ODD Description

The model description follows the ODD (Overview, Design concepts, Details) protocol for describing agent-based models (Grimm et al. 2010).

1. Purpose

We seek to improve understanding of the roles enzyme production play in soil food webs. Specifically, the importance of spatial heterogeneity in the production of enzymes is of interest. Well-established methods of analyses use equation-based representations of interactions, often using ordinary differential equations. In their typical use, those methods do not capture spatial variability in enzyme production or other interactions of soil biota. We created an agent-based simulation of a simple food web that includes enzymatic activity. Results from this spatial representation may be compared with the equation-based results to indicate the importance of spatial representations.

2. Entities, state variables, and scales

The simulation represents a thin layer of a $1/10^{th}$ cm^2 section of soil. The area is composed of 100 x 100 cells (10,000 cells), with the area a torus, such that cells on opposite edges of the area are considered neighbors. Agents that move off one side of the area appear on the opposite side. That is a common approach that eliminates edge effects. Time steps that are simulated may be thought of as seconds, but remain loosely defined.

The soil is populated by microbes and each has an energy value (in energy units). Two other state variables store the energy consumed and the portion of energy that went to respiration through the life of the microbe, but these accumulators play no role in system dynamics. Predators may be placed in the system. They have the same three variables, energy reflecting the condition of the predator and the accumulators storing consumed energy and energy lost to respiration.

Each cell contains three quantities: 1) detritus particles, which may be acted upon by 2) enzymes produced by microbes to yield quantities of 3) food. Some cells may be designated as roots, depending upon the scenario simulated. An accumulator in each cell is not active in soil processes and used for bookkeeping only, storing the quantity of enzyme lost to degradation. Enzymes and food units are scaled relative to other model components, so that enzymes are in 'enzyme units' and food is in 'food units.'

Detritus is influenced by global variables that, 1: describe how much detritus enters the environment each time step (units: particles); 2: the number of cells to which detritus is added each time step (units: cells); 3: the period between detritus additions (time step); and 4: the diffusion rate of the detritus through the environment (proportion). Variables 1 and 4 are used in all simulations, 2 is used in simulations of detritus rainfall but not root exudates, and 3 is used when detritus is added to the system periodically.

Microbes are influenced by the following global variables, 1: a speed (cells per time step); 2: an energy cost for each move made (energy units); 3: the value of a food item to the microbe (energy units); 4: the maximum rate detritus may be converted to food, known as vMax (time step); 5: the influence microbes on enzyme activity, beta (per microbe); 6: an enzymatic efficiency (unitless); 7: proportion of energy acquired that is put to respiration (energy units); 8:

the energy needed for the microbe to divide (energy units); and 9: the contribution of a microbe that dies to detritus (particles).

Predators have global variables analogous to microbes, including speed (1), movement cost (2), the value of a food item (a microbe) to the predator (3), proportion energy used for respiration(7), energy to divide (9), and contribution to detritus (9). There is also a maximum rate of capture of microbes (microbes per time step). Both microbes and predators store their current locations and internal identifiers as well.

Parameters controlling the production or influence of enzymes include, 1: a half-life for enzymatic decay (units: time steps); 2: the enzymatic concentration in a cell, above which a microbe will not produce more enzyme (enzyme units); 3: the cost to produce one enzyme (energy units); 4: an energy buffer, below which microbes will not produce enzyme (energy units); 5: enzymatic diffusion rate (proportion); and 6: the food produced per unit enzyme, given sufficient substrate (food units).

3. Process overview and scheduling

The model is implemented in NetLogo version 5.0 using a discrete representation of time. After initialization (see section 5), the following processes occur, with each process expanded upon in a discussion of the submodels used (see section 7) and the name of the process in parentheses.

