Small Potatoes: Microsimulation with Vivarium

Alec Deason

UNIVERSITY of WASHINGTON

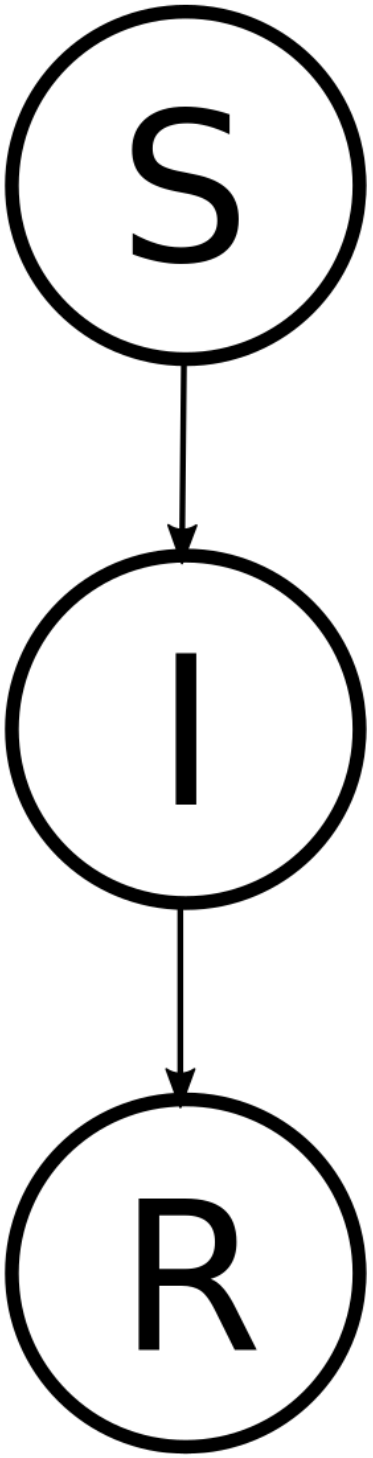
Institute for Health Metrics and Evaluation

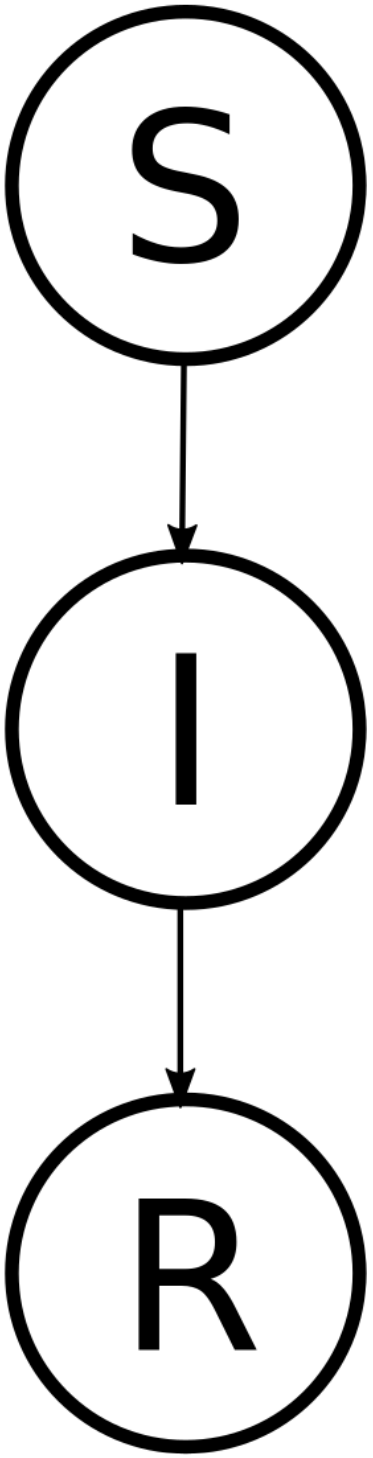
Why simulation?


The HUMAN CANNON Ball
"LOYAL"
FIRED FROM THE CANNON.

$$d = \frac{v^2 \sin(2\theta)}{g}$$

But, why microsimulation?

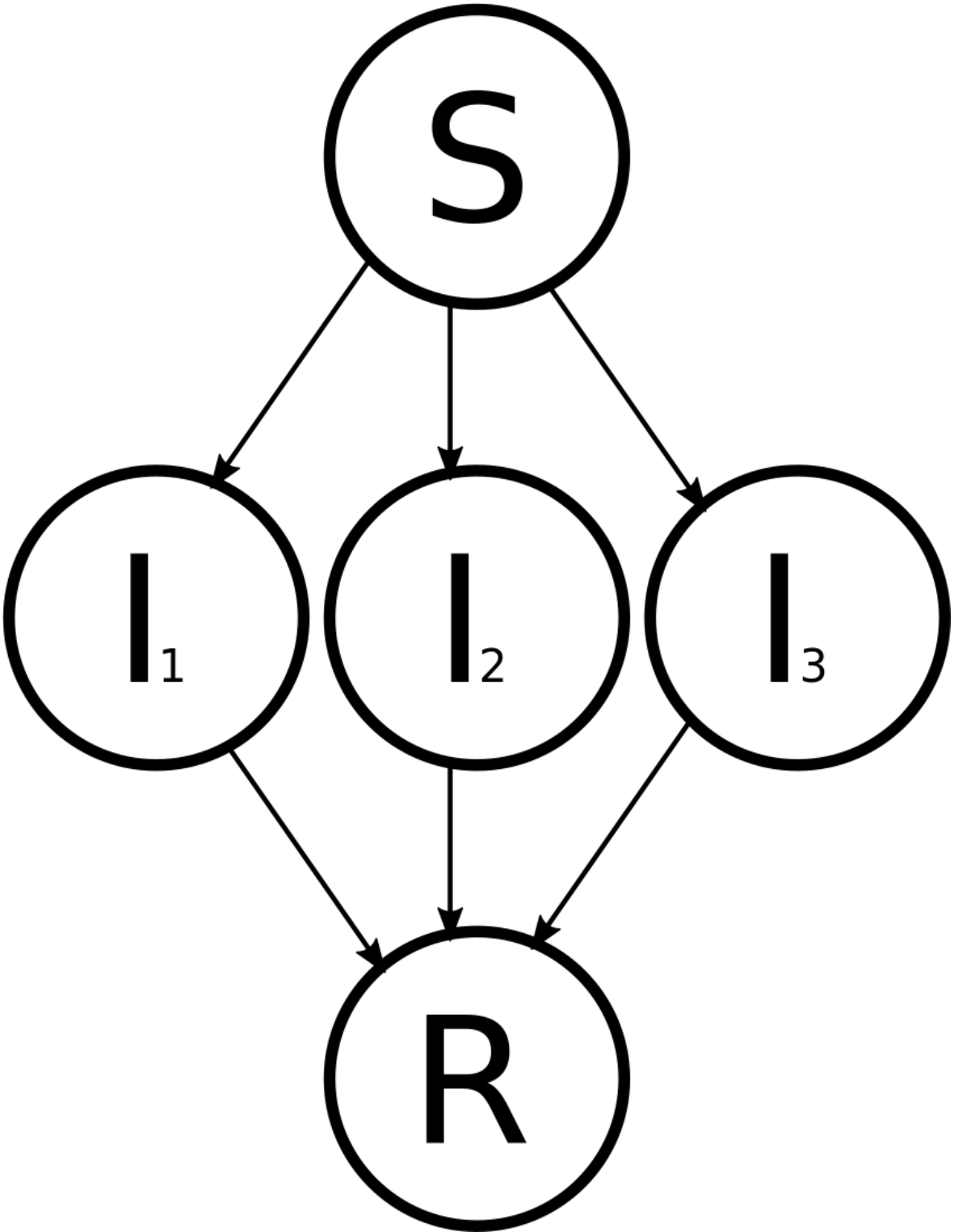

S
I
R



$$\frac{dS}{dt} = -\frac{\beta I S}{N}$$

$$\frac{dI}{dt} = \frac{\beta I S}{N} - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$


