Supporting Information

A.T. Tredennick, P.B. Adler, & F.R. Adler, "The relationship between species richness and..."

Ecology Letters

Section SI.1 R Code for Consumer-Resource Model

Below is the R code for our model function, which is represented mathematically in the main text in Equations 1-4. The same code, along with all the code to reproduce our results, has been archived on Figshare (link) and is available on GitHub (http://github.com/atredennick/Coexistence-Stability/releases).

```r
simulate_model <- function(seasons, days_to_track, Rmu,
                           Rsd_annual, sigE, rho,
                           alpha1, alpha2, alpha3, alpha4,
                           eta1, eta2, eta3, eta4,
                           r1, r2, r3, r4,
                           a1, a2, a3, a4,
                           b1, b2, b3, b4,
                           eps1, eps2, eps3, eps4,
                           D1, D2, D3, D4,
                           N1, N2, N3, N4, R) {

  require('deSolve') # for solving continuous differential equations
  require('mvtnorm') # for multivariate normal distribution functions

  ##  Assign parameter values to appropriate lists
  DNR <- c(D=c(D1,D2,D3,D4),   # initial dormant state abundance
           N=c(N1,N2,N3,N4),   # initial live state abundance
           R=R)                # initial resource level

  parms <- list (
    r   = c(r1,r2,r3,r4),          # max growth rate for each species
    a   = c(a1,a2,a3,a4),          # rate parameter for Hill function
    b   = c(b1,b2,b3,b4),          # shape parameter for Hill function
    eps = c(eps1,eps2,eps3,eps4)   # resource-to-biomass efficiency
  )


  ####
```

```
####  Sub-Model functions ------------------------------------------------
####
## Continuous model
updateNR <- function(t, NR, parms){
  with(as.list(c(NR, parms)), {
    dN1dt = N1*eps[1]*(uptake_R(r[1], R, a[1], b[1]))
    dN2dt = N2*eps[2]*(uptake_R(r[2], R, a[2], b[2]))
    dN3dt = N3*eps[3]*(uptake_R(r[3], R, a[3], b[3]))
    dN4dt = N4*eps[4]*(uptake_R(r[4], R, a[4], b[4]))
    dRdt  = -1 * (dN1dt/eps[1] + dN2dt/eps[2] + dN3dt/eps[3] + dN4dt/eps[4])
    list(c(dN1dt, dN2dt, dN3dt, dN4dt, dRdt)) # output as list
  })
} # end continuous function

## Discrete model
update_DNR <- function(t, DNR, gammas,
                       alpha1, alpha2, alpha3, alpha4,
                       eta1, eta2, eta3, eta4) {
  with (as.list(DNR),{
    g1    <- gammas[1]
    g2    <- gammas[2]
    g3    <- gammas[3]
    g4    <- gammas[4]
    D1new <- (1-g1)*(alpha1*N1 + D1)*(1-eta1)
    D2new <- (1-g2)*(alpha2*N2 + D2)*(1-eta2)
    D3new <- (1-g3)*(alpha3*N3 + D3)*(1-eta3)
    D4new <- (1-g4)*(alpha4*N4 + D4)*(1-eta4)
    N1new <- g1*(alpha1*N1 + D1)*(1-eta1)
    N2new <- g2*(alpha2*N2 + D2)*(1-eta2)
    N3new <- g3*(alpha3*N3 + D3)*(1-eta3)
    N4new <- g4*(alpha4*N4 + D4)*(1-eta4)
    Rnew  <- Rvector[t]
    return(c(D1new, D2new, D3new, D4new, N1new, N2new, N3new, N4new, Rnew))
  })
}

##  Resource uptake function (Hill function)
uptake_R <- function(r, R, a, b) {
  return((r*R^a) / (b^a + R^a))
```

```
}

##  Generate germination fractions
getG <- function(sigE, rho, nTime, num_spp) {
  varcov       <- matrix(rep(rho*sigE,num_spp*2), num_spp, num_spp)
  diag(varcov) <- sigE
  if(sigE > 0) { varcov <- Matrix::nearPD(varcov)$mat } # crank through nearPD to fix roundi
  varcov <- as.matrix(varcov)
  e      <- rmvnorm(n = nTime, mean = rep(0,num_spp), sigma = varcov)
  g      <- exp(e) / (1+exp(e))
  return(g)
}



####
#### Simulate model -----------------------------------------------------
####
days            <- c(1:days_to_track)
num_spp         <- length(parms$r)
nmsDNR          <- names(DNR)
dormants        <- grep("D", names(DNR))
NR              <- DNR[-dormants]
nmsNR           <- names(NR)
gVec            <- getG(sigE = sigE, rho = rho, nTime = seasons, num_spp = num_spp)
Rvector         <- rlnorm(seasons, Rmu, Rsd_annual)
saved_outs      <- matrix(ncol=length(DNR), nrow=seasons+1)
saved_outs[1,] <- DNR

##  Loop over seasons
for(season_now in 1:seasons) {
  # Simulate continuous growing  season
  output   <- ode(y = NR, times=days, func = updateNR, parms = parms)
  NR       <- output[nrow(output),nmsNR]
  dormants <- grep("D", names(DNR))
  DNR      <- c(DNR[dormants], NR)

  # Save end of season biomasses, before discrete transitions
  saved_outs[season_now+1,] <- DNR
```

```
    names(DNR) <- nmsDNR
    DNR        <- update_DNR(season_now, DNR, gVec[season_now,],
                             alpha1 = alpha1, alpha2 = alpha2,
                             alpha3 = alpha3, alpha4 = alpha4,
                             eta1 = eta1, eta2 = eta2, eta3 = eta3, eta4 = eta4)

    names(DNR) <- nmsDNR
    NR         <- DNR[-dormants]
    names(NR)  <- nmsNR
  } # next season

  return(saved_outs)

} #end simulation function
```