Simulate time step
- Store variables for use in phase space plots (i.e., values at t-1)
- Microbes move (move)
- Predators move (move-predators)
- Microbes produce enzymes (produce-enzyme)
- Diffuse enzymes across area (diffuse-enzyme)
- Convert detritus to food (convert-to-food)
- Microbes feed (feed)
- Predators feed (feed-predators)
- Mortality of microbes if energy depleted (mortality)
- Mortality of predators if energy depleted (mortality-predators)
- Reproduce microbes (reproduce)
- Reproduce predators (reproduce-predators)
- Allow enzymes to decay (decay-enzymes)
- Add detritus using one of the following four:
  - Rain detritus continuously (detritus-rain)
  - Rain detritus periodically (detritus-period)
  - Detritus from roots, continuously (detritus-root-continuous)
  - Detritus from roots, periodic (detritus-root-periodic)
- Diffuse detritus throughout the area
- Store variables for phase space plots (i.e., values at t)
- Store variables for long-term accumulators
- Update the user interface
- Stop if predators are extinct

End current time step and proceed with the next time step

4. Design concepts

Basic principles: The simulation is constructed to allow users to explore aspects of enzyme production in microbes, in a two-dimensional unit of soil that includes predators. Detritus is added to the soil in one of four combinations of two factors: continuous or periodic inflow, and distributed randomly or along roots. The configurations of roots may be changed. Microbes move randomly across the soil, and produce enzymes in the presence of detritus. The enzymes convert detritus to materials that the microbes may consume and gain energy (i.e., food). Energy is spent by microbes through enzyme production, movement, and reproduction, and if energy drops to zero for a given microbe, it dies. Predators are present in the soil in some simulations, and feed on microbes if they share the same location, up to some number of microbes that may be eaten. Like the microbes, predators gain energy from the microbes they eat, use energy to move and to reproduce, and if an individual's energy is depleted it dies. Respiration of microbes and predators is accumulated during simulations.

The model represents a stylized soil patch, with parameters controlling dynamics set through references to known relationships where possible. That said, the values or ranges of variables are sometimes not known. The simulation is intended to provide an experimental platform to help understand how system attributes relate, rather than represent a particular soil and biotic community. By adjusting parameters and rates, such as the rate of degradation of enzymes, rate of detritus inputs, energy required by microbes and predators to reproduce, and patterns of root formation, users may explore how the dynamics and spatial patterning of soil biota relate to primary system attributes.

Emergence: Two major emergent properties are of most interest, the population responses of microbes and predators and their interactions, and the spatial distribution of microbes, predators, detritus, food, and enzymes. The cyclic responses of biota are of interest, and especially the longevity of populations. The stability of the system components in four patterns of detritus delivery are of particular interest: with roots and continuous detritus delivery, with roots and periodic detritus delivery, with rain in randomly selected cells in continuous delivery, and with rain in randomly selected cells in a periodic delivery of detritus.

Adaptation, Objectives, and Learning: The organisms in the simulation do not adapt in the broadest sense in this application. There is no capacity in this application for the heredity of traits from one generation to the next, beyond the trivial inheritance of an energy level of a new organism during fission. Microbes may produce or not produce enzymes based the concentration of enzymes in a given cell. Microbes and predators have no inherent objectives. They share feeding behavior and will reproduce and populate a space, but have no intrinsic objectives beyond these. Learning is not part of the current simulation.

Prediction: The organisms represented in the simulation do not make predictions in any direct sense. Microbes do modify their environment for their future benefit by producing enzymes that convert detritus to food that may be consumed. Microbes move while producing enzymes, but the biased random walk they perform plus the diffusion applied to enzymes means that the individual producers of enzymes are likely to benefit from the investment.

Sensing: Microbes and predators can sense the food and other resources that occur on the cell they occupy. Specifically, microbes are aware of the presence of detritus and the concentration of enzymes. If the concentration of enzymes exceeds a threshold, microbes do not produce more enzymes. Microbes are aware of the presence of food, and predators are aware of the presence of microbes on their cell.

Interaction: Microbes compete for food resources, as do predators. Predators feed on microbes.

