Exploring chemotaxis in spatio-temporal stoichiometric models of metabolism: *Pseudomonas simiae* as a case study

Hui Shi,[1,2] Ercole LiPuma,[1,2,3] Haroon Qureshi,[2] Phineas McMillan,[2,3] Jing Zhang,[2] Melisa Osborne,[2,4] Ilija Dukovski,[2,4,5] Daniel Segrè[2,4,6,7,8,*]

1. Biochemistry & Molecular Biology Program, Boston University, Boston, MA, USA
2. Bioinformatics Program, Faculty of Computing and Data Sciences, Boston University, Boston, MA, USA
3. Boston University Academy, Boston, MA, USA
4. Biological Design Center, Boston University, Boston, MA, USA
5. Center for Advanced Interdisciplinary Research, Ss. Cyril and Methodius University, Skopje, N. Macedonia
6. Department of Physics, Boston University, Boston, MA, USA
7. Department of Biomedical Engineering, Boston University, Boston, MA, USA
8. Department of Biology, Boston University, Boston, MA, USA
* Corresponding Author: dsegre@bu.edu

Summary/Abstract

Chemotaxis is the movement of cells or organisms in the direction of an increasing or decreasing gradient of a chemical, i.e. a chemoattractant or chemorepellent, respectively. In bacteria, this movement is typically manifested through the presence of bias in the random walk generated by run-and-tumble motility. Often, the chemoattractant is consumed by the organism, thus coupling the biophysical, directed motion with the biochemical, metabolic activity in the growing bacterial colony. In this chapter, we describe our approach to simulating chemotaxis, coupled with detailed metabolic activity within the framework of our modeling software COMETS (Computation of Microbial Ecosystem in Time and Space). By combining genome-scale metabolic models, dynamic flux balance analysis and the newly implemented biophysical chemotaxis model, we built, tested and refined a chemotaxis module for COMETS. We applied this model to an automatically constructed genome-scale model of the plant commensal bacterium *Pseudomonas simiae WCS417*, which we hypothesized would perform chemotaxis on amino acids similar to its close relative *Pseudomonas putida*. Through agar plate experiments we confirmed that *P. simiae* displays chemotaxis to alanine, giving rise to a typical ring pattern. In addition, we used images to infer parameters needed for computer simulations, such as diffusivity and growth rate. Notably, with these parameters, COMETS was able to recapitulate the ring colony pattern, which is due to propagation at the colony front, indicative of bacteria growing on a chemoattractant-filled plate. By allowing for detailed examination of spatial distributions and time dynamics of the biomass, as well as concentration profiles of the metabolites involved in the growth and chemotaxis in bacterial colonies, the software method described here enables simulations of complex microbial patterns involving metabolism and directed motility. Beyond testing the appearance of the ring, future model validation could include comparisons of local growth rates and metabolic fluxes, including rates of metabolite uptake and secretion. The newly added COMETS chemotaxis capability integrates flux balance modeling with gradient-induced biomass propagation, paving the way for more realistic simulations of microbial ecosystem dynamics in complex environments, such as plant roots, where chemotaxis may play a fundamental role in microbiome-host dynamics.

Keywords: Microbial Communities, Chemotaxis, Flux Balance Analysis, Dynamic Flux Balance Analysis, Spatio-Temporal Simulation, Genome Scale Models

Introduction

Beyond possessing dynamic taxonomic composition and metabolic capabilities, microbiomes can be viewed as complex biophysical systems whose spatial architecture is strongly coupled with their functions. These biophysical and structural properties can affect the role of metabolism in the human microbiome, as well as in marine, soil and plant-associated microbial communities.[1–4] Much work has been done to study the spatio-temporal dynamics of bacterial colonies, i.e. the way populations of bacteria arrange into macroscopic spatial patterns that can change in time. These studies range from spatial scales as small as Petri dishes[5–10] to scales as large as complex root microbiomes.[11] They not only elucidate bacterial behavior in a plethora of conditions, but also provide information on spatio-temporal relationships that can help predict bacterial behavior for other phenomena.[12] For example, root microbiomes are composed of bacteria that can benefit from plant root exudates and secrete compounds beneficial to the plant.[13] The knowledge of the interactions between bacteria and the plant through exchange and sensing of these compounds could provide meaningful insights into rhizosphere microbiome dynamics, with important consequences for agricultural sustainability and biogeochemical cycles.

To understand the bacteria-bacteria and bacteria-host interactions in such spatially structured systems, we must understand the interplay between metabolic exchanges and biophysical processes related to the growth and spread of microbial populations. In prior work, we have developed a software package called COMETS (Computation of Microbial Ecosystem in Time and Space)[14] to simulate the growth of bacterial colonies by combining genome-scale models (GEMs) of metabolism[15] with biophysical models of cellular diffusion, motility and demographic fluctuations. In this chapter, we will focus on chemotaxis as one of the aspects of this type of coupling between metabolism and biophysics of bacterial colonies, newly implemented in COMETS.

Chemotaxis is the oriented (biased) motion of bacteria towards a chemical compound (chemoattractant) or away from it (chemorepellent). It is generally accepted that chemotaxis allows bacteria to sense gradients and move towards resources that enable them to survive,[16,17] or move away from harmful chemicals.[18] In the absence of chemotaxis or other gradients, bacterial motility is generally observed in the form of an unbiased, random walk (see Fig. 1A).[19] Bacteria perform runs, movement in straight lines, and tumbles, reorientations, often by switching directionality of flagella rotation. In an unbiased random walk, the distribution of run lengths and the frequency of tumbles does not change over time and is not sensitive to the chemical concentrations in the environment. In COMETS, the unbiased movement of bacteria is modeled as diffusion to simulate bacterial movement by swimming.[20] Chemotaxis, too, is a result of a random walk composed of a series of straight runs and random tumbles. However, in the case of chemotaxis, the random walk is biased by the presence of a chemical gradient. The bacteria sense local differences in the concentration of the chemical and change the frequency of tumbles.[20] In chemoattraction, if an increase in concentrations is sensed, a bacterium will tumble less frequently, leading to an overall movement biased toward higher concentrations of the chemical (Fig. 1B).

A variety of mathematical models have been proposed to explain and predict chemotaxis. A classical, early model for chemotaxis is the Keller-Segel model (KS model).[21] The KS model, originally developed to describe how slime molds move, was later adapted to model chemotaxis in multiple frameworks.[22–25] However, this model failed to reproduce the characteristic ring observed at the propagating front of bacterial colonies grown in the presence of a chemoattractant that is also metabolized.[20,26–28] A subsequent model, the Growth Expansion model (GE model)[29,30], which successfully predicted this phenomenon, is the one we implemented in COMETS in order to build a chemotaxis module.[28,29]

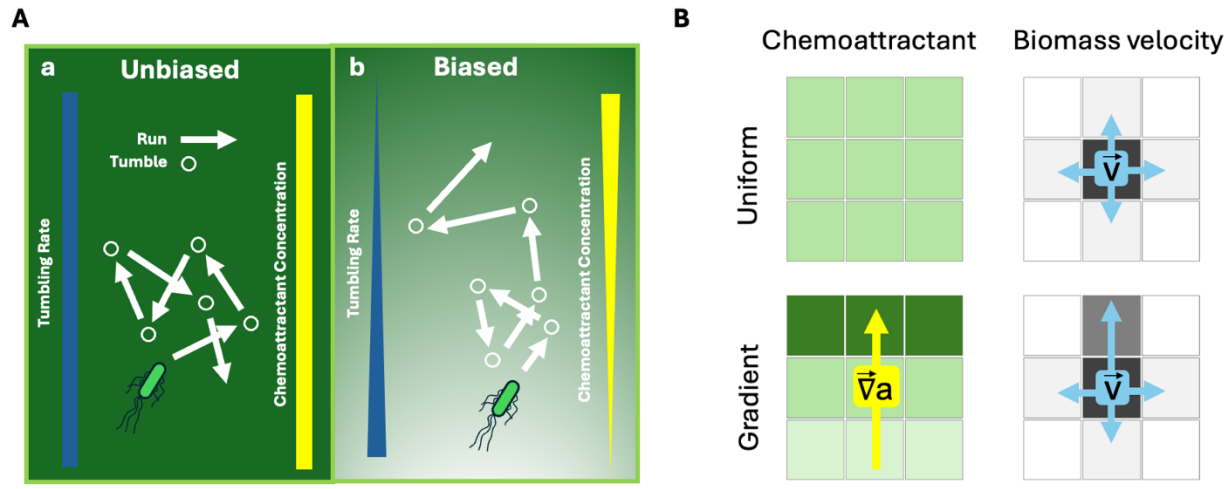

Figure 1: Chemotaxis as biased random walk (swim) of bacteria and its implementation in COMETS. A) a) Unbiased random walk consistent with normal bacterial motility in media with no gradient. The walk is performed as a stochastic series of straight directed flights (arrows) and sudden changes of direction, tumbles (circles). The average rate (frequency) of the tumbles is constant; **b)** Biased random walk consistent with chemotaxis, in the presence of a gradient of chemical concentration. When the bacterium senses an increasing concentration of the chemoattractant, the tumble frequency decreases, leading to longer flights, and to overall movement along the chemoattractant gradient.
B) The biomass propagation in COMETS is modeled by a differential equation with two terms: (i) term of unbiased diffusion and (ii) chemotaxis term proportional to the gradient of the chemoattractant. In the absence of a chemical gradient (top left), the biomass at every step of the numerical algorithm is propagating equally in all directions (top right). In the presence of a chemoattractant gradient (bottom left), the biomass chemotaxis velocity is proportional to the chemical gradient, and its motion is biased in the direction of higher chemoattractant concentration (bottom right).

