RESEARCH ARTICLE

The Community Simulator: A Python package for microbial ecology

Robert Marsland[1]*, **Wenping Cui**[1,2], **Joshua Goldford**[3], **Pankaj Mehta**[1]

1 Department of Physics, Boston University, Boston, Massachusetts, United States of America, **2** Department of Physics, Boston College, Chestnut Hill, Massachusetts, United States of America, **3** Bioinformatics Program, Boston University, Boston, Massachusetts, United States of America

* marsland@bu.edu

Citation: Marsland R, Cui W, Goldford J, Mehta P (2020) The Community Simulator: A Python package for microbial ecology. PLoS ONE 15(3): e0230430. https://doi.org/10.1371/journal.pone.0230430

Editor: Isaac Klapper, Temple University, UNITED STATES

Received: November 19, 2019

Accepted: February 28, 2020

Published: March 24, 2020

Data Availability Statement: All relevant data are within the manuscript.

Funding: This work was supported by NIH NIGMS grant 1R35GM119461 and a Simons Investigator in the Mathematical Modeling of Living Systems (MMLS) award to PM. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Abstract

Natural microbial communities contain hundreds to thousands of interacting species. For this reason, computational simulations are playing an increasingly important role in microbial ecology. In this manuscript, we present a new open-source, freely available Python package called Community Simulator for simulating microbial population dynamics in a reproducible, transparent and scalable way. The Community Simulator includes five major elements: tools for preparing the initial states and environmental conditions for a set of samples, automatic generation of dynamical equations based on a dictionary of modeling assumptions, random parameter sampling with tunable levels of metabolic and taxonomic structure, parallel integration of the dynamical equations, and support for metacommunity dynamics with migration between samples. To significantly speed up simulations using Community Simulator, our Python package implements a new Expectation-Maximization (EM) algorithm for finding equilibrium states of community dynamics that exploits a recently discovered duality between ecological dynamics and convex optimization. We present data showing that this EM algorithm improves performance by between one and two orders compared to direct numerical integration of the corresponding ordinary differential equations. We conclude by listing several recent applications of the Community Simulator to problems in microbial ecology, and discussing possible extensions of the package for directly analyzing microbiome compositional data.

Background

The last decade has seen a renewed interest in the study of microbial communities. Different environments can harbor diverse communities containing from hundreds to thousands of distinct microbes [1, 2]. A central goal of community ecology is to understand the ecological processes that shape these diverse ecosystems. The diversity and function of ecosystems are affected by a wide variety of factors including energy and resource availability [3, 4], ecological processes such as competition between species [5–8] and stochastic colonization [9–12].

Microbial ecosystems also present several new challenges specific to microbes that are not usually addressed in the theoretical ecology literature. Classical models of community ecology (especially niche-based theories) have traditionally considered ecosystems with a few species and resources [13, 14]. However, microbial ecosystems often have thousands of species and hundreds of small molecules that can be consumed. It is unclear how the intuitions and results from these low-dimensional settings scale to microbiomes. It is known that diverse ecosystems can exhibit distinct emergent features and phase transitions not found in low-dimensional systems [15–18]. Furthermore, classical ecological models usually assume a strict trophic layer separation, ignoring cross-feeding and syntrophy—the consumption of metabolic byproducts of one species by another species. It is now becoming clear that cross-feeding is a central component of microbial ecosystems [19–22] and any ecological model must account for this phenomenon.

For these reasons, there is a need for new ways of understanding microbial ecosystems. One powerful approach for understanding complex systems is through simulations. However, simulating diverse microbial ecosystems presents some unique challenges. First, most ecosystems are mathematically represented by complicated coupled, non-linear ordinary differential equations. Simulating these systems in ecosystems with hundreds to thousands of species and metabolites becomes computationally difficult and time-consuming. Second, these dynamical models have thousands of parameters. One needs a principled and biologically realistic way of choosing such parameters. Third, explaining real data requires incorporating ecological processes such as stochastic colonization that play an important role in shaping community structure and dynamics. Finally, we need to be able to incorporate spatial and population level structures in an experimentally realistic way.

Recently, we presented a powerful minimal model of microbial ecosystems that addresses these concerns [20, 22]. Furthermore, we have found a mathematical mapping between ecological dynamics and constrained optimization that can be used to accelerate simulations of many large ecosystems [23, 24]. In this paper, we present a new open-source Python package for microbial ecology called Community Simulator that implements these theoretical advances, making it easy to simulate complex microbial communities in a variety of experimentally relevant settings.

Implementation

The architecture of Community Simulator is inspired by the parallel experiments commonly performed with 96-well plates, as illustrated in Fig 1. The central object of the package is a `Community` class, whose instances are initialized by specifying the initial population sizes and resource concentrations for each parallel "well," along with the functions and parameters that define the population dynamics. The initial state, dynamical equations and parameters can all be generated automatically from a dictionary of modeling assumptions, or custom-built by the user. Each instance of this class represents an n-well plate, containing n well-mixed, non-interacting communities. Once initialized, the state of the plate can be updated in one of two ways. `Propagate(T)` propagates the system for a time T by integrating the supplied dynamical equations, and `Passage(f)` builds a replacement plate by adding a fraction $f_{\mu\nu}$ of the contents of each old well ν to each new well μ. In the final section below, we will discuss a third method `SteadyState()`, which can find the fixed point of the dynamics in some models without numerical integration.

The package also includes some functions for analysis of the simulation results, including a variety of measures of alpha diversity, as well as extraction of energy flux networks, effective interaction coefficients, and sensitivities to parameter perturbations.

In the following sections, we describe the functionality of each element of the package in turn. We particularly focus on the tools for generating the dynamical equations and the

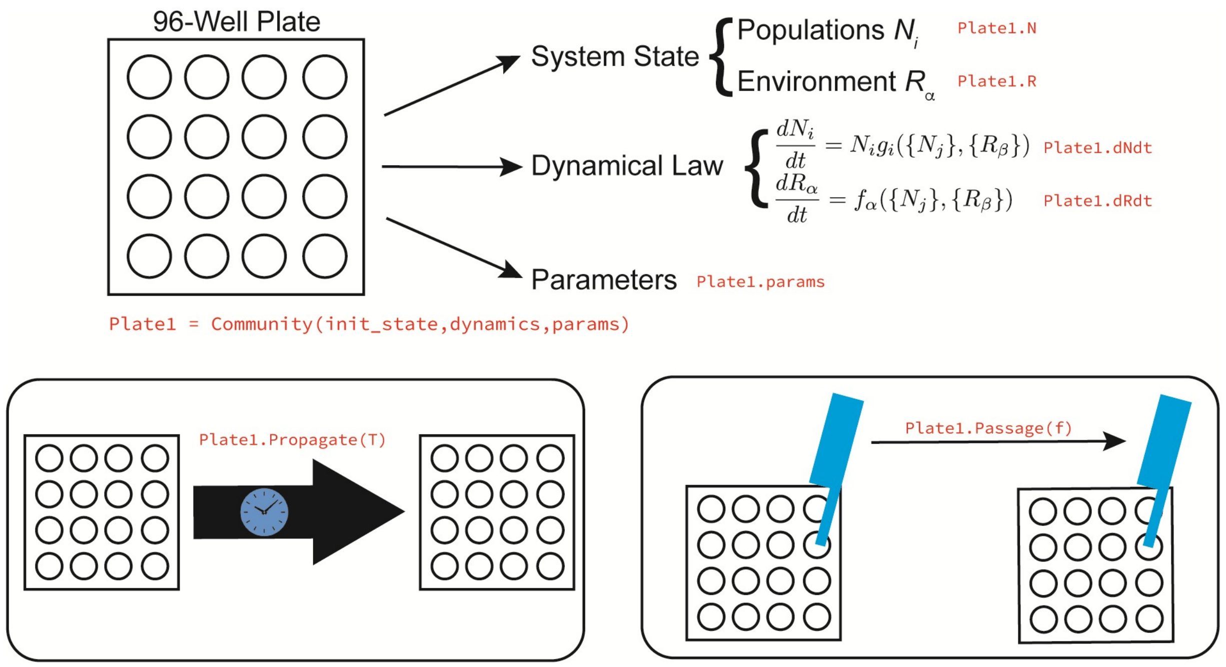


Fig 1. The five elements of the Community Simulator. The core object of the Community Simulator is a virtual n-well plate, holding n independent well-mixed microbial communities. This plate has three properties: its current state, a dynamical law for the population dynamics, and a set of parameters. Once a plate is initialized, two actions can be performed on it: propagation in time using the given dynamical law, and passaging of given fractions of the contents of each well to fresh wells on a replacement plate. For some models, the equilibrium state of the population dynamics can also be found directly using a new algorithm summarized in Fig 5 below.

https://doi.org/10.1371/journal.pone.0230430.g001

parameter sets, explaining how increasing levels of biological realism can be progressively incorporated. Each section has a corresponding segment in the Jupyter notebook `Tutorial.ipynb` included with the Community Simulator package. This notebook contains all the code and parameters for generating the figures found in the paper.

