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

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

02 May, 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)).

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

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

```
    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)
}
```

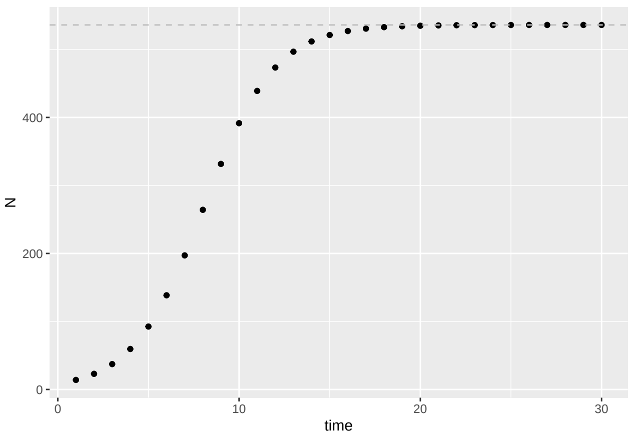


Figure 1: Population size N over time t for a discrete-time logistic growth model, with parameters $r_i = 1.67$, $N_{1,0} = 14$, and $\alpha_{11} = 0.00125$.

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:

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

```
# 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
```

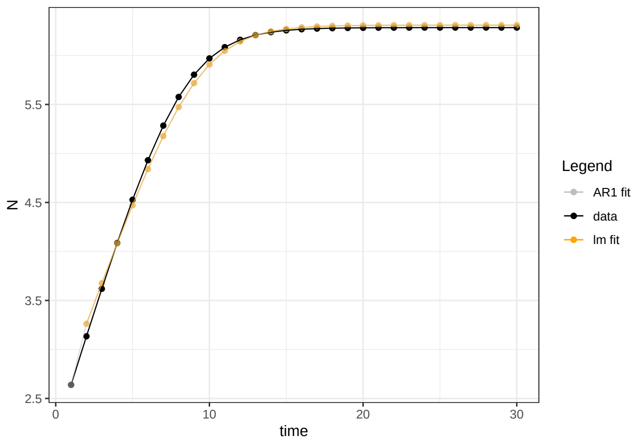


Figure 2: Population size over time (black line) with fitted values from a first-order autoregressive model (red dashed line).

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

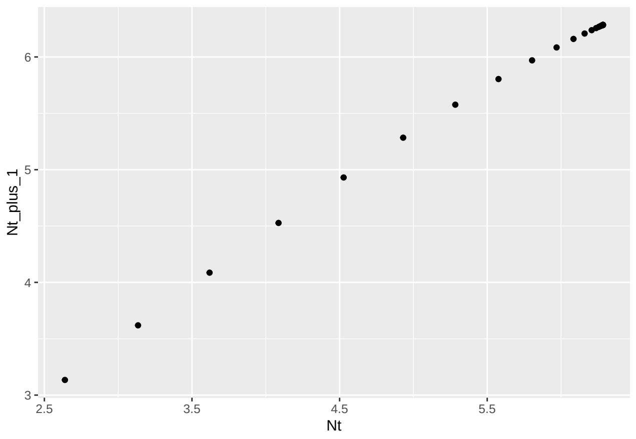


Figure 3: Population size (logarithm) at one time step N_{t+1} as a function of log-population size in the previous time step N_t.

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

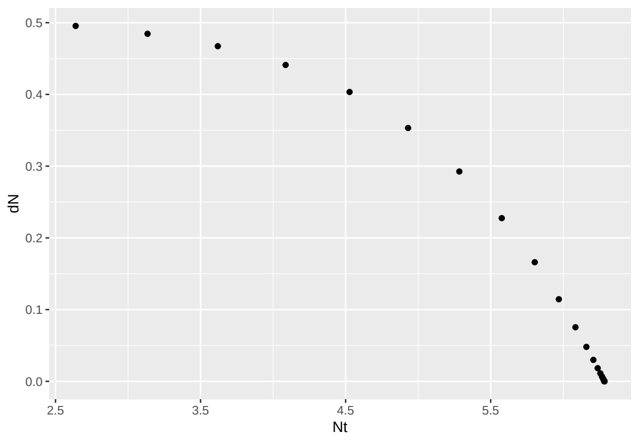


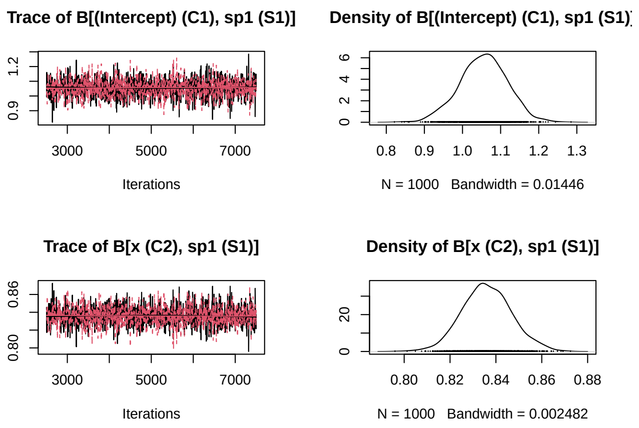
Figure 4: Change in population size from one time step to the next N_{t+1} as a function of N_{t+1}