Section SI.2 Exploring Parameter Space

Section SI.3 Eight-Species Storage Effect Model

In the main text we constrained our focus to four-species communities because getting more than four species to coexist by relative nonlinearity is tricky, and usually requires adding another coexistence mechanism on top off relative nonlinearity (Yuan & Chesson 2015). The storage effect does not suffer from this limitation, but we wanted our results in the main text to be easily comparable between coexistence mechanisms. Here, we show that our results are qualitatively similar if we simulate an eight-species community with species coexistence maintained by the storage effect. We conducted the same numerical simulations described in the main text for Figure 2. Quoting from the main text:

> *First, we allowed the variance of the environment to determine how many species can coexist, akin to a community assembly experiment with a species pool of four species. To do this, we simulated communities with all species initially present across a gradient of annual resource variability (for relative nonlinearity) or environmental cue variability (for the storage effect). Second, we chose parameter values that allowed coexistence of all four species and then performed species removals, akin to a biodiversity–ecosystem function experiment. The two simulation experiments correspond to (i) sampling ecosystem function across a natural gradient of species richness and (ii) sampling ecosystem function across diversity treatments within a site.*

From one to four species, the relationship is as presented in the main text: total community CV increases approximately linearly with environmental variability because (1) environmental variability promotes species coexistence *and* (2) environmental variability causes populations fluctuations to increase (Figure S1-1). However, after four species, the relationship saturates — species additions due to coexistence by the storage effect completely buffer ecosystem variability from further increases in environmental variability (Figure S1-1). Thus, our results provide novel theoretical explanations for positive and flat diversity–ecosystem variability relationships.