S
I1
I2
I3
R

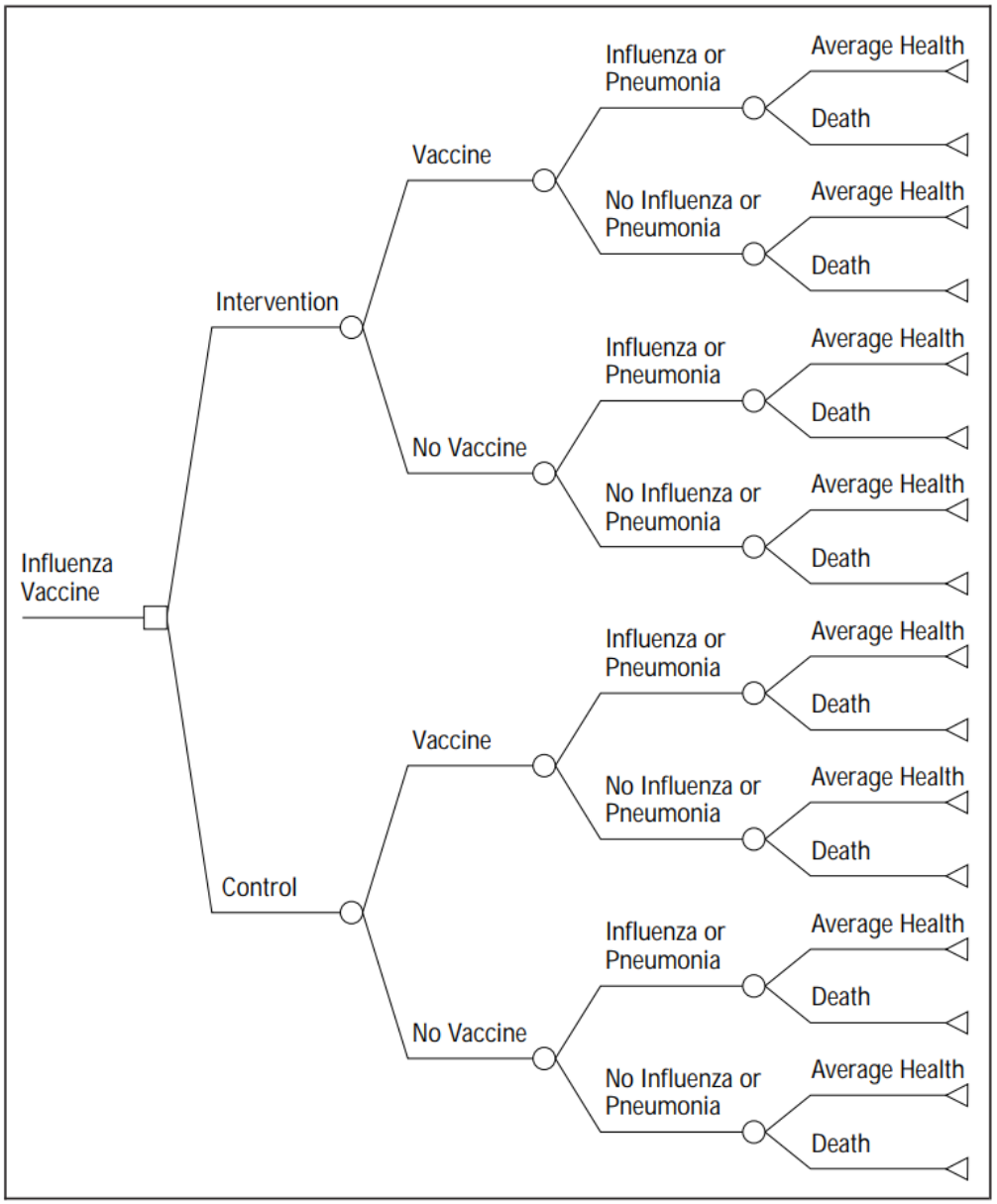
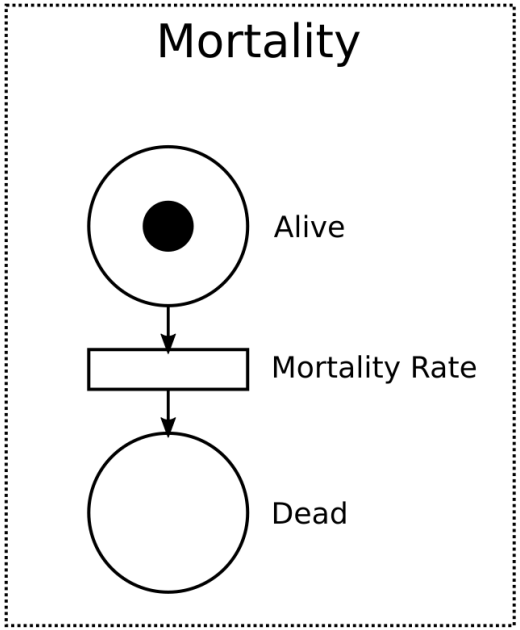
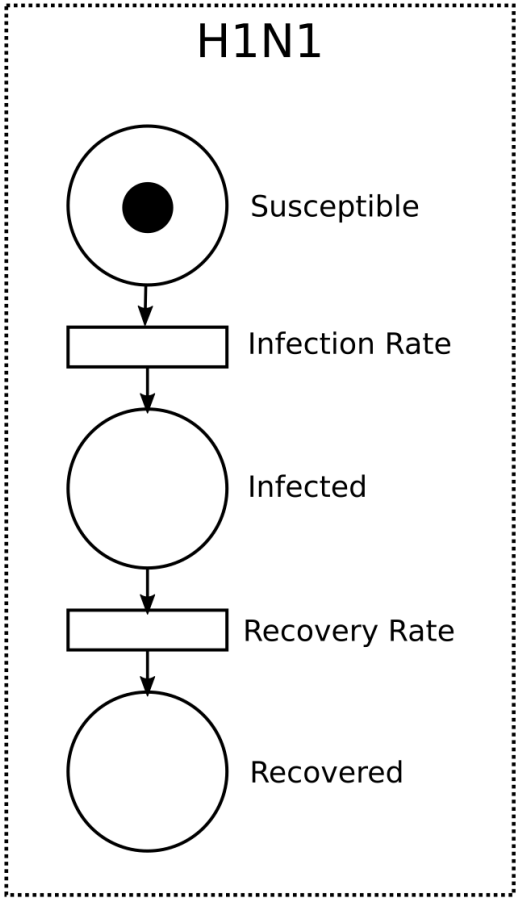
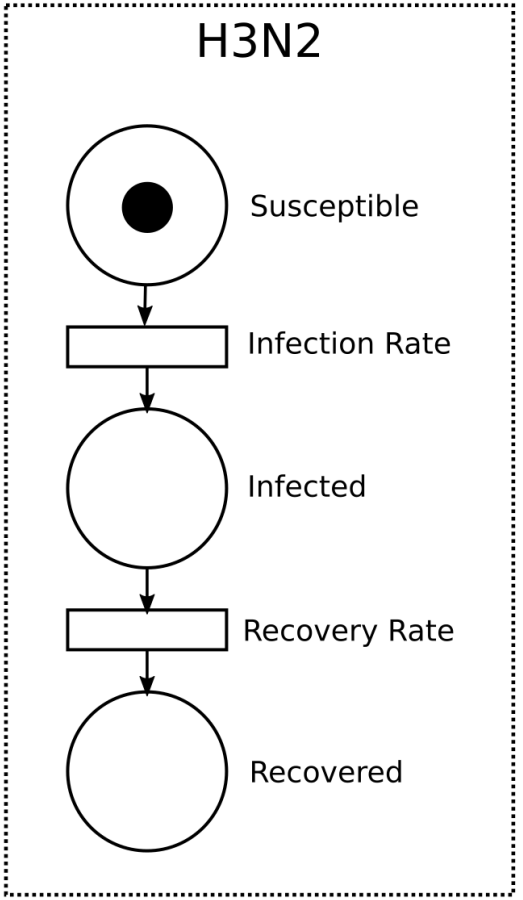