Stochasticity: Many aspects of the simulation include stochastic components. The locations of roots are random and their growth during initialization includes random turning angles and a stochastic probability of branching for roots based on a normally distributed branching angle and a deviation indicated in a global variable. The initial locations of microbes and predators are random (see Section 5).

The modeling tool used randomizes the order in which agents and cells perform operations. For microbes and predators, they turn a random heading prior to moving during a time step. When predators feed on in a cell that contains more than the maximum microbes that they can eat within a given time step, the microbes they do eat are selected randomly. When detritus is added to the simulation without roots present, patches that receive detritus are selected randomly, each time step if using a continuous addition of detritus, or each period if that option is used.

Observation: Populations of microbes and predators, and average concentrations of detritus, enzymes, and food are stored from each simulation. Simulation parameters are stored as well, such as the use of roots in modeling or not. As simulations progress, plots show the numbers of microbes and predators, plus the average detritus, enzymes, and food available across the cells. Another set of plots provide phase diagrams of microbes versus predators, microbes at the current and previous time step, and predators at the current and previous time step. Current values are displayed for populations of microbes and predators and their average energy, the length of roots, and average detritus, enzymes, and food in cells.

The main outputs of each simulation were summarized over the long term, defined in the model as 500 time steps. The number of microbes, predators, microbe respiration, predator respiration, and average enzymes, detritus, and food were stored in lists at each time step. After 500 time steps, the oldest entries were discarded from the lists and the newest added, always ensuring that the latest 500 time steps were retained. The average of these lists were then reported at the end of each simulation.

5. Initialization

The model is initialized by setting up the cells in the environment, initializing microbes and predators, and if part of the requested simulation, roots in the environment. Each is described below, with pseudo-code given for the sub-model that grows roots.

To setup cells in the environment, an initial quantity of detritus is assigned, if roots are not being simulated (in that instance, detritus is initialized in a following step). Food, enzymes, and the root indicator are set to zero. The color used to display the cell is initialized based on detritus present.

To setup microbes, a number of microbes are created based on an adjustable global variable. Their appearance is set, energy is initialized to a global variable describing initial microbe energy, and their location in the environment is set randomly. Two accumulators showing the energy consumed and energy respired are set to zero.

Predators are initialized in a manner analogous to microbes. The number of predators to create is controlled by their own global variable (i.e., initial predators), and the energy they begin with is set by a global parameter, initial predator energy.

Roots are grown within the environment to initialize the system. In summary, a number of root tips are created, and then those tips grow in response to a series of parameters. As the root tips move through the environment, the cells they move through are labeled as roots. In the current application, roots grow only during initialization; during simulations, root patches do not change. In pseudo-code, the method is:

```
Create a number of root tips, controlled by a global variable, and place them randomly
Assign to the root tips a random heading
While root length, represented by the sum of cells that are roots, -
  is less than an adjustable total root length
        For one randomly selected root tip
                Turn the heading of the tip by an adjustable angle, centered on zero -
                  meaning no turn
                Move forward one cell and label it as root
                Initialize detritus in that cell, based on a global setting
                At some small probability
                        Create a new root tip, representing a bifurcation of the root
                        Set the heading of new tip based on a normally distributed -
                          mean and standard deviation angle
        Update the root length
Continue while root length loop
```

The root tips that are created grow one cell at a time, each tip selected randomly for its opportunity to grow. This allows the initialization to match the number of cells that should be root precisely. A biased random walk is represented for the root tip. As roots grow, there is a probability the tip will branch, creating a new root tip. When that branching occurs, the heading of the new root tip is determined from a normal distribution with a specified mean and standard deviation.

With roots fully grown, the parameters controlling their initialization are no longer used, and roots are static during a simulation. The remaining step for initialization is for the system to calculate root clumpiness (see section 7). That clumpiness index indicates the linearity or contagion of roots.

6. Input data

No input data are used in the model. The dynamics in the simulation derive from the model structure, parameters provided, and spatial heterogeneity.

