Sparse interactions: Simulations

Contents

Code used to generate simulated communities for the manuscript "Disentangling key species interactions in diverse and heterogeneous communities: A Bayesian sparse modeling approach"

Functions

Single Beverton-Holt run

Function for a single Beverton-Holt time step within a single plot. Inputs:
- number of species in the plot
- vector of the species' starting populations
- vector of the species' λ_i values ($\lambda_{e,i}$ in the given plot)
- matrix of the $\alpha_{i,j}$ interactions between each species pair ($\alpha_{e,i,j}$ in the given plot)
Calculates new populations with a for loop over i species.

```
BH.run <- function(num.species, N.0, lambda, alpha){
  N.1 <- numeric(length = num.species) # blank vector to be filled in with the for loop
  for(i in 1:num.species){
    comp.sum <- sum(alpha[,i]*N.0)
    N.1[i] <- N.0[i]*lambda[i] / (1 + comp.sum)
  }

  return(N.1)
}
```

Multiple simulations through time

Function for the Beverton-Holt simulations on a given set of species parameters. Conducts multiple runs from a single set of input species' $\lambda_{e,i}$ intrinsic growth rates and species-pair $\alpha_{e,i,j}$ competition coefficients. Each run represents a unique plot with a given environmental condition X_e, and is run for an input number of time steps.

Inputs:
- Data frame with a row for each species. Columns are species ID number, λ parameters (either `lambda.mean` and `lambda.env` or `lambda.max`, `z.env`, and `sigma.env`; see below for details), and α parameters (for example, a 4-species simulation would have columns labeled `alpha.1`, `alpha.2`, `alpha.3`, `alpha.4`, and `alpha.env.gen` and `alpha.env.spec` coefficient-environment interaction terms)
- Number of runs
- Number of time steps
- Environmental condition for each run
- Demographic heterogeneity binary: whether poisson noise should be added to the end population values
- Binary `lambda.opt` term: If `lambda.opt` is false, $\lambda_{e,i}$ has a monotonic relationship with the environment and `df.in` should have two parameters, `lambda.mean` and `lambda.env`. If `lambda.opt` is true, $\lambda_{e,i}$ has an optimum environmental value and `df.in` should have three parameters: `lambda.max`, `z.env`, and `sigma.env`
- Data frame of starting population values for each species in each run. One column `run.count` for run ID and one for the population values `pop.size`

Dependencies:
- Calls the single BH run function above.

Outputs: - Data frame in a long tidy format with columns for species, run IDs, environmental conditions in the runs, time step, and population.

Steps:
- For each run, calculates the $\lambda_{e,i}$ values for each species in that given environment X_e and the $\alpha_{e,i,j}$ values for each species pair in the given environment.
- If using a monotonic lambda-environment relationship (`lambda.opt == FALSE`), the lambda value for each species $\lambda_{e,i}$ is calculated as $e^{\lambda_{i\ (mean)} + \lambda_{e,i} * env}$. If using an optimum lambda-environment relationship (`lambda.opt == TRUE`), $\lambda_{e,i}$ is calculated as $\lambda_{i\ max} e^{-\left(\frac{z_i - env}{2\sigma_i}\right)^2}$. *NOTE: this has the same issue as the table, with two $\lambda_{e,i}$ terms–still need to decide on how to disambiguate these and make sure it's consistent through the document*
- $\alpha_{e,i,j}$ is calculated as $e^{\alpha_i + \hat{\alpha}_{i,j} + (\alpha_{e,i} + \hat{\alpha}_{e,i,j}) X_e}$.
- Within each run, simulates the community through time with those λ and α parameters.
- If `dem.het == TRUE`, adds poisson noise at each time step by making each species population size a single poisson draw from a distribution with a rate equal to its deterministically-calculated population size.

```
BH.env <- function(df.in, n.runs, t.steps, p.start, env, dem.het = FALSE, lambda.opt = FALSE)
{
  # setting up empty data frame to store the output
  n.species <- length(df.in$species)
  species <- rep(factor(1:n.species), times = n.runs*(t.steps+1))
  run <- rep(1:n.runs, each = n.species*(t.steps+1))
  time <- rep(rep(0:t.steps, each = n.species), times = n.runs)
  pop <- rep(-100, times = n.species*(t.steps+1)*n.runs)
  run.env <- rep(env, each = n.species*(t.steps+1)) # env conditions in each run

  df.out <- data.frame(species, run, run.env, time, pop)


  # running across each run (environmental condition varies across runs)
  for(r in 1:n.runs){
    # initial populations
    start.rows <- df.out %>%
      with(run == r & time == 0) %>%
      which()
    start.pop <- with(p.start, pop.size[run.count == r])
    df.out$pop[start.rows] <- start.pop
```

```
    # calculating lambda for each species in the given environment
    if(lambda.opt == TRUE){
      # lambda with environmental optimum
      env.diff <- df.in$z.env - env[r] # diff bw species opt env and current env
      env.width <- 2*df.in$sigma.env # denom for below
      lambda <- df.in$lambda.max * exp(-(env.diff/env.width)^2)
    }
    else {
      # monotonic lambda-environment interaction
      lambda <- exp(df.in$lambda.mean + df.in$lambda.env*env[r])
    }

    # calculating alpha for each species pair in the given environment
    alpha.means <- df.in %>%
      select(starts_with('alpha') & !starts_with('alpha.env')) # mean alphas
    alpha.env.int <- (df.in$alpha.env.gen + df.in$alpha.env.spec) * env[r] # alpha x env interaction
    alpha <- alpha.means # make a same-sized df

    for(i in 1:n.species){
      alpha[i,] <- exp(alpha.means[i,] + alpha.env.int[i])
    }

    # simulating the community through time
    for(t in 0:t.steps){
      # where to store the data for the runs
      end.rows <- df.out %>%
        with(run == r & time == t+1) %>%
        which()

      end.pop <- BH.run(n.species, start.pop, lambda, alpha)
      end.pop[end.pop < 0] <- 0 # set negative numbers to zero

      # option to add in demographic stochasticity
      if(dem.het == TRUE) {
        end.pop <- rpois(length(end.pop), end.pop)
      }

      df.out$pop[end.rows] <- end.pop

      start.pop <- end.pop # updating populations for the next time step
    }
  }
  return(df.out)
}
```