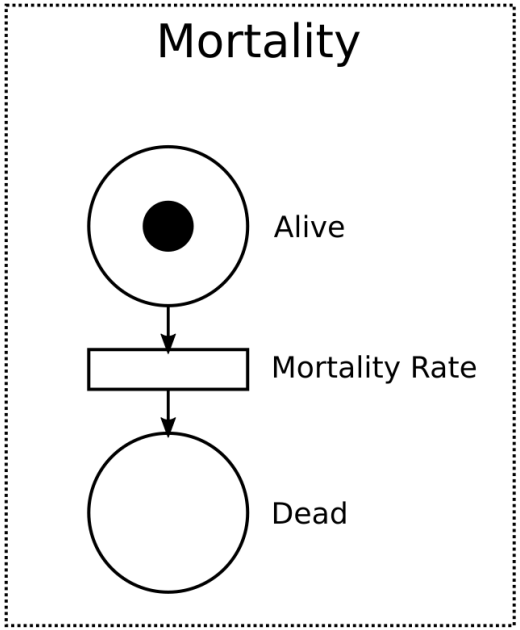
Figure 3. *Decision tree for an intervention to promote only the influenza vaccine.*

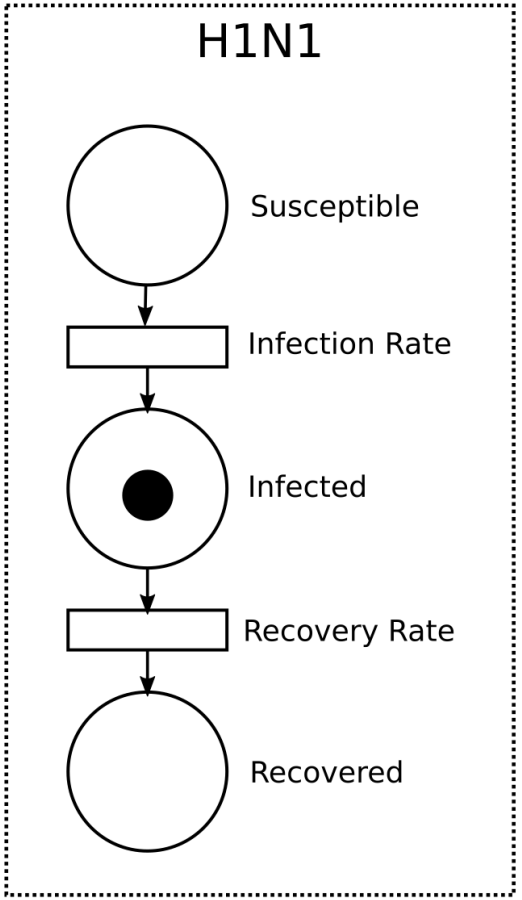
Weaver, et al. Annals of Internal Medicine, 2001

