



Shared (./..) | notebooks / HIV Mini Project.ipynb

Open in CoCalc (./../projects/27f4fe52-a1c1-4e36-8147-011565e6dcf1/files/notebooks/HIV Mini Project.ipynb?session=share)

Download (HIV Mini Project.ipynb?viewer=download), Raw (HIV Mini Project.ipynb?viewer=raw), Embed (HIV Mini Project.ipynb?viewer=embed)

Author: Lieselotte Heinrich (/share/users/4fe045cf-a441-464e-b487-3639647c04dc)

HIV Mini Project

Based on *Reduction of HIV Concentration During Acute Infection: Independence from a Specific Immune Response* by Andrew N. Phillips

```
In [1]: # Configure Jupyter so figures appear in the notebook
        %matplotlib inline
```

```
# import functions from the modsim.py module
from modsim import *
```

```
In [2]: def make_system(alpha, beta, gamma, delta, mu, pi, rho, sigma, tau):
        """Make a system object for the HIV model.

        alpha: rate of activation of latently infected cells [E/(Lt)]
        beta: rate of infection of uninfected cells per viron [1/(Vt)]
        gamma: rate of production of uninfected cells [R/t]
        delta: rate of removal of actively infected cells [1/t]
        mu: rate of cell death [1/t]
        pi: rate of production of virions by actively infected cells [V/(Et)]
        rho: proportion of cells that become latently infected upon infection [L/R]
        sigma: rate of removal of virions [1/t]
        tau: proportion of uninfected cells activated [1]

        returns: System object
        """
        init = State(R=200, L=0, E=0, V=4 * 10**-7)
        t_0 = 0
        t_end = 120 # days
        dt = 1.0/20.0 # 1.2 hour increments

        return System(init=init, t_0=t_0, t_end=t_end, dt=dt, alpha=alpha, beta=beta,
                      gamma=gamma, delta=delta, mu=mu, pi=pi, rho=rho, sigma=sigma, tau=tau)
```

```
In [3]: def update_func(state, t, system):
        """Update the HIV model.

        state: State with variables R, L, E, V
        t: time step
        system: System with parameters alpha, beta, gamma, delta, mu, pi, rho, sigma, and tau

        returns: State object
        """
        r, l, e, v = state

        # infections/activations
        infected = system.beta * r * v
        latentlyInfected = system.rho * infected
        activelyInfected = infected - latentlyInfected
        latentlyInfectedActivated = system.alpha * l

        # productions
        uninfectedProduction = system.gamma * system.tau
        vironProduction = system.pi * e

        # deaths/removals
        uninfectedDeath = system.mu * r
        latentlyInfectedDeath = system.mu * l
        activelyInfectedDeath = system.delta * e
        vironDeath = system.sigma * v

        dt = svstem.dt
```

```

r += (uninfectedProduction - infected - uninfectedDeath) *dt
l += (latentlyInfected - latentlyInfectedActivated - latentlyInfectedDeath) *dt
e += (activelyInfected + latentlyInfectedActivated - activelyInfectedDeath) *dt
v += (vironProduction - vironDeath) *dt

return State(R=r, L=l, E=e, V=v)

```

```

In [4]: def run_simulation(system, update_func):
        """Runs a simulation of the system.
        Adds a TimeFrame to the System: results

        system: System object
        update_func: function that updates state
        """
        init = system.init
        t_0, t_end, dt = system.t_0, system.t_end, system.dt
        frame = TimeFrame(columns=init.index)
        frame.row[t_0] = init
        ts = linrange(t_0, t_end, dt)

        for t in ts:
            frame.row[t+dt] = update_func(frame.row[t], t, system)

        return frame

```

```

In [5]: def plot_results(R, L, E, V):
        """Plot the results of the HIV model
        """
        xNums = [0,30,60,90,120]
        fig, (pl1, pl2) = plt.subplots(1, 2, figsize=(16, 8)) # create 2 subplots, right & left

        color = 'tab:green'
        pl1.set_xlabel('Days from infection')
        pl1.set_ylabel('CD4 lymphocytes', color=color)
        cells, = pl1.plot(R + E + L + 800, color=color)
        pl1.tick_params(axis='y', labelcolor=color)
        pl1.set_ylim([0, 1200])
        plt.xticks(xNums)

        ax1 = pl1.twinx() # instantiate a second axis on pl1 that shares the same x-axis
        color = 'tab:purple'
        ax1.set_ylabel('Virons V', color=color) # we already handled the x-label with ax1
        v, = ax1.semilogy(V, color=color)
        ax1.tick_params(axis='y', labelcolor=color)
        ax1.set_ylim([0.1, 10000])

        color = 'tab:red'
        pl2.set_xlabel('Days from infection')
        pl2.set_ylabel('R', color=color)
        r, = pl2.plot(R, color=color)
        pl2.tick_params(axis='y', labelcolor=color)
        pl2.set_ylim([0, 250])
        plt.xticks(xNums)

        ax2 = pl2.twinx() # instantiate a second axis on pl2 that shares the same x-axis
        color = 'tab:blue'
        ax2.set_ylabel('L and E', color=color) # we already handled the x-label with ax1
        l, = ax2.semilogy(L, color=color)
        e, = ax2.semilogy(E, color=color, linestyle='dashed')
        ax2.tick_params(axis='y', labelcolor=color)
        ax2.set_ylim([0.1, 100])

        pl1.legend((cells, v), ('CD4 lymphocytes', 'Cell-free virus'), fontsize=12, loc='upper right')
        pl2.legend((r, l, e), ('R', 'L', 'E'), fontsize=12, loc='upper right')
        fig.tight_layout() # otherwise the right y-label is slightly clipped

```

```

In [6]: alpha = 3.6 * 10**-2 # rate of activation of latently infected cells [E/(Lt)]
        beta = 0.0027 # rate of infection of uninfected cells per viron [1/(Vt)]
        gamma = 1.36 # rate of production of uninfected cells [R/t]
        delta = 0.33 # rate of removal of actively infected cells [1/t]
        mu = 1.36 * 10**-3 # rate of cell death [1/t]
        nu = 100 # rate of production of virions by actively infected cells [V/(Et)]

```

```

pi = 100 # rate of production of virions by actively infected cells [V/(ET)]
rho = 0.1 # proportion of cells that become latently infected upon infection [L/R]
sigma = 2 # rate of removal of virions [1/t]
tau = 0.2 # proportion of uninfected cells activated [1]

system = make_system(alpha, beta, gamma, delta, mu, pi, rho, sigma, tau)
results = run_simulation(system, update_func)
plot_results(results.R, results.L, results.E, results.V)

```

