IDLE Input

```
import sys
>>> sys.path.append('E:/Users/Ben/Epi/')
>>> from Epi_Model_4 import *
>>> m1=InfectModel (50000, 100, .1, .2, .3, .4, .5, .6, .4, .5, 6, 14, 20, False, False, False,
False, False, False, .1, .2, .3, .4, .5, .6)
>>> m1.run_steps(25)
```

Code

```
# Project 2017-2018 ATC - Flu Transmission
# Built on Mesa Agent Library from George Mason Univ.
# Written by Ben Thorp
# ATC-3 Ben Thorp, Alex Baten, Teddy Gonzales
# Version 1.1
#   Added start and stop of contagiousness
# Version 1.2
#   Added HealthCareAccess and HealthLifeStyle
# Version 1.3
#   Added community3 and buses
# Version 1.4
#   Added community4, community5, community6, and age groups

from mesa import Agent, Model
from mesa.time import RandomActivation
import random
from mesa.space import MultiGrid
from mesa.datacollection import DataCollector
import matplotlib.pyplot as plt
from math import exp, expm1

# Function that computs the number of infections for graphing
def compute_infections(model):
    total_inf_count = 0
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell
        for human in cell_content:
```

```
            if human.infected:
                total_inf_count += 1
    return total_inf_count

# Function that computes the number of infections in community 1 for graphing
def compute_infections_c1(model):
    total_inf_count = 0
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell
        for human in cell_content:
            if human.infected:
                if (human.community is 1):
                    total_inf_count += 1
    return total_inf_count

# Function that computs the number of infections in community 2 for graphing
def compute_infections_c2(model):
    total_inf_count = 0
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell
        for human in cell_content:
            if human.infected:
                if ( human.community is 2):
                    total_inf_count += 1
    return total_inf_count

# Function that computes the number of infections in community 3 for graphing
def compute_infections_c3(model):
    total_inf_count = 0
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell
        for human in cell_content:
            if human.infected:
                if (human.community is 3):
                    total_inf_count += 1
    return total_inf_count

# Function that computes the number of infections in community 4 for graphing
def compute_infections_c4(model):
    total_inf_count = 0
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell
        for human in cell_content:
```

```
            if human.infected:
                if (human.community is 4):
                    total_inf_count += 1
    return total_inf_count

# Function that computes the number of infections in community 5 for graphing
def compute_infections_c5(model):
    total_inf_count = 0
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell
        for human in cell_content:
            if human.infected:
                if (human.community is 5):
                    total_inf_count += 1
    return total_inf_count

# Function that computes the number of infections in community 6 for graphing
def compute_infections_c6(model):
    total_inf_count = 0
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell
        for human in cell_content:
            if human.infected:
                if (human.community is 6):
                    total_inf_count += 1
    return total_inf_count

# Function that computes the number of infections in the bus route for graphing
def compute_infections_b1(model):
    total_inf_count = 0
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell
        for human in cell_content:
            if human.infected:
                total_inf_count += 1
    return total_inf_count

# Function that computes the number of infections in the work for graphing
def compute_infections_work(model):
    total_inf_count = 0
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell
        for human in cell_content:
```

```
            if human.infected:
                total_inf_count += 1
    return total_inf_count

# Function that computes the number of infections in the school for graphing
def compute_infections_school(model):
    total_inf_count = 0
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell
        for human in cell_content:
            if human.infected:
                total_inf_count += 1
    return total_inf_count

# Function that computs the immunity for everyone for graphing
def compute_immunity(model):
    total_im_count = 0
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell
        for human in cell_content:
            if human.immunity:
                total_im_count += 1
    return total_im_count

class InfectModel(Model):
    """A Mesa Model to simulate the spread of disease through a home and work environment"""

    # __init__ creates the model
    def __init__(self, N,
                 hls_com1, hls_com2, hls_com3, hls_com4, hls_com5, hls_com6, #lowL, highL,
                 con_start, con_end_short, con_end_long,
                 com1_hca, com2_hca, com3_hca, com4_hca,