The GE model in COMETS is implemented by numerically solving the equations:

$$\frac{\partial \rho}{\partial t} = D\vec{\nabla}(\rho^k \vec{\nabla}\rho) - \chi \vec{\nabla}(\rho\, \vec{v}) + r\rho \qquad (1)$$

$$\vec{v} = \frac{\vec{\nabla} a}{a + a_m} \qquad (2)$$

on a discrete two-dimensional grid (see Fig. 1B). The time dynamics of the local colony biomass, ρ , is determined by its diffusion, chemotaxis and local growth. The first term on the right-hand side of Equation 1, $D\vec{\nabla}(\rho^k \vec{\nabla}\rho)$, models the propagation of a bacterial colony, irrespective of chemotaxis. The parameters of the model are the diffusivity D and the exponent k. In this work, $k = 0$, which simulates actively swimming bacteria through simple (i.e. linear) ρ. $-\chi \vec{\nabla}(\rho\, \vec{v})$ models chemotaxis. χ is the chemotaxis coefficient. The velocity of the biomass, calculated in equation (2), is proportional to the gradient of the concentration of the chemoattractant, $\vec{\nabla} a$. The velocity saturates at a concentration a_m, thus a_m is the sensitivity of the chemotactic response. For $a \gg a_m$ the bacteria lose their sensitivity to the gradient and chemotaxis stops.[29,30] The numerical recipe to introduce this mathematical model in COMETS was implemented.[14] The last term in equation (1) models the local growth of the biomass. This term couples the colony propagation and chemotaxis with the metabolic activity and growth. One crucial aspect of chemotaxis is its relation to the metabolic activities in a microbial community. Chemoattraction is orienting the propagation of the organisms to a food source, thus influencing its metabolic

response to the environment. Modeling the metabolic activities in a bacterial colony, therefore, must come hand in hand with the model of chemotaxis. This integration between chemoattraction and metabolism can be achieved by applying the method of flux balance analysis (FBA) to genome-scale models (GEMs), enabling computational predictions of bacterial metabolism. GEMs are reconstructions of the detailed stoichiometry for the set of all reactions in the metabolic network of an organism, usually generated from the organism's annotated genome and from literature information.[15] The first part of the methods section in this chapter describes the procedure of building a genome-scale model of bacterial metabolism from its sequenced genome. The last term in equation (1) models the local growth of the biomass. This term couples the colony propagation and chemotaxis with the metabolic activity and growth. One crucial aspect of chemotaxis is its relation to the metabolic activities in a microbial community. Chemoattraction is orienting the propagation of the organisms to a food source, thus influencing its metabolic response to the environment. Modeling the metabolic activities in a bacterial colony, therefore, must come hand in hand with the model of chemotaxis. This integration between chemoattraction and metabolism can be achieved by applying the method of flux balance analysis (FBA) to genome-scale models (GEMs), enabling computational predictions of bacterial metabolism. GEMs are reconstructions of the detailed stoichiometry for the set of all reactions in the metabolic network of an organism, usually generated from the organism's annotated genome and from literature information.[15] The first part of the methods section in this chapter describes the procedure of building a genome-scale model of bacterial metabolism from its sequenced genome.

FBA, which has been described in detail before, [31–35] is a mathematical approach that can be applied to GEMs, enabling predictions of the flow of metabolites through a metabolic network. In its most common formulation, FBA relies on two major assumptions: (i) that the metabolic network is at steady state under a given environmental condition and (ii) that the network has evolved towards the optimization of a given goal, captured mathematically in a linear objective function. Under these assumptions, FBA is expressed in the form of a convex optimization problem that can be solved using linear programming, yielding predictions of the biomass production rate (or growth rate of the cell), as well as intracellular and exchange (transport) fluxes for all reactions.[31–35]

A variant of FBA that is particularly valuable for microbial ecology studies is an extension called dynamic flux balance analysis (dFBA). dFBA still assumes a steady state for intracellular metabolism but treats environmental abundances of different microbes and metabolites as dynamic variables.[36–39] The dFBA approach, which is at the core of the COMETS software, is particularly useful for the study of metabolism in microbial communities. COMETS can take as input a set of GEMs and environmental conditions and use dFBA and diffusion equations to predict the dynamics of molecules and microbes in structured environments. COMETS is emerging as an open-source collaborative platform for modeling microbial ecosystems computationally.[14,40]

In this chapter, we present a detailed approach for generating COMETS simulations of chemotaxis and metabolism for a bacterial colony. We will illustrate the process starting from the construction a genome-scale stoichiometric model for *Pseudomonas simiae WCS417*. The model will be used to implement COMETS simulations of chemotaxis in Petri dishes inoculated in the presence of a chemoattractant, alanine. This specific bacterium was chosen for our case study because it is known to be able to colonize plants roots, promote growth, and provide overall benefits to its host.[41] The output of the simulations is the spatially detailed and time resolved dynamics of the biomass density, concentrations of all extracellular metabolites, as well as all reaction fluxes (rates), including the growth rate and chemoattractant uptake. In addition to describing the results, we will illustrate recommended steps for the visualization and analysis of simulation data. Finally, we will provide the detailed experimental protocol for growth of *P. simiae* on a plate with alanine, and describe the observed chemotaxis-induced pattern, which we used to fit model parameters and verify predictions.

Materials

COMETS and COMETSPy

COMETS (Computation of Microbial Ecosystems in Time and Space) is a software developed with the goal of computational spatio-temporal modeling the growth and propagation of microbial colonies in space. The current version (2.12.4) is limited to two-dimensional surfaces. COMETS is written in Java and utilizes two optimizer software packages: Gurobi and Glop. In addition to the core Java code, a Python interface, COMETSPy is available as a user interface (UI).[14] Detailed information about COMETS, including installation, is available at www.runcomets.org. COMETS can be downloaded and installed from https://www.runcomets.org/installation. The source code of COMETS is freely available at https://github.com/segrelab/comets. The COMETSPy source code is available at https://github.com/segrelab/cometspy. The COMETSPy scripts and the genome-scale model used in this chapter can be found at: https://github.com/segrelab/COMETS_Chemotaxis/

Kbase

Kbase (Knowledgebase) is a software developed by the US Department of Energy. It integrates several methods for systems biology research and data analysis. It can be accessed at: https://www.kbase.us/

We used the following Kbase Narrative: https://narrative.kbase.us/narrative/222319

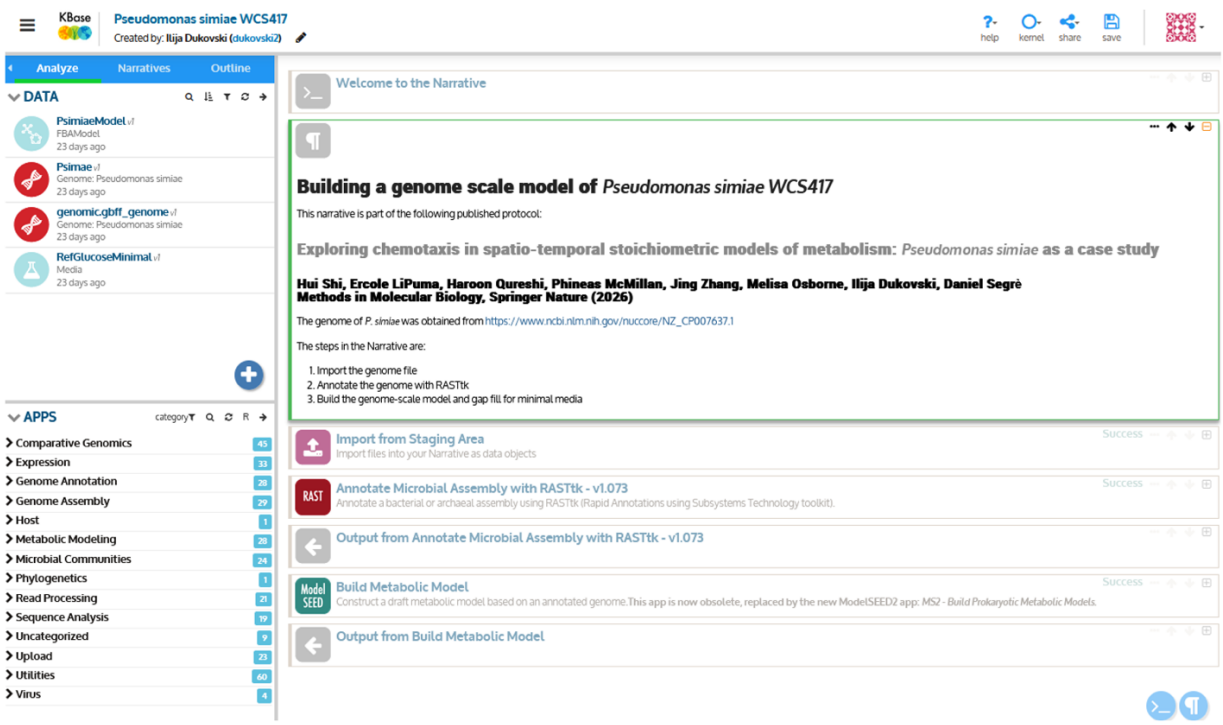

Figure 2: Kbase Narrative for building a genome-scale metabolic stoichiometric model of *P. simiae*.

