PhysiCell Mini-Course
Session 1: Getting Started

Paul Macklin, Ph.D.

Intelligent Systems Engineering
Indiana University

October 2, 2024

Session Goals

- Introduce key cell behaviors, phenotype, and signals
- Further explore response functions
- Introduce PhysiCell Studio
- Discuss typical modeling process
- First hands-on model (live modeling)

Key built-in cell behaviors

Built-in Reference Cell Behaviors

- PhysiCell has built-in reference models for key cell processes
  - Cycling (and division)
  - Apoptotic and necrotic death
  - Volume changes
  - Secretion and uptake
  - Cell-cell adhesion and "repulsion"
  - Migration
  - Type changes / differentiation
  - Phagocytosis
  - Fusion
  - Effector attack
  - Cell integrity

The key modeling work in PhysiCell is choosing which behaviors to modulate.

Reference behavior models: 1

- **Cycling**
  - Transition between cycle phases
  - Divide into two cells at end of last phase
  - Key parameter: cycle entry (rate of moving from phase 0 to phase 1)

- **A bit more detail:**
  - Each exit rate r_i is the transition rate to the next phase $r_{i,i+1}$
  - The mean duration T_i of a phase is related to the exit rate by $\mathrm{T_i} = \frac{1}{r_i}$

- **Several built-in cycle models are available:**
  - "Live" (single-phase)
  - Quiescent → Cycling
  - G0/G1 → S → G2/M
  - G0/G1 → S → G2 → M
  - $Ki67^-$ → $Ki67^+$
  - $Ki67^-$ → $Ki67^+$ (pre-mitotic) → $Ki67^+$ (post-mitotic)

Reference behavior models: 2

- **Apoptosis** (prototypical non-inflammatory death)
  - Gradually shrink, get removed
  - Key parameter: apoptotic death rate (rate of starting apoptosis)

- **Necrosis** (prototypical inflammatory death)
  - First swell, burst, then shrink
  - Key parameter: necrotic death rate

Reference behavior models: 3

- **motility**
  - biased random walk:
    - Move some time along a bias direction, then resample, move again
  - Key parameters: migration speed, persistence time, bias (directedness)

- **chemotaxis (basic)**
  - bias direction is along or against a chemical gradient
  - Key parameter: which chemical gradient

- **chemotaxis (advanced)**
  - bias direction is a weighted sum of chemical gradients
  - Key parameter: weights (-1 to 1) of each chemical gradient

Reference behavior models: 4

- **cell-cell adhesion (basic)**
  - Use potential functions for an attractive force
  - Key parameters: adhesive affinity, adhesion strength, max (relative) adhesion distance

- **cell-cell adhesion (elastic / advanced)**
  - form and break spring links to contacting cells
  - Key parameters: adhesive affinity, elastic constant, attachment rate, detachment rate, maximum number of adhesions

- **resistance to deformation and overlap**
  - Use potential function as a "repulsive" force
  - Key parameter: repulsive strength

Reference behavior models: 5

- **transformation (type change)**
  - Transition from type *i* to type *j*
    - Differentiation, Transdifferentiation, mutation, …
  - Key parameters: transition rates

- **fusion**
  - cells *i* and *j* combine volumes, re-center position
  - Key parameter: fusion rates (type *i* to type *j*)

- **phagocytosis**
  - Cell *i* consumes cell *j* (and acquires its volume)
  - Key parameters: rate of phagocytosing dead cells (separate rates for apoptotic, necrotic, and other dead cells), rates of phagocytosing live cell types

Reference behavior models: 6

- **effector attack**
  - Cell *i* attacks (damages) cell *j*
    - rate of initiating attack is a function of attack rate of *i* on *j* and immunogenicity of *j* to *i*
    - the attack increases damage of *j*
    - cells form adhesion during attack.
    - Attack has stochastic duration
    - requires an additional hypothesis to cause death in cell *j*
  - Key parameters: attack rates, immunogenicities, attack damage rate, attack duration

- **secretion**
  - cells can secrete, uptake (consume), and export diffusible substrates
  - Key parameter: secretion rates, secretion targets, uptake rates, export rates

$$\frac{\partial \rho}{\partial t} = D\nabla^2\rho - \lambda\rho + \sum_{\text{cells } i}\left(\delta(\boldsymbol{x}-\boldsymbol{x}_i)V_i\left[\overbrace{S_i\left(\rho_i^* - \rho\right)}^{\text{secretion}} - \overbrace{U_i\rho}^{\text{uptake}}\right] + \delta(\boldsymbol{x}-\boldsymbol{x}_i)\overbrace{E_i}^{\text{export}}\right)$$

Reference behavior models: 7

- **cell integrity**
  - Cell *i* can undergo and repair (generic) damage
    - damage rate (e.g., from a drug or alpha particles)
      - » Can use to increases cell death, increase mutations, block cell cycle, …
    - damage repair rate
  - Key parameters: damage rate, damage repair rate

$$\frac{d[\text{Damage}]}{dt} = [\text{damage rate}] + [\text{damage from effector attack}] - [\text{damage repair rate}][\text{Damage}]$$

- **reference behavior models in development**
  - Polarized cell adhesion
  - ECM interactions
  - Asymmetric cell division
  - Spontaneous variation
  - (and also built-in lineage tracking)

Cell Phenotype

Cell Phenotype

- The cell's key parameters are organized according to these processes:
  - Cycle
  - Death
  - Volume (and Geometry)
  - Mechanics
  - Motility
  - Secretion (and Uptake)
  - Interactions
  - Transformations
  - Integrity

- Think of it as a vector of phenotypic properties $\mathbf{p}(t)$ that can vary in time.
- Each cell has a "base" phenotype $\mathbf{p}_0$ (inherited from its cell definition)

Key built-in cell signals

Signal Dictionary

- Based on the cell types and diffusible substrates in a simulation, we can auto-generate dictionaries of available signals
- With standardized access, it's much easier to write cell rules
- This allows for a controlled vocabulary (an ontology)

Signal name	Biophysical meaning
{substrate X}	extracellular concentration of chemical factor X
intracellular {substrate X}	intracellular concentration of chemical factor X
{substrate X} gradient	slope of the extracellular concentration field of factor X
pressure	mechanical pressure (from other cells in close proximity)
volume	the cell's current total volume
contact with {cell type X}	number of cells of type X that are in physical contact
contact with live cell	number of live cells that are in physical contact
contact with dead cell	number of dead cells that are in physical contact
contact with basement membrane	1 if in contact with basement membrane. 0 otherwise.
damage	amount of damage (of any type)
dead	1 if the cell is dead (or dying). 0 otherwise.
total attack time	total amount of time the cell has been attacked.
time	current simulation time
custom:{X}	use a custom variable or symbol *X* to drive cell behavior

For current "dictionaries"

- Since language is evolving, you can easily get a current dictionary.

1. Build a model, and briefly run it.

2. Look for dictionaries.txt in your output directory

3. See a list of valid signals and behavioral parameters
   - These lists depend upon the names of diffusing factors & cell types in your simulation.