7. Submodels

Here we expand upon the submodels cited in section 3 and others, and the logic and assessment of their structures. Each is identified by the name used in simulation and their intent, as in section 3. Pseudocode summarizes the submodel, then notes are provided. For brevity we cite here that in every case entities are asked to perform a function, they do so asynchronously and with their orders determined randomly each time step. The values of parameters used in the model are discussed in more depth in the manuscript accompanying this ODD.

move **(Microbes move)**

```
For each microbe:
        Turn a random angle, to face a random direction
        Move forward equal to the value of the microbial speed variable
        Subtract microbial movement energy cost from the microbe's energy
Next microbe
```

A random walk model was adopted for microbes as the most parsimonious approach to representing microbial movements, and is a common approach (Duffy 1995). The bacteria represented by microbe agents are mobile but passive, changing location as they are carried along as ecosystem elements diffuse.

move-predators (Predators move)

Predators were simulated in a manner analogous to microbes, and follow the pseudocode shown under 'move.' Predators have their own speed and energy costs for movement. Future applications may include sensing and hunting behavior in predators, but for simplicity this application uses random walk movements.

produce-enzyme (Microbes produce enzymes)

```
For each microbe:
        If the quantity of enzyme in the cell is less than a threshold and -
        If the microbe has more energy than the cost of producing an enzyme
          times a buffer then
                Add an enzyme unit to the cell enzyme quantity
                Reduce microbe energy by the cost of producing an enzyme
        End if
Next microbe
```

A threshold is compared to the enzyme concentration in the cell, allowing a microbe on the cell to avoid producing enzyme if it would only add to an already abundant concentration. There is also a buffer internal to the microbe that dictates that the organisms' energy must be above some threshold prior to enzyme production. This allowed us to capture the idea that microbes may not produce enzymes if they are extremely low in energy.

diffuse-enzyme (Diffuse enzymes across the soil)

This is a single-line sub-model and so pseudocode is not needed. The sub-model diffuses enzymes across the cells in the model. The structure of that process is that a diffusion rate between 0.0 and 1.0 is provided, and that value influences the flow of quantities from each focal cell to its eight neighbors. If the value is 0.0, no diffusion occurs. If the value is 1.0, the contents of the cell flow completely into the neighboring cells, with each receiving 12.5% of the contents (8 cells x 12.5 = 100%). That said, material in neighboring cells may diffuse back into the focal cell. Most processes in the model are asynchronous, but diffusion is synchronous. In this context, that means that a quantity that flows from cell A to cell B does not then flow back from B into A during the same time-step. All flows are calculated for each cell in turn, and then quantities in the soil as a whole are updated.

convert-to-food (Convert detritus to food)

```
For each cell:
```

```
        If detritus is present and enzyme is present then
            The number of active enzymes is -
              the enzymes present times an enzyme efficiency times -
              the maximum detritus that can be converted divided by –
              beta plus the amount of detritus present
            For each active enzyme and while food is available:
                For the number of food units that an enzyme can convert:
                    Increase food in the cell by one food unit
                    Decrease detritus in the cell by one food unit
                    If detritus is negative, set to zero
                Next food unit
            Next active enzyme
    Next cell
```

This sub-model incorporates a commonly used model of the limitation of enzymatic activity on detritus. Enzymes are able to convert a number of detritus units into food units that may be adjusted prior to a simulation.

feed **(Microbes feed)**

```
    For each microbe:
        Determine the food present, up to an adjustable maximum quantity of food
        Calculate the energy of that food quantity
        Increase energy of the microbe by the energy in the food, minus respiration
        Decrease the food in the cell by the amount eaten
        Increase an accumulator by the energy acquired
        Increase an accumulator by the energy that is respired
    Next microbe
```

The amount of food that a microbe may consume is constrained to a maximum amount. This reflects the handling time the microbe may have of food particles. Respiration is a typical representation, where some proportion of energy acquired is lost to metabolic processes. Accumulators are used to store the sum of quantities through the duration of the simulation, and are not involved in ecosystem processes.

