Section S1.1   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 The same code, along with all the code to reproduce our results, has been archived on Figshare (link) and is available on GitHub (http://github.com/atredennick/Coexistence-Stability/releases).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,
                           beta1, beta2, beta3, beta4,
                           eta1, eta2, eta3, eta4,
                           theta1, theta2, theta3, theta4,
                           nu, 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
  )
```

```r
####
####  Sub-Model functions -----------------------------------------------
####
## Continuous model (Equations 1-2)
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 (Equations 3-4)
update_DNR <- function(t, DNR, gammas,
                       alpha1, alpha2, alpha3, alpha4,
                       eta1, eta2, eta3, eta4,
                       beta1, beta2, beta3, beta4,
                       theta1, theta2, theta3, theta4, nu) {
  with (as.list(DNR),{
    g1    <- gammas[1]
    g2    <- gammas[2]
    g3    <- gammas[3]
    g4    <- gammas[4]
    D1new <- alpha1*N1 + D1*(1-g1)*(1-eta1)
    D2new <- alpha2*N2 + D2*(1-g2)*(1-eta2)
    D3new <- alpha3*N3 + D3*(1-g3)*(1-eta3)
    D4new <- alpha4*N4 + D4*(1-g4)*(1-eta4)
    N1new <- beta1*(1-alpha1)*N1 + g1*(D1+(alpha1*N1))*(1-eta1)
    N2new <- beta2*(1-alpha2)*N2 + g2*(D2+(alpha2*N2))*(1-eta2)
    N3new <- beta3*(1-alpha3)*N3 + g3*(D3+(alpha3*N3))*(1-eta3)
    N4new <- beta4*(1-alpha4)*N4 + g4*(D4+(alpha4*N4))*(1-eta4)
    Rnew  <- theta1*(1-alpha1)*N1 + theta2*(1-alpha2)*N2 +
             theta3*(1-alpha3)*N3 + theta4*(1-alpha4)*N4 +
             nu*R + Rvector[t]
```

```r
    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

  # crank through nearPD to fix rounding errors
  if(sigE > 0) { varcov <- Matrix::nearPD(varcov)$mat }

  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
```

```r
  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,
                      beta1 = beta1, beta2 = beta2,
                      beta3 = beta3, beta4 = beta4,
                      theta1 = theta1, theta2 = theta2,
                      theta3 = theta3, theta4 = theta4,
                      nu=nu)

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


  return(saved_outs)