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")
```

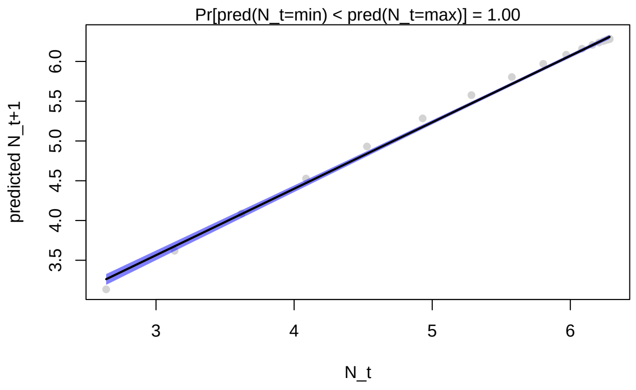


Figure 5: Observed (grey) and model-fit (blue) values for population size at time t (x-axis) and t+1 (y-axis).

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)
}
```

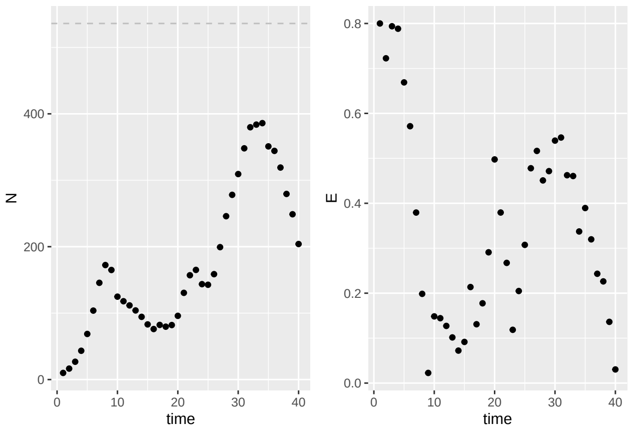


Figure 6: Population size N over time t for a discrete-time logistic growth model, with parameters $r_i = 1.67$, $N_{1,0} = 14$, and $\alpha_{11} = 0.00125$.

36 ## 2.2 Linear statistical model with environmental covariate

37 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)]) + 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
```

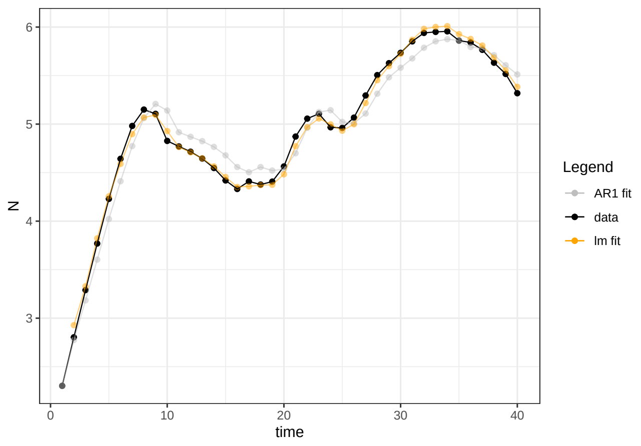


Figure 7: Population size over time (black line) with fitted values from a first-order autoregressive model (red dashed line).

The linear model is a good fit when including the environmental covariate. N_{t+1} and N_t are no longer well-represented by a linear function without taking E into account.

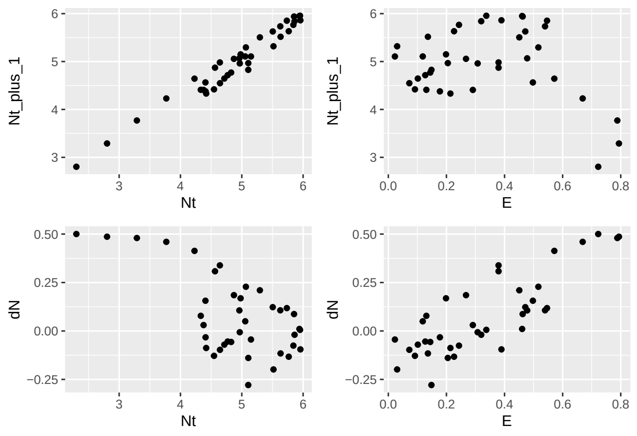


Figure 8: Population size (logarithm) at one time step N_{t+1} as a function of log-population size in the previous time step N_t.

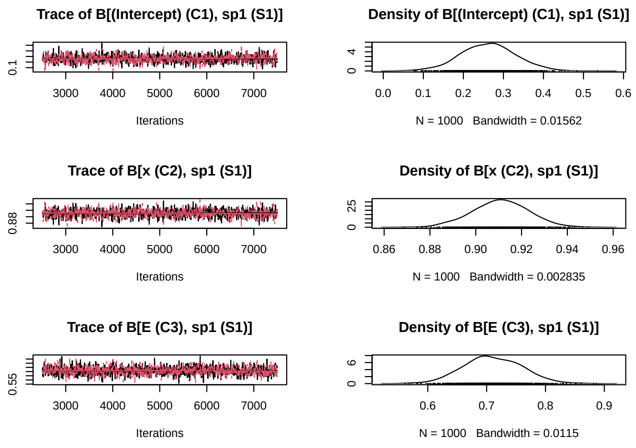
2.3 Bayesian linear statistical model: HMSC

We can estimate the same model parameters using HMSC:

```r
# prepare data in HMSC format
Y <- as.matrix(log(dat$N[2:t]))
XData <- data.frame(x = cbind(log(dat$N[1:(t - 1)])), E[1:(t - 1)])
colnames(XData)[2] <- "E"
m.2.hmsc <- Hmsc(Y = Y, XData = XData, XFormula = ~x + E)
# 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$Beta)
#>
#> Iterations = 2505:7500
#> Thinning interval = 5
#> Number of chains = 2
#> Sample size per chain = 1000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>                                   Mean      SD  Naive SE Time-series SE
#> B[(Intercept) (C1), sp1 (S1)] 0.2624 0.06856 0.0015330      0.0015334
#> B[x (C2), sp1 (S1)]           0.9108 0.01233 0.0002757      0.0002757
#> B[E (C3), sp1 (S1)]           0.7104 0.04962 0.0011094      0.0011097
#>
#> 2. Quantiles for each variable:
#>
#>                                 2.5%    25%    50%    75%  97.5%
#> B[(Intercept) (C1), sp1 (S1)] 0.1255 0.2165 0.2635 0.3068 0.3977
#> B[x (C2), sp1 (S1)]           0.8865 0.9027 0.9109 0.9191 0.9348
#> B[E (C3), sp1 (S1)]           0.6168 0.6772 0.7085 0.7455 0.8081
plot(m2.post.hmsc$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.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[1:3, 1:2]
#>                        Estimate Std. Error
#> (Intercept)           0.2601778 0.06652593
#> log(dat$N[1:(t - 1)]) 0.9111289 0.01191226
#> E[1:(t - 1)]          0.7120863 0.04601629
# Bayesian estimates
summary(m2.post.hmsc$Beta)$statistics[1:3, 1:2]
#>                                   Mean         SD
#> B[(Intercept) (C1), sp1 (S1)] 0.2624316 0.06856002
#> B[x (C2), sp1 (S1)]           0.9107787 0.01232761
#> B[E (C3), sp1 (S1)]           0.7104376 0.04961583
```

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(x = list(2)),
    ngrid = 39)
predY <- predict(m.2.sample, Gradient = Gradient, expected = TRUE)
# preds <-computePredictedValues(m.1.sample)
plotGradient(m.2.sample, Gradient, pred = predY, showData = T, measure = "Y", main = "",
    xlab = "E_t", ylab = "predicted N_t+1")
```

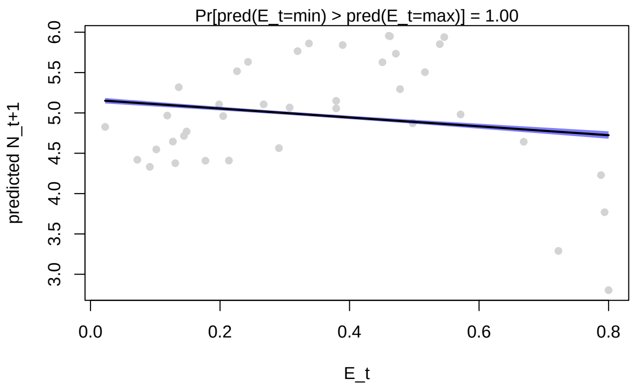


Figure 9: Observed (grey) and model-fit (blue) values for population size at time t (x-axis) and t+1 (y-axis).

2.4 Conclusions

In this example, the linear regression again works well to describe the impact of E_t for N_t. We can see the response of course depends on the population size value, as the gradient plot is generally linear 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.

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

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

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