                 com5_hca, com6_hca, vac_com1, vac_com2,
                 vac_com3, vac_com4, vac_com5, vac_com6):
        # N - the total number of humans
        # h - the number of houses or rooms in each environment

        # hls_com1 - the percentage of people in community1
        #   that are living a healthy lifestyle
        # hls_com2 - the percentage of people in communtiy2
        #   that are living a healthy lifestyle
        # hls_com3 - the percentage of people in communtiy3
        #   that are living a healthy lifestyle
```

```
# hls_com4 - the percentage of people in communtiy4
#   that are living a healthy lifestyle
# hls_com5 - the percentage of people in communtiy5
#   that are living a healthy lifestyle
# hls_com6 - the percentage of people in communtiy6
#   that are living a healthy lifestyle
# lowL - a low likelyhood to catch the disease (
#   assosiated with healthy lifestyle)
# highL - a high likelyhood to catch the disease (
#   assosiated with healthy lifestlye)

# con_start - when the disease starts to be contagious
# con_end_short - when the disease stops being contagious with health care
# con_end_long - when the disease stops being contagious without health care

# com1_hca - the health care access in community1
# com2_hca - the health care access in community2
# com3_hca - the health care access in community3
# com4_hca - the health care access in community4
# com5_hca - the health care access in community5
# com6_hca - the health care access in community6

# vac_com1 - percentage of vaccinated humans in community1
# vac_com2 - percentage of vaccinated humans in community2
# vac_com3 - percentage of vaccinated humans in community3
# vac_com4 - percentage of vaccinated humans in community4
# vac_com5 - percentage of vaccinated humans in community5
# vac_com6 - percentage of vaccinated humans in community6

self.num_agents = N
self.grid = MultiGrid(9, 78000, True)
self.schedule = RandomActivation(self)

self.community1_hca = com1_hca
self.community2_hca = com2_hca
self.community3_hca = com3_hca
self.community4_hca = com4_hca
self.community5_hca = com5_hca
self.community6_hca = com6_hca

self.con_end_short = con_end_short
self.con_end_long = con_end_long
```

```
self.vaccinated_com1 = vac_com1
self.vaccinated_com2 = vac_com2
self.vaccinated_com3 = vac_com3
self.vaccinated_com4 = vac_com4
self.vaccinated_com5 = vac_com5
self.vaccinated_com6 = vac_com6

self.healthy_lifestyle_com1 = hls_com1
self.healthy_lifestyle_com2 = hls_com2
self.healthy_lifestyle_com3 = hls_com3
self.healthy_lifestyle_com4 = hls_com4
self.healthy_lifestyle_com6 = hls_com5
self.healthy_lifestyle_com5 = hls_com6

#self.low_likelihood = lowL
#self.high_likelihood = highL

# Create N humans for the model
for i in range(self.num_agents):
    if i% 10000 == 0 :
        print(i)
    a = Human(i, self, con_start)
    self.schedule.add(a)

    #y = random.randrange(self.grid.height)
    self.grid.place_agent(a, (a.community, a.household))

    # Add the agent to a random grid cell
    #if (a.community == 1):
    #    x = 0
    #elif (a.community == 3):
    #    x = 4
    #elif (a.community == 4):
    #    x = 6
    #elif (a.community == 5):
    #    x = 7
    #elif (a.community == 6):
    #    x = 8
    #else:
    #    x = 1
```

```
        # Initialize timestep
        self.timestep=0
        #self.day=True
        self.transportation_cycle = 0

        # Initialize the software that collects the data each timestep
        self.datacollector = DataCollector(
            model_reporters={"Community1": compute_infections_c1,
                             "Community2": compute_infections_c2,
                             "Community3": compute_infections_c3,
                             "Community4": compute_infections_c4,
                             "Community5": compute_infections_c5,
                             "Community6": compute_infections_c6,
                             #"Work Place": compute_infections_work,
                             #"School": compute_infections_school,
                             #"Public Transport": compute_infections_b1,
                             "Immunity": compute_immunity,
                             "Total Infections": compute_infections,

                             }
            )

    def step(self):
        # Collects the data for this timestep
        self.datacollector.collect(self)

        self.schedule.step()
        self.timestep+=1
        self.transportation_cycle = (self.timestep % 4 )

    # run_steps steps the model forword for "steps" (days/nights)
    def run_steps(self, steps):
        for i in range(steps*4):
            if i%4 == 0:
                s='Day: ' + str(i//4)
                print(s)
            # Quadupling steps makes correct number of day/night/bus cycles
            # Because a day/night cycle takes two timesteps
            self.step()
        inf_data = self.datacollector.get_model_vars_dataframe()
        inf_data.plot()
        plt.show()
```

```
class Human(Agent):
    """An agent that represents one human in the model"""
