resevol: an R package for spatially explicit models of pesticide resistance given evolving pest genomes

A. Bradley Duthie[1][2], et al.

[1] Biological and Environmental Sciences, University of Stirling, Stirling, UK [2] alexander.duthie@stir.ac.uk, Biological and Environmental Sciences 3A149 University of Stirling Stirling, FK9 4LA, UK

Abstract

1. Pesticide resistance is a serious problem.
2. We introduce the resevol R package, which runs individual-based models of pests with quantitative and covarying traits and three mating systems.
3. Simulations are on a landscape where crops and pesticides are changed.
4. We give an example of simulating pests with covarying traits
5. The resevol R package is open source under GNU Public License; source code and documents are freely available on GitHub.

Key words: *pest management*, *food security*, *ecological modelling*

Introduction

- Pest resistance to pesticide is a serious wicked problem that affects food security.
- A quantitative genetic approach to pesticide resistance to maintaining resistance.
- The resevol R package addresses two problems: genetic architecture and landscape selection
- The R package can facilitate general questions and targetted ones

The resevol package uses individual-based modelling and a quantitative genetics approach to simulate the evolution of a pest population on a changing landscape. A focal goal of the software is to model traits with a pre-specified, but potentially evolving, covariance structure. To achieve this goal, each individual has a genome with L loci that underlie a set of T potentially evolving traits. Pleiotropic loci can vary in their effects on the direction and magnitude of polygenic trait, causing population-wide trait covariance to arise mechanistically from the underlying genetic architecture of individuals. To achieve this, two separate steps are necessary. First, a genetic algorithm is used to find a network of internal nodes that map standard random normal loci values to covarying traits. Values used to map loci to traits are incorporated into individual genomes. Second, a population of asexual or sexual individuals is initialised and simulated on a spatially explicit landscape separated into distinct units (e.g., farms). Land units can apply one of up to 10 pesticides, and one of up to 10 crops; pesticides and crops rotate independently in a pre-specified way over time. The resevol package can thereby model complex and evolving traits in agricultural pests over realistic landscapes that undergo different pesticide use and crop regimes.

Covarying pest quantitative traits

The first step of simulation with the resevol package is building individual genomes. This step is separate because building genomes is often time-consuming, and genomes that are built might need to be saved or inspected before actual simulation. High computation time is due to the mechanistic nature of how genomes and covarying traits are modelled. Instead of imposing a trait covariance structure directly, the objective is to use an evolutionary algorithm to find a network that maps standard random normal values (loci) to covarying values (traits). This is useful because it allows genomes to model potentially evolving physiological constraints, and trade-offs among traits, from the bottom up. Since multiple networks can potentially map loci to the same trait covariance structure, it is possible to replicate evolution with differently randomised genetic architectures. This approach to modelling individual genomes and traits thereby increases the complexity of questions that can be addressed for simulating evolution in agricultural pests.

