

Methods

Overview—simplex was developed to accomplish three primary objectives. First, simplex assembles and runs individual-based ecological models from random combinations of state variables and processes. Second, simplex stores the recorded output as R-style data frames for future analyses and/or as an image file; e.g., when the user chooses to generate a movie of a simplex model. Third, simplex provides R-based files for analyzing simulated; these will eventually be offered in Python as well.

Source code:—To save space, I refer the reader to the README.md file on the public simplex GitHub.com repository (<https://github.com/LennonLab/simplex/blob/master/README.md>) for descriptions of source code files and directories.

Model description following the ODD Protocol

The ODD protocol is standard for describing individual-based models (Grimm et al. 2006). ODD stands for Overview, Design concepts, and Details. Here, I describe simplex *largely* according to the ODD protocol. The ODD protocol for simplex can also be found on the public GitHub repository at <https://github.com/LennonLab/simplex>

Purpose

The purpose of simplex is to simulate life history of individual organisms, the assembly of ecological communities, and the evolution of traits in spatially explicit environments under stochastic conditions. The proximate goal of simplex is to generate high degrees of variation in the assembly and structure of populations and communities by assembling many different models from random combinations of state-variables and processes. The ultimate goal of simplex is to provide a simulation-based platform for examining conditions under which processes and constraints have a robust influence on eco-evolutionary dynamics and patterns of biodiversity.

Entities & their state variables

Individual organisms—Individuals are distinguished by collections of elements within lists. Individuals undergo changes when randomly sampled from lists. Each specific position in the list corresponds to the same individual. For example, in simulating growth, a simplex model chooses an individual from the list of cell quotas (the probability of reproducing is determined by endogenous resources). The first position in this list as well in all other lists of individual attributes corresponds to the same individual.

Example:

```
IndIDs = [1, 2, 33, 14]
Quota = [0.1, 0.99, 0.14, 0.05]
Xpos = [45, 23, 456, 1]
Ypos = [765, 87, 21, 34]
```

The individual with ID of 1 has a cell quota of 0.1 and is located at position x=45, y=765

These are the lists of attributes for individual organisms:

- time each particle spends in the system (aka residence time)
- species ID

- individual ID
- 2D spatial location
- endogenous levels of 3 resources (resource specific cell quotas)
- individual-level metabolic maintenance cost
- individual-level maximum dispersal rate
- pedigree, i.e., direct lineage
- individual resource use efficiency for each resource
- products of individual-level metabolism

Species—Each species is characterized by the individuals that share a common set of traits, such as maximum growth rate, metabolic maintenance cost. Species information is stored in Python dictionaries. In this way, if simplex requires the species ID of an individual it will access the species ID list where each element corresponds to a specific individual. But, if simplex requires the maximum growth rate for an individual, then it finds the species ID and then uses that to access the Python dictionary for maximum specific growth rates of species.

Example:

```
IndIDs = [1, 2, 33, 14]
spIDs = [12, 32, 11, 6]
MaxGrowthDict = {12: 0.9, 32: 0.5, 11: 0.8, 6: 0.4}
```

The individual with ID of 1 belongs to species 12, which has a theoretical maximum growth rate of 0.9.

The following are the types of species-level information that are stored in Python dictionaries:

- metabolic maintenance cost
- maximum dispersal rate
- maximum theoretical growth rate
- resource use efficiency for each resource

Resource particles—Individual resource particles are distinguished by collections of elements within lists. Each specific position in the list corresponds to the same resource particle.

Example:

```
resIDs = [4, 6, 17, 1]
size = [100, 87, 156.3, 0.001]
Xpos = [56, 34, 567, 2]
Ypos = [876, 98, 32, 45]
```

The particle with ID of 4 has a size of 100 and is located at position x=56, y=876

The following are the types of information stored about each resource particle:

- time in the system
- particle ID
- 2D spatial location
- whether the particle is Nitrogen, Carbon, or Phosphorus
- type of Nitrogen, Carbon, or Phosphorus the resource is
- size of the particle

Inert tracers particles—These are objects that move/flow into and through the environment. They only interact with physical barriers. The use of tracers particles allows for attributes of a system that flows or physically turns over to be quantified (e.g., hydraulic residence time). Like other individual-level objects in simplex models, tracers are distinguished by collections of elements within lists. Each specific position in the list corresponds to the same resource particle.

This information stored about each resource particle:

- time in the system
- 2D spatial location
- particle ID

Physical barriers—These objects are simulated as discrete 2D spatial coordinates that cannot be occupied by any individual entities. The number, size, and location of physical barriers are chosen at random by simplex at the start of each model.

