes(xt2, Nt2)) + geom_point()
p1 + p2 + p3 + p4
```



We do not hypothesize that the raw trait value is the best predictor of species abundances, because the impacts of this evolved trait depend on the context of the environment.

3.2 H2. Absolute value of trait distance from optimum drives species abundances

```r
# Plot simulation: ggplot
df$E <- E[2:t]
df$abs_Ex1 <- abs(df$E - df$xt1)
df$abs_Ex2 <- abs(df$E - df$xt2)
# Plot
p1 <- ggplot2::ggplot(df, aes(abs_Ex1, Nt1)) + geom_point()
p2 <- ggplot2::ggplot(df, aes(abs_Ex2, Nt1)) + geom_point()
p3 <- ggplot2::ggplot(df, aes(abs_Ex1, Nt2)) + geom_point()
p4 <- ggplot2::ggplot(df, aes(abs_Ex2, Nt2)) + geom_point()
p1 + p2 + p3 + p4
```

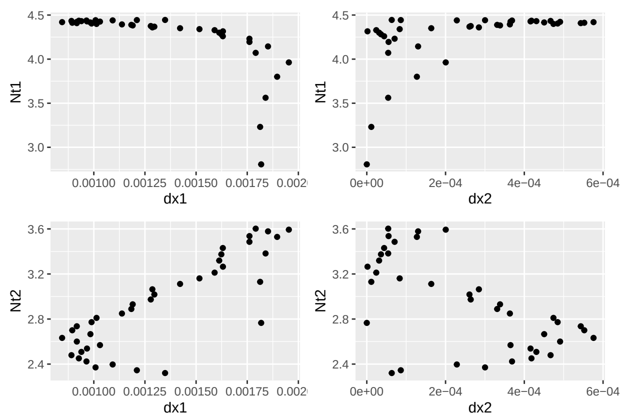


We hypothesize that this represents the true impact on species abundances, via the direct impact this measure has on fitness and consequent population growth. However, we also acknowledge this is likely difficult to measure in most empirical systems.

3.3 H3. Absolute value of trait change from one time to the next drives species abundances

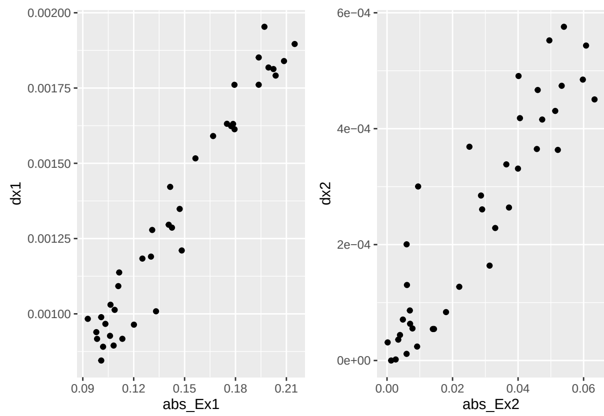
We now onsider the magnitude of trait change from one time step to the next as a driver for species abundances.

```
df$dx1 <- abs(dat$x1[2:t] - dat$x1[1:(t - 1)])
df$dx2 <- abs(dat$x2[2:t] - dat$x2[1:(t - 1)])
# Plot
p1 <- ggplot2::ggplot(df, aes(dx1, Nt1)) + geom_point()
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
```



The measures correlate highly with one another, as the amount of trait change (the amount of evolution) is determined by the distance to the optimum trait value (given that P and w remain constant in the simulation).

```
cor(df$dx1, df$abs_Ex1)
#> [1] 0.9733911
cor(df$dx2, df$abs_Ex2)
#> [1] 0.9144169
# Plot
p1 <- ggplot2::ggplot(df, aes(abs_Ex1, dx1)) + geom_point()
p2 <- ggplot2::ggplot(df, aes(abs_Ex2, dx2)) + geom_point()
p1 + p2
```



4 Statistical model for evolution as a driver of species abundances

We hypothesize that $|E_t - x_{i,t}|$ will be the best predictor of species abundances, and that $|x_{t+1} - x_{i,t}|$ will have similar explanatory power.