The Kbase Narrative for building genome-scale metabolic model of *P. simiae* consists of three applications: Import from Staging Area, Annotate Microbial Assembly with RASTtk and Build Metabolic Model. The imported data as well as the model output are in the Data panel on top left. The apps are loaded from the Applications panel on the bottom left.

Software requirements and the procedure for installing COMETS:

COMETS is written in Java. Java can be obtained and installed from https://www.java.com/en/download/manual.jsp

Python

The user interface to COMETS, COMETSPy, is written in Python. Although COMETS can be used without it, we recommend using COMETSPy with Jupyter notebooks, which have a convenient user interface. We recommend installing and using Python from the Anaconda distribution from: https://www.anaconda.com/download

CobraPy

CobraPy is a Python package that provides an interface to metabolic constraint-based modeling. We used this package to facilitate the handling of our genome-scale models. CobraPy can be downloaded and installed using the pip command in an Anaconda power shell terminal or a command line terminal: `pip install cobra`
or alternatively from: https://opencobra.github.io/cobrapy/

Pandas

Pandas is a Python package that enables users to manipulate data in the form of data frames. We use this package to handle and analyze output data (biomass, concentration, etc.) from simulation runs. Pandas can be downloaded and installed using the pip command: `pip install pandas` or alternatively from:
https://pandas.pydata.org/docs/getting_started/install.html

Linear optimization software

A crucial component of COMETS is the Flux Balance Analysis algorithm based on linear optimization of the metabolic resources' allocation. In COMETS, we use the following two optimizers.

Glop

The default linear optimizer in COMETS is Glop. It is part of the OR-Tool suite: https://developers.google.com/optimization. Glop is distributed with COMETS and the user does not need to perform any additional installation steps.

Gurobi

An alternative optimizer that can be used in COMETS is Gurobi.[14] The package can be downloaded and installed from http://www.gurobi.com/. The installation of Gurobi requires obtaining a license from http://www.gurobi.com/downloads/licenses/license-center. Academic users that will use COMETS on an individual basis may obtain a free academic license. When the installation of Gurobi is finished, it is very important to have the environment variable GUROBI_HOME set to the directory where Gurobi was installed. In Windows, this variable is set automatically during the installation process. In Linux and MacOS, however, depending on the system, sometimes this is not the case. It is therefore important to make sure that a line such as the following example:

```
export GUROBI_HOME=/usr/gurobi/gurobi902/linux64/
```

is included in the user's .bashrc file in Linux, or the corresponding file for an alternative shell. The version name and number in gurobi902 should be set to the one installed. Here, we use Gurobi version 9.0.2 (i.e. 'gurobi902') as a representative example version of Gurobi, but we anticipate that the following steps will continue to be valid for subsequent implementations.

In MacOS, the line:
`export GUROBI_HOME=/Library/gurobi902/mac64/`
should be included in the .bash_profile file in older versions of MacOS, or .zshrc in the latest version of MacOS. The version name and number in gurobi902 should be set to the one installed.

Chemotaxis Motility Assays

The agar plates were created with 1% agar, 2x RCH2 medium without carbon, and 80% glycerol. For the chemotaxis experiments, alanine was added as the chemoattractant. All materials are listed below. Prepare all solutions using distilled (DI) water. Solutions can all be kept at room temperature. We used 6 cm wide petri dishes.

All the materials used to create the agar plates are listed below. Volumes can be adjusted based on the number of plates to be made. One agar plate contained 10 mL.

Ingredient	Single plate (10 ml)	Master mix (x15)	Master mix 2 (x5)
1% agar	3 ml	45 ml	15 mL
2X RCH2 (-C)	5 ml	75 ml	25 mL
80% glycerol	2.5 ul	37.5 ul	12.5 uL
dH_2O	1.5 ml	22.5 ml	0
Amino acid (5mM)	0.5 ml	1 mL / amino acid	4 mL / amino acid

The 10x Salts and 1M PIPES buffer stock solutions should be made first as other solutions require them.

10x Salts

Total volume	1L
NH_4Cl	2.5 g
KCl	1 g
NaH_2PO_4 (sodium phosphate monobasic)	5.2 g

Measure the amount of each salt listed above separately using a spatula, weigh boat and scale. Add the compounds to a 1 L glass jar and add DI water up to 1 L. The solution should be mixed thoroughly with a magnetic mixer and stir bar. Autoclave when done.

1M PIPES buffer

Total volume	100 mL	150 mL
PIPES-Na (335.437g)	33.534 g	50.301 g

(https://goldbio.com/documents/3579/PIPES%20Sodium%20Buffer%201M%20Stock%20Solution%20with%20table.pdf)

Measure the amount of PIPES-Na as shown above and add DI water to get the total volume. For our stock solution of PIPES buffer, we created 250 mL. The amounts above can be adjusted based on need. The solution should be mixed thoroughly with a magnetic mixer and a stir bar.

1% Agar

Agar	5 g
DI water	500 ml

Weigh out 5 g of agar using a spatula, weigh boat and scale. Add this powder to a 1 L glass jar and fill it with 500 mL of DI water. The solution should be thoroughly mixed with a magnetic mixer with a stir bar. Autoclave the solution when done.

2x RCH2 medium without carbon

Total volume	0.5 L
10x Salts (**Step 2)**	100 mL
Vitamin Supplement	10 mL
Trace Mineral Supplement	10 mL
1M PIPES buffer	30 mL
DI water	350 mL

The 10x Salts, 1M PIPES buffer and DI water should be added first to the 1 L glass jar while keeping in mind the volume of NaOH and ATCC Vitamins and Minerals that will be needed in the total 0.5 L solution. DI water volume can be adjusted to be less or more based on the volume needed. The pH should be adjusted to 7.1. After the target pH is reached and thoroughly mixed using a stir bar and magnetic mixer, the solution should be autoclaved. ATCC Vitamin Supplement (https://www.atcc.org/products/md-vs) and ATCC Trace Mineral Supplement (https://www.atcc.org/products/md-tms) can then be added. The Vitamin Supplement and Trace Mineral Supplement should be kept refrigerated at 4℃ before use. The stock solution should be mixed using a stir bar and magnetic mixer.

100 mM Amino Acid

Amino Acid	Mass to Get 0.005 mol (g)
Alanine	0.45

We added 0.45 g of alanine to around 40 mL of water. In addition, we added enough NaOH to each bottle conical tube to reach a pH around 7. The conical tubes should be vortex in a way that the vortex reaches completely to the bottom of the tube to ensure thorough mixture.

80% Glycerol

The amount of glycerol used was 2.5 uL. There are a variety of ways to create this solution as long as the concentration is 80%.

Methods

Complete pipeline from genome to chemotaxis simulation in COMETS

1. Building a genome-scale metabolic model of Pseudomonas simiae WCS417 in Kbase (https://narrative.kbase.us/narrative/222319)

1.1 Download the genome of the strain *P. simiae* WCS417 from NCBI:
https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP007637.1
Click on ”Assembly” link in the bar on the right-hand side. This will take you to:
https://www.ncbi.nlm.nih.gov/datasets/genome/GCF_000698265.1/
Click on “Download.” A window will pop up to choose the sources and types of data to download.
On the popup window, choose “All” sources and “Genome sequences (FASTA),” “Sequence and annotation (GBFF)” and “Protein (FASTA).” Click on “Download.” Unzip the downloaded files.

1.2 Create an account and login to Kbase: https://narrative.kbase.us
Click on “Navigator” in the top left corner. Create a new narrative by clicking on “+ New Narrative” in the top right corner. Once a new narrative is created, rename it to your liking by clicking on “Unnamed” in the top left corner. In this case, name it “P. simiae WCS417.”

1.3 Click on “Add Data” on the “DATA” panel (stage) on the left (see Fig. 2). On the popup panel, click “Import” in the top right. Click on “Select” and select the file “genomic.gbff” from the genome folder that you downloaded in step 1.1.
The full path is: ncbi_dataset\ncbi_dataset\data\GCA_000698265.1\genomic.gbff.
The app “Import from Staging Area” will be added to your Narrative. Choose “GenBank Genome” in the “Import as” field and type “P. simiae” in “Name” and click on “Run.” This will import the file, and the genome and assembly files will show up on the left “DATA” area.

1.4 Add minimal media to your data, by clicking on the plus sign in the left panel. Click on “Public” on the top and choose “Media” as data type and search for “RefGlucoseMinimal.” Choose “RefGlucoseMinimal” and click on “Add” arrow. This will add “RefGlucoseMinimal” media to your data.

1.5 Open the “Annotate Microbial Assembly with RASTtk” app from the “Genome Annotation” app block. This will add the app to your Narrative.

1.6 Choose the file “genomic.gbff_genome” file. Add the scientific name P. simiae. Add a name of the output file. In this case it is “Psimiae.” Press “Run.” This will create a genome annotation file.

1.7 Open the “Metabolic Modeling” block in the Apps area on the bottom left, and choose the “Build Metabolic Model” app. This will add the app to your Narrative.