Constructing the initial state

The state of a `Community` instance is contained in a pair of Pandas data frames (https://pandas.pydata.org/, [25]), one of size $S_{\text{tot}} \times n$ for the microbial population sizes N_i ($i = 1, 2, \ldots S_{\text{tot}}$) and one of size $M \times n$ for the resource abundances R_α ($\alpha = 1, 2, \ldots M$). Each row of the data frame corresponds to a different species or resource type, while each column corresponds to a different well.

The function `MakeInitialState` automatically creates data frames `N0,R0` of initial population sizes and resource abundances corresponding to some common experimental scenarios, specified in a dictionary of assumptions. The initial species abundances are supplied by a stochastic process that is agnostic to species identity. This roughly captures the various dispersal mechanisms including mechanical disturbances and turbulent flow that convey microbial cells to new environments. Specifically, random subsets of S species from the regional pool of size S_{tot} are supplied to the n wells of the plate. The population sizes of these species are set to 1 by default, and can be rescaled afterwards if desired.

Initial resource abundances are generated by `MakeInitialState` based on a Biolog plate scenario, where each well is supplied with a single carbon source. The assumptions dictionary specifies the identity and quantity of the carbon source for each well. Arbitrarily chosen resource abundances can of course be directly supplied to the `Community` instance instead, to simulate more general conditions.

To capture coarse-grained metabolic structure, the M resources can be assigned to T classes (e.g. sugars, amino acids, etc.), each with M_A resources where $A = 1, \ldots T$ and $\sum_A M_A = M$. These class labels become functionally relevant by biasing the sampling of consumption preferences and byproduct stoichiometry, as will be described below. Likewise the S_{tot} species can be assigned to F families, with $F \leq T$, and each family preferentially consuming resources from a different resource class. A generalist family can also be included, with S_{gen} species and no preferred resource class, so that $S_{\text{gen}} + \sum_A S_A = S_{\text{tot}}$.

Generating the dynamical equations

Instances of the `Community` class can be initialized with any set of differential equations, which are specified as functions of the system state that return the time derivatives dN_i/dt and dR_α/dt. The package includes tools for constructing these functions automatically based on a dictionary of assumptions. These built-in dynamics are based on the recently introduced Microbial Consumer Resource Model (MicroCRM) [20, 22] illustrated in Fig 2, which

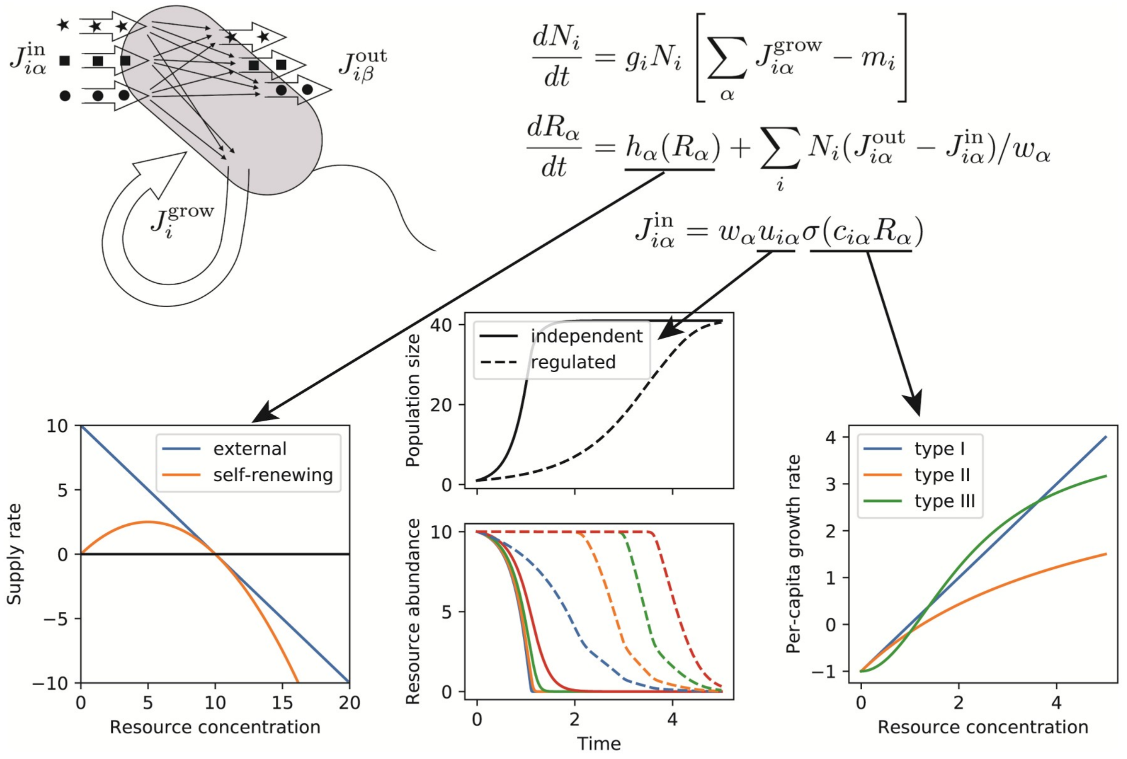


Fig 2. Constructing the dynamical law. The MicroCRM models the growth and metabolism of S microbial species in terms of energy fluxes $J_{i\alpha}^{\text{in}}, J_{i\beta}^{\text{out}}, J_i^{\text{grow}}$, mediated by import, export and chemical transformation of M substitutable resources. Specification of the resource dynamics and of the dependence of import rates on the resource concentrations requires three additional modeling choices, represented by the three arrows. First, the intrinsic dynamics of the resources can either be a linear model of a fixed external input flux and dilution rate, or a logistic model of self-renewing resources, which was employed in MacArthur's original CRM. The left-hand plot shows the supply rate as a function of resource concentration for these two options. Second, the import rates from the different resource types can be independent, or globally regulated in such a way as to preferentially consume the resource that is currently most abundant. The middle plot shows timeseries of consumer and resource abundances in the presence and absence of regulation, with all other parameters held fixed. Third, the dependence of import rates on resource concentration can take a linear (Type-I), Monod (Type-II) or Hill (Type-III) form. The right-hand plot shows the growth rate as a function of resource concentration for these three choices.

https://doi.org/10.1371/journal.pone.0230430.g002

Table 1. Parameters and units for the Microbial Consumer Resource Model.

N_i	population density of species i (individuals/volume)
R_α	Concentration of resource α (mass/volume)
$c_{i\alpha}$	Uptake rate per unit concentration of resource α by species i (volume/time)
$D_{\alpha\beta}$	Fraction of byproducts from resource β converted to α (unitless)
g_i	Conversion factor from energy uptake to growth rate (1/energy)
w_α	Energy content of resource α (energy/mass)
l_α	Leakage fraction for resource α (unitless)
m_i	Minimal energy uptake for maintenance of species i (energy/time)
R_α^0	Intrinsic equilibrium abundance of resource α (mass/volume)
τ_α	Timescale for externally supplied resource turnover (time)
r_α	Rate of resource self-renewal (volume/mass/time)
n	Hill coefficient for functional response (unitless)
$\sigma_{\max}$	Maximum input flux (mass/time)
n_{reg}	Hill coefficient for metabolic regulation (unitless)

https://doi.org/10.1371/journal.pone.0230430.t001

generalizes the classic consumer resource model of MacArthur and Levins [6] to the microbial context by allowing organisms to release metabolic byproducts. Table 1 lists all the parameters of this family of models, along with the corresponding units.

In order to provide a general-purpose set of models that produce physically reasonable results, the MicroCRM assumes that all resource types are substitutable, and can all be converted to a common energy currency. This allows us to enforce energy conservation, preventing communities from bootstrapping themselves to large population sizes using metabolic secretions with no external resource supply. It also eliminates the need to specify in detail how each resource type interacts with all the others within the consumer metabolism. If such interactions are important for capturing a given experimental phenomenon, the built-in dynamics cannot be used, and custom functions must be written for dN_i/dt and dR_α/dt. The tutorial notebook included with the package contains an example of this kind, using Liebig's Law of the Minimum to model phytoplankton dynamics.