```
# prepare data in HMSC format
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$x2[1:(t - 1)])
# Y matrix
Y <- as.matrix(cbind(df$Nt1, df$Nt2))
# X matrix
XData <- data.frame(cbind(dat$N1[1:(t - 1)], dat$N2[1:(t - 1)]), E[1:(t - 1)], E[1:(t -
    1)]^2, dat$x1[1:(t - 1)], dat$x2[1:(t - 1)], abs(dat$x1[2:t] - dat$x1[1:(t -
    1)]), abs(dat$x2[2:t] - dat$x2[1:(t - 1)]), abs(E[1:(t - 1)] - dat$x1[1:(t -
    1)]), abs(E[1:(t - 1)] - dat$x2[1:(t - 1)]))
colnames(XData) <- c("n1", "n2", "E", "Esq", "x1", "x2", "dx1", "dx2", "dEx1", "dEx2")
# Prepare HMSC model
studyDesign = data.frame(sample = as.factor(1:(t - 1)))
rL = HmscRandomLevel(units = studyDesign$sample)
# Design and fit 4 alternative HMSC models
models = list()
for (i in 1:4) {
    XFormula = switch(i, ~1, ~E + Esq + x1 + x2, ~E + Esq + dEx1 + dEx2, ~E + Esq +
        dx1 + dx2)
    m = Hmsc(Y = Y, XData = XData, XFormula = XFormula, studyDesign = studyDesign,
        ranLevels = list(sample = rL))
    models[[i]] = m
}
# Bayesian model parameters
nChains <- 2
thin <- 5
samples <- 2000
transient <- 100 * thin
```

```
verbose <- 500 * thin
for (i in 1:4) {
    models[[i]] = sampleMcmc(models[[i]], thin = thin, sample = samples, transient = transient,
        nChains = nChains, verbose = verbose)
}
#> 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)
#> 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)
#> 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)
#> 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)
```

We check the explanatory and predictive power of each model using cross-validation.

```
# After Ovaskainen & Abrego 2020, Chapter 7
partition = createPartition(m, nfolds = 2, column = "sample")
partition.sp = c(1, 2)
result = matrix(NA, nrow = 3, ncol = 4)
for (i in 1:4) {
```

```
    m = models[[i]]
    # Explanatory power
    preds = computePredictedValues(m)
    MF = evaluateModelFit(hM = m, predY = preds)
    result[1, i] = mean(MF$R2)
    # Predictive power based on cross-validation
    preds = computePredictedValues(m, partition = partition)
    MF = evaluateModelFit(hM = m, predY = preds)
    result[2, i] <- mean(MF$R2)
    # Predictive power based on conditional cross-validation
    preds = computePredictedValues(m, partition = partition, partition.sp = partition.sp,
        mcmcStep = 100)
    MF = evaluateModelFit(hM = m, predY = preds)
    result[3, i] = mean(MF$R2)
}
#> Cross-validation, fold 1 out of 2
#> 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)
#> Cross-validation, fold 2 out of 2
#> 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)
#> Cross-validation, fold 1 out of 2
#> 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)
#> Cross-validation, fold 2 out of 2
#> 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)
#> Cross-validation, fold 1 out of 2
#> 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)
#> Cross-validation, fold 2 out of 2
#> 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)
#> Cross-validation, fold 1 out of 2
#> 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)
#> Cross-validation, fold 2 out of 2
#> 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)
#> Cross-validation, fold 1 out of 2
#> Computing chain 1
#> Chain 1, iteration 2500 of 10500 (sampling)
```

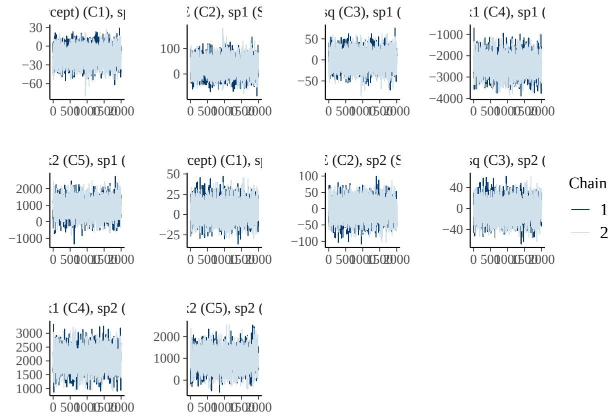
```
#> Chain 1, iteration 5000 of 10500 (sampling)
#> Chain 1, iteration 7500 of 10500 (sampling)
#> Chain 1, iteration 10000 of 10500 (sampling)
#> Computing chain 2
#> Chain 2, iteration 2500 of 10500 (sampling)
#> Chain 2, iteration 5000 of 10500 (sampling)
#> Chain 2, iteration 7500 of 10500 (sampling)
#> Chain 2, iteration 10000 of 10500 (sampling)
#> Cross-validation, fold 2 out of 2
#> 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)
#> Cross-validation, fold 1 out of 2
#> 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)
#> Cross-validation, fold 2 out of 2
#> 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)
#> Cross-validation, fold 1 out of 2
#> 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)
#> Cross-validation, fold 2 out of 2
```