Response functions

Using a response function

- If signal *S* increases / decreases behavior *B*
  - Vary behavioral parameter *p* with base value p_0 and maximal response value p_{M}

$$p(s) = p_0 + (p_{\mathrm{M}} - \mathrm{p}_0)R(s) = \big(1 - R(s)\big) \cdot p_0 + R(s) \cdot p_M$$

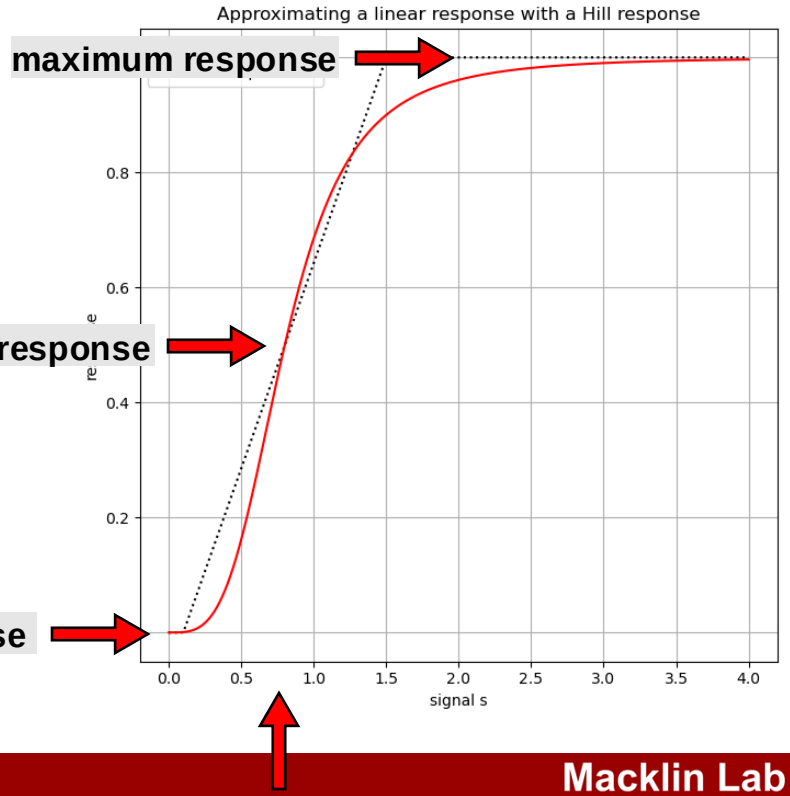
  - We generally use **Hill response functions:**

$$R(s) = \frac{s^h}{s_{\mathrm{half}}^h + s^h} = \frac{\left(\frac{s}{s_{\mathrm{half}}}\right)^h}{1 + \left(\frac{s}{s_{\mathrm{half}}}\right)^h} \quad \text{if } s \geq 0, \qquad \text{and } H(s) = 0 \text{ if } s < 0.$$

Hill response functions

- A widespread sigmoidal response curve in PKPD and systems biology
  - Varies from 0 (at signal=0) to 1 (as signal → infinity)
  - Completely characterized by:
    - half-maximum: Input value where curve reaches half of max effect
    - Hill power: How steeply it approaches 1

$$H(s; s_{\text{half}}, \text{h}) = \frac{s^h}{s_{\text{half}}^h + s^h} = \frac{\left(\frac{s}{s_{\text{half}}}\right)^h}{1 + \left(\frac{s}{s_{\text{half}}}\right)^h} \text{ if } s \geq 0, \quad \text{and } H(s) = 0 \text{ if } s < 0.$$



PhysiCell Studio

PhysiCell Studio Desktop

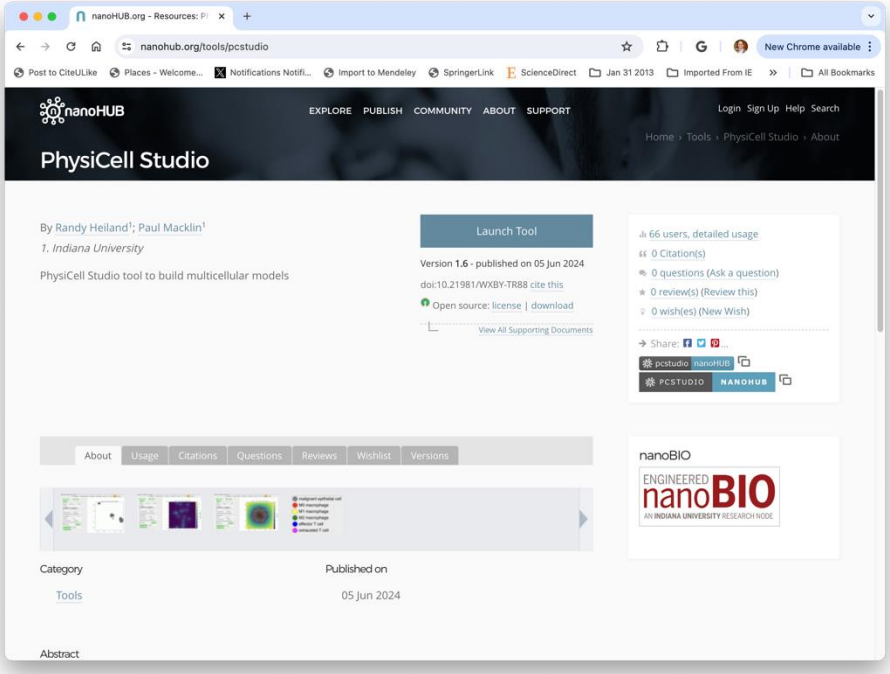
- Preferred method – less latency
  - Does not require a compiler
  - Does not require C++ or coding experience

- Make sure you have installed Python
  - Use anaconda if you're unsure: https://www.anaconda.com/download/success

- Follow the PhysiCell Studio Desktop installation instructions here:
  - https://github.com/physicell-training/institut-curie-2024/blob/main/PhysiCell-Studio-Setup.md

PhysiCell Studio Cloud

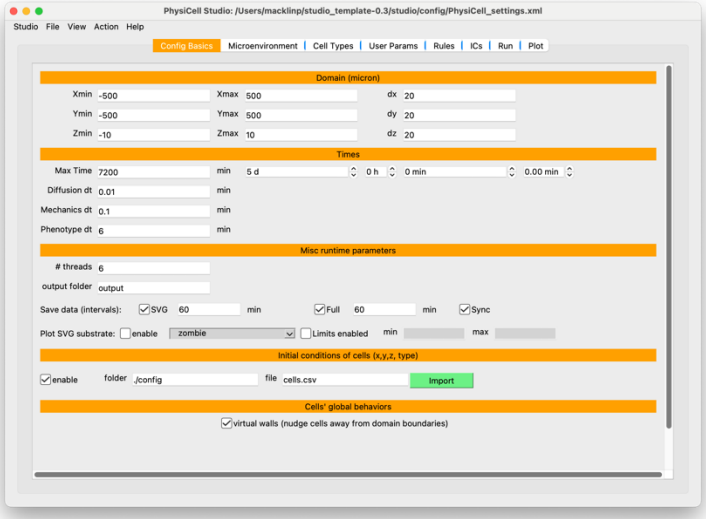
- Alternate method (just in case)
  - Fully runs in a web browser
  - No installation required
  - But … more latency ...
  - May lack some features of the Desktop edition
- Login to nanohub.org
- Go to:
  - https://nanohub.org/tools/pcstudio
- Click the blue "run tool" button



PhysiCell Studio: Overview

A graphical user interface (GUI) application to make it easier to build and explore PhysiCell models

- **Config basics:** Domain size, simulation duration, output
- **Microenvironment:** Diffusing substrates, boundary conditions
- **Cell types:** Define cell types and their base phenotypes
- **User params:** Model-specific parameters
- **Rules:** Hypothesis-based cell behaviors
- **ICs:** Initial cell positions
- **Run:** Use this to start executing the model
- **Plot:** Plot cells and diffusible substrates

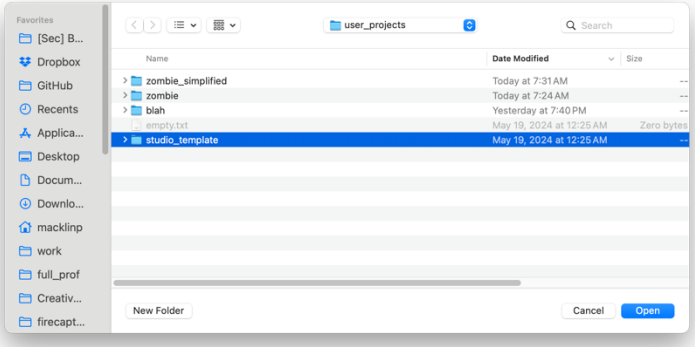


Cloud-based backup: https://nanohub.org/tools/pcstudio

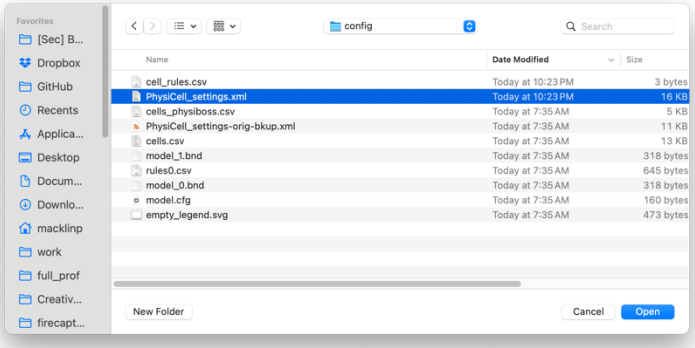
https://nanohub.org/tools/pcstudio

Loading a (blank) project

- Let's get the (blank) template project
  - File → Load user project
  - Choose **studio_template**
  - This loads all the files into the right place