1.8 In the “Build Metabolic Model” app, choose your genome annotation and media. Use “Automatic selection” for “Template reconstruction.” Choose “Aerobic glucose” for “Media for core ATP gapfilling”. Select “Gapfill model?” Choose a name for your model, in this case “PsimiaeModel.” Click on “Run.” This will build your model.

1.9 On the “DATA” panel on the left, click on the model “PsimiaeModel.” Choose the “export/download data” button, and download the model in SBML format. Rename the model as needed, in this case Psimiae_withATPM.xml.

2. COMETS Simulations of a *P. simiae* colony propagation with and without chemotaxis to alanine (Psimiae_chemotaxis_alanine.ipynb)

2.1. Download the scripts and all relevant files from https://github.com/segrelab/COMETS_Chemotaxis.

2.2 Start Jupyter Notebook, navigate to COMETS_Chemotaxis/simulation_notebooks/ and open the Jupyter notebook Psimiae_chemotaxis_alanine.ipynb.

2.3 Import the relevant python libraries. These libraries can be downloaded using “pip install” and the package name in terminal. They can also be installed with pip as shown in the first line. See the “Materials” section for more information.

```
#!pip install cobra # Install cobra if have not done it yet.
import cometspy as c
from matplotlib import pyplot as plt
import pandas as pd import numpy as np
#Import cobra libraries from the cobra toolbox.                #They are
needed to load model files in sbml or other formats.
from cobra.io import load_json_model, save_json_model, load_matlab_model,
save_matlab_model, read_sbml_model, write_sbml_model
```

2.4 Load the genome scale model (GEM) of the organism we want to model.
The argument should be the name of the file, or full path if not in the working directory. See Notes for more information about loading GEMs.

```
# First, load the genome-scale model of P. simiae
cobra_model = read_sbml_model("Psimiae_withATPM.xml")
```

2.5 The maintenance reaction lower bound, i.e. ATP maintenance requirement needs to be specified as needed. In this case we set it to zero.

```
#Modify the ATP maintenance lower limit as needed.
#In this example we will not impose any ATP maintenance limit.
cobra_model.reactions.get_by_id("ATPM").lower_bound = 0.0
```

2.6 Create a COMETS format metabolic model from the SBML Cobra model format. Use the command open exchanges to ensure that all exchange lower bounds are set to -1000.

```
#Create a COMETS model from the SBML/Cobra model format.
Psimiae = c.model(cobra_model)
Psimiae.id = "psimiae"
Psimiae.open_exchanges() # Make sure that exchanges are active.
```

2.7 Specify the model specific parameters of the two biophysical models of colony biomass propagation. First is the model of standard nonbiased swimming of bacteria. The second one is chemotaxis specific model. See Notes for detailed description of the input parameters.

```
#Set the parameters of the biomass propagation model.
#In this case, we set the parameter for simple, linear diffusion of the
biomass.
#This setting corresponds to active swimming bacteria.
Psimiae.add_nonlinear_diffusion_parameters(zero = 1.0e-7, n = 0.0, exponent =
0.0, hilln = 1.0, hillk = 0.0)
#Set the parameters of the chemotaxis model.
Psimiae.add_nonlinear_diffusion_chemotaxis_parameters( hilln = 1.0, hillk =
0.0)
```

2.8 Define the array that holds information about the amount and location of the bacteria. Users should consider a set of coordinates to serve as the midpoint of the circle and then determine the size of the circle. The current code defines (100, 100) to be the midpoint with a circle of radius 5 grid points. Each grid point within the circle is initialized with 1e-7 grams of biomass.

```
# Create an initial innoculum with P. simiae, that will be added to the
spatial layout in the next cell.
Psimiae.initial_pop = []
for i in range(95,105):
    for j in range(95,105):
        Psimiae.initial_pop.append([i,j,1e-7])
```

2.9 Set the desired size of the grid and layout.

```
# Create a spatial layout, a square of 201 by 201 grid points, with the P.
simiae innoculum created above.
# Later we will define each grid point to have a linear length of 0.03 cm.
# We are thus creating a 6 by 6 cm, two-dimensional Petri dish layout.

layout = c.layout([Psimiae])
layout.grid = [201, 201]
```

2.10 Define the amount of each compound to be used in the medium. See Notes for more information about compound naming conventions.

```
# Pour the media in the layout.
# The model of P. simiae created in Kbase uses ModelSeed names for the
metabolites.
# If you want to add additional metabolite, you must find its ModelSeed name.
# Here the unit is mmols per grid pixel in the layout.

layout.set_specific_metabolite('cpd00013_e0', 0.0000029)#NH3
```

```
layout.set_specific_metabolite('cpd00099_e0', 1000) #Cl
layout.set_specific_metabolite('cpd00205_e0', 1000) #K
layout.set_specific_metabolite('cpd00971_e0', 1000) #Na
layout.set_specific_metabolite('cpd00067_e0', 1000) #H
layout.set_specific_metabolite('cpd00009_e0', 0.0000027) #H2PO4
layout.set_specific_metabolite('cpd00254_e0', 1000) #Mg
layout.set_specific_metabolite('cpd00048_e0', 0.000018) #SO4
layout.set_specific_metabolite('cpd00030_e0', 1000)#Mn2+
layout.set_specific_metabolite('cpd00001_e0', 1000) #H2O
layout.set_specific_metabolite('cpd10515_e0', 1000)#Fe+2
layout.set_specific_metabolite('cpd00063_e0', 1000) #Ca
layout.set_specific_metabolite('cpd00149_e0', 1000)#Co2+
layout.set_specific_metabolite('cpd00034_e0', 1000) #Zn
layout.set_specific_metabolite('cpd00058_e0', 1000) #Cu
layout.set_specific_metabolite('cpd09225_e0', 0.00000010) #H3BO3
layout.set_specific_metabolite('cpd11574_e0', 0.000000076) #MoO4
layout.set_specific_metabolite('cpd00244_e0', 1000)#Ni2+
layout.set_specific_metabolite('cpd15574_e0', 0.000000048) #WO4
layout.set_specific_metabolite('cpd00104_e0', 1000) #biotin
layout.set_specific_metabolite('cpd00393_e0', 1000) #folic acid
layout.set_specific_metabolite('cpd00263_e0', 1000) #pyridoxine
layout.set_specific_metabolite('cpd00220_e0', 1000) #riboflavin
layout.set_specific_metabolite('cpd00305_e0', 1000) #thiamine
layout.set_specific_metabolite('cpd00218_e0', 1000) #nicotinic acid
layout.set_specific_metabolite('cpd00644_e0', 1000) #pantothenic acid
layout.set_specific_metabolite('cpd03424_e0', 1000) #vitamin b12
layout.set_specific_metabolite('cpd00443_e0', 1000) #4-Aminobenzoate
layout.set_specific_metabolite('cpd00541_e0', 1000) #thioctic acid (lipoate)
layout.set_specific_metabolite('cpd00027_e0', 2e-5) #D-glucose #1e-5
layout.set_specific_metabolite('cpd00007_e0', 1000) #O2
layout.set_specific_metabolite('cpd00035_e0', 1e-5) #alanine
```

2.11 Assign the chemotaxis coefficient and uptake parameters to the chemoattractant. See notes for the description of the input parameters.

```
# Set the chemotaxis model parameters. Here we define alanine (cpd00035_e0 is
alanine ModelSeed name) as chemoattractant.
layout.set_chemotaxis(0, "cpd00035_e0", 0.0001, 1e-6)
```

2.12 Define the diffusivity of the media.

```
# Set the diffusivity of the media. The unit is cm^2/sec.
layout.set_metabolite_diffusion(5e-6)
```

2.13 Modify the parameters of the simulation as needed. See Notes for a detailed description of the parameters.

```
#Now create a class of parameters for the simulation.
parameters = c.params()

#Populate the parameters with values.
parameters.all_params["maxCycles"] = 500 #The total numer of cycles to be
run.
parameters.all_params["timeStep"] = 0.05 # Unit hours.
parameters.all_params["spaceWidth"] = 0.03 # Unit cm. Linear size per layout
```

```
grid pixel.
parameters.all_params["biomassMotionStyle"] = "Nonlin Diff Chemotaxis 2D"
#The name of the biophysical model.
parameters.all_params["defaultKm"] = 0.069 # Michaelis constant for the
uptake. Unit mmol/cm^3.
parameters.all_params["defaultVmax"] = 7.1 # Maximum uptake flux. Unit
mmol/gh.
parameters.all_params["maxSpaceBiomass"] = 10 # This is a large number to
stop growth if the biomass surpasses it. Unit grams.
parameters.all_params["minSpaceBiomass"] = 1.e-11 # If the biomass falls
below this value, it will be reset to zero.
parameters.all_params["numRunThreads"] = 4 # Numer of parralel threads to run
in the simulation.
parameters.all_params["writeBiomassLog"] = True # Save the biomass spatio-
temporal data.
parameters.all_params["BiomassLogRate"] = 50 # The cycles period to save the
biomass data. This will save avary 50th time step.
parameters.all_params["writeTotalBiomassLog"] = True # Save the total
integrated biomass data.
parameters.all_params["totalBiomassLogRate"] = 1 # The cycles period to save
the total biomass data.
parameters.all_params["writeFluxLog"] = True # Save all the reaction fluxes
spatio-temporal data.
parameters.all_params["FluxLogRate"] = 50 # The cycles period to save the
fluxes data.
parameters.all_params["writeMediaLog"] = True # Save all the reaction media
(extracellular metabolites) spatio-temporal data.
parameters.all_params["MediaLogRate"] = 50 # The cycles period to save the
media data.
```