System level state variables

Each run of simplex begins with random choices for the values of:

- width (5 to 100)
- height (5 to 100)
- basal flow rate (1.0 to 0.0)
- number, size, and location of physical barriers
- number and direction of environmental gradients
- rate of stochastic disturbance
- rate of fluctuation in basal flow rate
- degree of fluctuation
- degree of synchronized flow of individual particles
 - completely in-sync or completely out of sync

For example

```
from randparams.py:
```

```
width = randint(5,100)
height = randint(5,100)
```

```
# number of barriers
barriers = randint(1,10)
```

```
# log-series alpha for metacommunity structure
alpha = np.random.uniform(0.99, 0.999)
```

```
reproduction = choice(['fission', 'sexual'])
speciation = choice(['yes', 'no'])
```

```
# size of starting community
seedCom = choice([10, 100, 1000])
```

```
# m = probability of immigration
m = choice([0.0, 0.0001, 0.0005, 0.001, 0.005])
...
```

Spatial and temporal scales

The two general aspects of scale are grain (a.k.a. resolution) and extent (e.g. total area).

Spatial extent—The environment of simplex models is two dimensional and can vary along each axis from 5 to 100 discrete units. This makes for a potential total extent of 25 to 10,000 discrete patches, each with a grain of 1 square unit.

Note that all particles move in decimal units the limit of which is determined by Python’s decimal precision. This means that individual particles can occupy practically infinite locations within patches and likewise, squeeze through barriers (which only occupy integer x-y coordinates).

Temporal extent—Extent of time in simplex models refers to residence time, i.e., the average amount of time that individual particles spend in the system. Residence time for inert tracer particles can vary across five orders of magnitude.

Grain—Grain is the smallest unit over which change can happen. For example, as per the original ODD documentation: “One time step represents one year and simulations were run for 100 years. One grid cell represents 1 ha and the model landscape comprised 1,000 x 1,000 ha; i.e., 10,000 square kilometers”. In contrast, the smallest grain achievable by simplex is determined by slowest rate at which individuals can undergo BIDE (birth, immigration, death, emigration) processes. For example, under high residence times, an individual can move across 0.00001% of the x or y axis in one time step. Under low residence times, an individual can move across 10% or greater of the x or y axis in one time step.

Process overview and scheduling

Assembly—The user runs a program that chooses random values for system-level state variables including whether disturbance, immigration, speciation, fluid dynamics, etc. will occur and at what rates.

Core simulation process—simplex models begin simulation immediately after assembly from random combinations of state-variables and processes. Instead of operating by definitive time steps (i.e. days, generations), simplex models advance turnover of the environmental matrix according to the initial rate of flow. If the initial rate of flow is 1.0, then the environmental matrix and inert particles would flow 1.0 units of distance. After each iteration of flow, each individual is given the chance to consume, grow, reproduce, starve, die, and to disperse towards resources and environmental optima.

Duration: A run to mean reversion—Once assembled, a simplex model simulates ecological processes (birth, death, dispersal, growth, consumption, etc.) until the system reaches a fluctuating equilibrium determined by a point of mean reversion (quantified by Hurst’s exponent). Mean reversion captures the tendency of a system to repeatedly reverse a directional change in, say, total abundance. The system examines whether a point of mean reversion has occurred by recording the total abundance of the system each time a tracer particle exits the system. This ensures that, at the least, enough time has passed for an inert particle to enter and exit the system.

Fluid dynamics

simplex uses an efficient and powerful method for simulating fluid flow, i.e., a Lattice-Boltzmann Method (LBM). An LBM discretizes the environment into a lattice and attaches to each position in the lattice a number of particle densities and velocities for each of nine directions of movement possible in a 2D environment (N, S, E, W, NE, NW, SE, SW, current position).

Active dispersal—simplex models allow individuals to move towards their environmental optima. Rather than a single environmental optima resulting from a single environmental gradient, simplex allows environmental optima to occur as intersections among environmental gradients. Hence, individuals potentially have multiple optima resulting from unique and equally optimal intersection of up to 10 environmental gradients.

Simulated life history—simplex models simulate growth, reproduction, and death via weighted random sampling. This simulate the partly probabilistic and partly deterministic nature of environmental filtering and individual-level interactions.

Inflow/Entrance: Resources and individuals enter from any point in the environment. Species identities of inflowing propagules are chosen at random from a log-series distribution, which often approximates the distribution of abundance among species in ecological communities (see Hubbell 2001). Along with a species ID and species-specific maximum rates of resource uptake, active dispersal, resource efficiencies, cell maintenance, and environmental optima, each propagule was given a unique ID and a multi-resource cell quota that represent the state of internal resources. The average of these cell quotas determine the probability of reproduction.