- Next let's load it into the studio:
  - File → open
  - Browse to **config**
  - Load **PhysiCell_settings.xml**



Modeling Steps

Key modeling steps

1. Plan the modeling problem.
   - What are the important things we want to learn?
   - What cell types are important?
     - What are their key behaviors?
     - What diffusible factors drive their behaviors?
     - Any other important interactions?
   - Can we estimate parameters (at least to order of magnitude)?
   - Can we build the model sequentially? (Add components one at a time?)
2. Set up diffusing factors.
3. Set up cell types.
4. Add rules.
5. Simulate and assess.
6. Iterate and improve.

Sample Problem:

Villagers and Zombies

Plan the problem (1)

- What are the important things we want to learn?
  - How do the behaviors of zombies and villagers affect their population dynamics?

- What cell types are important?
  - Villagers
  - Zombies

- What are their key behaviors?
  - Villagers:
    - Aggregate
    - Reproduce
    - Flee from Zombies
    - Either die or transform into zombies after attack
  - Zombies:
    - Chase villagers
    - Attack villagers
    - Move away from other Zombies (optional)

Plan the problem (2)

- What diffusible factors drive their behaviors?
  - Let's use a quorum factor for villager aggregation
  - Let's use a "zombie" factor to help locate zombies

- Any other important interactions?
  - We might want to let villagers counter-attack. Stretch goal

- Can we estimate parameters?
  - Not a huge focus for this problem.
  - We'll choose decay and diffusion parameters based on diffusion length scale

Plan the problem (3)

- Can we build the model sequentially?
  - First, let's build a model of villagers:
    - Aggregation with a quorum factor
    - Proliferation based on contact
    - **Test:** aggregation and proliferation
  - Next, let's add zombies
    - Zombies chase villagers
    - Villagers flee from zombies (need to add diffusing zombie factor)
    - **Test:** chasing and avoidance
    - Zombies attack villagers
    - Damage causes villagers to die
    - **Test:** Villager death
    - Damage causes villagers to transform to zombies
    - **Test:** Villagers
    - Add Zombies avoid Zombies
    - **Test:** Zombies get spaced out more evenly.

Diffusion length scale

- For any diffusing factor we should choose:
  - Diffusion coefficient (D) and decay rate (λ)

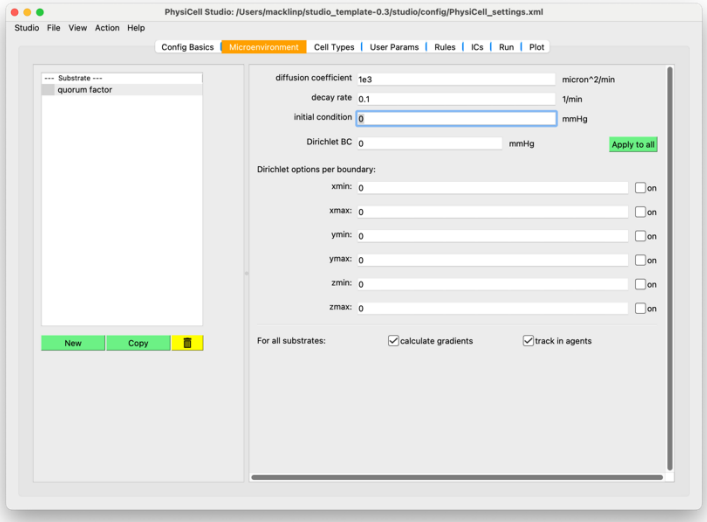
- We use the **diffusion length scale** from physics / applied mathematics:
  - Penetration into a tissue is competition between effects:
    - Diffusion (D) increases spread
    - Uptake (U) and decay (λ) tend to halt spread

$$L = \sqrt{\frac{D}{U + \lambda}}$$

- For our signaling factors:
  - Suppose they linger for ~10 min. That gives $\lambda \sim 0.1$ min^{-1}
  - If we want $L \sim 100\ \mu\text{m}$, then: $D \sim 10^3\ \mu\text{m}^2$/min

Villager model: Diffusing Factor

- Go to **microenvironment tab**
- Double-click on **substrate**
- Rename it to **quorum**
- Set the **diffusion coefficient** to 1e3

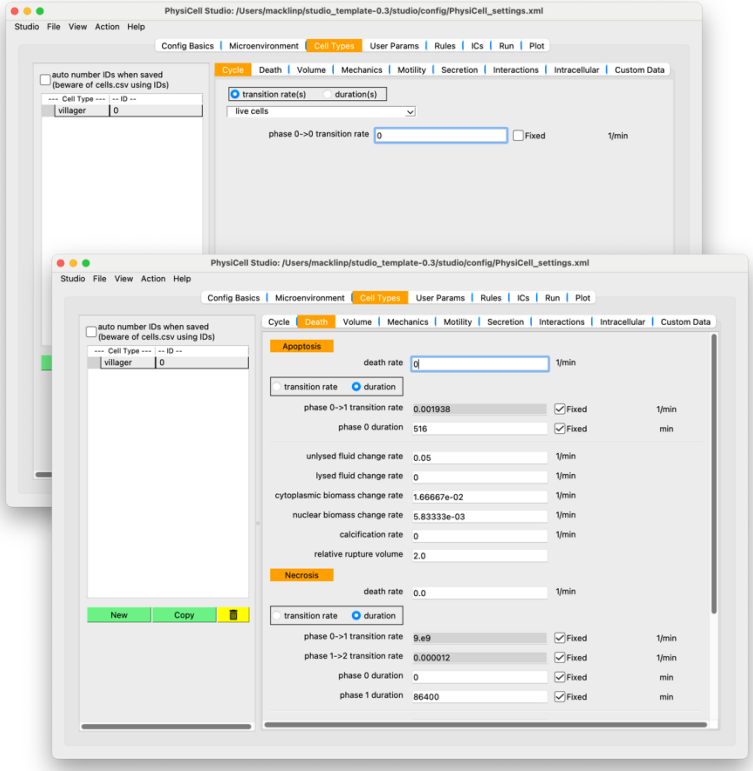


Villager model: Villagers (1)

- Go to **cell types tab**
- Double-click on **default**
- Rename it to **villager**

- Let's turn off (baseline) cycling and death
  - In the **Cycle** tab, choose the **live cells** model
  - Choose the **transition rate** description
  - Set the transition rate to 0