} ##  End simulation function
```
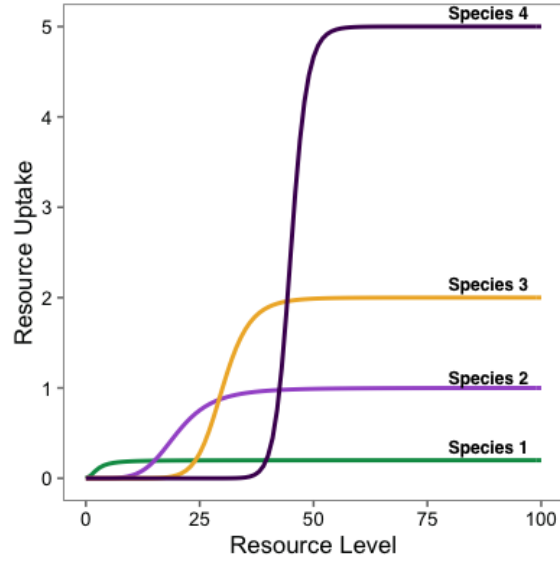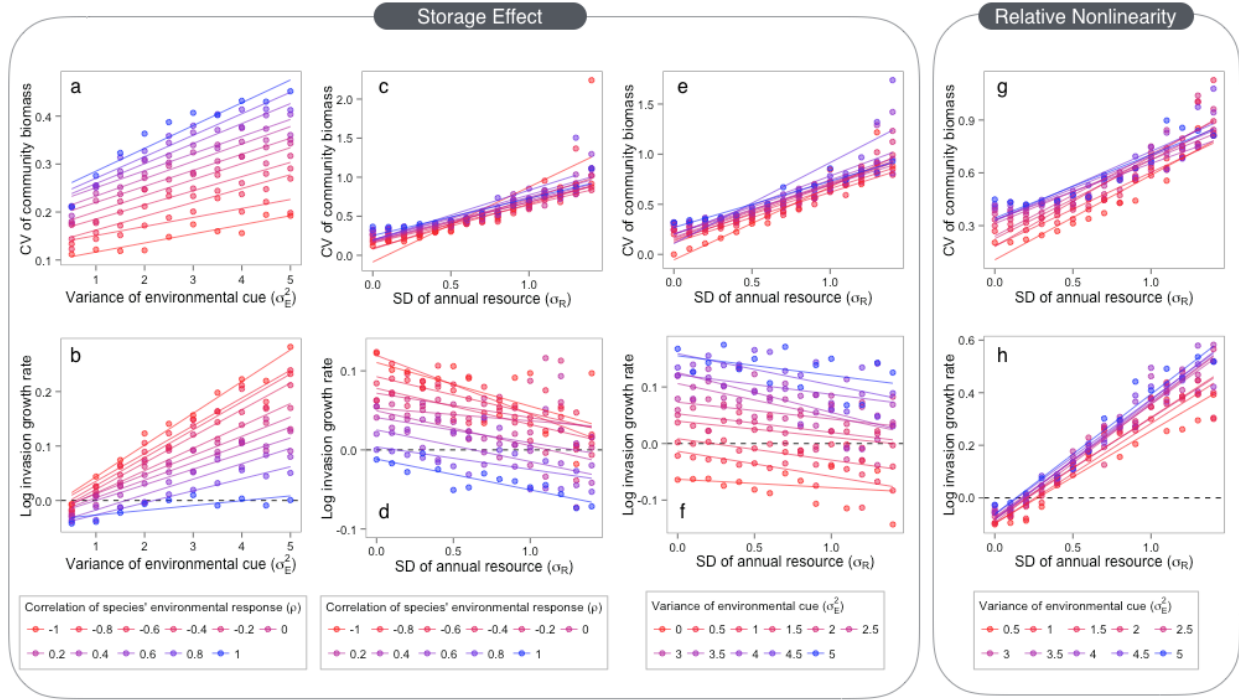
Section S1.2 Additional Figures

Figure S1-1 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 S1-2 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 10,000 growing seasons and lines are linear fits to show trends. In **Storage Effect** plots, resource supply is held constant between growing seasons, whereas resource supply varies each year in **Relative Nonlinearity** simulations.
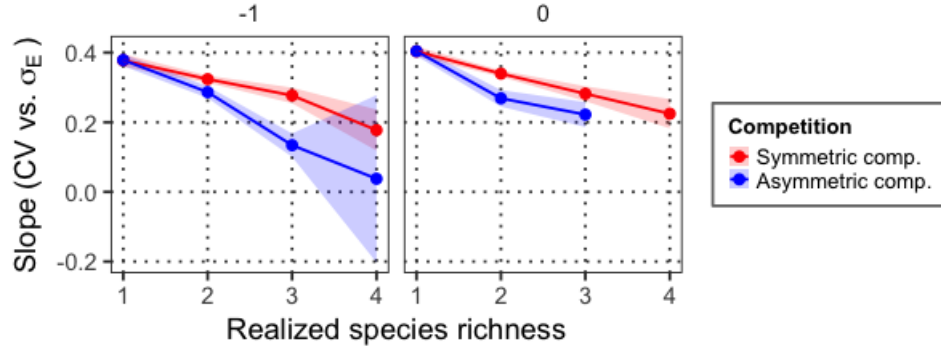
Figure S1-3 Slopes of linear fits for the relationship between $\log(CV)$ and $\log(\sigma_E)$ at different levels of realized species richness from storage effect simulations. The slopes come from linear models fit to log-transformed versions of Figure 3 in the main text. For these simulations, "symmetric competion" (•) refers to similar live-to-dormant biomass allocation fractions ($\alpha = [0.5, 0.495, 0.49, 0.485]$ for the four species), and "asymmetric competition" (•) refers to more dissimilar live-to-dormant biomass allocation fractions ($\alpha = [0.5, 0.49, 0.48, 0.47]$ for the four species).
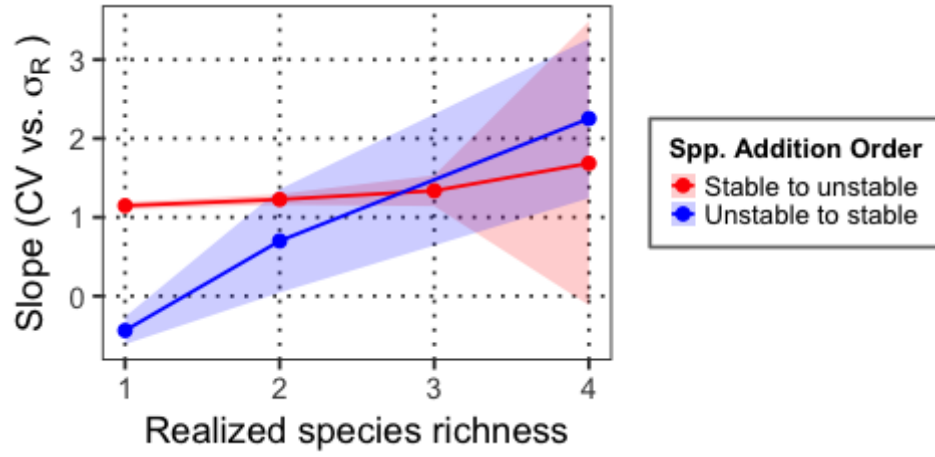
Figure S1-4 Slopes of linear fits for the relationship between $\log(CV)$ and $\log(\sigma_R)$ at different levels of realized species richness from relative nonlinearity simulations. The slopes come from linear models fit to log-transformed versions of Figure 4 in the main text.