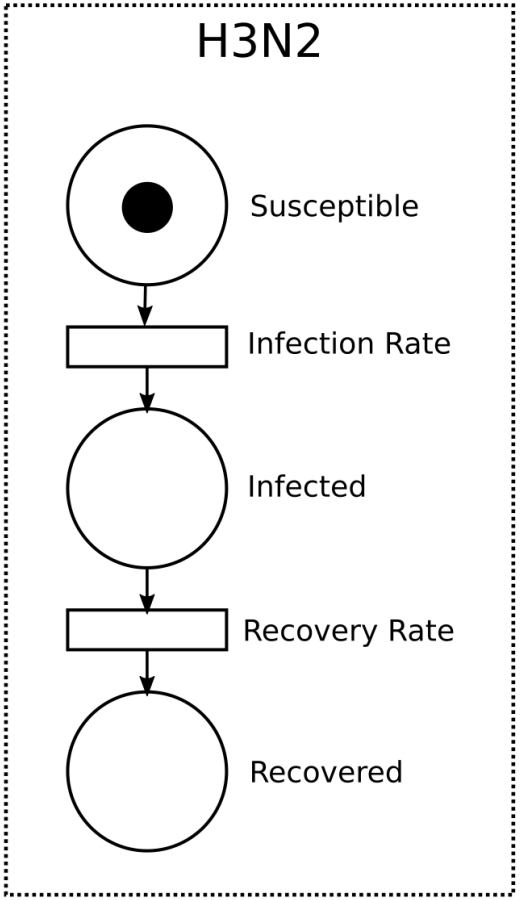

Mortality
Alive
Mortality Rate
Dead

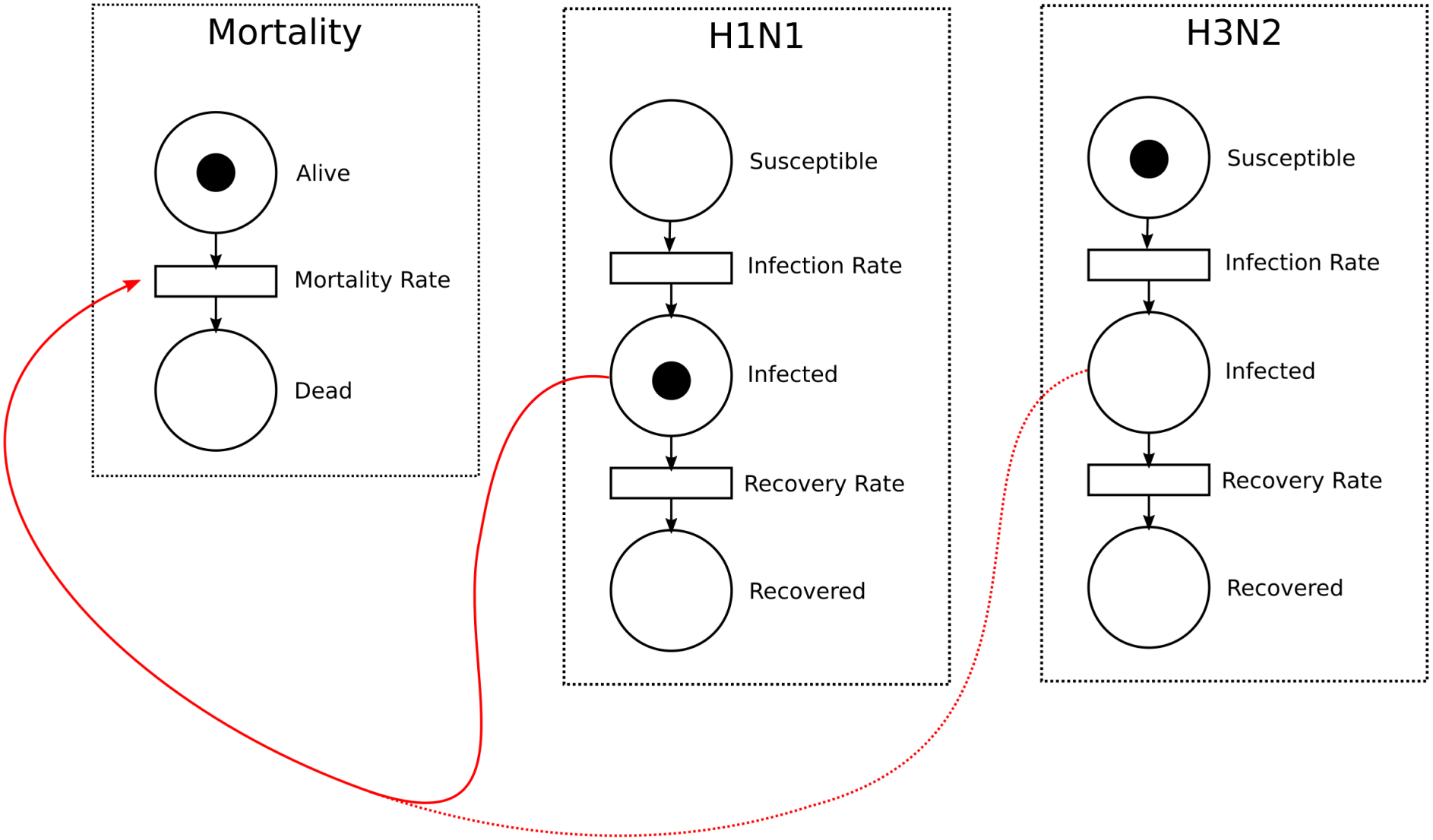

H1N1
Susceptible
Infection Rate
Infected
Recovery Rate
Recovered

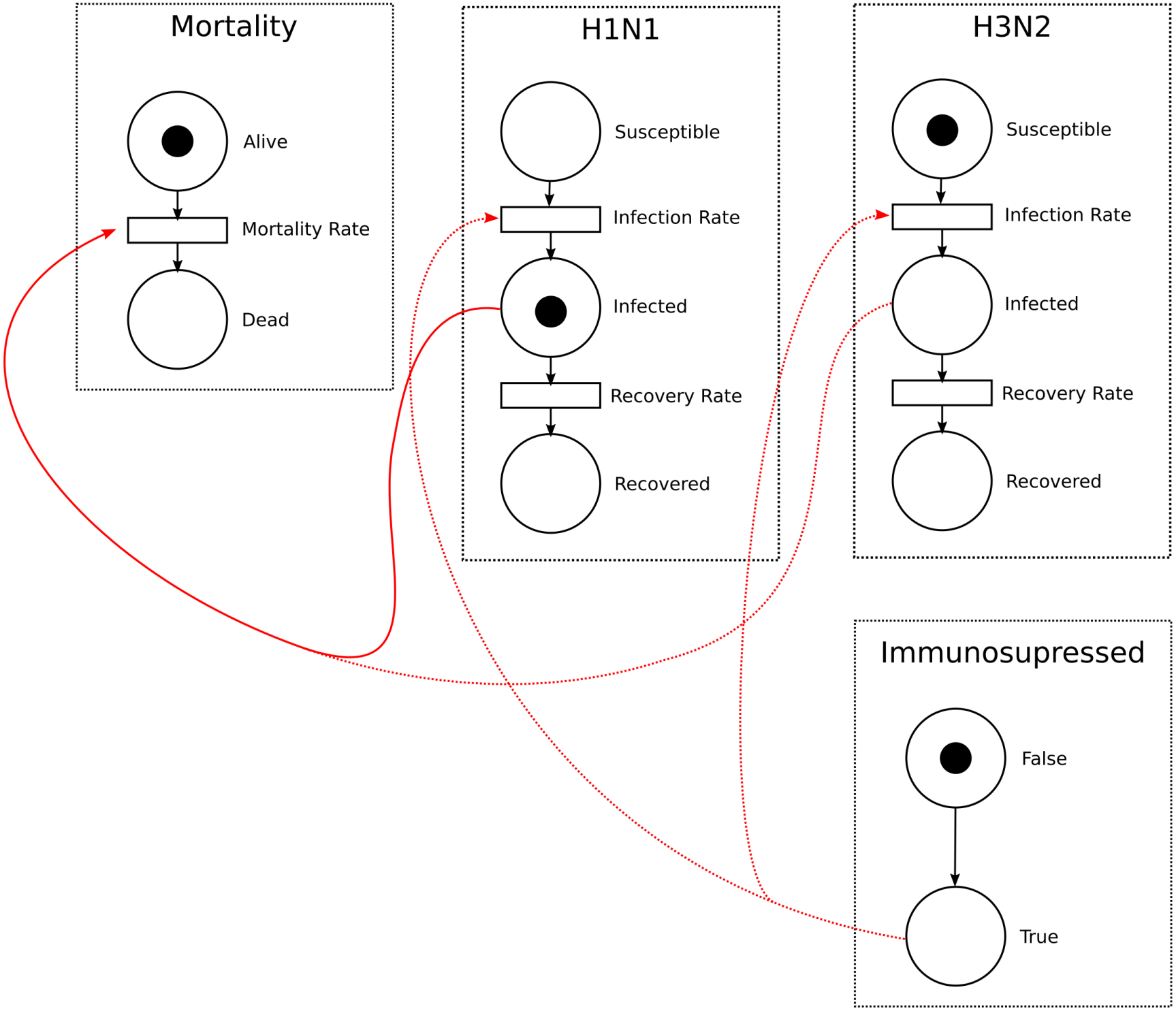

H3N2
Susceptible
Infection Rate
Infected
Recovery Rate
Recovered

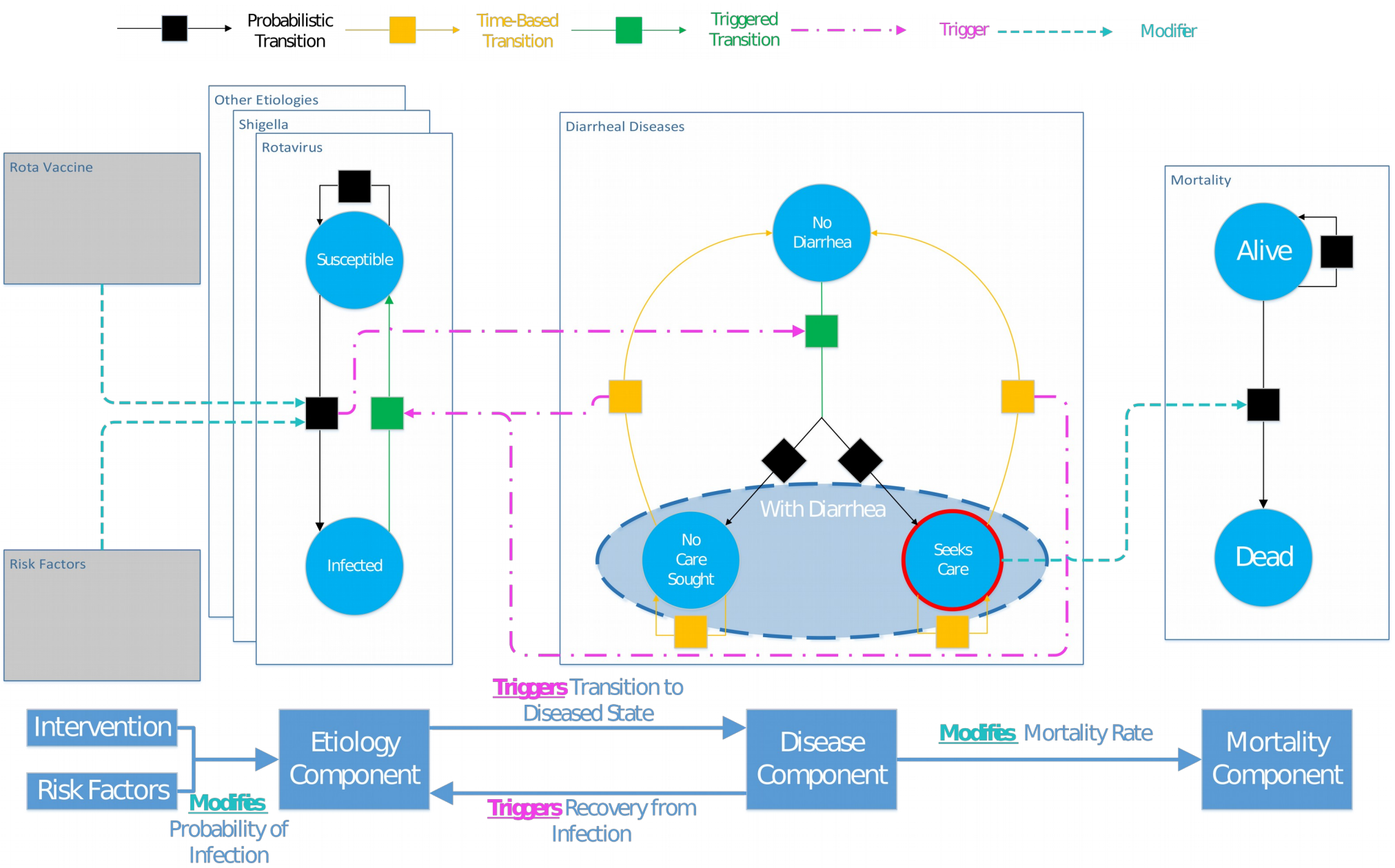

Mortality
Alive
Mortality Rate
Dead


H1N1
Susceptible
Infection Rate
Infected
Recovery Rate
Recovered

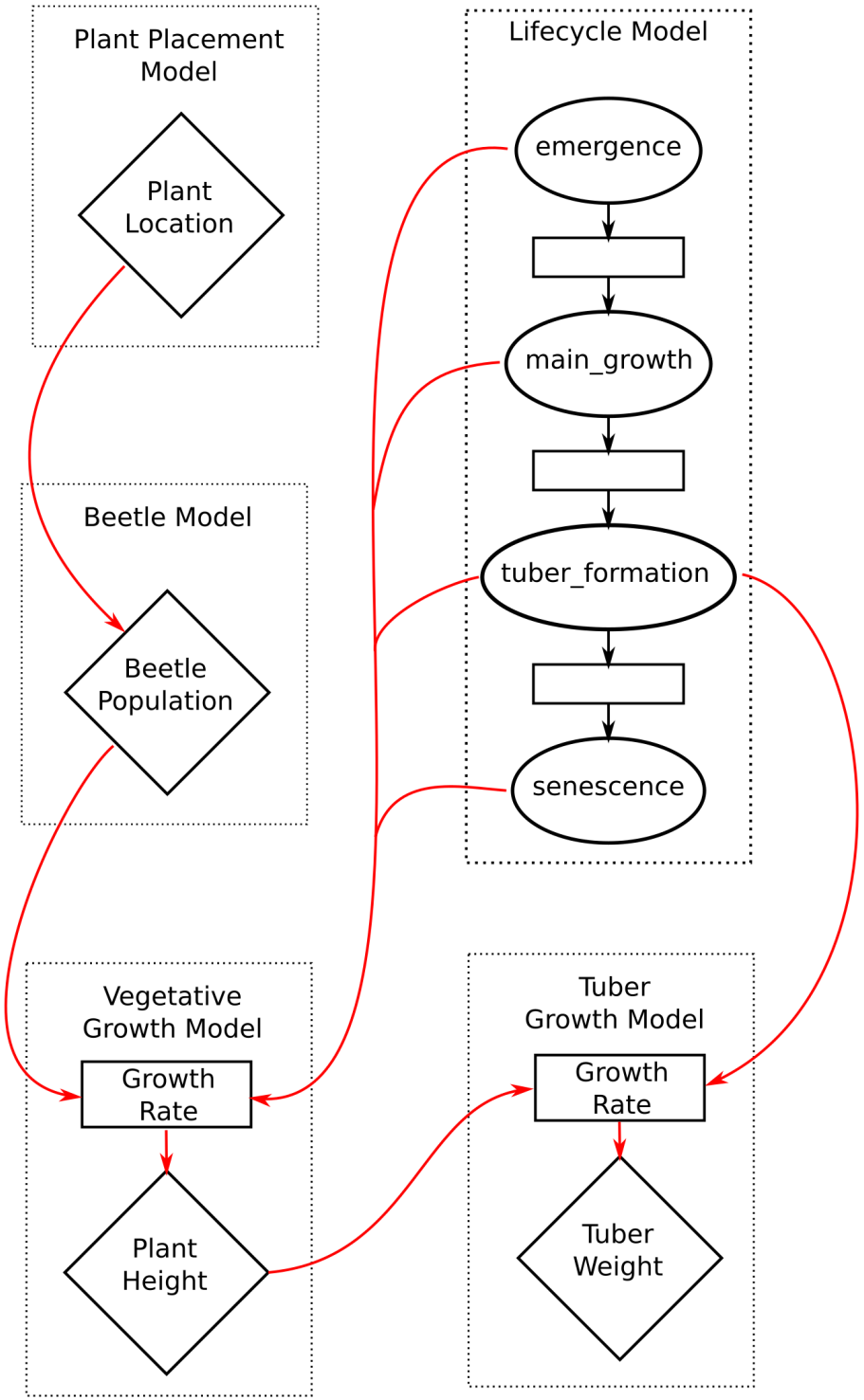

H3N2
Susceptible
Infection Rate
Infected
Recovery Rate
Recovered


Mortality
Alive
Mortality Rate
Dead
H1N1
Susceptible
Infection Rate
Infected
Recovery Rate
Recovered
H3N2
Susceptible
Infection Rate
Infected
Recovery Rate
Recovered


Mortality
Alive
Mortality Rate
Dead
H1N1
Susceptible
Infection Rate
Infected
Recovery Rate
Recovered
H3N2
Susceptible
Infection Rate
Infected
Recovery Rate
Recovered