2.14 Consolidate the layout and parameters into a COMETS object and then run the simulation.

```
#Put together the layout and the simulation parameters in a single comets
simulation.
simulation = c.comets(layout, parameters)

#Run the simulation. The argument False provides that the output files are
not deleted after the simulation.
#Setting the parameter to True, will clean the working directory clean of all
output.
simulation.run(False)
```

2.15 Print the output of the simulation for progress monitoring and debugging purposes if needed.

```
#Check the command line output to confirm that the run finished with no
errors.
print(simulation.run_output)
```

3. Visualization and analysis of the simulation output

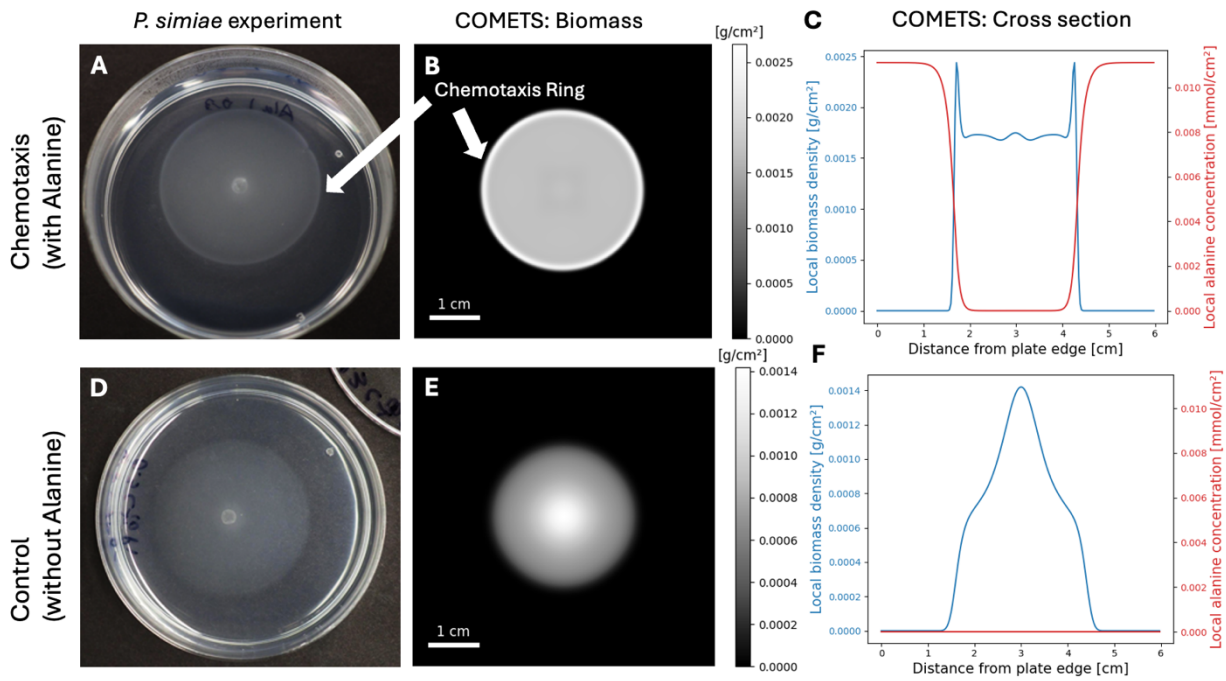

Figure 3: COMETS simulations and experiments of chemotaxis (and its absence) of *P. simae* in the presence (and the absence) of alanine in the media.
P. simiae colonies were **A)** grown in the presence of alanine and **B)** simulated in the equivalent model layout. The chemotaxis ring at the edge of the colony is evident both in the experimental and simulated colonies. **C) T**he cross section of the simulated colony, with the leading peak coinciding with the gradient of alanine, created due to the consumption of alanine by the bacteria. **D)** An experiment without the presence of alanine, and **E)** its equivalent simulation. The chemotaxis ring is absent in both cases. **F)** The absence of a leading peak in the cross section.

Biomass

3.1 Define the function that will help with biomass visualization

```
#This function will load the file and return a matrix of spatial biomass
distribution at a given time

import numpy as np
def read_biomass_file_and_process(filename, Nx, Ny, time_cycle):
  """
  Reads a file with the specified format and processes each line.

  Args:
    filename: The path to the file to be read.
    Nx: Size of the spatial grid in x direction.
    Ny: Size of the spatial grid in y direction.
    time_cycle: The simulation time step to be loaded.
  """

  matrix_data=np.zeros((Nx+1,Ny+1))  #Create an empty matrix with the spatial
grid size for the biomass data.
```

```
  # Read the file and select the biomass grid content at the time_cycle.
  with open(filename, 'r') as file:
    for line in file:
      line = line.strip()
      if not line:
        continue

      values = line.split()

      try:
        val1 = int(values[0])   #time cycle
        val2 = int(values[1])   #x coordinate
        val3 = int(values[2])   #y coordinate
        string_val = values[3]  #name of the model
        scientific_val = float(values[4])  #biomass content in grams

        #select the specified time cycle and populate the matrix with biomass
content
        if val1==time_cycle:
            matrix_data[val2,val3]=scientific_val

      except (ValueError, IndexError) as e:
        print(f"Error processing line: {line} - {e}")

  return matrix_data
```

3.2 Load in the biomass file. See notes for details on input file name.

```
# Now we will load the biomass file.
# Change the filename string as it was produced by the simulation

filename = "biomasslog_0x14c6a7706410"
biomass_data=read_biomass_file_and_process(filename, Nx=201, Ny=201,
time_cycle=250)
```

3.3 Convert the units in the original biomass file to our preferred version.

```
#In this step we will transform the data from biomass content per pixel in
grams,
#to biomass density per cm^2. The linear size of a pixel in this simulation
was L=0.03cm.
#The linear size should be changed according to the one set in the
simulation.

L=0.03
biomass_data=biomass_data/(L*L)
```

3.4 Import relevant libraries and graph the biomass. The results are shown in Figs. 3B and 3E.

```
#Now we are ready to visualize the biomass.

import matplotlib.pyplot as plt
import matplotlib.patches as patches

fig, ax = plt.subplots()
```

```
# Display the data as an image and get the mappable object.
# The cmap (colormap) was chosen to produce a grayscale image.
im = ax.imshow(biomass_data, cmap='Greys_r')

# Create the colorbar
cbar = fig.colorbar(im)

# Set the title of the colorbar
cbar.ax.set_title('[g/cm\u00b2]', loc='center') # loc='center' is default but
can be specified
plt.axis('off') #do not plot the axis

# Show a size bar with a "1cm"label
plt.text(15, 180, '1 cm', color='white', fontsize=12)
# Define rectangle parameters: (x_lower_left, y_lower_left, width, height)
rect_x = 10
rect_y = 185
rect_width = 33
rect_height = 2
# Create the Rectangle patch
rect = patches.Rectangle((rect_x, rect_y), rect_width, rect_height,
                            linewidth=2, edgecolor='w', facecolor='w')
# Add the patch to the Axes
ax.add_patch(rect)
#Add title
plt.title("Biomass density")
# First save the image, then display the plot
plt.savefig("biomass.png")
plt.show()
```

Media

3.5 Define the function that will help with media visualization.

```
#This function will load the file and return a matrix of spatial biomass
distribution at a given time

def read_media_file_and_process(filename,metabolite, Nx, Ny, time_cycle):
  """
  Reads a file with the specified format and processes each line.

  Args:
    filename: The path to the file to be read.
    metabolite: The name of the metabolite data to be loaded.
    Nx: Size of the spatial grid in x direction.
    Ny: Size of the spatial grid in y direction.
    time_cycle: The simulation time step to be loaded.
  """

  matrix_data=np.zeros((Nx+1,Ny+1)) #Create an empty matrix with the spatial
grid size for the media data.

  # Read the file and select the media grid content at the time_cycle.
  with open(filename, 'r') as file:
    for line in file:
```

```
      line = line.strip()
      if not line:
        continue

      values = line.split()

      try:
        string_val = values[0] # Name of the metabolite
        val1 = int(values[1])  # time cycle
        val2 = int(values[2])  # x coordinate
        val3 = int(values[3])  # y coordinate
        scientific_val = float(values[4]) # metabolite content (in mmol)
        if string_val==metabolite and val1==time_cycle:
            matrix_data[val2,val3]=scientific_val

      except (ValueError, IndexError) as e:
        print(f"Error processing line: {line} - {e}")

  return matrix_data
```

3.6 Filter the media file for the media of interest and load the data. See notes on how to handle large files.

```
# Now load the data. In this case the file was too large, so we filtered it
only for alanine
# with the following command in a terminal:
# cat medialog_0x14c6a7706410 |grep cpd00035_e0 >alanine
# The input file is named 'alanine' in this case

filename = "alanine"
media_data=read_media_file_and_process(filename,metabolite='cpd00035_e0',Nx=2
01,Ny=201,time_cycle=250)
```

3.7 Calculate the concentration of alanine based on the amount listed in the files.

```
# As for the biomass, we will calculate the concentration of alanine

L=0.03 #cm
media_data=media_data/(L*L)
```

3.8 Import pertinent libraries and visualize the media. The result is shown in Fig. 4A.

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import seaborn

fig, ax = plt.subplots()

# Display the data as an image and get the mappable object
im = ax.imshow(media_data[1:201,1:201], cmap='viridis')

# Create the colorbar
cbar = fig.colorbar(im)

# Set the title of the colorbar
```

```
cbar.ax.set_title('[mmol/cm\u00b2]', loc='center') # loc='center' is default
but can be specified

#Do not display the axis
plt.axis('off')

# Include a size bar and a label
plt.text(15, 180, '1 cm', color='black', fontsize=12)
# Define rectangle parameters: (x_lower_left, y_lower_left, width, height)
rect_x = 10
rect_y = 185
rect_width = 33
rect_height = 2

# Create the Rectangle patch
rect = patches.Rectangle((rect_x, rect_y), rect_width, rect_height,
                            linewidth=2, edgecolor='k', facecolor='k')

# Add the patch to the Axes
ax.add_patch(rect)
#Add title
plt.title("Alanine concentration")
# Save the image
plt.savefig('alanine.png')
# Display the plot
plt.show()
```

Cross Section of Biomass and Alanine

3.9 Define the x coordinate and the range of y coordinates from which to take a cross section.