```
#> 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)
#> Cross-validation, fold 1 out of 2
#> 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)
#> Cross-validation, fold 2 out of 2
#> 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)
```

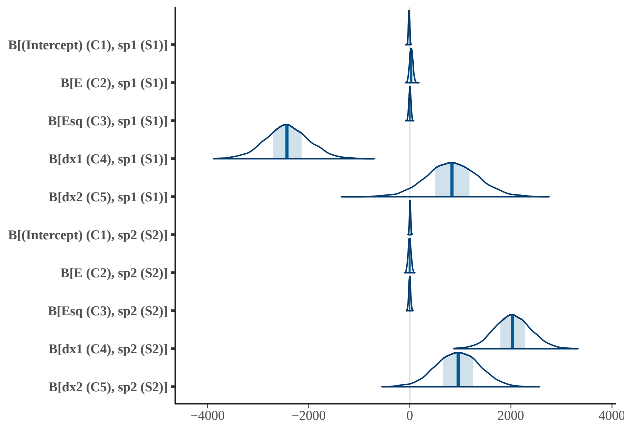
```
colnames(result) <- c("null", "xt", "dEx", "dx")
rownames(result) <- c("R2_expl", "R2_pred", "R2_pred_cross")
result
#>                      null        xt       dEx        dx
#> R2_expl        0.88571029 0.9994385 0.9992831 0.9992807
#> R2_pred       -0.07968265 0.5751829 0.3681523 0.3699655
#> R2_pred_cross  0.03203046 0.7198686 0.5857806 0.5878659
```

An important conclusion from this analysis is that the $|E_t - x_{i,t}|$ and $|x_{t+1} - x_{i,t}|$ do have similar explanatory power, and that - even though the species traits were the stronger predictor of species abundances - the model with total change in trait value can be used to assess the question "How does evolution impact species abundances?"

```
m.post.hmsc <- convertToCodaObject(models[[4]])
bayesplot::mcmc_trace(m.post.hmsc$Beta)
```

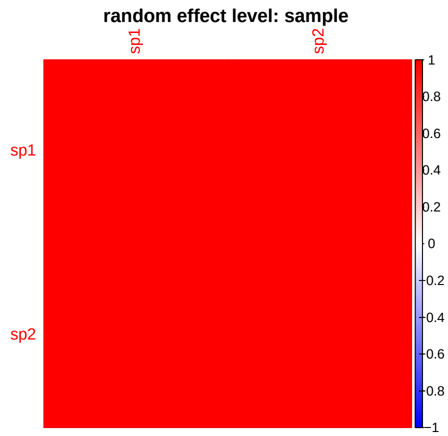



```r
bayesplot::mcmc_areas(m.post.hmsc$Beta, area_method = c("equal height"))
```