Immunosupressed
False
True


Probabilistic Transition
Time-Based Transition
Triggered Transition
Trigger
Modifier
Other Etiologies
Shigella
Rotavirus
Rota Vaccine
Risk Factors
Susceptible
Infected
Diarrheal Diseases
No Diarrhea
With Diarrhea
No Care Sought
Seeks Care
Mortality
Alive
Dead
Intervention
Risk Factors
Modifies Probability of Infection
Etiology Component
Triggers Transition to Diseased State
Triggers Recovery from Infection
Disease Component
Modifies Mortality Rate
Mortality Component

What’s Vivarium then?

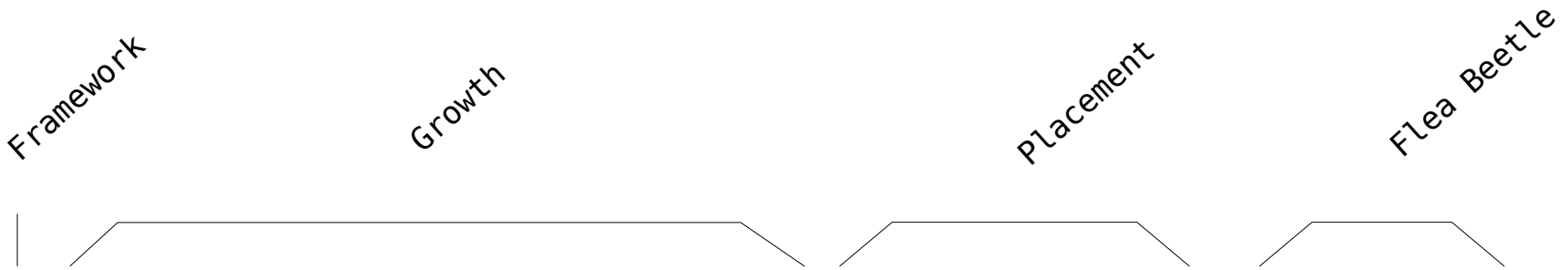

Plant Placement Model
Plant Location
Beetle Model
Beetle Population
Lifecycle Model
emergence
main_growth
tuber_formation
senescence
Vegetative Growth Model
Growth Rate
Plant Height
Tuber Growth Model
Growth Rate
Tuber Weight

```
do_some_setup()

while current_time < end_time:
    execute_all_components()

output_results()
```