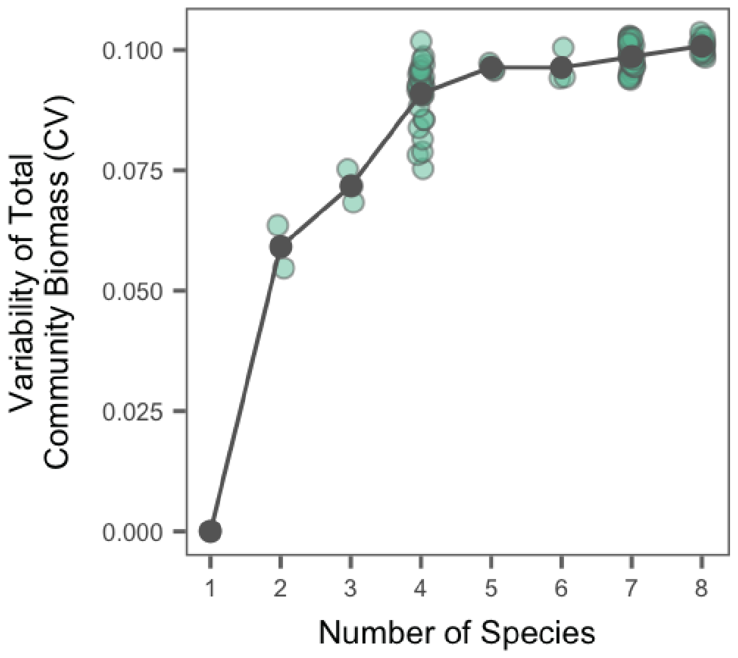

Figure S1-1 A storage effect result.

35 Section SI.4 Additional Figures

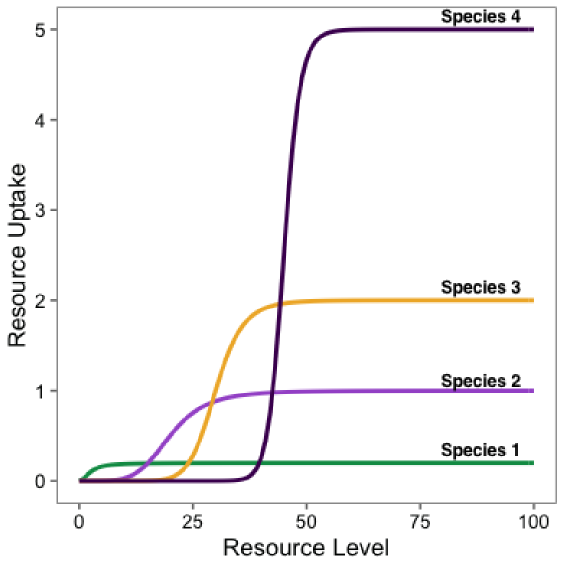

Figure S1-2 Resource uptake curves for each species (represented by different colors) as used in relative nonlinearity simulations. The equation for resource uptake is: $f_i(R) = r_i R^{a_i}/(b_i^{a_i} + R^{a_i})$. Parameter values are as follows. Species 1: $r = 0.2$, $a = 2$, $b = 2.5$; Species 2: $r = 1$, $a = 5$, $b = 20$; Species 3: $r = 2$, $a = 10$, $b = 30$; Species 4: $r = 5$, $a = 25$, $b = 45$.

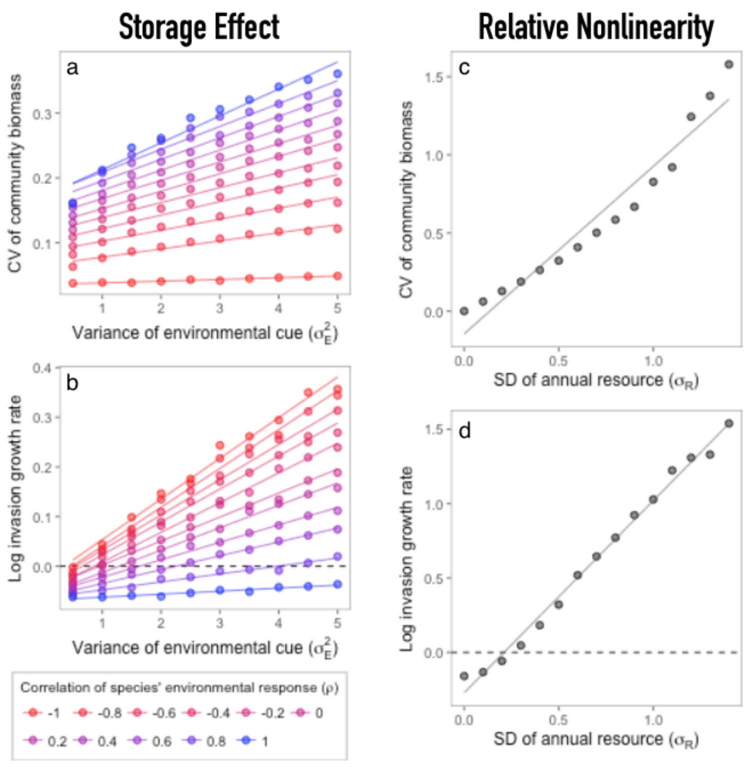

Figure S1-3 Variability of community biomass and invasion growth rates of the inferior competitor in a two-species community under different parameter combinations. Points are mean values from 5,000 growing seasons and lines are linear fits to show trends. In **Storage Effect** plots (a,b), resource supply is held constant between growing seasons. Resource supply varies each year in **Relative Nonlinearity** simulations (c,d), while the environmental cue variance is set to 0.