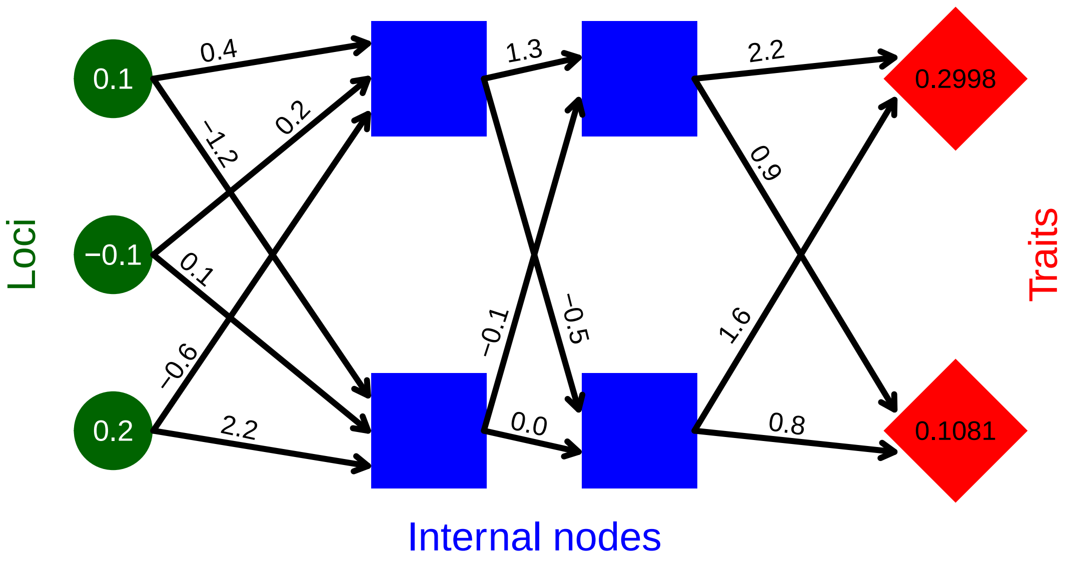

Figure 1: Example network mapping loci (green circles) to traits (red diamonds) through an intermediate set of hidden layers (blue squares) in the mine_gmatrix function. Individual genomes in the resevol R package consist of standard random normal values for at loci, real values for black arrows linking nodes, and real values of traits. Values shown for loci and arrows are an example for illustration.

Conceptually, the relationship between individual genotypes and traits is defined by a network connecting loci to traits through a set of hidden internal nodes (Figure @ref(fig:network)). Values for loci are randomly drawn from a standard normal distribution $\mathcal{N}(0, 1)$. Links between loci, internal nodes, and traits, can take any real value and are represented by black arrows in Figure @ref(fig:network). Traits can take any real value, and are calculated as the summed effects of all preceding nodes (i.e., the blue squares immediately to the left of the traits in Figure @ref(fig:network)). Mathematically, loci are represented by row a vector of length L. Effects of loci on the first layer of internal nodes (black arrows emanating from loci in Figure @ref(fig:network)) are represented by an $L \times T$ matrix, and transitions between internal nodes, and between the last set of internal nodes and the final traits, are represented by $T \times T$ matrices. For the example in Figure @ref(fig:network),

$$\begin{pmatrix} 0.1, & -0.1, & 0.2 \end{pmatrix} \begin{pmatrix} 0.4, & 0.1 \\ -1.2, & -0.6 \\ 0.2, & 2.2 \end{pmatrix} \begin{pmatrix} 1.3, & -0.5 \\ -0.1 & 0.0 \end{pmatrix} \begin{pmatrix} 2.2, & 0.9 \\ 1.6 & 0.8 \end{pmatrix} = \begin{pmatrix} 0.2998, & 0.1081 \end{pmatrix}.$$

Values mapping loci to traits become part of an individual's genome, so the genome for the individual represented by Figure @ref(fig:network) is stored in the model as below:

```
0.1, -0.1, -0.2, 0.4, 0.1, -1.2, -0.6, 0.2, 2.2, 1.3, -0.5, -0.1, 0.0, 2.2, 0.9, 1.6, 0.8
```

Individuals with different loci can therefore have different covarying traits that are constrained by the network structure encoded in each genome. Individuals can be haploid (as in Figure @ref(fig:network)) or diploid (in which case, allele values are summed at a locus).

An evolutionary algorithm is used to find appropriate values that produce covarying traits from loci (see Supporting Information for details). Evolutionary algorithms are heuristic tools can simulate adaptive evolution to find solutions for a broad range of problems (Hamblin 2013; Duthie et al. 2018). In the resevol package, the `mine_gmatrix` function runs an evolutionary algorithm, and requires the argument `gmatrix`, which specifies the desired trait covariance matrix. The function runs a genetic algorithm that initialises a population of `npsize` networks (Figure @ref(fig:network)), and this population evolves until some maximum generation number (`max_gen`) or minimum expected network stress (`term_cri`) is met. In a single generation of the algorithm, network values crossover and mutate. Next, trait covariances produced for each network are estimated by initalising `indivs` individuals with loci sampled from a standard normal distribution. Network stress is calculated as the logged mean squared deviation between estimated covariances and those in `gmatrix`. Tournament selection (Hamblin 2013) is then used to determine the networks for the next generation of the algorithm. Networks with the lowest estimated stress have the highest fitness, so these networks are disproportionately represented in the next generation. Throughout the evolutionary algorithm, the lowest stress network is saved and output upon network termination. The robustness of this network's stress to sets of individuals with different loci values can be tested using the `stress_test` function.