Energy fluxes and growth rates. We begin by defining an energy flux into a cell J^{in}, an energy flux that is used for growth J^{growth}, and an outgoing energy flux due to byproduct secretion J^{out}. Energy conservation requires

$$J^{\text{in}} = J^{\text{growth}} + J^{\text{out}} \tag{1}$$

for any reasonable metabolic model. It is useful to denote the input and output energy fluxes that are consumed/secreted in metabolite β by J_β^{in} and J_β^{out} respectively. We can define corresponding mass fluxes by

$$v_\beta^{\text{out}} \equiv J_\beta^{\text{out}}/w_\beta \tag{2}$$

and

$$v_\beta^{\text{in}} \equiv J_\beta^{\text{in}}/w_\beta \tag{3}$$

where the conversion factor w_β measures the energy density of metabolite β. In general, all these fluxes depend on the consumer species under consideration, and will carry an extra Roman index i indicating the species.

We assume that a fixed quantity m_i of power per cell is required for maintenance of species i, and that the per-capita growth rate is proportional to the remaining energy flux $(J^{\text{growth}} - m_i)$, with proportionality constant g_i. Under these assumptions, the time-evolution of the population size N_i of species i can be modeled using the equation

$$\frac{dN_i}{dt} = g_i N_i (J_i^{\text{growth}} - m_i). \tag{4}$$

We can model the resource dynamics by functions of the form

$$\frac{dR_\alpha}{dt} = h_\alpha(R_\alpha) - \sum_j N_j v_{j\alpha}^{\text{in}} + \sum_j N_j v_{j\alpha}^{\text{out}}, \tag{5}$$

where the function h_α describes the resource dynamics in the absence of consumers. The Community Simulator has two kinds of default resource dynamics: externally supplied and self-renewing. For externally supplied resources, we take a linearized form of the dynamics:

$$h_\alpha^{\text{external}}(R_\alpha) = \tau_\alpha^{-1}(R_\alpha^0 - R_\alpha) \tag{6}$$

while for self-renewing we take a logistic form

$$h_\alpha^{\text{self-renewing}}(R_\alpha) = r_\alpha R_\alpha (R_\alpha^0 - R_\alpha). \tag{7}$$

Finally, the intrinsic resource dynamics can also be turned off, with $h_\alpha^{\text{off}} = 0$, to simulate resource depletion in a closed community with no resupply.

Input fluxes and output partitioning. We now specify the form of the input fluxes v_β^{in}, and of the relationships among input, output and growth that define the metabolism. We start by assuming that all resource utilization pathways are independent, resulting in input fluxes of the form

$$v_{i\alpha}^{\text{in}} = \sigma(c_{i\alpha} R_\alpha) \tag{8}$$

where σ is a single-valued function encoding the relationship between resource availability and uptake rates. The community simulator implements three kinds of response functions: Type-I, linear response functions where

$$\sigma_I(x) = x, \tag{9}$$

a Type-II saturating Monod function,

$$\sigma_{II}(x) = \frac{x}{1 + \frac{x}{\sigma_{\max}}} \tag{10}$$

and a Type-III Hill or sigmoid-like function

$$\sigma_{III}(x) = \frac{x^n}{1 + \frac{x^n}{\sigma_{\max}}}, \tag{11}$$

where $n > 1$.

To obtain the output fluxes, we define a leakage fraction l such that

$$J^{\text{out}} = l J^{\text{in}}. \tag{12}$$

We allow different resources to have different leakage fractions l_α. A direct consequence of energy conservation (Eq (1)) is that

$$J_i^{\text{growth}} = \sum_\alpha (1 - l_\alpha) J_{i\alpha}^{\text{in}} = \sum_\alpha (1 - l_\alpha) w_\alpha \sigma(c_{i\alpha} R_\alpha) \tag{13}$$

Finally, we denote by $D_{\beta\alpha}$ the fraction of the output energy that is contained in metabolite β when a cell consumes α. Note that by definition $\sum_\beta D_{\beta\alpha} = 1$. The total energy output in metabolite β is thus

$$J^{\text{out}}_{i\beta} = \sum_\alpha D_{\beta\alpha} l_\alpha J^{\text{in}}_{i\alpha} = \sum_\alpha D_{\beta\alpha} l_\alpha w_\alpha \sigma(c_{i\alpha} R_\alpha). \tag{14}$$

This also yields

$$v^{\text{out}}_{i\beta} = \sum_\alpha D_{\beta\alpha} l_\alpha \frac{w_\alpha}{w_\beta} \sigma(c_{i\alpha} R_\alpha) \tag{15}$$

We are now in position to write down the full dynamics in terms of these quantities:

$$\begin{aligned} \frac{dN_i}{dt} &= g_i N_i \left[\sum_\alpha (1 - l_\alpha) w_\alpha \sigma(c_{i\alpha} R_\alpha) - m_i \right] \\ \frac{dR_\alpha}{dt} &= h_\alpha(R_\alpha) - \sum_j N_j \sigma(c_{j\alpha} R_\alpha) + \sum_{j\beta} N_j \sigma(c_{j\beta} R_\beta) \left[D_{\alpha\beta} \frac{w_\beta}{w_\alpha} l_\beta \right] \end{aligned} \tag{16}$$

Notice that when σ is Type-I (linear) and $l_\alpha = 0$ for all α (no leakage or byproducts), this reduces to MacArthur's original model [6].

Metabolic regulation. The package can also generate dynamics for active metabolic regulation, which allocates a higher fraction of import capacity to nutrients with higher available energy flux. This regulation is implemented through a series of weight functions for resource α that reflect how much of the utilizable energy in the environment is in resource α

$$u^{\text{in}-w}_{i\alpha} = \frac{(w_\alpha c_{i\alpha} R_\alpha)^{n_{\text{reg}}}}{\sum_\beta (w_\beta c_{i\beta} R_\beta)^{n_{\text{reg}}}}, \tag{17}$$

with n_{reg} a Hill coefficient that tunes steepness. Another option is to regulate based on the fraction of biomass contained in resource α,

$$u^{\text{in}-v}_{i\alpha} = \frac{(c_{i\alpha} R_\alpha)^{n_{\text{reg}}}}{\sum_\beta (c_{i\beta} R_\beta)^{n_{\text{reg}}}} \tag{18}$$

For the metabolically regulated model, we define the input fluxes by

$$v^{\text{in}}_\beta = u^{\text{in}}_{i\beta} \sigma(c_{i\beta} R_\beta) \tag{19}$$

Then, we can follow the exact same procedure as above. This yields the equations

$$\begin{aligned} \frac{dN_i}{dt} &= g_i N_i \left[\sum_\alpha (1 - l_\alpha) w_\alpha u^{\text{in}}_{i\alpha} \sigma(c_{i\alpha} R_\alpha) - m_i \right] \\ \frac{dR_\alpha}{dt} &= h_\alpha(R_\alpha) - \sum_j N_j u^{\text{in}}_{j\alpha} \sigma(c_{j\alpha} R_\alpha) + \sum_{j\beta} N_j u^{\text{in}}_{j\beta} \sigma(c_{j\beta} R_\beta) \left[l_\beta D_{\alpha\beta} \frac{w_\beta}{w_\alpha} \right] \end{aligned} \tag{20}$$

These equations are generated by the functions `MakeConsumerDynamics` and `MakeResourceDyanamics`, based on the user's specification of the resource replenishment mode h, the response function σ, and the regulation mode u.

Sampling the parameters

The MicroCRM contains a large number of parameters: the S_{tot}-dimensional vectors g_i and m_i, the M-dimensional vectors R^0_α, l_α, w_α and τ_α or r_α, the $S_{\text{tot}} \times M$ consumer preference matrix $c_{i\alpha}$ and the $M \times M$ metabolic matrix $D_{\alpha\beta}$. Some modeling choices require a small number of additional parameters: the maximal uptake rate σ_{max} for Type-II and Type-III growth, and the exponents n for Type-III growth and n_{reg} for metabolic regulation. A dictionary containing all these parameters must be supplied to the `Community` instance upon initialization. A list of dictionaries may be supplied instead, to allow different wells to have different parameters.

The package contains a function `MakeMatrices` for generating the two matrices, which contain most of the ecological structure, based on a dictionary of modeling assumptions summarized in Table 2. The output of this function is illustrated in Fig 3 and described in detail below.