```
# Here we take a cross section in the middle of the layout
media_cross_section=media_data[100,1:201]
biomass_cross_section=biomass_data[100,1:201]
```

3.10 Plot the cross sections. The results are shown in Figs. 3C and 3F.

```
# Now we can plot the cross section profile

import matplotlib.pyplot as plt
import numpy as np

# Get the Data first
x=np.arange(0.0,6.0,0.03) #This is the x distance from the plate edge
y1=biomass_cross_section[0:200] #Biomass cross section
y2=media_cross_section[0:200]   #Alanine cross section

# Create the first plot (left y-axis)
fig, ax1 = plt.subplots()

color = 'tab:blue'
ax1.set_xlabel('Distance from plate edge [cm]', fontsize=15)
ax1.set_ylabel('Local biomass density [g/cm\u00b2]', color=color,
fontsize=15)
ax1.plot(x, y1, color=color)
ax1.tick_params(axis='y', labelcolor=color)
```

```
# Create the second plot (right y-axis)
ax2 = ax1.twinx()  # Instantiate a second Axes that shares the same x-axis

color = 'tab:red'
ax2.set_ylabel('Local alanine concentration [mmol/cm\u00b2]', color=color,
fontsize=15)
ax2.plot(x, y2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

# Adjust layout
fig.tight_layout()  # Adjust layout to prevent labels from overlapping
#Add title
plt.title("Biomass and alanine cross section ")

#Save the plot and display it
plt.savefig("biomass_alanine.png")
plt.show()
```

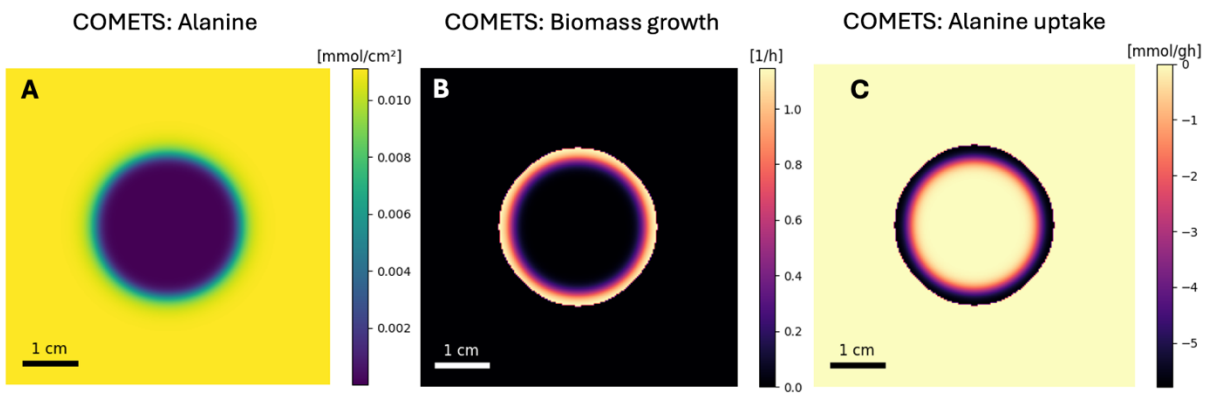

Figure 4: A typical COMETS output consists of a time series of all spatial distributions of metabolite concentrations and reaction fluxes, including biomass growth rate.
A) Simulated alanine concentration showing the depletion region at the center of the layout and its gradient at the colony edge. **B)** The biomass growth is localized at the edge of the colony and coincides with **C)** the region of alanine uptake. The growth region where the uptake is taking place is where the alanine gradient is produced, which in turn triggers chemotaxis and produces the leading biomass ring, shown in Fig. 3.

Uptake Fluxes and Biomass Growth Rate

3.11 Define the function that will read the fluxes from the file.

```
#Function to read the fluxes output file
def read_flux_file_and_process(filename,flux,Nx,Ny,model_number,time_cycle):
  """
  Reads a file with the specified format and processes each line.

  Args:
    filename: The path to the file to be read.
    flux: The name of the reaction flux to be processed.
    Nx: Size of the spatial grid in x direction.
    Ny: Size of the spatial grid in y direction.
```

```
    time_cycle: The simulation time step to be loaded.
  """

  matrix_data=np.zeros((Nx+1,Nx+1))
  with open(filename, 'r') as file:
    for line in file:
      line = line.strip()
      if not line:
        continue

      values = line.split()

      try:
        val0 = int(values[0])
        val1 = int(values[1])
        val2 = int(values[2])
        val3 = int(values[3])
        scientific_val = float(values[flux+3])
        if val0==time_cycle and val3==model_number:
            matrix_data[val1,val2]=scientific_val


      except (ValueError, IndexError) as e:
        print(f"Error processing line: {line} - {e}")

  return matrix_data
```

3.12 Load in the flux data and get an array of alanine uptake flux.

```
#Before loading the fluxes data, you need to know what is the location of the
flux of interest in the array of fluxes.
#Here we will load the alanine uptake flux.

filename = "fluxlog_0x149264947d30"
flux_data=read_flux_file_and_process(filename,flux=1439,Nx=201,Ny=201,model_n
umber=1,time_cycle=250)
```

3.13 Load in relevant libraries and visualize the flux. The result is shown in Fig. 4C.

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import seaborn

fig, ax = plt.subplots()

# Display the data as an image and get the mappable object
im = ax.imshow(flux_data[1:201,1:201], cmap='magma')

# Create the colorbar
cbar = fig.colorbar(im)

# Set the title of the colorbar
cbar.ax.set_title('[mmol/gh]', loc='center') # loc='center' is default but
can be specified
plt.axis('off')
plt.text(15, 180, '1 cm', color='black', fontsize=12)
```

```
# Define rectangle parameters: (x_lower_left, y_lower_left, width, height)
rect_x = 10
rect_y = 185
rect_width = 33
rect_height = 2

# Create the Rectangle patch
rect = patches.Rectangle((rect_x, rect_y), rect_width, rect_height,
                            linewidth=2, edgecolor='k', facecolor='k')

# Add the patch to the Axes
ax.add_patch(rect)

#Add title
plt.title("Alanine uptake")
plt.savefig('Flux_EX_alanine.png')
# Display the plot
plt.show()
```

3.14 Read the biomass growth rate.

```
#Now we can do the same for biomass growth rate
biomass_growt_rate=read_flux_file_and_process(filename,flux=1417,Nx=201,Ny=20
1,model_number=1,time_cycle=250)
```

3.15 Load in the relevant libraries and visualize the biomass flux. The result is shown in Fig. 4B.

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import seaborn

fig, ax = plt.subplots()

# Display the data as an image and get the mappable object
im = ax.imshow(biomass_growt_rate[1:201,1:201], cmap='magma')

# Create the colorbar
cbar = fig.colorbar(im)

# Set the title of the colorbar
cbar.ax.set_title('[mmol/gh]', loc='center') # loc='center' is default but
can be specified
plt.axis('off')
plt.text(15, 180, '1 cm', color='black', fontsize=12)
# Define rectangle parameters: (x_lower_left, y_lower_left, width, height)
rect_x = 10
rect_y = 185
rect_width = 33
rect_height = 2

# Create the Rectangle patch
rect = patches.Rectangle((rect_x, rect_y), rect_width, rect_height,
                            linewidth=2, edgecolor='k', facecolor='k')