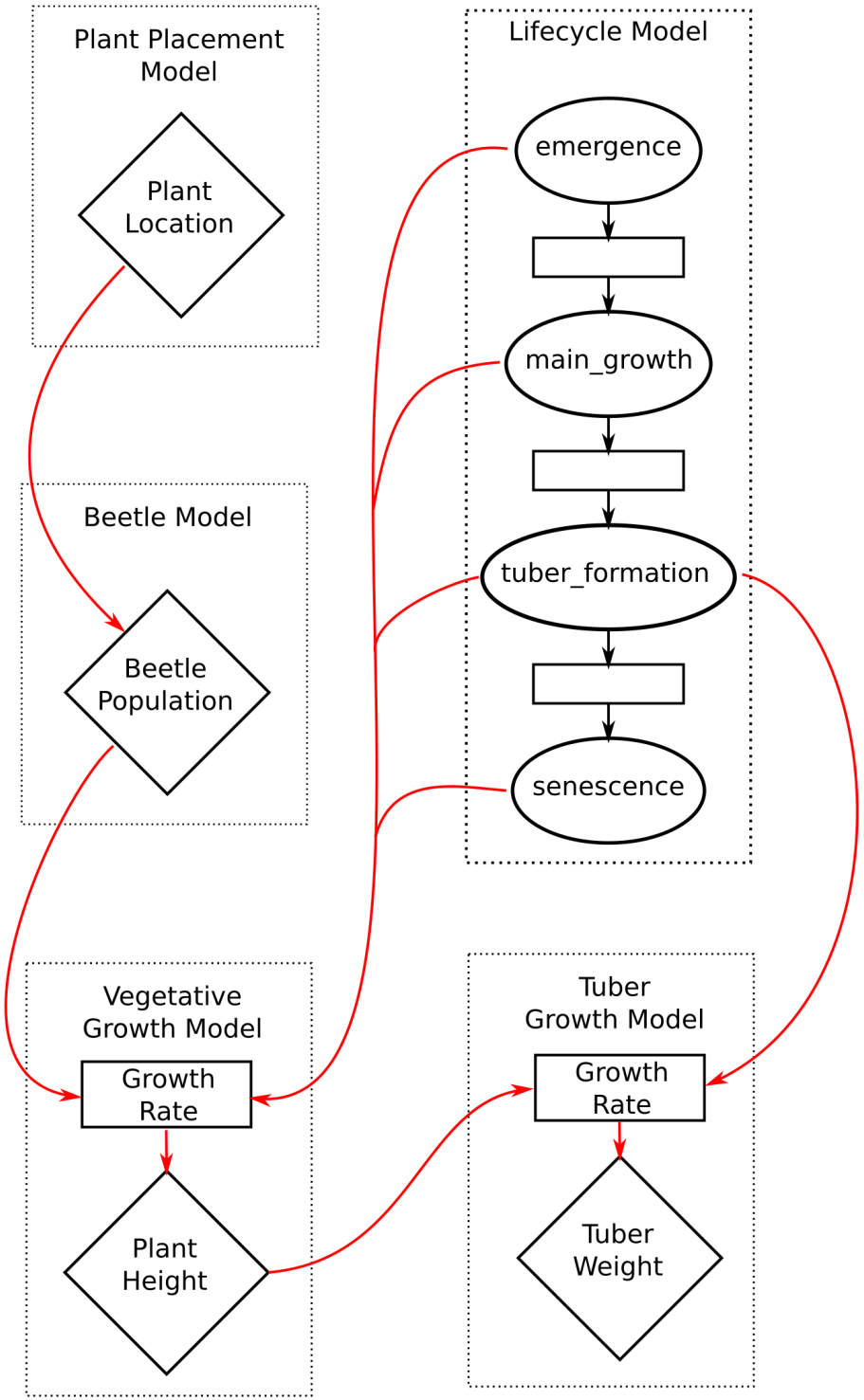
```
    @listens_for('time_step')
    @uses_columns(['plant_height'])
    def growth(self, event):
        effective_daily_growth = self.daily_growth(event.index)

        population = event.population

        population['plant_height'] += effective_daily_growth

        event.population_view.update(population)
```

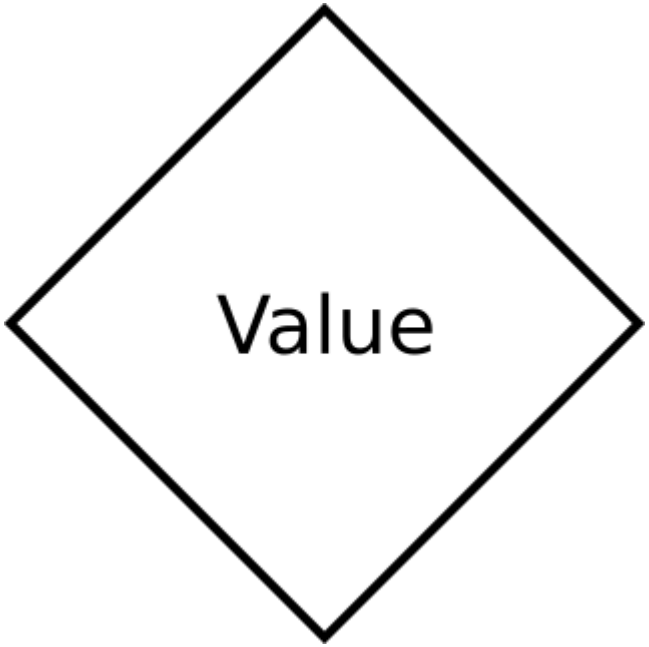
id	plant_height	growth_phase	tuber_weight	location_x	location_y	beetle_population
0	16	main_growth	0	0	0	2
1	20	tuber_formation	10	10	10	104
2	15	main_growth	0	11	10	30

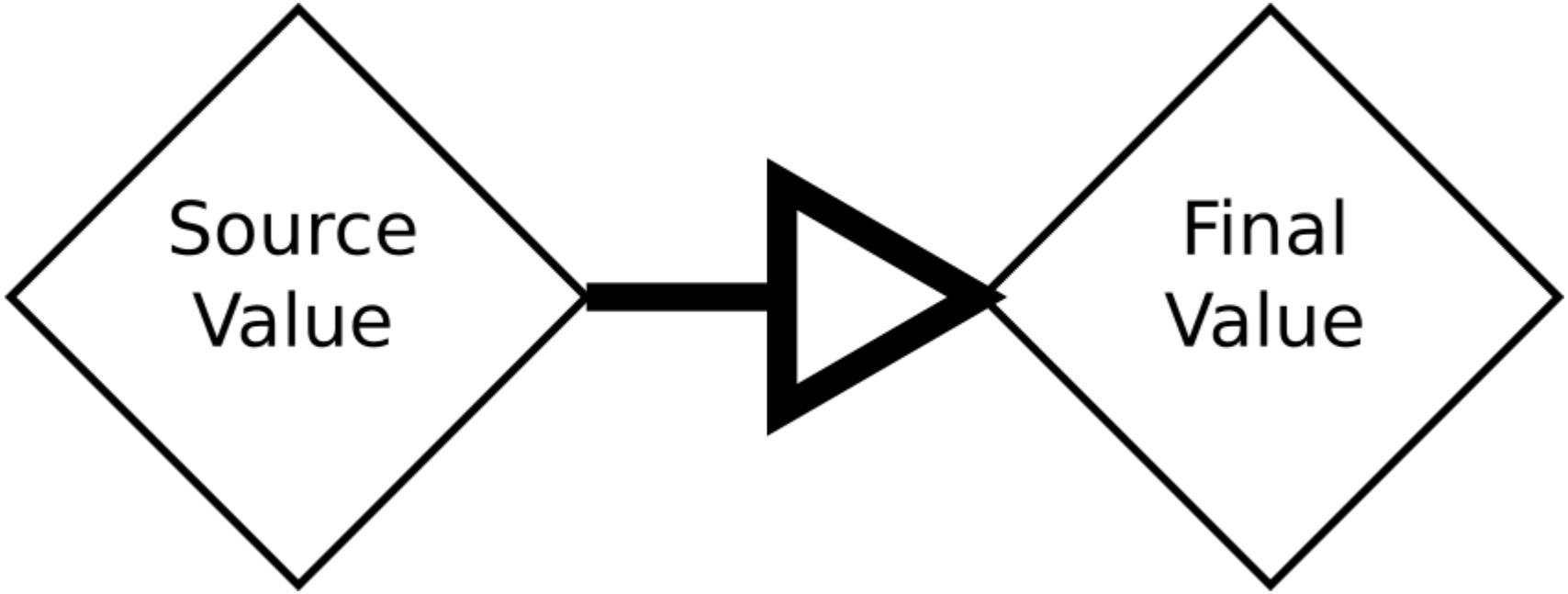
```
    @listens_for('initialize_simulants')
    @uses_columns(['tuber_weight'])
    def create_initial_weight(self, event):
        event.population_view.update(pd.Series(0.0, index=event.index))
```