Consumer preferences $c_{i\alpha}$. We choose consumer preferences $c_{i\alpha}$ as follows. As stated earlier, we assume that each specialist family has a preference for one resource class A (where $A = 1...F$) with $0 \leq F \leq T$, and we denote the consumer coefficients for this family by $c^A_{i\alpha}$. We also consider generalists that have no preferences, with consumer coefficients $c^{\text{gen}}_{i\alpha}$. The $c^A_{i\alpha}$ can be drawn from one of three probability distributions: (i) a Normal/Gaussian distribution, (ii) a Gamma distribution (which ensure positivity of the coefficients), and (iii) a Bernoulli distribution with binary preference levels. Fig 3 shows examples of all three models.

The Gaussian model is parameterized in terms of the mean $\mu_c = \langle \sum_\alpha c_{i\alpha} \rangle$ and variance $\sigma_c^2 = \text{var}(\sum_\alpha c_{i\alpha})$ of the total consumption capacity, and a parameter q that controls how specialized each family is for its preferred resource class. In the generalist family, the mean and variance of $c_{i\alpha}$ are the same for all resources, and are given by

$$\langle c^{\text{gen}}_{i\alpha} \rangle = \frac{\mu_c}{M} \tag{21}$$

$$\langle (\delta c^{\text{gen}}_{i\alpha})^2 \rangle = \frac{\sigma_c^2}{M}. \tag{22}$$

where $\delta c^{\text{gen}}_{i\alpha} = c^{\text{gen}}_{i\alpha} - \langle c^{\text{gen}}_{i\alpha} \rangle$ is the deviation from the mean value. The specialist families sample

Table 2. Definitions of global parameters used for constructing random ecosystems. Values of these parameters are supplied as a Python dictionary to the function `MakeMatrices`, which generates randomly sampled consumer preference and metabolic matrices.

M	Number of resources
T	Number of resource classes
S_{tot}	Number of microbial species in regional pool
F	Number of specialist families
S	Number of microbial species initially present in each local community
μ_c	Mean sum over a row of the preference matrix $c_{i\alpha}$
σ_c	Standard deviation of sum over a row for Gaussian or Gamma $c_{i\alpha}$
c_0	Low consumption level for Binary $c_{i\alpha}$
c_1	High consumption level for Binary $c_{i\alpha}$
q	Fraction of consumption capacity allocated to preferred resource class
s	Sparsity of metabolic matrix
f_w	Fraction of secreted byproducts allocated to "waste" resource class
f_s	Fraction of secreted byproducts allocated to same resource class

https://doi.org/10.1371/journal.pone.0230430.t002

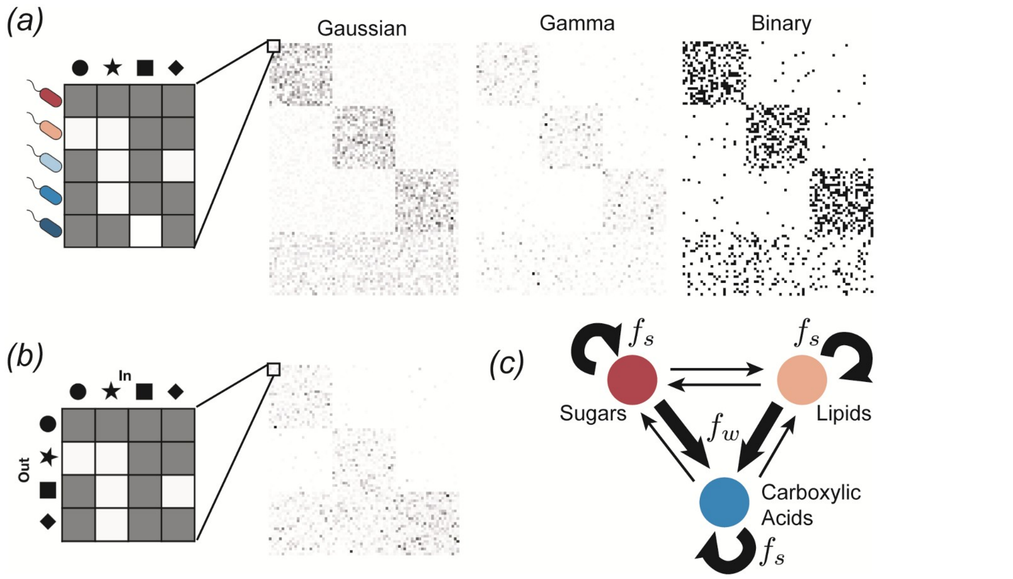


Fig 3. Sampling parameters and adding metabolic structure. *(a)* Sampling the consumer preference matrix $c_{i\alpha}$. Each row corresponds to a different microbial species, and the value of each entry in the row specifies the preference level of that species for a given resource. An example of each of the three sampling choices is shown, with white pixels representing $c_{i\alpha} = 0$ and darker pixes representing larger values. The examples have $F = 3$ consumer families with specialism level $q = 0.9$, each with $S_A = 25$ species, plus a generalist family with $S_{\text{gen}} = 25$ species. *(b)* Sampling the metabolic matrix $D_{\alpha\beta}$. Each column represents the allocation of output fluxes resulting from metabolism of a given input resource. This example has $T = 3$ resource classes, and an effective sparsity $s = 0.05$. *(c)* Diagram of three-tiered metabolic structure. A fraction f_s of the output flux is allocated to resources from the same resource class as the input, while a fraction f_w is allocated to the "waste" class (e.g., carboxylic acids). In the example of the previous panel, allocation fractions were $f_s = f_w = 0.49$.

https://doi.org/10.1371/journal.pone.0230430.g003

from a distribution with a larger mean for resources in their preferred class:

$$\langle c_{i\alpha}^{A}\rangle = \begin{cases} \frac{\mu_c}{M}\left[1 + \frac{M - M_A}{M_A} q\right], & \text{if } \alpha \in \mathbf{A} \\ \frac{\mu_c}{M}(1-q), & \text{otherwise,} \end{cases} \tag{23}$$

where M_A is the number of resources in class A and $\mathbf{A}$ is the set of resource indices in class A. The variances are likewise larger for the preferred class:

$$\langle (\delta c_{i\alpha}^{A})^2\rangle = \begin{cases} \frac{\sigma_c^2}{M}\left[1 + \frac{M - M_A}{M_A} q\right], & \text{if } \alpha \in \mathbf{A} \\ \frac{\sigma_c^2}{M}(1-q), & \text{otherwise.} \end{cases} \tag{24}$$

This makes it possible to construct pure specialist families with no off-target consumption by setting $q = 1$. Note that this is different from the original version of this model in [22], where all the variances were chosen to be identical.

We also consider the case where consumer preferences are drawn from Gamma distributions, which guarantee that all coefficients are positive. Since the Gamma distribution only has two parameters, it is fully determined once the mean and variance are specified. We parameterize the mean and variance for this model in the same way as for the Gaussian model.

In the binary model, there are only two possible values for each $c_{i\alpha}$: a low level $\frac{c_0}{M}$ and a high level $\frac{c_0}{M} + c_1$. The elements of $c^A_{i\alpha}$ are given by

$$c^A_{i\alpha} = \frac{c_0}{M} + c_1 X_{i\alpha}, \tag{25}$$

where $X_{i\alpha}$ is a binary random variable that equals 1 with probability

$$p^A_{i\alpha} = \begin{cases} \frac{\mu_c}{Mc_1}\left[1 + \frac{M - M_A}{M_A} q\right], & \text{if } \alpha \in A \\ \frac{\mu_c}{Mc_1}(1-q), & \text{otherwise} \end{cases} \tag{26}$$

for the specialist families, and

$$p^{\text{gen}}_{i\alpha} = \frac{\mu_c}{Mc_1} \tag{27}$$

for the generalists. Note that the variance in each family is $\langle (\delta c^A_{i\alpha})^2 \rangle = c_1^2 p^A_{i\alpha}(1 - p^A_{i\alpha}) \sim c_1^2 p^A_{i\alpha}$ for large M, which depends on q in the same way as the variances in the Gaussian case.