  - In the **death** tab, set the **apoptosis rate** to 0

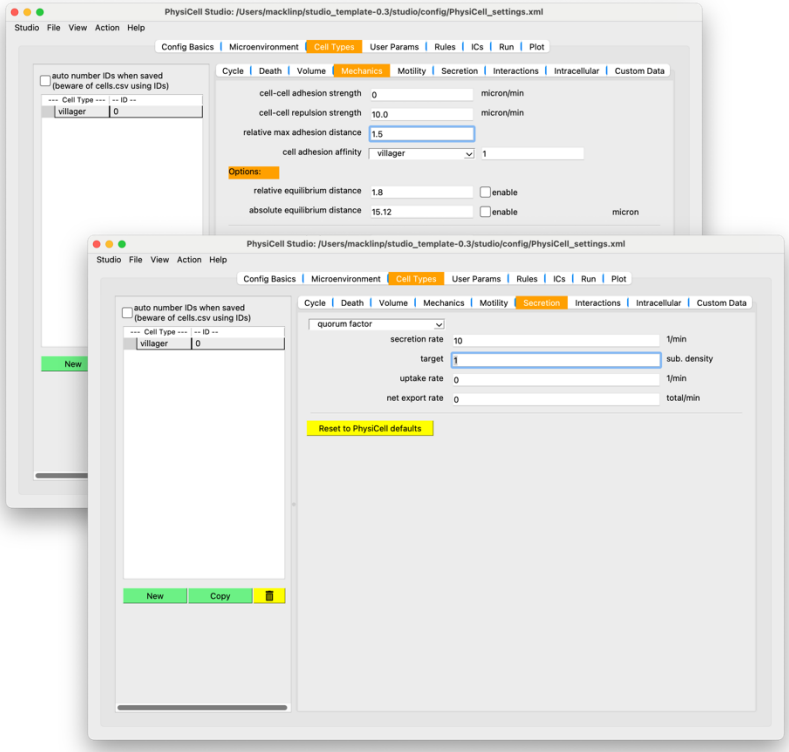


Villager model: Villagers (2)

- Let's turn off (baseline) adhesion
  - Go to the **Mechanics** tab
  - Set the **cell-cell adhesion strength** to 0

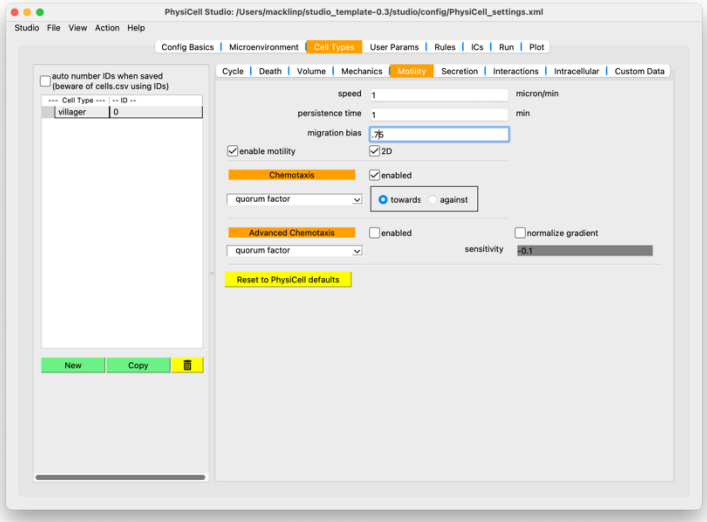
- And let's set the cell-cell max adhesion distance to 1.5 cell radii
  - Set **relative max adhesion distance** to 1.5

- Let's turn on secretion of quorum factor
  - Go to **Secretion** tab
  - Choose **quorum factor** in the drop-down
  - Set **secretion rate** to 10 (strong forcing)
  - Make sure **target** is 1



Villager model: Villagers (3)

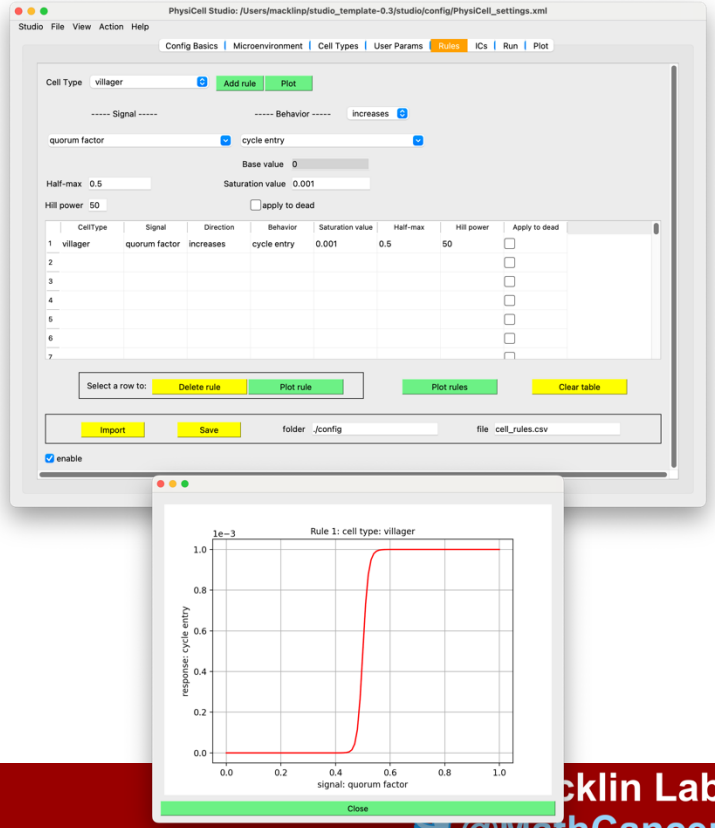
- Let's make sure chemotaxis is ready
  - Go to the **motility** tab
    - Set the **speed** to 1 (micron/min)
    - Keep **persistence time** at 1 min
    - Set **migration bias** a bit higher to 0.75
    - Make sure the **enable motility**
  - Then, make sure that we use **chemotaxis** for that motility
    - Choose **enabled** under motility
    - chose the **quorum factor** and choose **towards**



Villager model: Villager rules

- Let's make these villagers reproduce
  - Villagers need to be in contact with other villagers to reproduce
    - Rule: contact with villager increases cycle entry
    - We'll use a max rate of 0.001
    - We'll use a half-max of 1 and a steep hill power

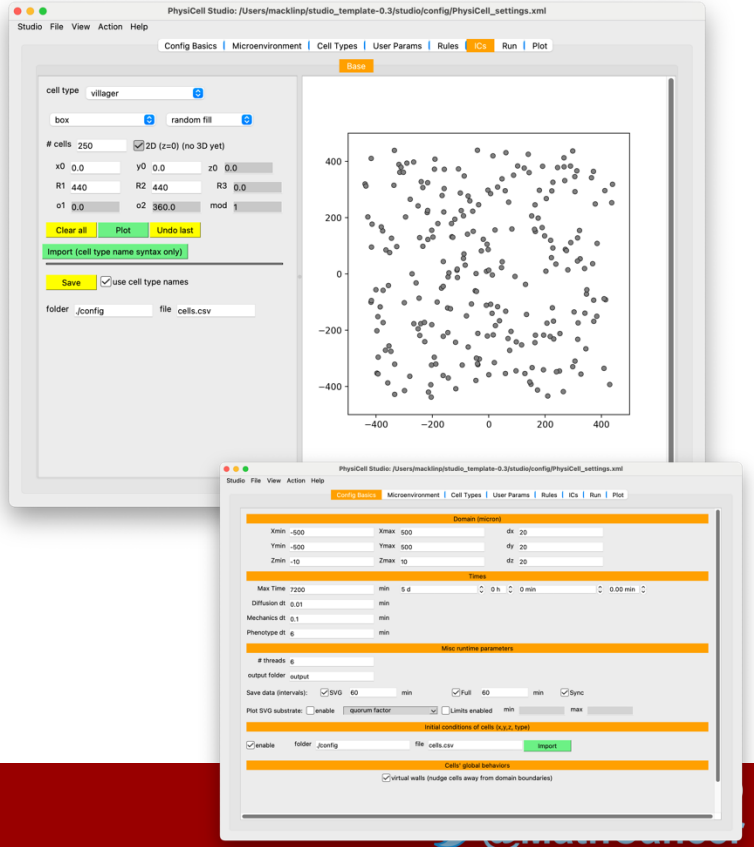
- Go to **rules** tab
  - Add rule:
    - select **villager** as type
    - select **quorum factor** as signal
    - choose **cycle entry** as behavior, and **increases** as response
    - Choose **saturation value** is 0.001
    - Choose **half-max** of 0.5
    - Choose **Hill power** of 50
    - View the response function
    - Click **add rule**
    - Save the rules