Dispersal: Individuals are allowed to actively move along environmental gradients (sometimes against the direction of environmental flow) and towards their optima. A better match to one's environmental optima increases the chance of reproduction and the individual's ability to perform (consume, grow).

Consumption & growth: Sampled individuals consume resources according to their specific maximum rates of uptake and grow according to specific resource efficiencies and maintenance costs. Uptake increases the individual cell quotas and decreases ambient resources. Individual cell quotas are then decreased according to a specific physiological maintenance cost.

Reproduction: Reproduction in simplex is currently limited to clonal reproduction with the possibility of mutation. Individuals reproduce with a probability determined by the combination of mean cell quota and the proportional match to the environmental optima. The cell quota of each resource is evenly divided between the individual and its daughter. The daughter is given a unique individual ID and the species ID of its mother, unless in the case of speciation, but is allowed small mutations in individual-level state variables.

Speciation: Speciation is simulated within simplex as a discrete event but is accompanied by mutations in the values of species-level state variables. This allows for diversity to arise within the system, which the environmental filter can then select on.

Death: Individuals sampled at random will die if their smallest resource specific cell quota (i.e., N , C , P) is equal to or less than 0.

Emigration: Individuals, resource particles, and inert tracers are considered to have left or to have flowed out when they pass beyond edges of the environment.

Design concepts

Basic principles. *Ecological complexity:* simplex assembles models from random combinations of constraints (state-variables) and processes to generate output data that allow the user to test the general influence of a particular state-variable, process, or combination thereof using univariate and multivariate tests.

Nutrient limited growth: All models assembled by simplex employ the universal concept that individual growth and activity is fueled and limited by resources.

Resource diversity & heterogeneity: simplex allows the user to explore the influence of the number and abundances of different resources on ecological diversity and ecosystem processes.

Theories *Constraint-based theory:* simplex was originally built to explore the influence of ecosystem residence time (volume/flow rate) on community assembly and structure. That is, the idea that both ecological processes and constraints shape ecological diversity.

Chemostat theory: simplex operates much like an unhinged bioreactor or chemostat. That is, particles flow through a system of a defined size at an average rate, and are limited in growth by their residence time.

Ecological neutral theory: simplex operates via random sampling and can vary from being completely neutral (all individuals having equal vital rates) to completely idiosyncratic (all individual and species are as different as possible). The one aspect of neutral theory that simplex adopts without question is the importance of stochastic life history processes (i.e. weighted or unweighted random fluctuations in population sizes).

Hypotheses * These are entirely up to the user to formulate and test according to simplex’s capabilities and analytical tools.

Modeling approaches Simplex operations via two main modeling approaches, other than being individual-based, i.e., random sampling and computational fluid dynamics.

Emergence simplex uses random sampling and random assembly its models to avoid imposing strong constraints on the properties that emerge and to allow unanticipated combinations of traits and ecological structure to emerge.

- Abundance
 - Total community abundance
 - Trait-related population size
 - Effective population size
 - Abundance-biomass relation
- Community assembly
 - Species turnover
 - Biomass turnover
 - Extinction rate
 - Succession
- Community structure
 - Species richness
 - Species evenness
 - Trait diversity
- Population structure
 - Demography
 - Subpopulation trait variation
- Life history tradeoffs
 - Mobility vs. metabolic maintenance
 - Growth rate vs. metabolic maintenance
 - Generalist vs. specialist
 - R vs. K selection

Adaptation Individuals can move towards their environmental optima. Populations can become aggregated in areas that provide favorable intersections of species optima. Species can evolve by the action of the environmental filter on subpopulation variation in state variables.

Objectives Individual seek conditions that match them to the environment (e.g., positions along environmental gradients). Individuals also seek to acquire resources through active searching. In the future, individuals will seek to avoid predation.

Learning There is no aspect of individual-based learning in simplex

Prediction Individuals in simplex do not have the ability to anticipate conditions.

Sensing Individuals only sense in the sense that they can move towards environmental optima and, in the future, resources. Otherwise, all encounters are the result of random walks or fluid flow.

Interaction At the moment, individuals only interact indirectly through excluding each other from resources (e.g. preemption). In the future, individuals will interact as predator-prey, mutualists, resource-dependents, etc. Likewise, there is currently no communication, though quorum sensing would be cool.

Stochasticity The occurrence of nearly all processes of birth, death, life, immigration, dispersal, emigration, consumption, etc. are conducted via random sampling. In this way, population and community dynamics result, in part, from demographic stochasticity. Likewise, the emergence of life history traits proceeds from initially random combinations of traits.