# Add the patch to the Axes
ax.add_patch(rect)
```

```
#Add title
plt.title("Biomass growth rate")
plt.savefig('Flux_EX_alanine.png')
# Display the plot
plt.show()
```

Experimental Methods

4. Experimental Chemotaxis Motility Assays (in vitro)

1. One liter of RCH2 medium without a carbon source contained 10x salts (100 mL), ATCC Vitamin Supplements (10 mL, stored at 4℃), ATCC Mineral Supplements (10 mL, stored at 4℃), 1M PIPES buffer (30 mL) and DI water (850 mL). Each batch of RCH2 medium was adjusted to pH 7.1. All solutions were added to a 1 L glass jar and mixed thoroughly with a stir bar and magnetic mixer.

2. PIPES buffer and 10x salts were also created. PIPES salt buffer for 150 mL and 1 M was created by dissolving 0.15 moles of PIPES-Na into enough DI water to create a 150 mL solution. 10x salts for 1 L contained 2.5 g of NH_4Cl, 1 g of KCl and 5.2 g of NaH_2PO_4.

3. Stock solutions with a concentration of 100 mM were created for alanine in a conical tube and vortex. Agar solutions of 1% were also created in a 1 L glass jar.

4. A 0.3% agar plate of 10 mL would contain 3 mL of 1% agar, 5 mL of RCH2 medium, 2.5 µL of 80% glycerol, 0.5 mL alanine and adjusted to 10 mL with DI water. For each plate, 50% of the solution must be the RCH2 medium. The volume of amino acid added will depend on the desired concentration of amino acid in the plate. The same applies for the agar amount added. The volume will then be adjusted to 10 mL with DI water. 2.5 µL of 80% glycerol is spiked in.

5. To create the plates, each solution was added into a new conical tube with the exception of the agar. At pouring time, hot agar solution was added to the master mix, vortexed and then poured into 6 cm culture plates.

6. Plates were let sit for a day.

7. Pseudomonas simiae 1602 was inoculated into R2A solution and let grow for a day. When the agar hardened and *P. simiae* grew, *P. simiae* were spotted on the plates with a P20 pipette dispensing 2 µL per plate and then incubated at 30℃.

8. 2 plates of 0.3% agar were created for alanine. For the control, two plates of 0.3% agar without alanine were created. Plates are created in duplicate for verification.

9. Plates were imaged every three hours for 24 hours using a Nikon camera. When plates were not imaged, they were kept in the incubator. The results are shown in Figs. 3A and 3C.

Notes

Notes for building a genome-scale metabolic model of *Pseudomonas simiae* WCS417 in Kbase

1. When building Genome Scale Models (GEMs) the quality of the sequenced genome is crucial. Typically, the genomes are not complete, but contain missing parts, gaps. The model building algorithms can perform a stochastic gap filling of the genome, but that process may be very inaccurate if the genome is of poor quality. It is therefore extremely important to choose a good quality genome, one with as few gaps as possible when building a genome scale metabolic model.

2. In Kbase, one has three choices for an Annotation application: RASTtk, Prokka and DRAM. These three applications may result in different annotations. When building a GEMs from a sequence, it is recommended to try all three apps, and after the building the models check their quality. In this case we decided that the RASTtk app gave us the best result in terms of completeness of the metabolic network when compared to other GEMs such as one for *E. coli*. Prokka and DRAM produced much smaller annotations, resulting in smaller GEMs.

3. When building a GEM, typically we perform a gap-filling step. Gap filling is dependent on the media chosen. In essence, the gap-filling algorithm adds reactions to the GEM, to make sure that the model produces biomass given media with nutrients. The problem here is that if one supplies a rich medium, such as LB, the algorithm will not include reactions for production of essential metabolites, if they are supplied in the media, thus will produce a relatively smaller model compared to one built with minimal media. In this case we used minimal media with supplied glucose, which resulted in a relatively large model. The model building typically does not stop here, but manual curation is necessary. We did not present manual curation protocol, since the emphasis here is on the simulations of chemotaxis rather than producing a high-quality GEM.

4. The final model can be downloaded in several different formats. Our software COMETS uses the systems biology package Cobra to load models. Cobra supports most of these formats. In this case we used the SBML format, but json file would be perfectly fine too.

Notes on COMETS Simulations of a *P. simiae* colony propagation with and without chemotaxis to alanine

1. Cobra is a systems biology software package that is not part of COMETS and needs to be installed separately. Here we use Cobra only to load the SBML format model file, in Methods 2.4. COMETS has its specific model format, and a translation step is necessary, shown in Methods 2.6.

2. The flow of ATP in the metabolic networks is often restricted to being higher than a certain preset level. This minimal ATP flux requirement reflects the fact that production of ATP is needed even in cells that are not dividing, only maintaining their metabolic state. By setting a nonzero lower bound for the ATP maintenance reaction, one can regulate the maintenance requirements in the bacterial colony. In this case, for the sake of simplicity, we set it to zero in Methods 2.4.

3. In step 2.7 we specify the parameters of the two model terms in equation (1). The first function, `add_nonlinear_diffusion_parameters`, sets the parameters of the model of simple diffusion to simulate the unbiased free swimming of bacteria. This model is numerically solved in the form: $H(r)\vec{\nabla}(D_0 + D_k\rho^k\vec{\nabla}\rho)$ Here $H(r) = r^n/(r^n + K)$ is a Hill function that limits the movement of the cells only to the ones that are growing for K other than zero. The first parameter in the function is D_0 and we set it to $D_0 = 10^{-7}cm^2/s$. To simulate simple diffusion

(free swimming) of all of the biomass, regardless of the growth, we set $D_k = 0$, $k = 0$, $n = 1$, and $K = 0$. The second function `add_nonlinear_diffusion_chemotaxis_parameters` adds a Hill function to prevent chemotaxis for bacteria that are not growing, similar to the one above. In this case we set the parameters to allow all of the biomass to chemotax. The user, however, can experiment with the Hill function parameters to explore the relationship between the growth of the colony and its propagation.

4. Model Seed vs BiGG Compound Naming Conventions
There are different naming conventions for compounds in GEMs. COMETS can handle both naming conventions, but the convention used depends on the model and the way metabolites are named in it. If you use a model with the Model Seed naming convention ("cpd"), then all your metabolites must begin with "cpd" and have the correct ID. If the compounds are in BiGG convention, it is more readable and may have IDs such as "h2o." It is crucial to match the metabolite ID with the name from the GEM. A mismatch will result in an erroneous simulation.

5. The metabolites amounts per pixel of the simulation grid are set in 2.11. The unit is mmol per grid pixel. These are not concentrations. The reason for that is that the grid is 2D, and stating volume concentrations can be misleading here. If a metabolite is considered unlimiting, it is set to a very high number, typically 1000 mmol.

6. The chemoattractant (or repellent) specific parameters for the chemotaxis model are set in 2.11 with the function `layout.set_chemotaxis(0, "cpd00035_e0", 0.0001, 1e-6)`. Here the first parameter is the ID number of the model. In this case we have only one model, so it is set to zero. The second parameter is the name of the chemoattractant in the model. In this case, the ID of alanine in the model is `cpd00035_e0`. The third parameter is the chemotaxis coefficient χ, and the last one is the sensitivity a_m as specified in equations (1) and (2). To switch from a chemoattractant to chemorepellent, one needs to change the sign of χ.

7. The diffusivity of the metabolites can be either globally set the same for all metabolites, or for each metabolite separately. More info can be found in Dukovski et al. (2021).

8. The parameters of the simulation are organized in a class on their own. The time step and linear grid pixel size need to be chosen in a way that produces a numerically stabile simulation. A too large time step in combination with a too small grid pixel size will result in numerical instability.

9. The model for maximum uptake rate in COMETS used here is the Michaelis-Menten formula $V = V_{max}[c]/([c] + K_M)$, where $[c]$ is the concentration of the metabolite, and V_{max} and K_M are parameters of the model. These parameters can be either globally set equal for all metabolites, or separately, specifically for each metabolite. More info can be found in Dukovski et al. (2021).

10. The results of a simulation are total integrated biomass, biomass, media concentrations and fluxes spatial distributions for selected time steps. They are saved as Pandas data frames in the simulation class. They can be accessed as `simulation.total_biomass`, `simulation.biomass`, `simulation.media` and `simulation.fluxes_by_species`. If the parameters to save them into files are set to True, they will be printed out into files that can be analyzed out of the simulation script.

Notes for Visualization and analysis of the simulation output

1. The scripts in this section are specific to this simulation layout. They can easily be customized for different layouts and simulation parameters.