Villager model: Initial conditions

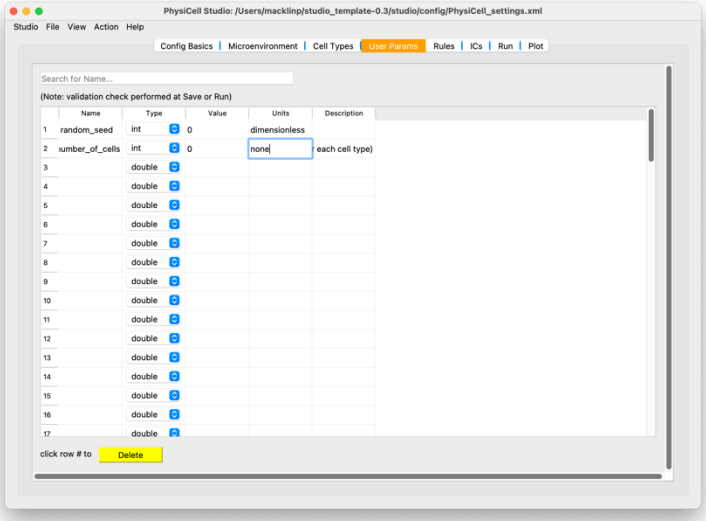
- Go to the **ICs** tab
  - Choose **villager** from the drop-down
  - choose **box** centered at (0,0), and radii **440**
  - Place **250** cells wtihj the **plot** button
  - Click **save**

- Make sure this config is used
  - Go to **config basics**
  - Click **enable** under "initial conditions of cells (x,y,z, type)



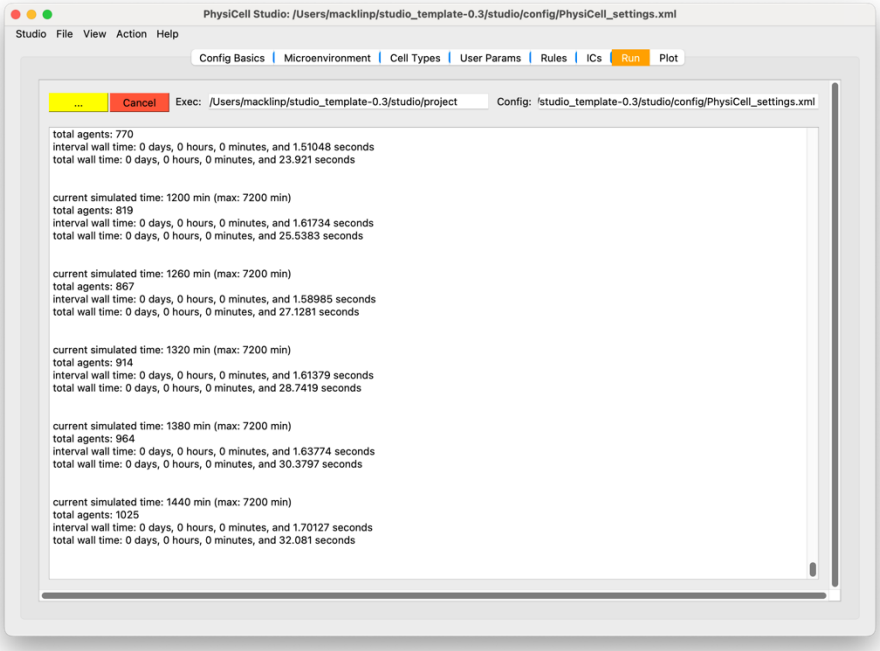
Villager model: Initial conditions (2)

- Let's disable the default random cell seeding
  - Go to **User Params**
  - Set **number_of_cells** to 0



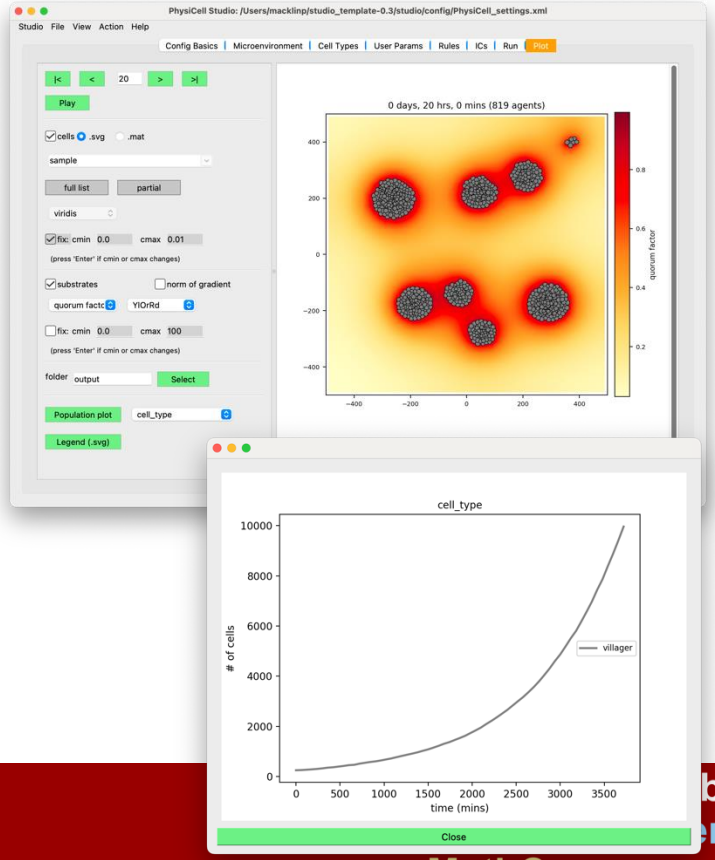
Villager model: run!

- Go to the **Run** tab
- Click **run simulation**



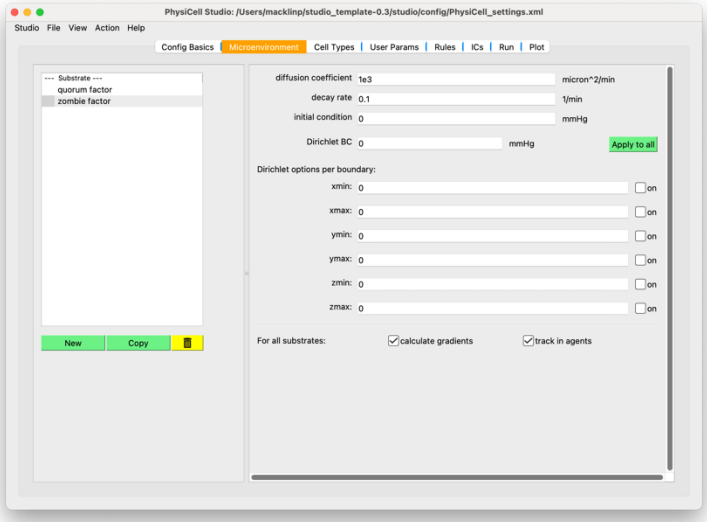
Villager model: Visualize

- Go to the **Plots** tab
- Make sure **cells** is checked
  - Use the **svg** visualization for no
- Plot the substrates
  - Choose **quorum factor**
  - Let it auto-range
- Click **population plot** to see growth curves
- What we should see:
  - Aggregates of villagers
  - Highest *q* of quorum factor around villagers