```r
# Bayesian estimates
summary(m.post.hmsc$Beta)$statistics[1:10, 1:2]
#>                                      Mean        SD
#> B[(Intercept) (C1), sp1 (S1)]   -15.125572  13.34014
#> B[E (C2), sp1 (S1)]              27.774288  32.48726
#> B[Esq (C3), sp1 (S1)]             1.437631  20.50185
#> B[dx1 (C4), sp1 (S1)]         -2425.485246 436.82617
#> B[dx2 (C5), sp1 (S1)]           840.535509 510.39539
#> B[(Intercept) (C1), sp2 (S2)]     5.176535  11.87201
#> B[E (C2), sp2 (S2)]              -4.548271  29.06227
#> B[Esq (C3), sp2 (S2)]            -2.757809  18.37233
#> B[dx1 (C4), sp2 (S2)]          2036.758427 365.57220
#> B[dx2 (C5), sp2 (S2)]           952.564461 432.35834
```

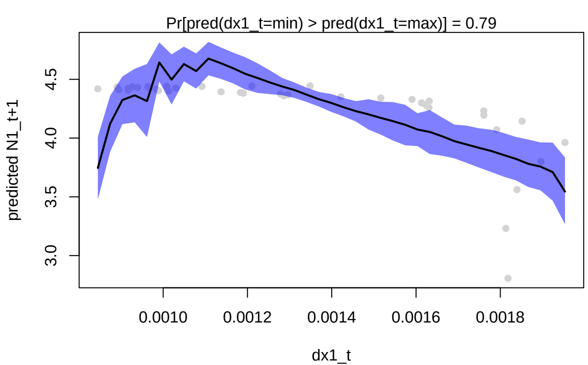
```r
OmegaCor = computeAssociations(models[[4]])
supportLevel = 0.7
toPlot = ((OmegaCor[[1]]$support > supportLevel) + (OmegaCor[[1]]$support < (1 -
    supportLevel)) > 0) * OmegaCor[[1]]$mean
corrplot::corrplot(toPlot, method = "color", col = colorRampPalette(c("blue", "white",
    "red"))(200), title = paste("random effect level:", models[[4]]$rLNames[1]),
    mar = c(0, 0, 1, 0))
```



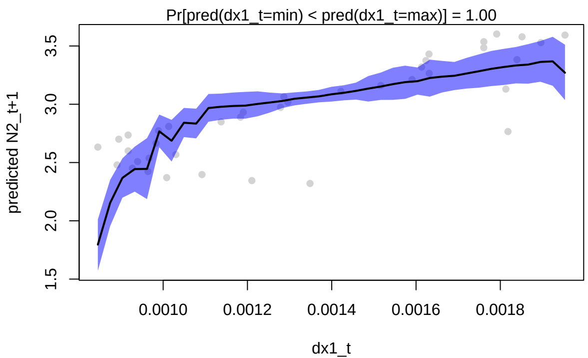

```
OmegaCor
#> [[1]]
#> [[1]]$mean
#>           sp1       sp2
#> sp1 1.0000000 0.9986902
#> sp2 0.9986902 1.0000000
#>
#> [[1]]$support
#>     sp1 sp2
#> sp1   1   1
#> sp2   1   1
```

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

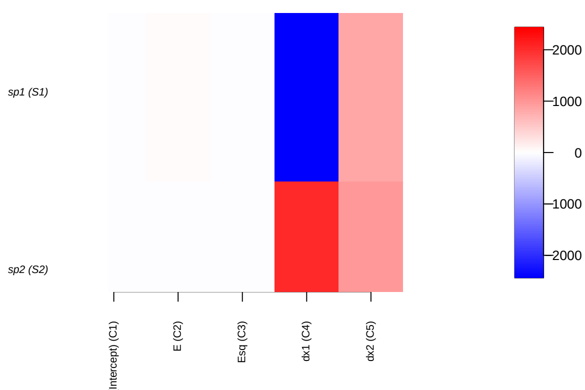
```
Gradient <- constructGradient(models[[4]], focalVariable = "dx1", non.focalVariables = list(E = list(2)
    Esq = list(2), dx2 = list(2)), ngrid = 39)
Gradient$XDataNew$Esq <- Gradient$XDataNew$E^2
predY <- predict(models[[4]], XData = Gradient$XDataNew, expected = TRUE)
a <- plotGradient(models[[4]], Gradient, pred = predY, showData = T, measure = "Y",
    index = 1, main = "", xlab = "dx1_t", ylab = "predicted N1_t+1")
```