Collectives Individuals belong to species. Species belong to communities. In the future, simplex will allow communities to belong to trophic levels.

Observation Many simplex models should be run to examine trends in the variation generated. The following is recorded for each simplex model:

- Values of randomly chosen state variables
- Total abundance, N
- Species richness, S
- Avg growth rate (per species & per capita)
- Avg maintenance cost (per species & per capita)
- Avg resource efficiency (per species & per capita)
- Avg active dispersal rate (per species & per capita)
- Compositional turnover
 - Bray-Curtis
 - Sorensen's
- Species turnover
 - Whittaker's β
- Species evenness
 - Smith and Wilson's evenness, E_{var}
 - Simpson's evenness, $E_{1/D}$
- Species diversity
 - Shannon's diversity, H'
 - Simpson's diversity, $D_{1/D}$
- Dominance
 - Absolute, N_{max}
 - Relative, N_{max}/N
- Productivity
 - Individuals
 - Carbon, Nitrogen, Phosphorus
- Residence time
 - Ecosystem, $(length * width) / flowrate$
 - Individual, avg time spent in the system before dying or emigrating
 - Resource, avg time spent in the system before washing out or being consumed
 - Tracer, avg time spent in the system before washing out
- Individual residence time distribution (RTD)
- Resource particle RTD
- Tracer particle RTD

These data are stored in file as R-formatted data.frames. These files can be directly imported into an R or Python environment.

Initialization

The model initiates with a random set of values for state- variables, 100 to 10,000 randomly drawn individuals from a theoretical log-series metacommunity. These values are saved, so that a simplex model could be programmed to replicate an analysis.

Input data

simplex models require no input data, but it might be cool to use an api to grab environmental data or other data from a website to parameterize a simplex model.

Submodels & Equations

Cell quota model of Droop In simplex models, individuals grow according to their amounts of endogenous resources (cell quota).

Droop (1968, 1983) gave a relationship between specific growth rate (μ) and cell quota (Q):

$$\mu = \mu'_m(1 - k_q/Q)$$

where k_q is the minimum cell quota needed for life, also referred to as the subsistence quota. μ' is the

Maintenance cost of Pirt Pirt (1965) states “The variation, with growth rate, of the yield of organism from the substrate used as energy source is attributed to consumption of energy at a constant rate for cell maintenance.” He derives a relationships between the growth yield (biomass), the growth rate, and metabolic maintenance.

simplex models use Pirt’s concept of a constant maintenance requirement. simplex also draws from Pirt’s simple relation for substrate use:

$$use(total) = use(maintenance) + use(growth)$$

Respiration and activity without growth is not accounted for.

Log-series metacommunity simplex models draw immigrating individuals from a theoretical log-series distribution. Hubbell (2001) states that the regional community (i.e., metacommunity) often has a log-series species abundance distribution. The probability density function (pdf) of the log-series is:

$$f(k) = -1/\ln(1 - p) * p^k/k$$

Hubbell (2001) provides explicit detail of the log-series, which is also covered in most ecological diversity texts and even on Wikipedia: https://en.wikipedia.org/wiki/Logarithmic_distribution

Notes on simplex source code

simplex models operate primarily on lists in a programmatic way, e.g., quickly sorting lists, and removing and returning an element from lists with very little overhead. Likewise, simplex models generate and hold a lot of information about all the particles and elements in the system, which can become a computationally intensive task. To this end, simplex modeling coded is written in Python.

Python is an easy to interpret high-level programming language that has many scientific, plotting libraries, and animation libraries. Python gives greater control over the operating system than data analysis languages (e.g. R, Matlab). Likewise, data analysis languages tend to be comparatively slow at purely computational

tasks and can greatly limit the amount of memory held in any data object and fail to import large amounts of data. Python can also obtain C-like speeds when implementing certain software, e.g., Cython, Sage.

The output of simplex is a broad array of information held in seven .csv files. The most important of these is SimData.csv, and is intended to be analyzed in the freely available R (<https://www.r-project.org/>) and RStudio (<https://www.rstudio.com/>) environments. The R statistical computing language is well-suited to the analysis of simplex output and contains many packages for multivariate analysis and higher-order statistical analysis that Python is only beginning to accumulate. Consequently, we provide R source code in .R files and .Rmd (RMarkdown) files, complete with basic and advanced statistical analyses for analyzing diversity, regression models, ordination, variance partitioning, and for generating pdf documents (via Knitr) that integrate prose, code, and figures for manuscripts.

The reader can view example R-based analysis files that users can use to examine simplex's simulated data: <https://github.com/LennonLab/simplex/tree/master/results/analyses>.