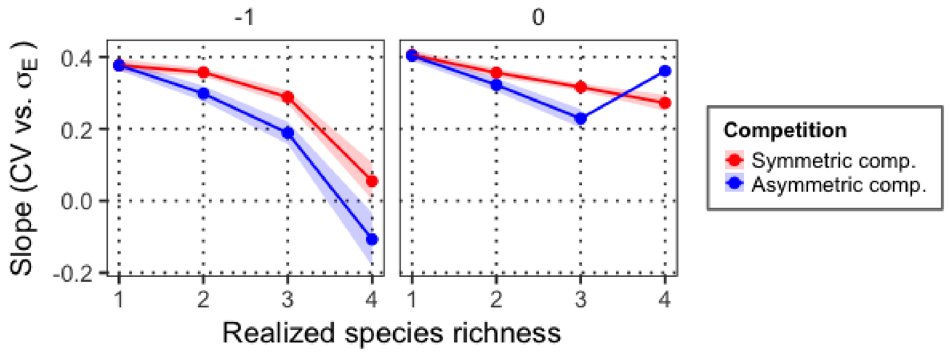

Figure S1-4 Slopes of linear fits for the relationship between $\log(CV)$ and $\log(\sigma_E)$ at different levels of realized species richness from storage effect simulations. The slopes come from linear models fit to log-transformed versions of Figure 3 in the main text. For these simulations, "symmetric competion" (●) refers to similar live-to-dormant biomass allocation fractions ($\boldsymbol{\alpha} = [0.5, 0.495, 0.49, 0.485]$ for the four species), and "asymmetric competition" (●) refers to more dissimilar live-to-dormant biomass allocation fractions ($\boldsymbol{\alpha} = [0.5, 0.49, 0.48, 0.47]$ for the four species).

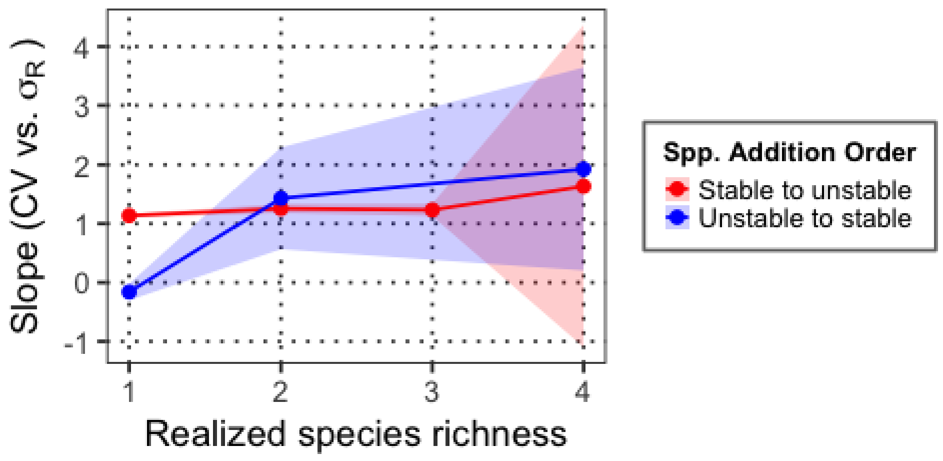

Figure S1-5 Slopes of linear fits for the relationship between $\log(CV)$ and $\log(\sigma_R)$ at different levels of realized species richness from relative nonlinearity simulations. The slopes come from linear models fit to log-transformed versions of Figure 4 in the main text.

Yuan, C. & Chesson, P. (2015). The relative importance of relative nonlinearity and the storage effect in the lottery model. *Theoretical Population Biology*, 105, 39–52.