An example run of `mine_gmatrix` with the same number of loci and internal nodes is shown below with traits that do not covary:

```r
trait_covs  <- matrix(data = c(1, -0.4, -0.4, 1), nrow = 2, ncol = 2);
new_network <- mine_gmatrix(loci = 3, layers = 2, gmatrix = trait_covs,
                            max_gen = 1000, term_cri = -6.0, prnt_out = FALSE);
```

The code above found a genome that produced the following expected trait covariances:

```
##            [,1]       [,2]
## [1,]  0.9496718 -0.4763565
## [2,] -0.4763565  1.0101641
```

The mean deviation between elements of the above matrix and the identity matrix provided by `trait_covs` is 0.0035742. Lower values of `term_cri` and higher values of `max_gen` will result in a lower stress, but this will require more computation time, especially if the number of traits is high. Similarly, higher values of `indivs` will result in more accurate estimations of true stress, but this also requires more computation time. Additional arguments to `mine_gmatrix` can also be used to improve the performance of the evolutionary algorithm (see Supporting Information).

Simulating landscape-level pesticide resistance

The full output of `mine_gmatrix` is passed to the `run_farm_sim` function, which initialises and simulates an evolving population of pests on a changing landscape for a natural number of time steps (`time_steps`). In this section, we explain the landscape, pest ecology, and evolving pest traits.

Landscape

Landscapes are spatially explicit and initialised in `run_farm_sim` in one of two ways. First, a landscapes can be built from the arguments `xdim`, `ydim`, and `farms`. These arguments specify the dimensions of the landscape and the number of farms on it. Contiguous rectangular farms of roughly equal size are generated on the landscape using a splitline algorithm. Second, a custom landscapes can be input using the `terrain` argument, which takes a matrix with elements that includes integers 1 to `farms`. Each value defines a unique farm, but values do not need to be contiguous. These 'farms' could even model non-farmland (e.g., water, roads), if pesticides and crops on them are invisible to pests (see below). This `terrain` customisation therefore allows for a high degree of landscape detail, and offers the potential for modelling real-world landscapes from raster images.

Each farm can hold one pesticide and one crop type at any time step. The `crop_init` and `pesticide_init` arguments initialise one of `crop_number` crops and one of `pesticide_number` pesticides for each farm, respectively (maximum of 10 each). Initialisation can be random for each farm with equal probability, or can be set using a vector of length `farms` in which vector elements define the initialised crop or pesticide number. After initialisation, crops and pesticides rotate once ever `crop_rotation_time` and `pesticide_rotation_time` time steps, respectively. Each of these rotation time arguments can take either an integer value from 1-3, or a matrix. Integer values specify (1) no rotation, (2) random transition from one type to another, or (3) cycling through each available crop or pesticide in order. Square matrices can be used to define the probability that a given crop or pesticide in row i transitions to that in column j. Hence, any possible Markov chain can be used to transition between crop or pesticide types on farms.

Pest ecology

Individual pests can be modelled to have several reproductive systems and life histories. Pest reproductive system can be specified in the `run_farm_sim` function using the `repro` argument, which accepts arguments “asexual” (haploid), “sexual” (monoecious), and “biparental” (dioecious). For “sexual” pests, the `selfing` argument specifies if self-fertilisation is (`TRUE`) or is not (`FALSE`) allowed. At the start of a simulation, pests are initialised in a random location. Initialised pests are age of zero if `rand_age = FALSE` or a random age from zero to `max_age` if `rand_age = TRUE`. Following initialisation, a single time step proceeds with landscape change (see above), pest aging, feeding, pesticide consumption, movement, reproduction, mortality, and immigration. Feeding, pesticide consumption, movement, and reproduction all depend on pest age. Pests feed and consume pesticide from ages `min_age_feed` to `max_age_feed`, move from ages `min_age_move` to `max_age_move`, and reproduce from ages `min_age_reproduce` to `max_age_reproduce` (all inclusive). Food accumulated is lost during aging if `baseline_metabolism > 0` and pest age is within `min_age_metabolism` and `max_age_metabolism`. The option to set minimum and maximum ages for events makes it possible to model pests with much different life histories (e.g., Sudo et al. 2018).