Metabolic matrix $D_{\alpha\beta}$. We choose the metabolic matrix $D_{\alpha\beta}$ according to a three-tiered secretion model illustrated in Fig 3. The first tier is a preferred class of 'waste' products, such as carboyxlic acids for fermentative and respiro-fermentative bacteria, with M_w members. The second tier contains byproducts of the same class as the input resource. For example, this could be attributed to the partial oxidation of sugars into sugar alcohols, or the antiporter behavior of various amino acid transporters. The third tier includes everything else. We encode this structure in $D_{\alpha\beta}$ by sampling each column β of the matrix from a Dirichlet distribution with concentration parameters $d_{\alpha\beta}$ that depend on the byproduct tier, so that on average a fraction f_w of the secreted flux goes to the first tier, while a fraction f_s goes to the second tier, and the rest goes to the third. The Dirichlet distribution has the property that each sampled vector sums to 1, making it a natural way of randomly allocating a fixed total quantity (such as the total secretion flux from a given input). To write the expressions for these parameters explicitly, we let $A(\alpha)$ represent the class containing resource α, and let w represent the 'waste' class. We also introduce a parameter s that controls the sparsity of the reaction network, ranging from a dense network with all-to-all connection when $s \to 0$, to maximal sparsity with each input resource having just one randomly chosen output resource as $s \to 1$. With this notation, we have

$$D_{\alpha\beta} = \text{Dir}(d_{1\beta}, d_{2\beta}, d_{3\beta}, \cdots, d_{M\beta})_\alpha \tag{28}$$

$$d_{\alpha\beta} = \begin{cases} \frac{f_w}{sM_w}, & \text{if } A(\beta) \neq w \text{ and } A(\alpha) = w \\ \frac{f_s}{sM_{A(\beta)}}, & \text{if } A(\beta) \neq w \text{ and } A(\alpha) = A(\beta) \\ \frac{1-f_s-f_w}{s(M-M_{A(\beta)}-M_w)}, & \text{if } A(\beta), A(\alpha) \neq w \text{ and } A(\alpha) \neq A(\beta) \\ \frac{f_w+f_s}{sM_w}, & \text{if } A(\beta) = w \text{ and } A(\alpha) = w \\ \frac{1-f_w-f_s}{s(M-M_w)}, & \text{if } A(\beta) = w \text{ and } A(\alpha) \neq w. \end{cases} \tag{29}$$

The final two lines handle the case when the 'waste' type is being consumed. For these columns, the first and second tiers are identical. This led to an ambiguity in the expression presented in the Supporting Information of [22], which we have now clarified by treating this case separately.

Propagation in time

Once an instance of the `Community` class is initialized, its state can be propagated forward in time using the `Propagate` method, as illustrated in Fig 1. Since the dynamical equations and parameters were supplied at initialization, the only required argument for this method is the time *T*. When the method is invoked, the state and parameters for each well are sent to different CPU's (as many as are available) using the `Pool.map` function from the `multiprocessing` module in the Python standard library. Then the dynamical equations are integrated using the `odeint` function from SciPy, which calls the LSODA solver from the FORTRAN library ODEPACK [26, 27].

If the initial population size of a species equals zero, but its per-capita growth rate is positive under current environmental conditions, the limited precision of any numerical solver will enable that species to spontaneously invade the community. To prevent this from happening, the `Propagate` method comes with an option `compress_species` (set to `True` by default), which reduces the dimensionality of the state and parameters before invoking the solver by removing all references to extinct species. Compression requires deciding whether each dimension of each parameter array corresponds to species or resources. This information is built in to the package for the MicroCRM, so `c`, `D`,`w`, `g`, `m`, `l`, `R0`, `tau` and `r` are all automatically handled correctly according to their definitions in that context. If custom dynamics are used, a dictionary of parameter dimensions must be supplied to the optional argument `dimensions` of `Community` when the instance is initialized, listing which parameters are of length S, length M, or of shape SxM, SxS, or MxM, respectively.

Passaging to fresh plates

The second method that can be invoked on a `Community` instance is `Passage`, which simulates pipetting of cultures to fresh wells in a typical 96-well plate experiment. The only required argument for `Passage` is a two-dimensional array `f`, whose elements $f_{\mu\nu}$ specify the fraction of the contents of well ν from the old plate that should be transfered to well μ on the new plate. The method also contains an option `refresh_resource`, set to `True` by default, that supplies the new plate with the same initial resource concentrations as the original one. This is the most direct way of simulating actual 96-well plate experiments, where the resources are resupplied in discrete intervals at each passaging step.

This method facilitates simulation of various kinds of mixing or coalescence experiments, as well as metacommunity dynamics of weakly coupled local communities. Fig 4 illustrates how this feature can capture coarse-grained spatial structure, following an experimental protocol developed for mimicking range expansions in 96-well plates [28].

In addition, passaging helps to stabilize long simulations, and simulate demographic noise, by setting all cell counts to integer values. The species abundances N_i are converted to absolute cell counts through a conversion factor `scale`, which is by default set to 10^6. Then the new integer cell counts are generated by multinomial sampling based on the average number of cells $\sum_\nu f_{\mu\nu}(N_i)_\nu$ of species i transferred to well μ. One source of numerical instability in ecological models is the exponentially small values reached by population sizes headed for extinction. The multinomial sampling ensures that these population sizes are fixed at zero when they become significantly smaller than 1 in absolute units. Thus a simple way of avoiding instability in a continuously resupplied chemostat model is to periodically call `Passage` with `f` set to the identity matrix and `refresh_resource` set to `False`.

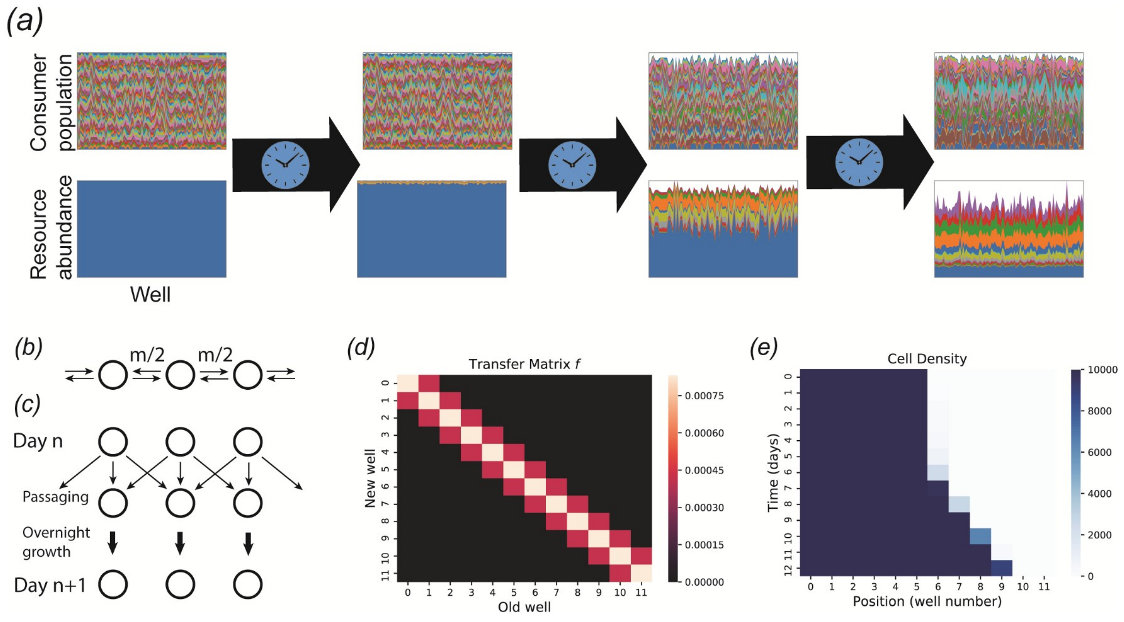


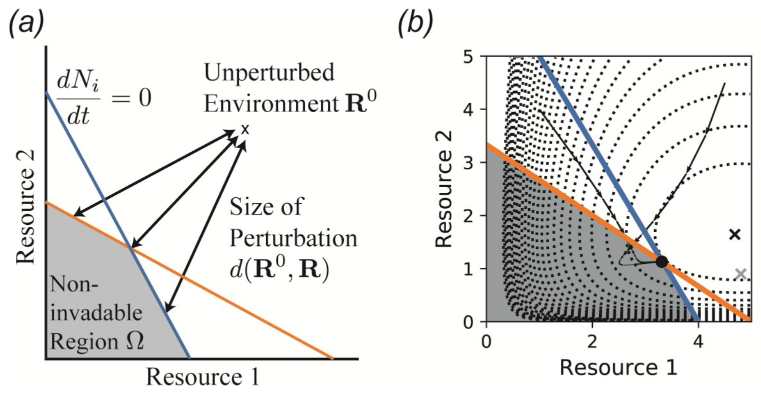
Fig 4. Propagating and passaging. *(a)* System state after successive applications of the method to a plate with $n = 100$ wells, with a single externally supplied resource (blue). Each column of a panel represents a different well, and the height of each colored patch represents the abundance of a different consumer species or resource type. Each panel is normalized so that the sample with the largest total biomass or total resource concentration spans the entire panel. As time passes, the resources become more diverse due to the generation of metabolic byproducts, while the consumers become less diverse through competitive exclusion. *(b)* Modeling spatial structure with a stepping stone model. At each time step, each cell in a given well can migrate to neighboring wells with probability m. *(c)* Implementation of stepping stone model in a 96-well plate. Every day, the communities are passaged to fresh wells, with a fraction $f_0(1 - m)$ transferred to the corresponding position in the new set of wells, and $f_0\, m$ divided equally between the two nearest neighbors, where f_0 is an overall dilution factor. *(d)* Transfer matrix f implementing the stepping stone protocol. *(e)* Simulated range expansion using successive applications of the and methods, with the transfer matrix from the previous panel. See the Jupyter notebook included with the package for all simulation details.

https://doi.org/10.1371/journal.pone.0230430.g004

Since the most common experiments involve many iterations of identical `Passage` and `Propagate` steps, the package also includes a method `RunExperiment(f,T,np)`, which applies a given transfer matrix `f` and propagation time `T` for `np` iterations, saving a snapshot of the plate after each propagation step. If passaging is being used simply to stabilize the integration as discussed above, and not to simulate a batch culture setting, then the value of T does not affect the results (as long as it is short enough to successfully eliminate instabilities). In this case, this variable mainly serves to control the time-resolution of the resulting timeseries.