Zombie model: Diffusing Factor

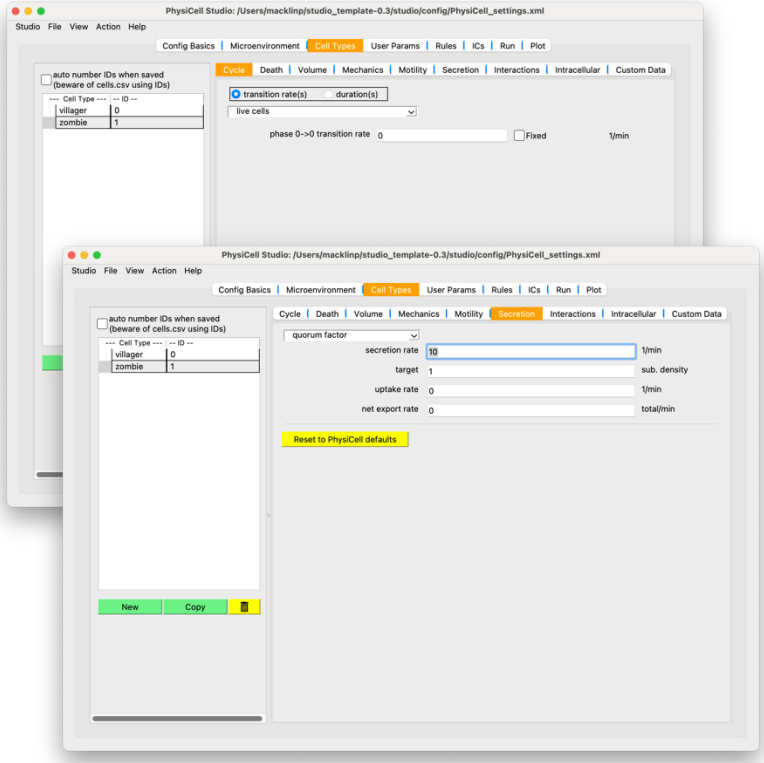
- Go to **microenvironment tab**
- Click on **substrate**
- Click on **copy**
- Rename it to **zombie factor**



Zombie model: Zombies (1)

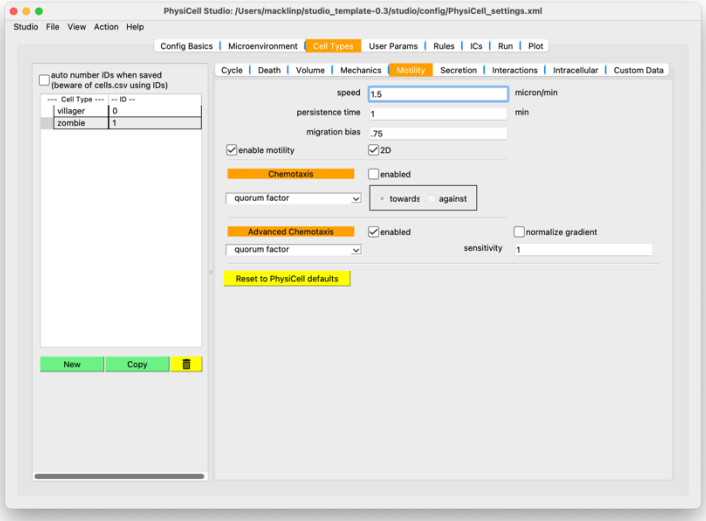
- Go to **cell types tab**
  - Click on **villager**
  - Copy it, and rename to **zombie**

- Let's make sure secretion is right
  - Go to the **secretion** tab
  - Choose **quorum factor** from the drop-down
    - Set its **secretion rate** to 0.0
  - Choose **zombie factor** from the drop-down
    - Set its **secretion rate** to 10.0



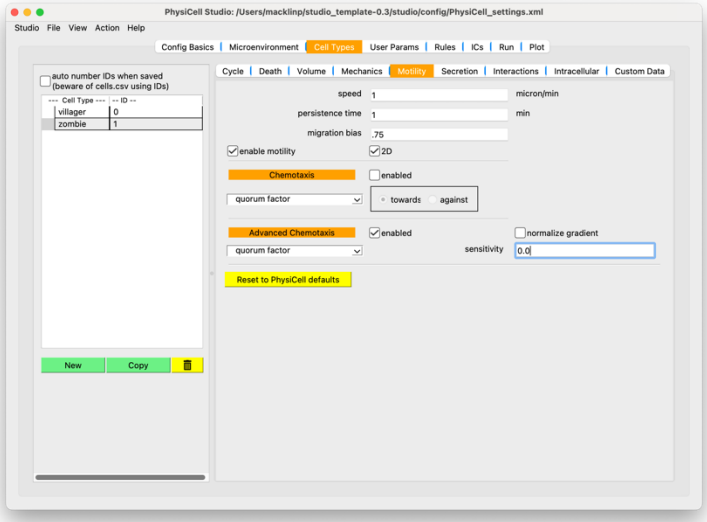
Villager model: Zombies (2)

- Let's adjust chemotaxis
  - go to the **motility** tab
  - Set **migration speed** to 1.5
  - uncheck **enabled** for **chemotaxis**
  - check **enabled** for **advanced chemotaxis**
  - choose **quorum factor** from the drop-down
    - Set **sensitivity** to **1.0**
  - choose **zombie factor** from the drop-down
    - Set **sensitivity** to -**1.0**



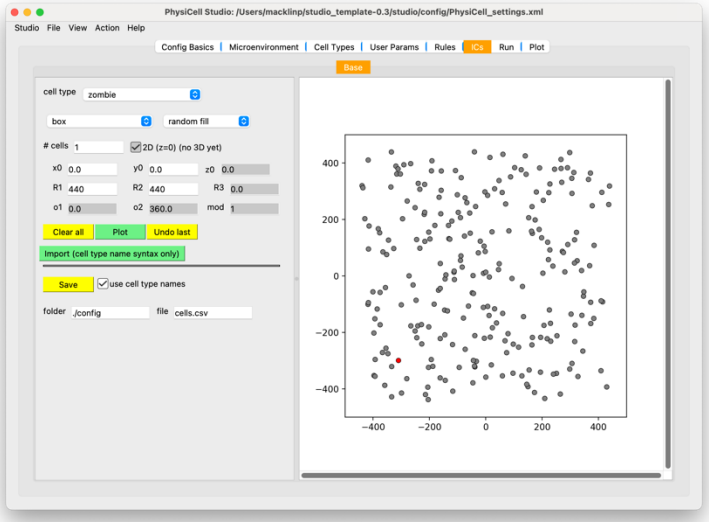
Villager model: Adjust Villagers

- Let's adjust chemotaxis on villagers
- Select **villagers** on the far-left drop-down
  - go to the **motility** tab
  - uncheck **enabled** for **chemotaxis**
  - check **enabled** for **advanced chemotaxis**
  - choose **quorum factor** from the drop-down
    - Set **sensitivity** to **1.0**
  - choose **zombie factor** from the drop-down
    - Set **sensitivity** to -**10.0**



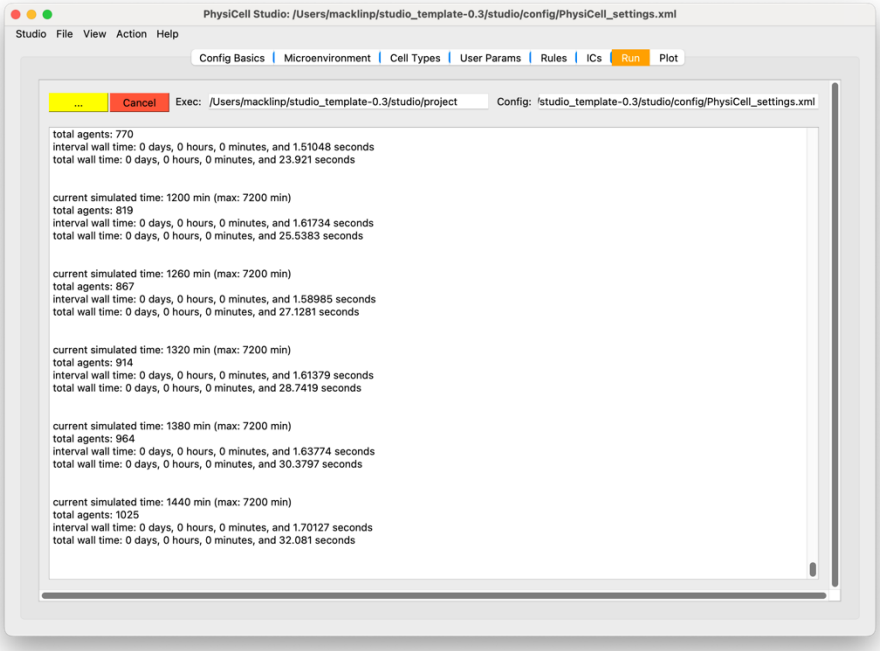
Zombie model: Initial conditions

- Go to the **ICs** tab
  - Choose **zombie** from the drop-down
  - choose **box** centered at (0,0), and radii **440**
  - Place **1** cells wtihj the **plot** button
  - Click **save**