Running simulations

Uses the functions above to simulate plot communities deterministically to near-equilibrium values, perturb species populations, and then look at stochastic population change from that perturbed state to the following time step.

Set up input parameters

Setting number of species in the simulation, number of runs with different environments, number of steps for the warm-up simulation in each run (20 recommended), strength of environmental variation among plots (0 = no variation, 1 = variation), and parameters for initial population distributions (these are pretty flexible).

```
num.species <- 15 # number of species
num.runs <- 500 # number of separate runs
pre.time.steps <- 20 # steps for the warm-up run
env.variation <- 1 # environmental variation
pop.start.mean <- 80 # mean initial population size
pop.start.sd <- 50 # standard deviation in initial population size
```

Determining type of relationship between λ and the environment. If `lambda.optimum == FALSE`, simulation uses the monotonic lambda-environment relationship. If `lambda.optimum == TRUE`, simulation uses an optimum environment for λ. `env.response` parameter sets the strength of the variation in species' environmental responses in λ as the standard deviation for the Gaussian distribution of slopes (monotonic) or location of optima (optimum). Use 0.5 for a decently strong variation.

```
lambda.optimum <- TRUE
env.response <- 0.5
```

Parameters for $\alpha_{e,i,j}$ values:
- Intraspecific competition α_intra (-4.6 corresponds to $\sim$ 0.01 after exponentiating)
- Generic intraspecific competition mean and standard deviation for α_i (-6.9 ~> 0.001)
- Non generic species number (this can also be set using a poisson draw) and range for uniform distribution of $\hat{\alpha}_{i,j}$ component (-2 to 2)
- Standard deviation for generic $\alpha_{e,i}$ environmental interaction (sd try 0.3, mean is 0)
- Number of species whose alphas vary with the environment (0 for none) and range of values (-1 to 1)

```
a.intra <- -4.6 # intraspecific competition
a.gen.mean <- -6.9 # generic competition mean
a.gen.sd <- 0.1 # generic competition sd

a.diff.num <- 4 # number of non-generic species
a.diff.range <- c(-2, 2) # range of non-generic competition strength

a.env.gen.sd <- 0.3 # range for generic alpha-env interaction
a.env.num <- 4 # number of species with specific alpha-env interactions
a.env.range <- c(-1, 1) # range of specific alpha environmental interaction strength
```

Setting coefficients for the simulation

Setting the environmental condition in each plot X_e

```
env.condition <- rnorm(n = num.runs, mean = 0, sd = env.variation)
```

Setting lambda values for each species ($\lambda_{i\ (max)}$, z_i and σ_i for optimum; $\lambda_{i\ (mean)}$ and $\lambda_{e,i}$ for monotonic). Note that because of the different functional forms, $\lambda_{i\ (max)}$ is drawn from a range of 1 to 5, while $\lambda_{i\ (mean)}$ is drawn from a range of 0 to 1.5 (which after exponentiating corresponds to a similar range of 1 to 4.5)

```
if(lambda.optimum == TRUE){
  # optimum environmental condition for species
  # max intrinsic growth
  lambda.max <- runif(n = num.species, min = 1, max = 5)
  # environmental response
  z.env <- rnorm(n = num.species, mean = 0, sd = env.variation)
```

```
  sigma.env <- rexp(n = num.species, rate = 0.5)
} else {
  # montonic environmental relationships for species
  lambda.mean <- runif(n = num.species, min = 0, max = 1.5)
  # environmental response
  lambda.env <- rnorm(n = num.species, mean = 0, sd = env.response)
}
```

Setting up α values for each species pair. First the matrix is filled with all generic values α_i drawn from Gaussian distribution with mean and sd set above. Deviation for non-generic species $\hat{\alpha}_{i,j}$ is drawn from a uniform distribution with the range set above and added to the generic α_i. Intraspecific α_{intra} terms are set separately to the value specified above.