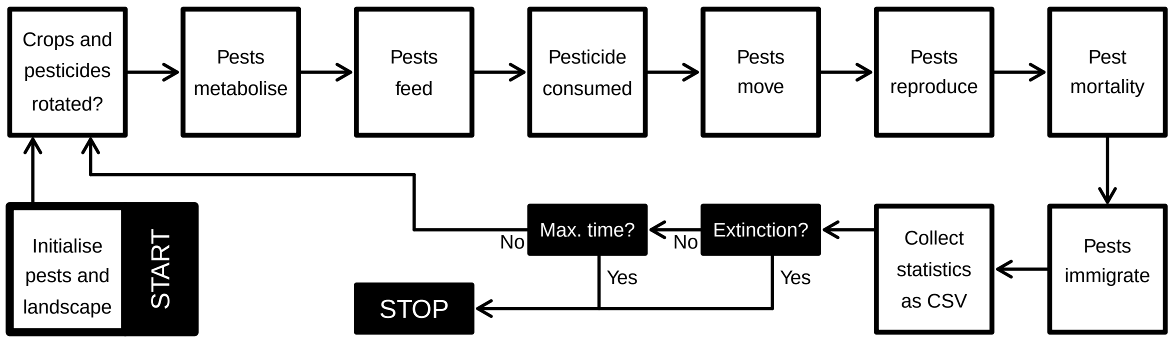

Figure 2: Overview of simulated events in the resevol R package. Note that metabolism, feeding, pesticide consumption, movement, and reproduction are all subject to a minimum and maximum pest age. Consequently, simulation order might not reflect the order of events from the perspective of a focal pest (e.g., pests might move from ages 1-2, but only feed from ages 2-4). Crops and pesticides are also not necessarily rotated in each time step (see Landscape). Statistics collected within a time step are printed in a CSV file.

Pests feed by

Pesticide is consumed during feeding

Movement is modelled by

Pests reproduce by

Immigration occurs.

- We want to model evolution of pesticide resistance across heterogenous landscape
- Track entire genomes, traits, and yield over time and across space
- Use covarying traits from `mine_gmatrix` to make evolving pests
- Options for 10 crops and 10 pesticides simultaneously, and rotation
- Function `run_farm_sim` runs simulations. Note unusual timing structure

Pest evolution

Example of individual-based simulations

- Example of evolving species with different crop and pesticide applications

Conclusions

Availability

The resevol R package can be downloaded from CRAN (https://cran.r-project.org/package=resevol) or GitHub (https://bradduthie.github.io/resevol/). The package is open source under GNU Public License.

Conclusions

Acknowledgements

This software was developed as part of the project for Enhancing Diversity to Overcome Resistance Evolution (ENDORSE). The ENDORSE project is a joint Newton funded international partnership between the Biotechnology and Biological Sciences Research Council (BBSRC) in the UK and the São Paulo Research Foundation (FAPESP) in Brazil under BBSRC award reference BB/S018956/1 and FAPESP award reference 2018/21089-3. ENDORSE is a partnership among Universidade Estadual Paulista (UNESP), the University of Stirling (UoS), and the Centre for Agricultural and Biosciences International (CABI).

References

Duthie, A Bradley, Jeremy J Cusack, Isabel L Jones, Erlend B Nilsen, Rocio A Pozo, O. Sarobidy Rakotonarivo, Bram Van Moorter, and Nils Bunnefeld. 2018. "GMSE: an R package for generalised management strategy evaluation." *Methods in Ecology and Evolution* 9: 2396–2401. https://doi.org/10.1101/221432.

Hamblin, Steven. 2013. "On the practical usage of genetic algorithms in ecology and evolution." *Methods in Ecology and Evolution* 4 (2): 184–94. https://doi.org/10.1111/2041-210X.12000.

Sudo, Masaaki, Daisuke Takahashi, David A Andow, Yoshito Suzuki, and Takehiko Yamanaka. 2018. "Optimal management strategy of insecticide resistance under various insect life histories: Heterogeneous timing of selection and interpatch dispersal." *Evolutionary Applications* 11 (2): 271–83. https://doi.org/10.1111/eva.12550.