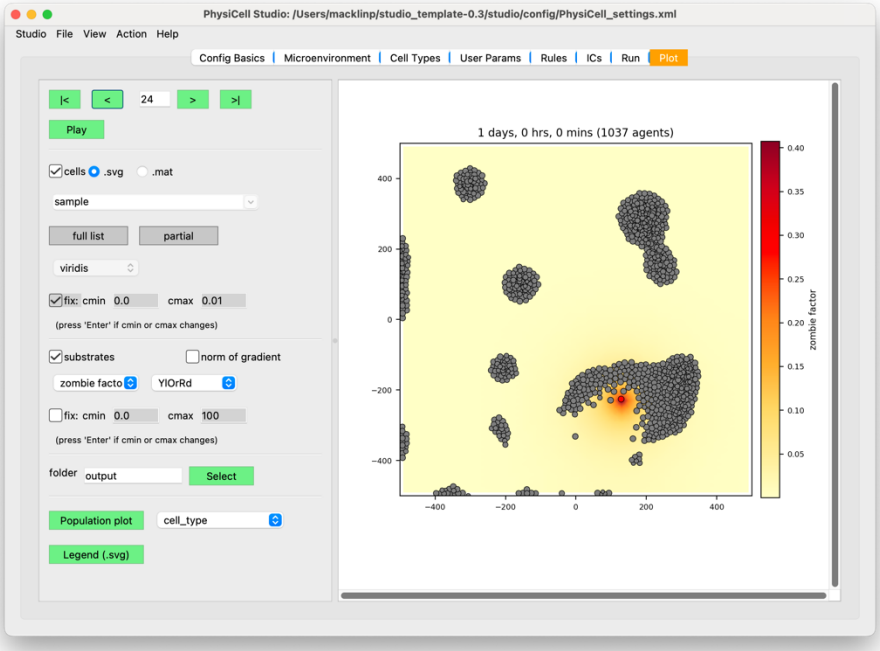
Villager model: run!

- Go to the **Run** tab
- Click **run simulation**



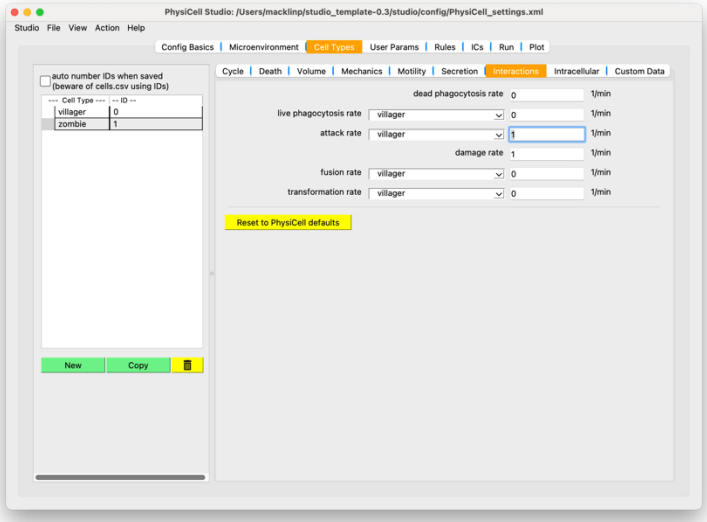
Villager model: Visualize

- Go to the **Plots** tab
- Make sure **cells** is checked
  - Use the **svg** visualization for no
- Plot the substrates
  - Choose **quorum factor**
  - Let it auto-range
- Click **population plot** to see growth curves
- What we should see:
  - Aggregates of villagers
  - Concentration of zombie factor aroudn the zombie
  - Zombie chases villagers, whoh steer clear



Zombie model: Adjust Zombies (1)

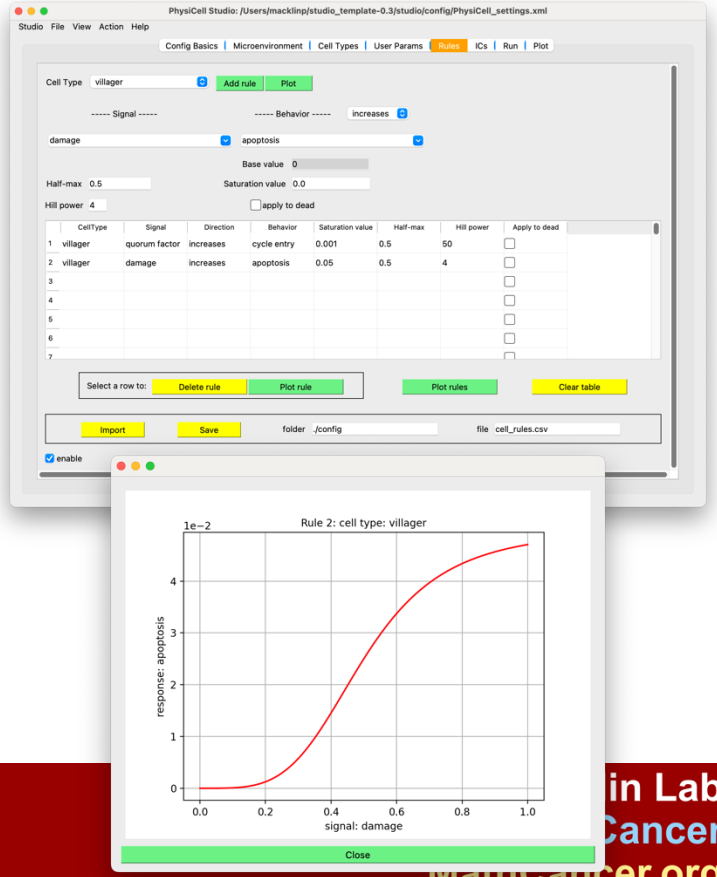
- Let's add the effector attack
  - Go to **cell types tab**
  - Click on **zombie**
  - Go to the **interactions** tab
  - Choose **villager** and set the **attack rate** to 1



Zombie model: Add villager rule (1)

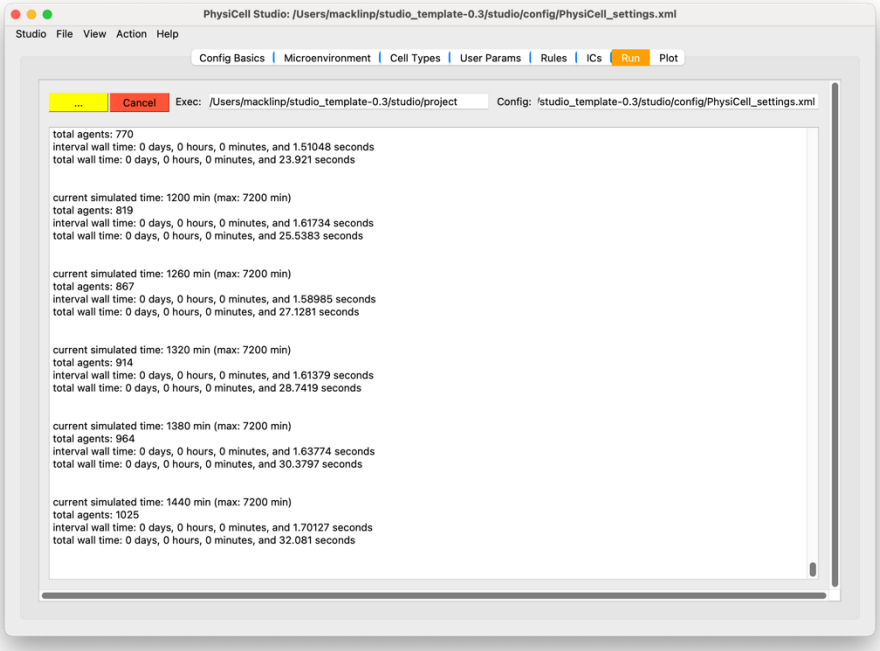
- Let's make these villagers die from damage