    def __init__(self, unique_id, model, con_start):
        # unique_id - the human's id number
        # model - the Mesa simulation class
        # likelihood - the chance that the infection spreads from one human to another
        # con_start - when the disease starts to be contagious

        # Call the Mesa agent setup
        super().__init__(unique_id, model)

        # Initialize the human variables
        tempa = [1, 2, 3, 4, 5, 6]
        com_size_weights = [.34, .10, .14, .22, .13, .07]
        temp = random.choices(tempa, weights=com_size_weights, k=1)
        self.community = temp[0]
        #self.household=random.randrange(model.grid.height)
        tempb = model.num_agents
        tempc = com_size_weights[(self.community) - 1]
        self.household=random.randrange(int(tempb * tempc * 0.3))
        #self.schoolroom=random.randrange(model.grid.height)
        self.schoolroom=random.randrange(int(model.num_agents/25))
        #self.workroom=random.randrange(model.grid.height)
        self.workroom=random.randrange(int(model.num_agents/100))
        #self.bus_number=random.randrange(model.grid.height)
        self.bus_number=random.randrange(int(model.num_agents/30))

        # Initialize the age to match Albuquerque's age profile
        self.age = self.my_age()

        # Initializes the age bracket
        self.ic = [[0.00061,0.00033,0.00080], [0.00053,0.00032,0.00080],
[0.00057,0.00029,0.00102]]

        self.bracket_youth = 0
        self.bracket_adult = 1
        self.bracket_elderly = 2

        if self.age >= 5 and self.age <= 18:
            self.age_bracket = self.bracket_youth

        elif self.age >= 19 and self.age <= 65:
            self.age_bracket = self.bracket_adult
```

```
  elif self.age >= 66 and self.age <= 95:
     self.age_bracket = self.bracket_elderly

  else:
     print ("Something went wrong when assigning ages")



  # Initialize disease variables
  # self.likelihood = likelihood
  #tmp_rnd = random.random()
  #if self.community == 1:
  #    if (tmp_rnd <= model.healthy_lifestyle_com1):
  #        self.likelihood = model.low_likelihood
  #    else:
  #        self.likelihood = model.high_likelihood

  #elif self.community == 2:
  #    if (tmp_rnd <= model.healthy_lifestyle_com2):
  #        self.likelihood = model.low_likelihood
   #   else:
   #       self.likelihood = model.high_likelihood

  #elif self.community == 3:
  #    if (tmp_rnd <= model.healthy_lifestyle_com3):
  #        self.likelihood = model.low_likelihood
  #    else:
  #        self.likelihood = model.high_likelihood

  #elif self.community == 4:
  #    if (tmp_rnd <= model.healthy_lifestyle_com4):
  #        self.likelihood = model.low_likelihood
  #    else:
  #        self.likelihood = model.high_likelihood

  #elif self.community == 5:
  #    if (tmp_rnd <= model.healthy_lifestyle_com5):
  #        self.likelihood = model.low_likelihood
   #   else:
   #       self.likelihood = model.high_likelihood

#  elif self.community == 6:
```

```
#     if (tmp_rnd <= model.healthy_lifestyle_com6):
#         self.likelihood = model.low_likelihood
 #    else:
 #        self.likelihood = model.high_likelihood
 #else:
 #    print ("Something went wrong witht the hls")

 self.con_timer = 0
 self.con_start = con_start

 # change length of disease based on community health care access
 if self.community == 1:
    if model.community1_hca == True:
       self.con_end = model.con_end_short
    else :
       self.con_end = model.con_end_long

 elif self.community == 2:
    if model.community2_hca == True:
       self.con_end = model.con_end_short
    else :
       self.con_end = model.con_end_long

 elif self.community == 3:
    if model.community3_hca == True:
       self.con_end = model.con_end_short
    else :
       self.con_end = model.con_end_long

 elif self.community == 4:
    if model.community4_hca == True:
       self.con_end = model.con_end_short
    else :
       self.con_end = model.con_end_long

 elif self.community == 5:
    if model.community5_hca == True:
       self.con_end = model.con_end_short
    else :
       self.con_end = model.con_end_long

 elif self.community == 6:
    if model.community6_hca == True:
```

```
        self.con_end = model.con_end_short
    else :
        self.con_end = model.con_end_long

else :
    print ("Something went wrong with assinging the healthcare access")

self.immunity = False
tmp_rnd = random.random()
if self.community == 1:
    if (tmp_rnd  <= model.vaccinated_com1):
        self.immunity = True

elif self.community == 2:
    if (tmp_rnd  <= model.vaccinated_com2):
        self.immunity = True

elif self.community == 3:
    if (tmp_rnd  <= model.vaccinated_com3):
        self.immunity = True

elif self.community == 4:
    if (tmp_rnd  <= model.vaccinated_com4):
        self.immunity = True

elif self.community == 5:
    if (tmp_rnd  <= model.vaccinated_com5):
        self.immunity = True

elif self.community == 6:
    if (tmp_rnd  <= model.vaccinated_com6):
        self.immunity = True

else:
    print ("Something went wrong with the self.immunity")