feed-predators **(Predators feed)**

```
    For each predator:
        Determine the number of microbes in the cell the predator occupies
        If there are microbes present then
            Set the number of microbes to eat if it exceeds a maximum number of –
              prey by randomly selecting the maximum number from those present
            Increase energy of the predator by the accumulated energy of the –
              microbes consumed, minus a portion used in respiration
            Increase an accumulator storing the energy consumed and stored
            Increase an accumulator storing the energy that was respired
            Remove the microbes that were eaten
        End if
    Next predator
```

The number of microbes that predators may eat is limited, reflecting the handling time of the prey by the predator. Predators accumulate the energy that is in the microbes they consume, so their feeding on 'healthy' microbes will yield more energy than feeding on emaciated microbes. As elsewhere, accumulators store sums during each simulation, in case those quantities are of interest. They do not play a role in ecosystem processes.

mortality **(Mortality of microbes, energy depleted)**

```
For each microbe:
    If the energy of the microbe is less than zero then
        Contribute some material to detritus representing the microbe carcass
        Kill the microbe
    End if
Next microbe
```

If a microbe depletes its energy, which occurs when it moves about but cannot locate food, the microbe dies. This causes a small quantity to be added to detritus of the cell that was occupied, representing the cell walls, etc. of the microbe.

mortality-predators **(Mortality of predators, energy depleted)**

Mortality in predators is represented analogously to that in microbes, and so pseudo-code is not provided. Predators have their own variable representing the quantity of detritus their death causes to be added to the detritus pool of the cell.

reproduce **(Reproduce microbes)**

```
For each microbe:
    If the microbe has more energy than the threshold for reproduction then
        Divide the energy of the microbe by two
        Create a new microbe
    End if
Next microbe
```

Reproduction in the microbes is represented in a straightforward way. If the microbe has sufficient energy, it fissions into two identical microbes, each with half the energy of the original microbe. The division of energy in two prior to creating a new microbe reflects the method used in the modeling platform. When a new individual is created by another individual (it's parent), that individual inherits the properties of the parent. Therefore, by setting the energy of a 'parent' to one-half its current energy, then having that parent spawn an offspring that has an equal amount of energy, we conclude with two identical organisms with the total energy divided between them.

reproduce-predators **(Reproduce predators)**

Predators reproduce in a manner analogous to microbes, and so pseudo-code is not shown. Predators have their own global variable representing the energy required to divide.

decay-enzymes **(Allow enzymes to decay)**

```
    Calculate decay rate of enzymes as 0.50 divided by a parameter controlling half-life
```

```
For each cell:
        Increase an accumulator of enzymes that were lost by the enzyme present -
          times the decay rate
        Decrease enzyme in the cell by its quantity times 1 minus the decay rate
Next cell
```

Enzymatic decay is represented in a straightforward way, once the decay rate is calculated. In practice , a decay-half-life of 10 yields a rate of 0.5 / 10 or a decay rate of 0.05. A very short half-life such as 1 yields 0.5 / 1 and a rate of 0.5, and a very long half-life such as 1000 yields 0.5 / 1000 and a rate of 0.0005.

detritus-rain (Rain detritus continuously)

```
Calculate the detritus added per patch, which in this case is the amount of detritus -
  to be added per time step divided by the number of cells to receive detritus
For a randomly selected number of cells indicated by a global variable:
        Increase detritus by the detritus to be added per cell
        Increment an accumulator storing the total detritus added to the system
Next cell
```

This sub-model represents continuous detritus contributions to a subset of cells that are selected randomly each time step. That number of cells is controlled by a global variable.

detritus-period (Rain detritus periodically)

```
Calculate the detritus added per patch, which in this case is the amount of detritus -
  to be added per time step times the number of time steps defining the period -
  divided by the number of cells to receive detritus
For a randomly selected number of cells indicated by a global variable:
        Increase detritus by the detritus to be added per cells
        Increment an accumulator storing the total detritus added to the system
Next cell
```