- Go to **rules** tab
  - Add rule:
    - select **villager** as type
    - select **damage** as signal
    - choose **apoptosis** as behavior, and **increases** as response
    - Choose **saturation value** is 0.05
    - Choose **half-max** of 0.5
    - Choose **Hill power** of 4
    - View the response function
    - Click **add rule**
    - Save the rules



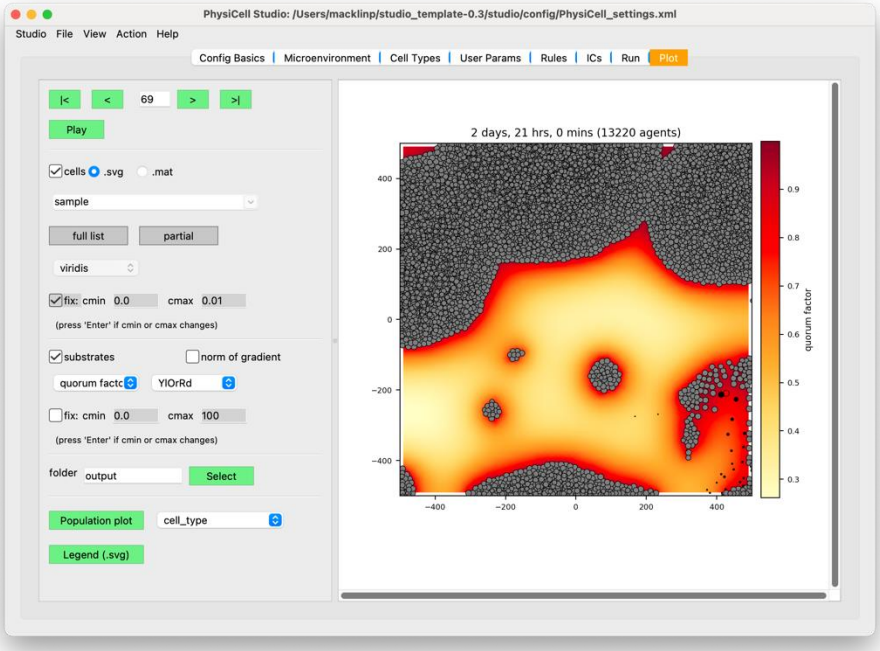
Villager model: run!

- Go to the **Run** tab
- Click **run simulation**



Villager model: Visualize

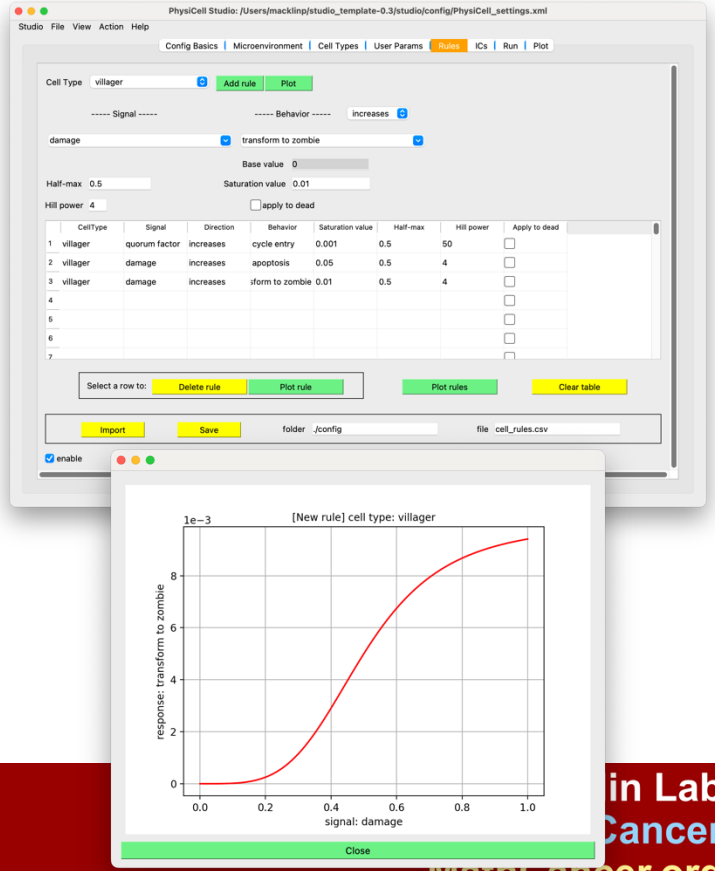
- Go to the **Plots** tab
- Make sure **cells** is checked
  - Use the **svg** visualization for no
- Plot the substrates
  - Choose **quorum factor**
  - Let it auto-range
- Click **population plot** to see growth curves
- What we should see:
  - Aggregates of villagers
  - Concentration of zombie factor around the zombie
  - Zombie chases villagers, who steer clear
  - Some villagers are killed.



Zombie model: Add villager rule (2)

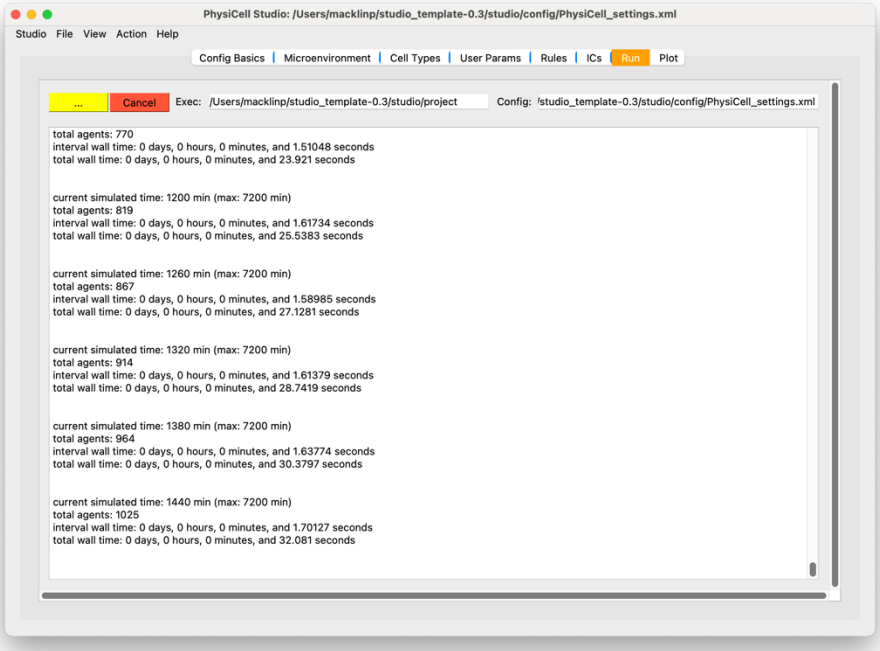
- Let's make sure damage kills makes some villagers turn to zombies

- Go to **rules** tab
  - Add rule:
    - select **villager** as type
    - select **damage** as signal
    - choose **transform to zombie** as behavior, and **increases** as response
    - Choose **saturation value** is 0.01
    - Choose **half-max** of 0.5
    - Choose **Hill power** of 4
    - View the response function
    - Click **add rule**
    - Save the rules



Villager model: run!

- Go to the **Run** tab
- Click **run simulation**



Villager model: Visualize

- Go to the **Plots** tab
- Make sure **cells** is checked
  - Use the **svg** visualization for no
- Plot the substrates
  - Choose **quorum factor**
  - Let it auto-range
- Click **population plot** to see growth curves
- What we should see:
  - Aggregates of villagers
  - Concentration of zombie factor around the zombie
  - Zombie chases villagers, who steer clear
  - Some villagers are killed.
  - Many villagers turn to zombies
  - Eventually all villagers wiped out
  - Zombies evenly space themselves