2. It is important to note that the file reading functions are specific to the format of the output files. The output of a COMETS simulation can be saved by the COMETS utilities, if the save output parameters are set to True. However, one can save the output by using the Pandas utility, `simulation.biomass.to_csv('biomass.csv')` for example. In that case, instead of using the load functions presented here, one can load the file directly in Pandas with its utility `pandas.read_csv` and perform custom analysis.

Notes for Agar Plate Motility Assays

1. 10x Salts, PIPES buffer and DI water should be added in and adjusted to pH 7.1 and autoclaved before adding in the ATCC Vitamin and Mineral Supplements. It took us approximately 6 mL of 2 N NaOH for the pH to reach 7.1. pH adjustment can be done using a pH probe. Extent of mixing can be determined visually. When the solution looks to be homogeneous, mixing can stop.

2. The 10x Salts should be autoclaved. We autoclaved it for 45 minutes based on the amount of solution and the material of its container.

3. Vortexing must be done so that the vortex reaches all the way to the bottom to ensure a thorough mixture.

4. The amount of glycerol is negligible to the total volume, and it is not calculated in the total 10 mL. However, the amount of glycerol should be added proportionally to the total volume with 2.5 µL glycerol added per 10 mL of desired solution. For different agar concentrations, the volume can be adjusted.

5. Vortexing must be done so that the vortex reaches all the way to the bottom to ensure a thorough mixture. Vortexing was also done quickly to ensure that the more concentrated agar solutions do not harden before being poured. Plates should be poured in a hood or in front of an open Bunsen burner to ensure sterility. Air bubbles can be avoided by pouring slowly and shuffling the plates from side to side.

6. We set the plates on a lab bench at room temperature.

7. To allow a smoother workflow, the *P. simiae* can be inoculated the day the plates are made so that they are ready for spotting the next day. We obtained our *P. simiae* by selecting a cluster from a cultured dish of bacteria using a pipette tip. The tip was dropped into the R2A solution in a conical tube. New culture should be inoculated each time plates are made. We spotted the plates in a hood and parafilm-wrapped each plate before placing it in the incubator. Parafilm wrapping can be made easier by obtaining strips of parafilm, using one thumb to hold down a side of the parafilm and then using the other to stretch it around the plates.

8. To better keep track of the plates, it is easier to take out the number of plates you need and mark out the conditions on the lid and bottom of the agar plate. This smoothens the process when agar is poured.

9. The colony grew to fill up most of the plate in 24 hours. There are many ways to image these plates. They can be done by a Bunsen burner on a lab bench, in a hood or in a black box. For our images, we imaged them in a hood although some measure of tilting may be used to avoid glare from the plates. One can also put a black piece of paper below to plate to better see the colony morphology. For more advanced imaging, a microscope or confocal microscope may be used.

Acknowledgements

We would like to thank Dr. Kirill Korolev for his advice on modelling chemotaxis, Dr. Konrad Herbst for his help in agar experiments, and Alvin Lu and Suraj Prabhu for their help with the genome-scale model. This work was partially supported by the U.S. Department of Energy, Office of Science, Office of Biological & Environmental Research through the Microbial Community Analysis and Functional Evaluation in Soils SFA Program (m-CAFEs) under contract number DE-AC02-05CH11231 to Lawrence Berkeley National Laboratory; the National Science Foundation (NSF-BSF 2246707 and the NSF Center for Chemical Currencies of a Microbial Planet, Award Number 2019589); the Human Frontiers Science Program (RGP0060/2021); and the Boston University Biological Design Center Kilachand Multicellular Design Program.

References

1. Matilla, M. A. & Krell, T. The effect of bacterial chemotaxis on host infection and pathogenicity. *FEMS Microbiol Rev* 42, (2018).
2. Kundu, P., Blacher, E., Elinav, E. & Pettersson, S. Our Gut Microbiome: The Evolving Inner Self. *Cell* 171, 1481–1493 (2017).
3. Scharf, B. E., Hynes, M. F. & Alexandre, G. M. Chemotaxis signaling systems in model beneficial plant-bacteria associations. *Plant Mol. Biol.* 90, 549–559 (2016).
4. Keegstra, J. M., Carrara, F. & Stocker, R. The ecological roles of bacterial chemotaxis. *Nat. Rev. Microbiol.* 20, 491–504 (2022).
5. Shou, W., Ram, S. & Vilar, J. M. G. Synthetic cooperation in engineered yeast populations. *Proc. Natl. Acad. Sci. U. S. A.* 104, 1877–1882 (2007).
6. Vorholt, J. A., Vogel, C., Carlström, C. I. & Müller, D. B. Establishing Causality: Opportunities of Synthetic Communities for Plant Microbiome Research. *Cell Host Microbe* 22, 142–155 (2017).
7. Kehe, J. *et al.* Massively parallel screening of synthetic microbial communities. *Proc. Natl. Acad. Sci. U. S. A.* 116, 12804–12809 (2019).
8. Venturelli, O. S. *et al.* Deciphering microbial interactions in synthetic human gut microbiome communities. *Mol. Syst. Biol.* 14, e8157 (2018).
9. Johns, N. I., Blazejewski, T., Gomes, A. L. & Wang, H. H. Principles for designing synthetic microbial communities. *Curr. Opin. Microbiol.* 31, 146–153 (2016).
10. Grosskopf, T. & Soyer, O. S. Synthetic microbial communities. *Curr. Opin. Microbiol.* 18, 72–77 (2014).
11. Andreote, F. D. & Pereira E Silva, M. de C. Microbial communities associated with plants: learning from nature to apply it in agriculture. *Curr. Opin. Microbiol.* 37, 29–34 (2017).
12. Morrison, J. M. *et al.* Spatiotemporal analysis of microbial community dynamics during seasonal stratification events in a freshwater lake (Grand Lake, OK, USA). *PLoS One* 12, e0177488 (2017).

13. Feng, H. *et al.* Chemotaxis of Beneficial Rhizobacteria to Root Exudates: The First Step towards Root-Microbe Rhizosphere Interactions. *Int. J. Mol. Sci.* 22, (2021).
14. Dukovski, I. *et al.* A metabolic modeling platform for the computation of microbial ecosystems in time and space (COMETS). *Nat. Protoc.* 16, 5030–5082 (2021).
15. Passi, A. *et al.* Genome-Scale Metabolic Modeling Enables In-Depth Understanding of Big Data. *Metabolites* 12, (2021).
16. Wadhwa, N. & Berg, H. C. Bacterial motility: machinery and mechanisms. *Nat. Rev. Microbiol.* 20, 161–173 (2022).
17. Miyata, M. *et al.* Tree of motility - A proposed history of motility systems in the tree of life. *Genes Cells* 25, 6–21 (2020).
18. Karmakar, R. State of the art of bacterial chemotaxis. *J Basic Microbiol* 61, 366–379 (2021).
19. Lauffenburger, D., Aris, R. & Keller, K. H. Effects of random motility on growth of bacterial populations. *Microb. Ecol.* 7, 207–227 (1981).
20. Sampedro, I., Parales, R. E., Krell, T. & Hill, J. E. Pseudomonas chemotaxis. *FEMS Microbiol. Rev.* 39, 17–46 (2015).
21. Tindall, M. J., Maini, P. K., Porter, S. L. & Armitage, J. P. Overview of mathematical approaches used to model bacterial chemotaxis II: bacterial populations. *Bull. Math. Biol.* 70, 1570–1607 (2008).
22. Keller, E. F. & Segel, L. A. Initiation of slime mold aggregation viewed as an instability. *J Theor Biol* 26, 399–415 (1970).
23. Keller, E. F. & Segel, L. A. Model for chemotaxis. *J. Theor. Biol.* 30, 225–234 (1971).
24. Keller, E. F. & Segel, L. A. Traveling bands of chemotactic bacteria: a theoretical analysis. *J Theor Biol* 30, 235–248 (1971).
25. Burger, M., Di Francesco, M. & Dolak-Struss, Y. The Keller–Segel model for chemotaxis with prevention of overcrowding: Linear vs. nonlinear diffusion. *SIAM J. Math. Anal.* (2006).
26. Brenner, M. P., Levitov, L. S. & Budrene, E. O. Physical Mechanisms for Chemotactic Pattern Formation by Bacteria. *Biophys.* 74, 1677–1693 (1998).
27. Bhattacharjee, T., Amchin, D. B., Alert, R., Ott, J. A. & Datta, S. S. Chemotactic smoothing of collective migration. *Elife* 11, (2022).
28. Curk, T., Marenduzzo, D. & Dobnikar, J. Chemotactic Sensing towards Ambient and Secreted Attractant Drives Collective Behaviour of E. coli. *PLoS One* 8, e74878 (2013).
29. Cremer, J., Honda, T., Tang, Y., Wong-Ng Jerome and Vergassola, M. & Hwa, T. Chemotaxis as a navigation strategy to boost range expansion. *Nature* 575, 658–663 (2019).
30. Narla, A. V, Cremer, J. & Hwa, T. A traveling-wave solution for bacterial chemotaxis with growth. *Proc. Natl. Acad. Sci. U. S. A.* 118, (2021).
31. Orth, J. D., Thiele, I. & Palsson, B. Ø. What is flux balance analysis? *Nat. Biotechnol.* 28, 245–248 (2010).
32. Covert, M. W. *et al.* Metabolic modeling of microbial strains in silico. *Trends Biochem. Sci.* 26, 179–186 (2001).
33. Edwards, J. S., Covert, M. & Palsson, B. Metabolic modelling of microbes: the flux-balance approach. *Environ. Microbiol.* 4, 133–140 (2002).
34. Palsson, B. Ø. *Systems Biology: Properties of Reconstructed Networks*. (Cambridge University Press, 2006).

35. Price, N. D., Reed, J. L. & Palsson, B. Ø. Genome-scale models of microbial cells: evaluating the consequences of constraints. *Nat. Rev. Microbiol.* 2, 886–897 (2004).
36. Henson, M. A. & Hanly, T. J. Dynamic flux balance analysis for synthetic microbial communities. *IET Syst. Biol.* 8, 214–229 (2014).
37. Chen, J., Gomez, J. A., Höffner, K., Phalak Poonam and Barton, P. I. & Henson, M. A. Spatiotemporal modeling of microbial metabolism. *BMC Syst. Biol.* 10, 21 (2016).
38. Mahadevan, R., Edwards, J. S. & Doyle 3rd, F. J. Dynamic flux balance analysis of diauxic growth in Escherichia coli. *Biophys. J.* 83, 1331–1340 (2002).
39. Höffner, K., Harwood, S. M. & Barton, P. I. A reliable simulator for dynamic flux balance analysis. *Biotechnol. Bioeng.* 110, 792–802 (2013).
40. Harcombe, W. R. *et al.* Metabolic resource allocation in individual microbes determines ecosystem interactions and spatial dynamics. *Cell Rep.* 7, 1104–1115 (2014).
41. Pieterse, C. M. J. *et al.* Pseudomonas simiae WCS417: star track of a model beneficial rhizobacterium. *Plant Soil* 461, 245–263 (2021).