This sub-model represents periodic detritus contributions to a subset of cells that are selected randomly each time step. That number of cells is controlled by a global variable. The simulation uses the amount of detritus multiplied by the period to judge how much detritus should be added at the beginning of the period.

detritus-root-continuous (Detritus from roots, continuously)

```
Calculate the detritus added per patch, which in this case is the amount of detritus -
  to be added per time step divided by the number of root cells to receive detritus
If a cell is a root then
        Increase detritus by the detritus to be added per cell
        Increment an accumulator storing the total detritus added to the system
End if
```

This sub-model represents continuous detritus contributions to the cells that are defined as roots. Cells do not change their status as roots during a simulation.

detritus-root-periodic (Detritus from roots, periodic)

```
Calculate the detritus added per patch, which in this case is the amount of detritus -
  to be added per time step times the number of time steps defining the period -
  divided by the number of root cells receiving detritus
If a cell is a root then
    Increase detritus by the detritus to be added per cells
    Increment an accumulator storing the total detritus added to the system
End if
```

This sub-model represents periodic detritus contributions to a subset of cells that are defined as roots. The simulation uses the amount of detritus multiplied by the period to judge how much detritus should be added at the beginning each period. The status of cells as root or non-root does not change during a simulation in the current model.

get-clumpiness (Calculate clumpiness index for roots)

```
Calculate the proportion of the cells in roots; call the result P1
For each cell that is a root:
    For the four neighbors of the cell:
        If the neighbor is a root, increment root touching root by one
        If the neighbor is not a root, increment root touching non-root by one
    Next neighbor
Next cell

Calculate as the number of root cells as neighbors divided by the number of -
  root cells with roots or non-roots as neighbors; call the result G1

If G1 is greater than or equal to P1, calculate clumpiness as G1 minus P1 -
  divided by 1 minus P1
If G1 is less than P1 and P1 is greater than 0.5, calculate clumpiness as G1 minus P1 -
  divided by 1 minus P1
If G1 is less than P1 and P1 is less than 0.5, calculate clumpiness as P1 minus G1 -
  divided by -1 times P1
```

This sub-model calculates the class-level clumpiness index found in McGarigal et al. (2012), with roots as the focal class. If roots were distributed more regularly than random, which is unlikely, the clumpiness metric would approach -1. If the distribution of roots are random, the index would be about 0. If there is clustering of the roots, the index approaches 1.

draw-patches (Update cell colors displayed for the environment)

```
Calculate mean values for detritus, food, and enzymes within the cells
Calculate standard deviations for detritus, food, and enzymes within the cells
Calculate the range (maximum minus minimum) for detritus, food, and enzymes
Calculate a stretching factor for detritus, food, and enzymes such that -
  each is defined by the mean minus twice the standard deviation
For each cell
    Calculate a color value for detritus, food, and enzymes based on -
      value in cell minus the factor calculated divided by the range times -
      the desired range
    Trim the result to be between zero and the desired range
```

```
            Paint the display using the color values calculated
        Next cell
```

This sub-model is not used to represent soil biology, but rather updates the simulation display. An element on the simulation interface allows users to choose to show individual results for one of detritus, food, or enzymes, or a composite image. In the composite, detritus amounts are shown in red, food in green, and enzymes in blue. The algorithm described above is a typical stretching used by popular geographic software, using twice the standard deviation to identify low and high values of the linear stretch.

Literature Cited

Duffy, K.J., P.T. Cummings, and R.M. Ford. 1995. Random walk calculations for bacterial migration in porous media. Biophysical Journal 68:800-806.

McGarigal, K., SA Cushman, and E Ene. 2012. FRAGSTATS v4: Spatial Pattern Analysis Program for Categorical and Continuous Maps. Computer software program produced by the authors at the University of Massachusetts, Amherst. Available at the following web site: http://www.umass.edu/landeco/research/fragstats/fragstats.html