Model code, general instructions

The model presented in the main text of the Methods in Ecology and Evolution paper “Inferring community assembly processes from macroscopic patterns using dynamic eco-evolutionary models and Approximate Bayesian Computation (ABC)” is implemented in its basic form as MATLAB (version R2017b) code. In order to run the model “main_ecoevo.m” should be executed in the same directory as the m-files briefly presented below. The code is commented within each m-file, below we present general features and key components of the implementation.

main_ecoevo.m

This is the main function that is executed in order to run the model. Default model parameters and initial conditions are initiated (lines 25-69), or parameters can be assigned as input to the function (see details below). Initiation is essential for the scenario that one wants to model. Habitat variables dictate the number of habitats and their position in resource/ trait space. Competition variables initiate the number of competitor populations, their position, and abundance in the habitats, their traits, niche width, dispersal propensity, and evolvability. Similarly, predator variables dictate the number of predator populations, their position, and abundance in the habitats, their traits, niche width, dispersal propensity, and evolvability. This flexibility provide possibility to run multiple model scenarios. For example, like the scenarios presented in the case study of this paper, it is possible to reduce the general model to a model of predator only and predator-prey adaptive radiations in one habitat. This is done through modification of parameters N0 and P0, such that the system is seeded with prey only or with prey and predators in one habitat only and by setting dispersal (mN and mP) to zero. The program can be run from the MATLAB command line without any input arguments (default parameter values will used) or as a function with input arguments for:

- Prey abundance (N0) in habitat 1,2, and 3 (first 3 input arguments)
- Prey dispersal (mN, fourth input argument)
- Prey niche width (sigma_a, fifth argument)
- Predator abundance (P0) in habitat 1, 2, and 3 (sixth to the eighth argument)
- Predator dispersal (mP, ninth argument).
- Predator niche width (sigma_b, tenth argument)
- Index for replicate (the eleventh argument, will show in saved output)

This allows for runs with different niche widths as in the case study and the model can be run with prey only or with prey and predators in one, two or three habitats respectively. For example, if the function is called as “main_ecoevo(2,0,0,0,0.5,1,0,0,0,0.4,1)” the model will start with 2 prey individuals and 1 predator individual in habitat 1, there will be no dispersal, and prey and predator will have niche width 0.5 and 0.4 respectively. Note that if no arguments are assigned then default values will be run. The program will let the user know if default values are used at startup. Also, if input arguments are assigned, all 11 arguments listed above need to be assigned or else default values will be used. The program will let the user know what input arguments have been assigned by the user, or if default values are used, at startup.

At execution, the code computes equilibrium population sizes for the initiated populations and removes populations that do not have positive population size at equilibrium. Thereafter, the code moves into the evolutionary part, looping over evolutionary time. Each evolutionary time step: 1) Populations mutate (phenotypic trait change). 2) Invasion fitness for mutants is computed. 3) Mutants with positive invasion fitness are allowed to invade. 4) Mutual invisibility between mutant and resident morphs is computed. 5) Mutants are allowed to invade alongside the resident (if mutual invisibility exists) or replace the resident (if mutual invisibility does not exist). 6) Equilibrium population size is re-computed, species are defined and extinct populations are removed. Thereafter the algorithm moves into the next evolutionary step. The main script outputs (see lines473- 485) trait distribution data, abundance data, phylogenetic data, and ecological variables (e.g. interaction strengths).

The possibility for the function to take arguments as input also facilitates atomized model output for different parts of parameter space or for different simulation realizations (e.g. replicates). For example, in the case study of this paper, for Approximate Bayesian Computation, we called the function with input for the prey and predator niche widths, drawn randomly from the prior distribution and we did this for multiple replicates on a computer cluster. Different scenario and replicate combinations will commonly be initiated on nodes of the cluster by the cluster-specific initiation script. Running replicates on a cluster will also require some individualization of output name and bookkeeping of outputs for further ABC analysis (see example in **extract_SS_fromSimData.m** below). Output individualization is facilitated by the “Index for replicate” -argument (eleventh argument) which will be specified in saved output.

Below we explain the functions that are called by main_ecoevo.m in the order that they are called.

pop_dyn_func.m and **ode_sys_popeq.m**

The function pop_dyn_func.m computes the population dynamics and population abundances at equilibrium (input argument defines how long time the dynamics should be solved for).

This is thus where the equations of the ecological model are implemented. The function outputs 1) time series of the dynamics and 2) ecological variables (e.g. ecological interaction strengths and carrying capacities) that are conditions on the current system (e.g. traits of the populations). Ecological variables are arranged in matrix form according to the form of the general Generalized Lotka-Volterra (defined in ode_sys_popeq.m) and population dynamics is then solved using an ODE solver provided by MATLAB.

PergeFunc.m

Takes equilibrium population size data as input and removes populations that are extinct, trait vectors and other data variables are also cleared from extinct population data.

cons_fitness_func.m and **ode_sys_mutfit_consumer.m**

This function sets up the model for a consumer mutant dynamics in the context of the full resident community and then computes initial growth (fitness) of the mutant while rare. This is done both analytically and with simulation. The simulation (used in our analysis) is based on solving population dynamics for the initial growth of the mutant, using an ODE solver. The ecological equations presented in the main text are implemented here. The function ode_sys_mutfit_consumer.m determine the general model structure for the ODE solver. If the mutant grows initially fitness is positive, else it is negative.

pred_fitness_func.m and **ode_sys_mutfit_predator.m**

Same description as for cons_fitness_func.m but for a predator mutant.

PhyloCheck.m

Registers speciation events by looking for gaps in the consumer and predator trait distributions.

extract_SS_fromSimData.m

This script extracts summary statistics for “true” data and for each simulation of a community that was generated by the **main_ecoevo.m**. For the script to work the use needs to generate the true data using the model. This true data is called and SS are computed on lines 25-64. Then on lines 67 and onwards multiple simulation realizations are collected and SS are computed. For this to work the user need to make sure that all simulated data are in the same directory as this script, or the code needs to be modified such that a specific directory is specified. Finally, the distance between true data and each simulated realization is computed on lines 132 and onwards.

The output will be a matrix containing the parameters (first two collumns) of the simulation, followed by values on:

No. prey

Mean abundance of prey

Width of the prey trait distribution

Mean trait distance MTD

mean nearest trait distance MNTD

Mean phylogenetic distance

Nearest neighbor phylogenetic distance

extraxt_SS_plotResults.m

This script loads the output from **extract_SS_fromSimData.m** and plots posterior distributions, given acceptance thresholds that are specified by the user in the script. Note that the names of the outputs from the previous script needs to match the names of the files that are loaded in this script.