Finding equilibrium points with convex optimization and expectation maximization

In many experimental contexts, one is interested in the stable community structure reached after a long period of constant environmental conditions. Numerical integration of the dynamical equations is an inefficient way to identify these equilibrium points. When the species diversity and number of resource types are high, the equations typically contain complex transient dynamics spanning a large range of time scales. This transient behavior is often irrelevant to the identification of the final equilibrium point, and wastes significant computation time. For a typical implementation of the MicroCRM with Type-I response and a $1,280 \times 1,280$ binary consumer matrix, integrating to the steady state takes about 37 hours on standard


(a)
$\frac{dN_i}{dt} = 0$
Unperturbed Environment $\mathbf{R}^0$
Size of Perturbation $d(\mathbf{R}^0, \mathbf{R})$
Non-invadable Region Ω
Resource 2
Resource 1
(b)
Resource 2
Resource 1

(c)

```
while $||\mathbf{R}^*_t - \mathbf{R}^*_{t-1}|| > \delta$ do
    {Minimize environmental perturbation}
    $\mathbf{R}^*_t \leftarrow \underset{\mathbf{R}}{\operatorname{argmin}}\, d(\tilde{\mathbf{R}}^0_{t-1}, \mathbf{R})$
    {Update effective unperturbed state}
    $\tilde{\mathbf{R}}^0_t \leftarrow \tilde{\mathbf{R}}^0(\mathbf{R}^*_t)$
    {Update counter}
    $t \leftarrow t + 1$
end while
```

Fig 5. An expectation-maximization algorithm for finding noninvadable stationary states. *(a)* Noninvadable states by definition can only exist in the region Ω of resource space where the growth rate dN_i/dt of each species i is zero or negative. Here, the blue and orange lines represent the combinations of resource abundances leading to zero growth rate for two different consumer species, so the noninvadable region is the space beneath both of the lines. Within this region, a recently discovered duality implies that the stationary state $\mathbf{R}^*$ locally minimizes the dissimilarity $d(\mathbf{R}^0, \mathbf{R})$ with respect to the fixed point $\mathbf{R}^0$ of the intrinsic environmental dynamics [23, 24]. *(b)* Metabolic byproducts move the relevant unperturbed state from $\mathbf{R}^0$ (gray 'x') to $\tilde{\mathbf{R}}^0(\mathbf{R})$ (black 'x'), which is itself a function of the current environmental conditions. Dotted contour lines represent $d(\tilde{\mathbf{R}}^0(\mathbf{R}^*), \mathbf{R})$, and arrows are two trajectories of the population dynamics starting from the unperturbed environmental state with two different sets of initial consumer population sizes. See main text and Appendix for model details and parameters. *(c)* Pseudocode for self-consistently computing $\mathbf{R}^*$ and $\tilde{\mathbf{R}}^0(\mathbf{R}^*)$, which is identical to standard expectation-maximization algorithms employed for problems with latent variables in machine learning.

https://doi.org/10.1371/journal.pone.0230430.g005

hardware. The computation time appears to scale asymptotically as M^4 when the number of species S and the number of resources M are changed simultaneously, as shown in Fig 6.

To address this problem, we have developed an algorithm for identifying equilibrium points directly, without integration through the transient. This algorithm is implemented in Community Simulator as the method `SteadyState`, and is illustrated in Fig 5. Under the same test conditions, this algorithm converges between one and two orders of magnitude faster than numerical integration, as shown in Fig 6, facilitating rapid hypothesis evaluation and iteration

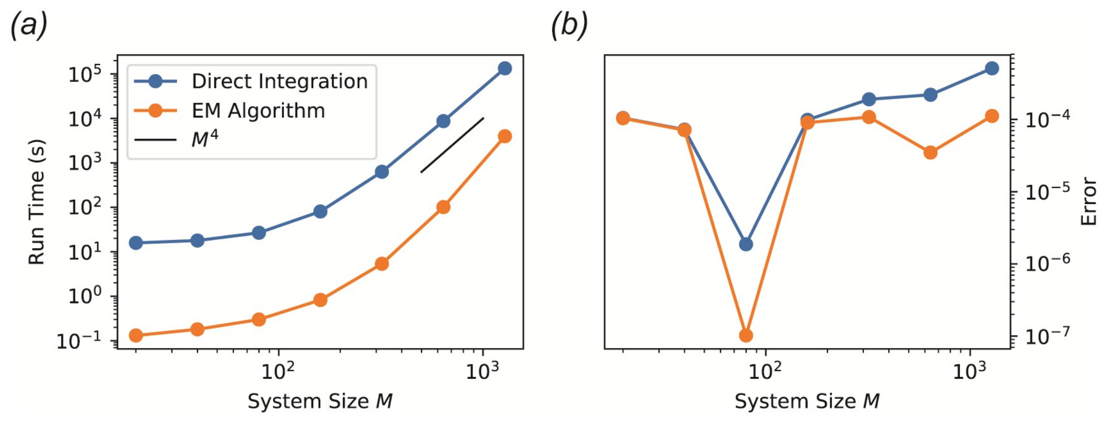


Fig 6. Performance of EM algorithm versus ODE integration. The steady state of the MicroCRM was computed by direct ODE integration and with our new EM algorithm for a range of values of the number of resource types M. The initial number of species S was set equal to M, and a single resource type was externally supplied with intrinsic fixed point $R^0_1 = 10M$ ($R^0_i = 0$ for all $i > 1$). The absolute error tolerance of the integrator was set to 10^{-4}, and the convergence tolerance for the EM algorithm was set to $\delta = 10^{-7}$. See 'scripts' folder in the 'EM-algorithm' branch of the GitHub repository for the rest of the parameters, which were held fixed for all simulations. (a) Total computation time for 10 realizations. *(b)* Final root-mean-square per-capita deviation of the growth rate from zero ('Error') over all surviving species in all 10 samples.

https://doi.org/10.1371/journal.pone.0230430.g006 in large ecosystems. Note that while all other features of the Community Simulator depend only on packages included with a standard Anaconda installation (https://www.anaconda.com/), `SteadyState` additionally requires prior installation of a convex optimization package called CVXPY (https://www.cvxpy.org/).

The algorithm exploits a recently discovered duality between consumer resource models and constrained optimization over resource space [23, 24], which generalizes a minimization principle originally identified by MacArthur in the context of his original Consumer Resource Model [6]. This duality applies to a wide class of consumer-resource type models, requiring only that the environmentally mediated interactions between pairs of consumer species are symmetric [24].

For models in this class, it was shown that the vector of resource abundances $\mathbf{R}^*$ in every stable equilibrium state locally minimizes a measure $d(\mathbf{R}^0, \mathbf{R})$ of the dissimilarity between the current resource abundances $\mathbf{R}$ and the supply point $\mathbf{R}^0$ (defined in general by $h_\alpha(R^0_\alpha) = 0$), subject to the constraint that all consumer growth rates are zero or negative ($dN_i/dt \leq 0$ for all i). Instances of the MicroCRM with Type I resource consumption, no metabolic regulation and $l_\alpha = 0$ fall into this class. For externally supplied resources, d turns out to be a weighted Kullback-Leibler (KL) divergence:

$$d^{\text{external}}(\mathbf{R}^0, \mathbf{R}) = \sum_\beta w_\beta \tau_\beta^{-1} \left[R^0_\beta \ln \frac{R^0_\beta}{R_\beta} - (R^0_\beta - R_\beta) \right] \tag{30}$$

while for self-renewing resources, it is a weighted Euclidean distance [24]:

$$d^{\text{self-renewing}}(\mathbf{R}^0, \mathbf{R}) = \sum_\beta w_\beta r_\beta (R^0_\beta - R_\beta)^2. \tag{31}$$

The equilibrium consumer populations N_i^* are the Lagrange multipliers that enforce the constraints. For models with Type-I response, the non-invadable region is convex, allowing for efficient solution of the optimization problem using the Python package CVXPY [29, 30].