```r
b <- plotGradient(models[[4]], Gradient, pred = predY, showData = T, measure = "Y",
    index = 2, main = "", xlab = "dx1_t", ylab = "predicted N2_t+1")
```

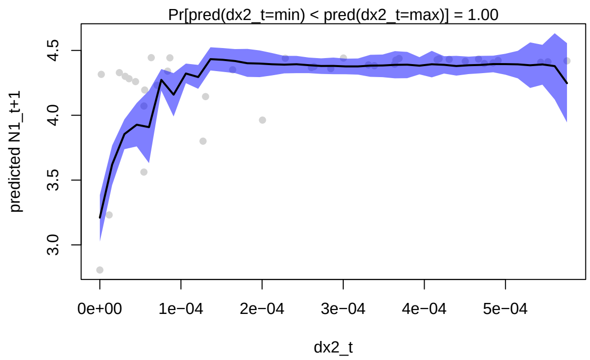



```r
postBeta = getPostEstimate(models[[4]], parName = "Beta")
plotBeta(models[[4]], post = postBeta, param = "Mean", supportLevel = 0.7)
```

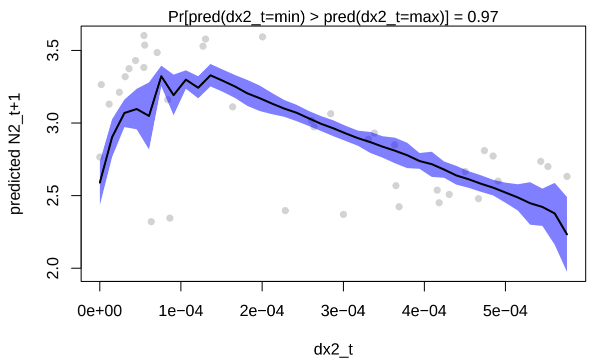


We repeat this plot for the effect of evolution in Species 2 for abundance in Species 1 and Species 2.

```r
Gradient <- constructGradient(models[[4]], focalVariable = "dx2", non.focalVariables = list(E = list(2)
    Esq = list(2), dx1 = list(2)), ngrid = 39)
Gradient$XDataNew$Esq <- Gradient$XDataNew$E^2
predY <- predict(models[[4]], XData = Gradient$XDataNew, expected = TRUE)
a <- plotGradient(models[[4]], Gradient, pred = predY, showData = T, measure = "Y",
    index = 1, main = "", xlab = "dx2_t", ylab = "predicted N1_t+1")
```

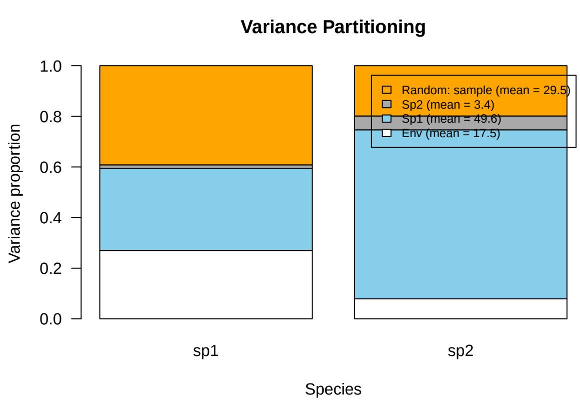



```r
b <- plotGradient(models[[4]], Gradient, pred = predY, showData = T, measure = "Y",
    index = 2, main = "", xlab = "dx2_t", ylab = "predicted N2_t+1")
```



To evaluate the relative importance of evolution for species abundances, we use variation partition:

```r
VP <- computeVariancePartitioning(models[[4]], group = c(1, 1, 1, 2, 3), groupnames = c("Env",
    "Sp1", "Sp2"))
plotVariancePartitioning(models[[4]], VP, cols = c("white", "skyblue", "darkgrey",
    "orange"), args.legend = list(cex = 0.75, bg = "transparent"))
```



We see that trait evolution in Species 1 is an important driver of abundances in both Species 1 and Species 2.