$\alpha_{e,i}$ generic environment interaction term is drawn from a normal distribution with mean 0 and sd set above. $\hat{\alpha}_{e,i,j}$ environment interaction terms are stored in a separate vector with one term per selected environmentally responsive species drawn from a uniform distribution with the range set above.

```
# setting up competition matrix with all generic values
alphas.generic <- rnorm(num.species^2, mean = a.gen.mean, sd = a.gen.sd)
alpha <- matrix(alphas.generic, nrow = num.species, ncol = num.species)

# selecting non-generic species
a.diff <- sample(1:num.species, a.diff.num)
print(c('a.diff', a.diff))

# filling in alphas for non-generic species
for(i in 1:a.diff.num){
  all.species <- 1:num.species
  competitor <- a.diff[i]
  responders <- all.species[!(all.species %in% competitor)]
  comp.diff <- runif(1, min = a.diff.range[1], max = a.diff.range[2])
  alpha[competitor, responders] <- alpha[competitor, responders] + comp.diff
  print(comp.diff)
}

# filling in alphas for intraspecific variaiton
diag(alpha) <- a.intra

# environmentally-variable alpha terms
alpha.env.gen <- rep(rnorm(1, mean = 0, sd = a.env.gen.sd), time = num.species)
alpha.env.spec <- rep(0, time = num.species) # default of 0 leaves alphas at their means
if(a.env.num > 0){
  a.env.id <- sample(1:num.species, a.env.num)
  print(c('a.env.id', a.env.id))
  for(i in 1:a.env.num){
    competitor <- a.env.id[i]
    alpha.env.spec[competitor] <- runif(1, min = a.env.range[1], max = a.env.range[2])
  }
}
```

Creating a coefficient dataframe with one row per species and columns for species ID, λ parameters, and α parameters filled in from the above.

```
# input dataframe of coefficients
if(lambda.optimum == TRUE){
  df.coef <- data.frame(species = factor(1:num.species),
```

```
                        lambda.max, z.env, sigma.env,
                        alpha, alpha.env.gen, alpha.env.spec) %>%
    rename_with(.,~gsub("X","alpha.", .x, fixed = TRUE))
} else {
  df.coef <- data.frame(species = factor(1:num.species),
                        lambda.mean, lambda.env,
                        alpha, alpha.env.gen, alpha.env.spec) %>%
    rename_with(.,~gsub("X","alpha.", .x, fixed = TRUE))
}
```

Running the functions

Setting up the initial populations

```
pop.size <- as.integer(rnorm(num.species*num.runs,
                             mean = pop.start.mean,
                             sd = pop.start.sd))
pop.size[pop.size < 0] <- 0 # set negative numbers to zero
run.count <- rep(1:num.runs, each = num.species)
pop.start <- data.frame(run.count, pop.size)
```

Warm-up steps: deterministic simulations to near-equilibrium

```
df.eq <- BH.env(df.in = df.coef,
                n.runs = num.runs, t.steps = pre.time.steps,
                p.start = pop.start,
                env = env.condition,
                dem.het = FALSE,
                lambda.opt = lambda.optimum)
```

Perturbing equilibrium populations with poisson noise (single draw for each population from a poisson distribution with rate parameter equal to that population size + 2. We add 2 to allow extinct populations to potentially re-colonize in our simulation.)

```
pop.eq <- filter(df.eq, time == max(time)) # final time step
pop.eq.dist <- rpois(length(pop.eq$pop), pop.eq$pop + 2)
pop.eq.dist.df <- data.frame(run.count, pop.eq.dist)
```

Running the simulation stochastically for one time step from those perturbed population values. This is the final data used for the sparse model.

```
df.result <- BH.env(df.in = df.coef,
                    n.runs = num.runs, t.steps = 1,
                    p.start = pop.eq.dist.df,
                    env = env.condition,
                    dem.het = TRUE,
                    lambda.opt = lambda.optimum)
```

Visualizing and saving output

Storing the output

```
# write.csv(df.coef, file = "parameters_x.csv")
# write.csv(df.result, file = "simulation_x.csv")
# write.csv(df.eq, file = 'warmup_x.csv')
```

Visualizing warmup

Only the first 20 runs–check that we are converging to equilibrium

```
ggplot(filter(df.eq, run < 21), aes(x = time, y = pop, color = species)) +
  facet_wrap(vars(run)) +
  geom_line() +
  theme_classic() +
  xlab('Timestep') +
  ylab('Population')

# ggsave('warmup_x.pdf', width = 6, height = 4, units = 'in')
```

Environmental interactions

Rough visualization of output of final two time steps and the input coefficients

```
df.result.wide <- df.result %>%
  pivot_wider(names_from = time, names_prefix = "time.", values_from = pop) %>%
  left_join(df.coef, by = 'species')

df.result.wide$Fec <- (df.result.wide$time.1 - df.result.wide$time.0)/(df.result.wide$time.0 + 1)

ggplot(filter(df.result.wide, species %in% c(2, 9, 13)),
       aes(x = run.env, y = Fec, color = species)) +
  geom_point() +
  geom_smooth(se = FALSE)

df.coef
```