The duality does not strictly apply to other variants of the MicroCRM. In particular, byproduct secretion breaks the symmetry of the effective interactions between consumers whenever $l_\alpha > 0$ for some resource α. The duality can be recovered, however, if the equilibrium point $\mathbf{R}^0$ of the intrinsic resource dynamics is changed to a new value $\tilde{\mathbf{R}}^0(\mathbf{R}^*)$, which accounts for the extra resources produced by the consumer species when the system is at its equilibrium state $\mathbf{R}^*$ [24]. This accounting can be done in a variety of ways that all successfully recover the duality, with varying degrees of computational efficiency. The currently implemented form is

$$\tilde{R}^0_\alpha(\mathbf{R}^*) = R^0_\alpha + \sum_{\beta \neq \alpha} \frac{Q^{-1}_{\alpha\beta}}{Q^{-1}_{\beta\beta}} (R^0_\beta - R^*_\beta) \tag{32}$$

where

$$Q_{\alpha\beta} = \delta_{\alpha\beta} - l_\beta D_{\alpha\beta} \frac{w_\beta}{w_\alpha} \tag{33}$$

and $Q^{-1}_{\alpha\beta}$ are the elements of the matrix inverse of $Q_{\alpha\beta}$, satisfying $\sum_\beta Q^{-1}_{\alpha\beta} Q_{\beta\gamma} = \delta_{\alpha\gamma}$. With these definitions, one can show that the MicroCRM with Type I consumption and no regulation minimizes the objective function

$$d^{\text{byproducts}}(\tilde{\mathbf{R}}^0, \mathbf{R}) = \sum_\beta \tilde{w}_\beta \tau_\beta^{-1} \left[\tilde{R}^0_\beta \ln \frac{\tilde{R}^0_\beta}{R_\beta} - (\tilde{R}^0_\beta - R_\beta) \right] \tag{34}$$

where

$$\tilde{w}_\alpha = Q_{\alpha\alpha}^{-1}(1 - l_\alpha)\tau_\alpha^{-1} w_\alpha. \tag{35}$$

To find $\mathbf{R}^*$, one must now self-consistently solve the following equation:

$$\mathbf{R}^* = \underset{\mathbf{R}}{\operatorname{argmin}}\, d(\tilde{\mathbf{R}}^0(\mathbf{R}^*), \mathbf{R}). \tag{36}$$

The structure of this problem is mathematically equivalent to a standard task in machine learning, where one attempts to infer model parameters from partial data [31]. These parameters θ specify a multivariate probability distribution $p(\mathbf{y}|\theta)$ for a set of measurements $\mathbf{y}$. A standard way of estimating the parameters is to compute the values $\hat{\theta}$ that maximize the likelihood of the data: $\hat{\theta} = \operatorname{argmax}_\theta\, p(\mathbf{y}|\theta)$. But if one actually has access to only a subset $\mathbf{x}$ of the measurement results, then the values $\mathbf{z}$ of the remaining quantities must also be estimated in order to perform this optimization. Ideally, one would use the statistical model with the optimal parameters $\hat{\theta}$ for this task. In the simplest case, where the value of $\mathbf{z}$ can be inferred with certainty given θ and $\mathbf{x}$, this results in the following self-consistency equation:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, p(\mathbf{x}, \mathbf{z}(\hat{\theta})|\theta). \tag{37}$$

This is identical in form to Eq 36, where the parameters θ become the resource concentrations $\mathbf{R}$ and the estimated latent variables $\mathbf{z}$ become the effective unperturbed state $\tilde{\mathbf{R}}^0$. The observed data $\mathbf{x}$ become the model parameters, which are implicitly used in the calculation of d and $\tilde{\mathbf{R}}$. We can think of the environmental perturbation d as a statistical potential or "free energy" $-\ln p$, which is minimized when p is maximized.

Eq 37 can be solved by a standard iterative approach called Expectation Maximization [31]. At each iteration t, the latent variable $\mathbf{z}_t$ is computed from the previous estimate $\hat{\theta}_{t-1}$ of $\hat{\theta}$, and then the new parameter estimate $\hat{\theta}_t$ is found by maximizing $p(\mathbf{x}, \mathbf{z}_t|\theta)$. Fig 5(c) contains pseudocode for this algorithm as applied to our ecological problem, which was also reported previously in [24].

This algorithm fails to converge at low resource supply levels, because both arguments must be positive when d is a weighted KL divergence, but $\tilde{R}_t^0$ can temporarily become negative under these conditions. To solve this issue, we replaced update step for $\tilde{R}_t^0$ by

$$\tilde{\mathbf{R}}_t^0 \leftarrow \alpha\tilde{\mathbf{R}}^0(\mathbf{R}_t^*) + (1-\alpha)\tilde{\mathbf{R}}_{t-1}^0$$

where α is a constant rate, equivalent to the "learning rate" in machine learning [31].

Default values of the tolerance δ and learning rate α are set to 10^{-7} and 0.5, respectively, which give robust convergence for typical simulation scenarios. They can be adjusted as optional arguments of the `SteadyState` method.

This algorithm can be applied to any consumer-resource type model, including models beyond the MicroCRM framework, with non-substitutable resources [24]. But the enhanced efficiency of the new approach requires that the optimization problem be convex. In the Community Simulator package, the algorithm is only implemented for Type-I response with no metabolic regulation, where convexity is guaranteed. For more complex models, the differential equations must be numerically integrated using the `Propagate` method discussed above.

In the tutorial notebook included with the package, we show that the MicroCRM can be bistable if the externally supplied resources are insufficient to directly support growth of any consumer species. In this scenario the state with all consumers extinct is a stable equilibrium of the dynamics, and another stable equilibrium with persisting consumers is also possible that relies on the metabolic byproducts. In this scenario `SteadyState` method can find either of the two equilibria, depending on the initial estimate of $\tilde{\mathbf{R}}^0$, which can be set by an optional argument. If this initial condition is sufficiently close to the actual equilibrium state $\mathbf{R}^0$ of the intrinsic resource dynamics, the method ends in the state with the consumers extinct, where $\tilde{\mathbf{R}}^0 = \mathbf{R}^0 = \mathbf{R}^*$. But if the initial condition is not deliberately tuned to be close to $\mathbf{R}^0$, we find that the method typically finds the other state where some consumers survive.

Conclusions

We hope that the Community Simulator will become a valuable resource for the microbial ecology community. It has already played an important role in our own work. The package initially facilitated the systematic evaluation of the robustness of results to different modeling assumptions in a study of the effects of total energy influx on community structure, diversity and function [24]. More recently, the convex optimization approach has made it possible to perform more than 100,000 independent simulations in a reinterpretation and extension of Robert May's classic work on diversity and stability [32, 33]. We have also employed the package to reproduce large-scale patterns in microbial biodiversity from the Human Microbiome Project, Earth Microbiome Project, and similar surveys [34]. Finally, the random matrix approach implemented in this package is amenable to analytic calculation in the limit of large numbers of species and resources, using cavity methods from the physics of disordered systems [35, 36]. It is our belief that the Community Simulator will facilitate the further development of these mathematical techniques through efficient testing of new conjectures.

One interesting future direction to explore is integrating the Community Simulator with methods for directly analyzing Microbiome sequencing data. For example, there has been a renewed interest in statistical techniques such as Approximate Bayesian Computation (ABC) for understanding ecology and evolution [37]. In ABC, the need to exactly calculate complicated likelihood functions—often a prerequisite for many statistical techniques—is replaced with the calculation of summary statistics and numerical simulations. For this reason, the Community Simulator Python package is ideally suited to form the backbone of new inference techniques for trying to related ecological processes to observed abundance patterns in microbial ecosystems.