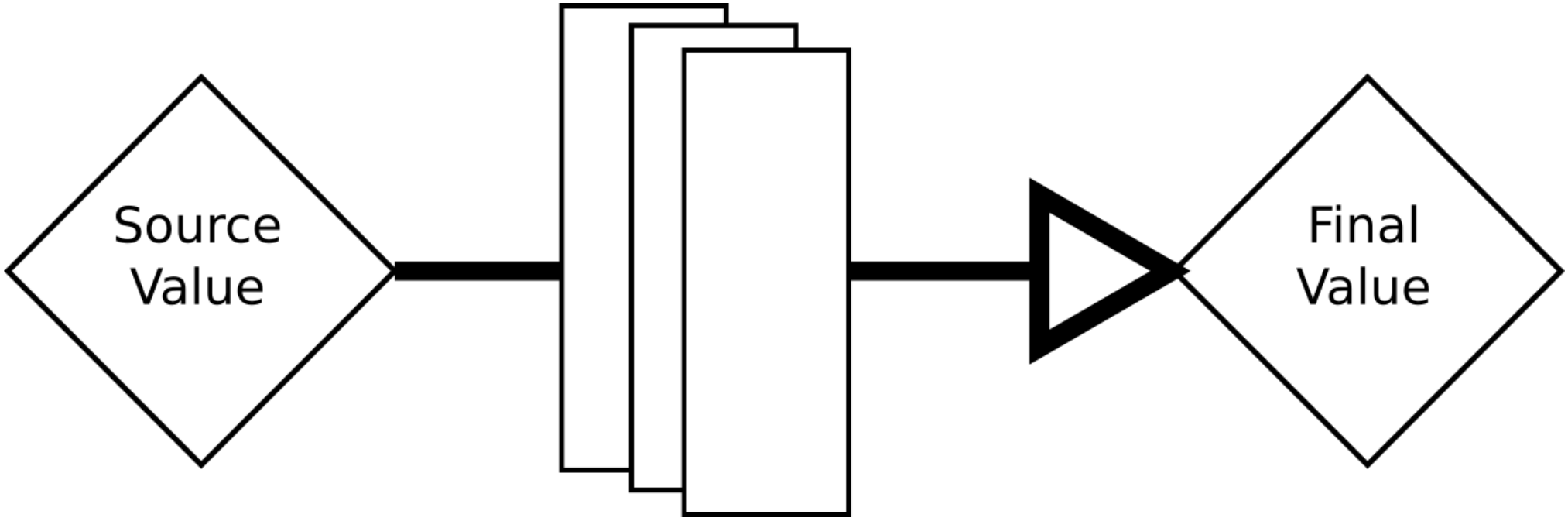


id	plant_height	growth_phase	tuber_weight	location_x	location_y	beetle_population
0	16	main_growth	0	0	0	2
1	20	tuber_formation	10	10	10	104
2	15	main_growth	0	11	10	30


Plant Placement Model
Plant Location
Beetle Model
Beetle Population
Lifecycle Model
emergence
main_growth
tuber_formation
senescence
Vegetative Growth Model
Growth Rate
Plant Height
Tuber Growth Model
Growth Rate
Tuber Weight

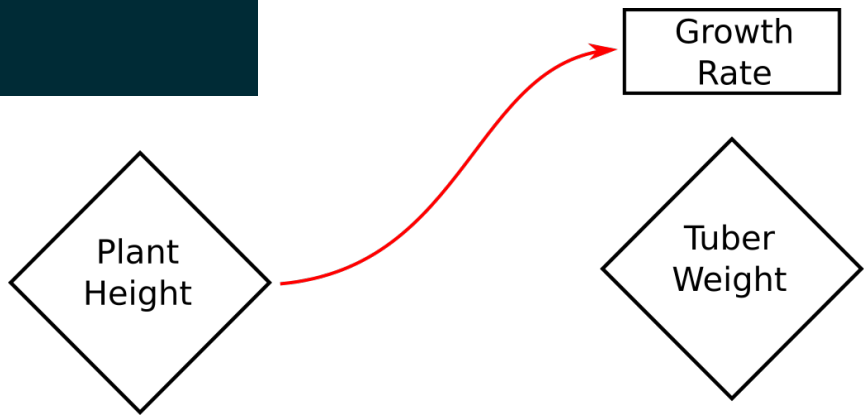

Value


Source
Value
Final
Value


Source
Value
Final
Value

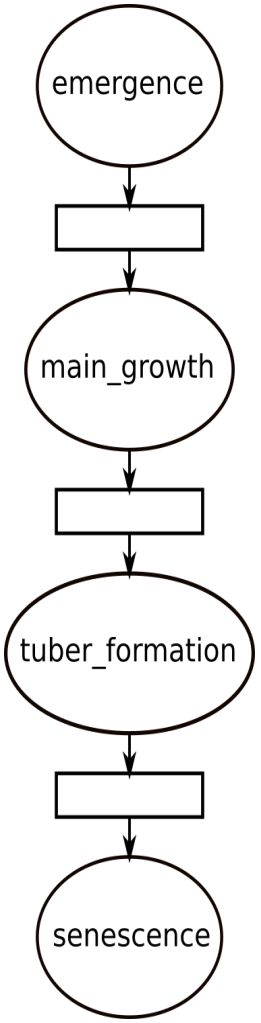
```python
@produces_value('potato.daily_tuber_growth')
def tuber_growth_base_rate(self, index):
    population = self.population_view.get(index)

    daily_growth_in_ounces = 1.8
    return pd.Series((population[self._model] == self.state_id) * daily_growth_in_ounces, index=index)
```

```python
    @modifies_value('potato.daily_tuber_growth')
    @uses_columns(['plant_height'])
    def modify_tuber_growth(self, index, rate, population_view):
        pop = population_view.get(index)
        multiplier = pop.plant_height / 24

        return rate * multiplier
```

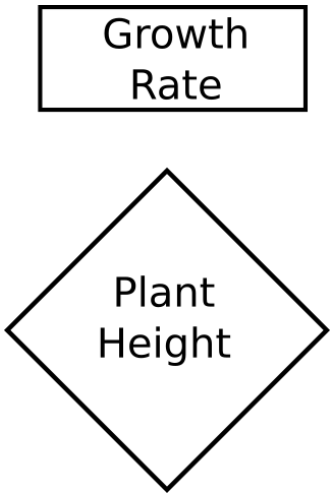



```python
    @listens_for('time_step')
    @uses_columns(['tuber_weight'])
    def growth(self, event):
        effective_daily_growth = self.daily_growth(event.index)

        population = event.population

        population['tuber_weight'] += effective_daily_growth

        event.population_view.update(population)
```

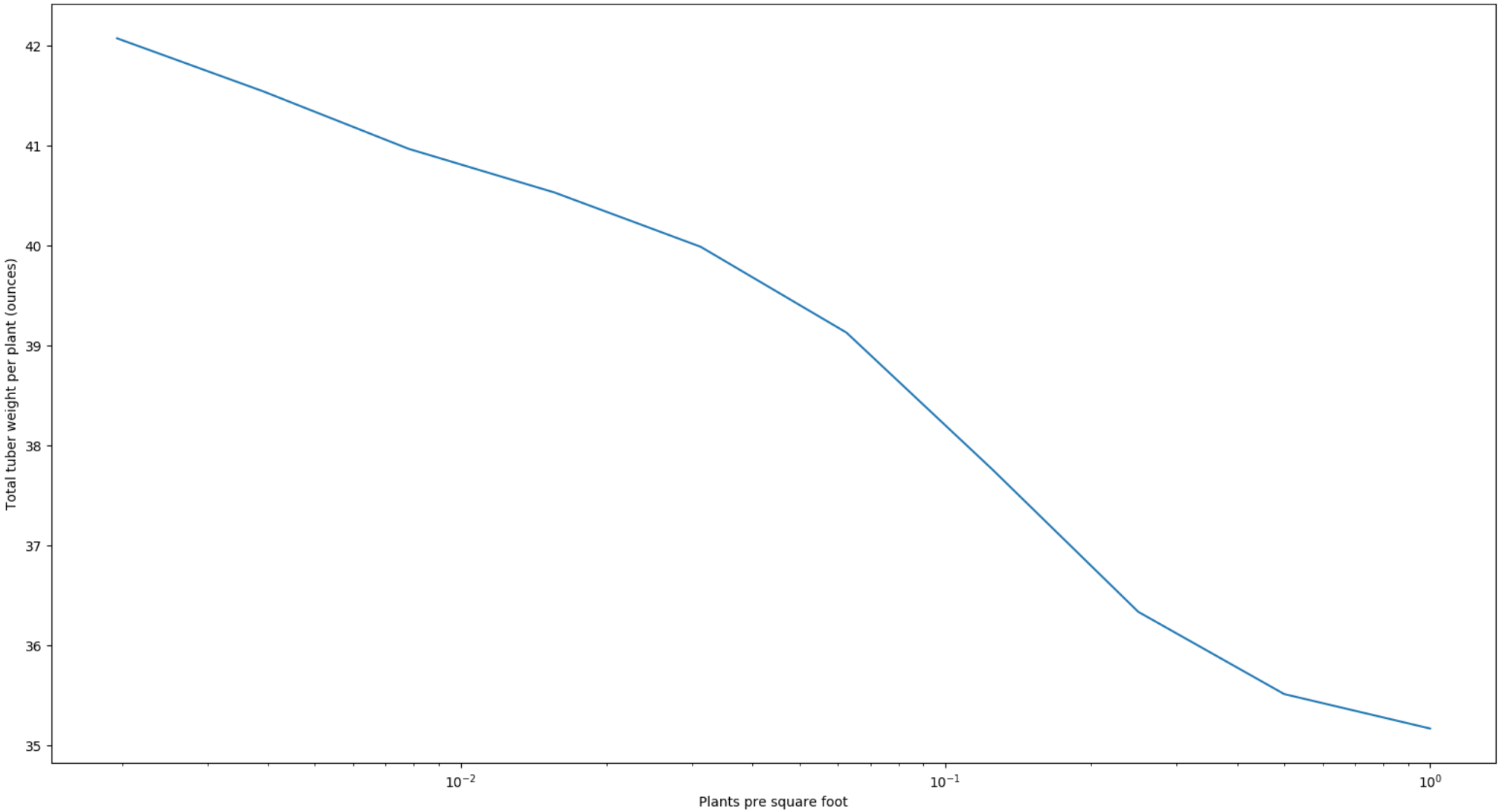
```
emergence = GrowthPhaseState('emergence', 0.0)
main_growth = GrowthPhaseState('main_growth', 1.0)
tuber_formation = TuberFormationPhase(0.1)
senescence = GrowthPhaseState('senescence', 0.0)

emergence.transition_set.append(RateTransition(emergence, main_growth, main_growth_rate))
main_growth.transition_set.append(RateTransition(main_growth, tuber_formation, tuber_formation_rate))
tuber_formation.transition_set.append(RateTransition(tuber_formation, senescence, senescence_rate))
```