self.infected = False
if unique_id==1:
    self.infected=True
if unique_id==2:
    self.infected=True
if unique_id==3:
    self.infected=True
```

```
        if unique_id==4:
            self.infected=True
        if unique_id==5:
            self.infected=True
        if unique_id==6:
            self.infected=True

    def my_age(self):
        # a = random.randint(6, 95)
        p = random.random()
        if p < 0.25:
            a = 12

        elif p < .88:
            a = 45

        else:
            a = 75
        return a

    def move(self):
        # Placement of the communities rows
##          community0_row=1
##          community1_row=2
##          community3_row=3
##          community4_row=4
##          community5_row=5
##          community6_row=6

        # Placement of the day place rows
        #schoolroom_row=7
        #workroom_row=8

        # Placement of the bus
        #bus_row=0

        # Beginning cycle - Night
        if (self.model.transportation_cycle == 0):
            self.exposure_time = 840

            # Assigns the human to its correct community
            new_position = (self.community, self.household)
##              if (self.community == 1):
```

```
##             new_position = (community0_row, self.household)
##         elif (self.community == 2):
##             new_position = (community1_row, self.household)
##         elif (self.community == 3):
##             new_position = (community3_row, self.household)
##         elif (self.community == 4):
##             new_position = (community4_row, self.household)
##         elif (self.community == 5):
##             new_position = (community5_row, self.household)
##         elif (self.community == 6):
##             new_position = (community6_row, self.household)
##         else :
##             print ('Incorrect assigned community')

        # Morning Commute
        elif (self.model.transportation_cycle == 1):
            self.exposure_time = 60
            new_position = (0, self.bus_number) #bus_row

        # Day
        elif (self.model.transportation_cycle == 2):
            self.exposure_time = 560
            # This if-statement controls where the ages will go during the 'day'
            if (self.age_bracket == self.bracket_adult): # Ages go to work
                # workroom_row - the work environment
                # self.workroom - the room in the work environment
                new_position = (8, self.workroom) #workroom_row
            else : # Kids go to school
                new_position = (7, self.schoolroom) #schoolroom_row

        # Afternoon Commute
        elif (self.model.transportation_cycle == 3):
            self.exposure_time = 60
            new_position = (0, self.bus_number) #bus_row

        # Error Message
        else :
            print ("Something went wrong with the night/day cyle")

        self.model.grid.move_agent(self, new_position)

##    def infect_others_old(self):
##        cellmates = self.model.grid.get_cell_list_contents([self.pos])
```

```
##        if len(cellmates) > 1:
##            for other in cellmates:
##                if other.immunity is False:
##                    if other.infected is False:
##                        if(random.random() <= self.likelihood):
##                            other.infected = True

    def infect_others(self):
        cellmates = self.model.grid.get_cell_list_contents([self.pos])
        if len(cellmates) > 1:
            n = len(cellmates)
            for other in cellmates:
                if other.immunity is False:
                    if other.infected is False:
                        if(random.random() <= (1-(exp(-(self.ic[self.age_bracket][other.age_bracket] *
self.exposure_time))))):
                            other.infected = True

    def step(self):
        self.move()
        if self.infected:
            self.con_timer +=1
            if (( self.con_timer >= self.con_start) and (self.con_timer <= self.con_end)):
                    self.infect_others()
            else:
              if ( self.con_timer > self.con_end):
                  self.immunity = False
                  if self.community == 1 :
                      if self.model.community1_hca == True:
                          self.immunity = True
                  elif self.community == 2:
                      if self.model.community2_hca == True:
                          self.immunity = True
                  elif self.community == 3:
                      if self.model.community3_hca == True:
                          self.immunity = True
                  elif self.community == 4:
                      if self.model.community4_hca == True:
                          self.immunity = True
                  elif self.community == 5:
                      if self.model.community5_hca == True:
                          self.immunity = True
                  elif self.community == 6:
```

```
    if self.model.community6_hca == True:
        self.immunity = True
else:
    print ("Incorrect Community ID")

self.infected = False
```