Availability and requirements

- **Project name**: Community Simulator
- **Project home page**: https://github.com/Emergent-Behaviors-in-Biology/community-simulator
- **Operating system(s)**: Linux or Mac preferred. Parallelization scheme is currently incompatible with Windows, and must be deactivated (set `parallel = False` when initializing a plate) for the code to run.
- **Programming language**: Python 3
- **Other requirements**: Numpy 1.15+, Pandas 0.23.0+, Matplotlib 2.2.3+, SciPy 1.1.0+. `SteadyState` method additionally requires CVXPY 1.0+.
- **License**: MIT
- **Any restrictions to use by non-academics**: None

Acknowledgments

We are grateful to Kirill Korolev, Alvaro Sanchez, and Daniel Segrè for many useful conversations, and to Matti Gralka for testing cross-platform compatibility of the package. The performance evaluation reported in Fig 6 was performed on the Shared Computing Cluster which is administered by Boston University Research Computing Services.

Author Contributions

Conceptualization: Joshua Goldford, Pankaj Mehta.

Funding acquisition: Pankaj Mehta.

Investigation: Robert Marsland, Wenping Cui.

Software: Robert Marsland.

Supervision: Pankaj Mehta.

Validation: Wenping Cui.

Writing – original draft: Robert Marsland.

Writing – review & editing: Wenping Cui, Pankaj Mehta.

References

1. Thompson LR, Sanders JG, McDonald D, Amir A, Ladau J, Locey KJ, et al. A communal catalogue reveals Earth's multiscale microbial diversity. Nature. 2017; 551:457. https://doi.org/10.1038/nature24621 PMID: 29088705
2. Huttenhower C, Gevers D, Knight R, Abubucker S, Badger JH, Chinwalla AT, et al. Structure, function and diversity of the healthy human microbiome. Nature. 2012; 486:207. https://doi.org/10.1038/nature11234
3. Loreau M. Consumers as maximizers of matter and energy flow in ecosystems. The American Naturalist. 1995; 145:22. https://doi.org/10.1086/285726
4. Embree M, Liu JK, Al-Bassam MM, Zengler K. Networks of energetic and metabolic interactions define dynamics in microbial communities. Proceedings of the National Academy of Sciences. 2015; 112:15450. https://doi.org/10.1073/pnas.1506034112
5. Gause GF, Witt AA. Behavior of Mixed Populations and the Problem of Natural Selection. The American Naturalist. 1935; 69:596. https://doi.org/10.1086/280628
6. MacArthur R. Species Packing and Competitive Equilibrium for Many Species. Theoretical Population Biology. 1970; 1:1. https://doi.org/10.1016/0040-5809(70)90039-0 PMID: 5527624
7. Levin SA. Community equilibria and stability, and an extension of the competitive exclusion principle. The American Naturalist. 1970; 104:413. https://doi.org/10.1086/282676
8. Chesson P. MacArthur's consumer-resource model. Theoretical Population Biology. 1990; 37:26. https://doi.org/10.1016/0040-5809(90)90025-Q
9. Chase JM. Community assembly: when should history matter? Oecologia. 2003; 136:489. https://doi.org/10.1007/s00442-003-1311-7 PMID: 12836009
10. Jeraldo P, Sipos M, Chia N, Brulc JM, Dhillon AS, Konkel ME, et al. Quantification of the relative roles of niche and neutral processes in structuring gastrointestinal microbiomes. Proceedings of the National Academy of Sciences. 2012; 109:9692. https://doi.org/10.1073/pnas.1206721109
11. Kessler DA, Shnerb NM. Generalized model of island biodiversity. Physical Review E. 2015; 91:042705. https://doi.org/10.1103/PhysRevE.91.042705
12. Vega NM, Gore J. Stochastic assembly produces heterogeneous communities in the *Caenorhabditis elegans* intestine. PLoS Biol. 2017; 15:e2000633. https://doi.org/10.1371/journal.pbio.2000633 PMID: 28257456
13. Tilman D. Resource competition and community structure. Princeton University Press; 1982.
14. Chesson P. Mechanisms of maintenance of species diversity. Annual review of Ecology and Systematics. 2000; 31:343. https://doi.org/10.1146/annurev.ecolsys.31.1.343

15. Fisher CK, Mehta P. The transition between the niche and neutral regimes in ecology. PNAS. 2014; 111:13111. https://doi.org/10.1073/pnas.1405637111 PMID: 25157131

16. Dickens B, Fisher CK, Mehta P. Analytically tractable model for community ecology with many species. Physical Review E. 2016; 94:022423. https://doi.org/10.1103/PhysRevE.94.022423 PMID: 27627348

17. Bunin G. Ecological communities with Lotka-Volterra dynamics. Physical Review E. 2017; 95:042414. https://doi.org/10.1103/PhysRevE.95.042414 PMID: 28505745

18. Barbier M, Arnoldi JF, Bunin G, Loreau M. Generic assembly patterns in complex ecological communities. Proceedings of the National Academy of Sciences. 2018; 115:2156. https://doi.org/10.1073/pnas.1710352115

19. Pacheco AR, Moel M, Segrè D. Costless metabolic secretions as drivers of interspecies interactions in microbial ecosystems. Nature Communications. 2019; 10:103. https://doi.org/10.1038/s41467-018-07946-9 PMID: 30626871

20. Goldford JE, Lu N, Bajić D, Estrela S, Tikhonov M, Sanchez-Gorostiaga A, et al. Emergent Simplicity in Microbial Community Assembly. Science. 2018; 361:469. https://doi.org/10.1126/science.aat1168 PMID: 30072533

21. Muscarella ME, O'Dwyer JP. Species dynamics and interactions via metabolically informed consumer-resource models. bioRxiv. 2019;518449.

22. Marsland R III, Cui W, Goldford J, Sanchez A, Korolev K, Mehta P. Available energy fluxes drive a transition in the diversity, stability, and functional structure of microbial communities. PLOS Computational Biology. 2019; 15:e1006793. https://doi.org/10.1371/journal.pcbi.1006793

23. Mehta P, Cui W, Wang CH, Marsland R III. Constrained optimization as ecological dynamics with applications to random quadratic programming in high dimensions. Physical Review E. 2018; 99:052111. https://doi.org/10.1103/PhysRevE.99.052111

24. Marsland III R, Cui W, Mehta P. The Minimum Environmental Perturbation Principle: A new perspective on niche theory. arXiv. 2019;1901.09673.

25. McKinney W. Data Structures for Statistical Computing in Python. In: van der Walt S, Millman J, editors. Proceedings of the 9th Python in Science Conference; 2010. p. 51—56.

26. Jones E, Oliphant T, Peterson P, et al. SciPy: Open source scientific tools for Python; 2001–. Available from: http://www.scipy.org/.

27. Hindmarsh AC. ODEPACK, a systematized collection of ODE solvers. Scientific computing. 1983; p. 55–64.

28. Datta MS, Korolev KS, Cvijovic I, Dudley C, Gore J. Range expansion promotes cooperation in an experimental microbial metapopulation. Proceedings of the National Academy of Sciences. 2013; 110:7354–7359. https://doi.org/10.1073/pnas.1217517110

29. Diamond S, Boyd S. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. Journal of Machine Learning Research. 2016; 17:1.

30. Agrawal A, Verschueren R, Diamond S, Boyd S. A Rewriting System for Convex Optimization Problems. Journal of Control and Decision. 2018; 5:42. https://doi.org/10.1080/23307706.2017.1397554

31. Mehta P, Bukov M, Wang CH, Day AG, Richardson C, Fisher CK, et al. A high-bias, low-variance introduction to machine learning for physicists. Physics Reports. 2019; 810:1. https://doi.org/10.1016/j.physrep.2019.03.001 PMID: 31404441

32. Cui W, Marsland III R, Mehta P. Diverse communities behave like typical random ecosystems. arXiv. 2019;1904.0261.

33. May R. Will a Large Complex System be Stable? Nature. 1972; 238:413. https://doi.org/10.1038/238413a0 PMID: 4559589

34. Marsland R III, Cui W, Mehta P. A minimal model for microbial biodiversity can reproduce experimentally observed ecological patterns. Scientific Reports. 2020; 10:3308. https://doi.org/10.1038/s41598-020-60130-2

35. Advani M, Bunin G, Mehta P. Statistical physics of community ecology: a cavity solution to MacArthur's consumer resource model. Journal of Statistical Mechanics. 2018;033406. https://doi.org/10.1088/1742-5468/aab04e PMID: 30636966

36. Cui W, Marsland III R, Mehta P. The effect of resource dynamics on species packing in diverse ecosystems. arXiv. 2019;191102595.

37. Csilléry K, Blum MG, Gaggiotti OE, François O. Approximate Bayesian computation (ABC) in practice. Trends in Ecology & Evolution. 2010; 25:410. https://doi.org/10.1016/j.tree.2010.04.001