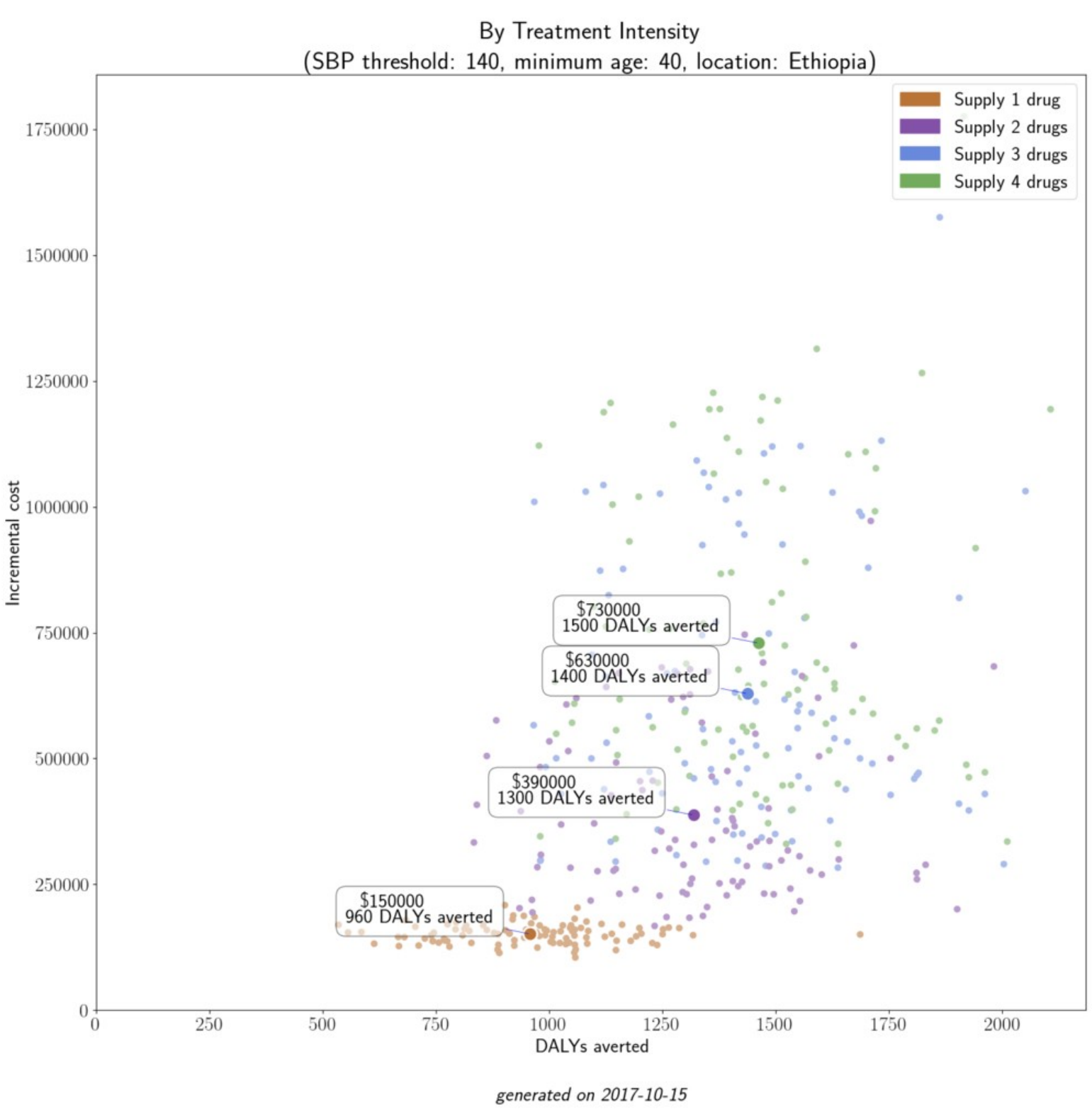


```python
class VegetativeGrowthComponent:
    def setup(self, builder):
        self.daily_growth = builder.value('potato.daily_vegetative_growth')
        self.daily_growth.source = lambda index: pd.Series(0.2, index=index)

    @listens_for('initialize_simulants')
    @uses_columns(['plant_height'])
    def create_initial_height(self, event):
        event.population_view.update(pd.Series(0.0, index=event.index))

    @listens_for('time_step')
    @uses_columns(['plant_height'])
    def growth(self, event):
        effective_daily_growth = self.daily_growth(event.index)

        population = event.population

        population['plant_height'] += effective_daily_growth

        event.population_view.update(population)

    @modifies_value('potato.daily_tuber_growth')
    @uses_columns(['plant_height'])
    def modify_tuber_growth(self, index, rate, population_view):
        pop = population_view.get(index)
        multiplier = pop.plant_height / 24

        return rate * multiplier

    @listens_for('simulation_end')
    @uses_columns(['plant_height'])
    def metrics(self, event):
        print(f'Average Plant Height: {event.population.plant_height.mean()} in')
```



```
class PlantingComponent:
    configuration_defaults = {
        'planting': {
            'density': 0.05 # Plants per square foot
        }
    }

    def setup(self, builder):
        self.density = builder.configuration.planting.density
        self.randomness = builder.randomness('placement')

    @listens_for('initialize_simulants')
    @uses_columns(['location_x', 'location_y'])
    def create_initial_population(self, event):
        garden_edge_length = np.sqrt(len(event.index) / self.density)
        x = self.randomness.get_draw(event.index, 'x') * garden_edge_length
        y = self.randomness.get_draw(event.index, 'y') * garden_edge_length
        placements = pd.DataFrame({'location_x': x, 'location_y': y})
        event.population_view.update(placements)
~
```


Total tuber weight per plant (ounces)
42
41
40
39
38
37
36
35
10^{-2}
10^{-1}
10^{0}
Plants pre square foot


By Treatment Intensity
(SBP threshold: 140, minimum age: 40, location: Ethiopia)
Supply 1 drug
Supply 2 drugs
Supply 3 drugs
Supply 4 drugs
Incremental cost
DALYs averted
$730000
1500 DALYs averted
$630000
1400 DALYs averted
$390000
1300 DALYs averted
$150000
960 DALYs averted
generated on 2017-10-15