**<부록 3> 소스 코드 (Jupyter Notebook 형식, 핵심 부분만 발췌)**

전체 소스는 https://github.com/leejs209/MesaABM/blob/master/MesaABM.ipynb에서 확인할 수 있다.

**1. 준비**

필요한 패키지 설치, 관련 설정 및 import

(코드 생략)

**2. 모델 구현**

**각 스텝에서 S,E,I,R 상태의 행위자의 수를 제시하는 함수 정의**

def s\_count(model):

cnt = 0

for agent in model.schedule.agents:

if agent.status == "S":

cnt += 1

return cnt

(i\_count, e\_count, r\_count생략)

def affected(model):

return model.N - s\_count(model)

**Student 및 School Class (모델) 구현**

class Student(Agent):

""" A Student that moves depending on the time, space, and other students"""

def \_\_init\_\_(self, unique\_id, group\_no, status, infection\_duration, model, dinner):

super().\_\_init\_\_(unique\_id, model)

self.group\_no = group\_no

(생략)

self.dinner = dinner

def meal\_normal(self):

if len(self.model.order) != 0:

t = self.model.order[self.unique\_id - 1]

else:

t = self.unique\_id

k = ((t - 1) // 5) % 46

if k % 2 == 0:

xpos = k \* 3 / 2 + 1

else:

xpos = (k \* 3 + 1) / 2

ypos = (t - 1) % 5 + 31 + 6 \* ((t - 1) // 460)

self.model.grid.move\_agent(self, (int(xpos), int(ypos)))

def meal\_distanced(self):

(생략 – 실제 보고서에서는 사용하지 않는 함수이다.)

def move\_to\_group(self, group\_no):

x1 = ((group\_no - 1) % 14) \* 10 + 1

x2 = x1 + 8

y1 = ((group\_no - 1) // 14) \* 10 + 1

y2 = y1 + 8

new\_pos = (self.random.randint(x1, x2), self.random.randint(y1, y2))

self.model.grid.move\_agent(self, new\_pos)

def move\_within\_bound(self, group\_no):

x1 = ((group\_no - 1) % 14) \* 10 + 1

x2 = x1 + 8

y1 = ((group\_no - 1) // 14) \* 10 + 1

y2 = y1 + 8

possible\_steps = self.model.grid.get\_neighborhood(self.pos, moore=True, include\_center=False, radius=1)

bounded\_steps = []

for x in possible\_steps:

if x1 <= x[0] <= x2 and y1 <= x[1] <= y2:

bounded\_steps.append(x)

if len(bounded\_steps) == 0:

return

new\_pos = self.random.choice(bounded\_steps)

self.model.grid.move\_agent(self, new\_pos)

def spread\_infection(self, multiplier):

# Spread infection only when self is Infected or Exposed

if self.status != "I" and self.status != "E":

return

cellmates = self.model.grid.get\_cell\_list\_contents([self.pos])

if len(cellmates) <= 1:

return

if self.status == "I":

for x in cellmates:

if x.status == "S" and self.random.random() <= self.model.infection\_prob\_per\_contact \* multiplier:

x.status = "E"

else:

for x in cellmates:

if x.status == "S" and self.random.random() <= self.model.exposed\_infection\_prob\_per\_contact \* multiplier:

x.status = "E"

def recovery\_countdown(self):

if self.status == "E":

if self.exposed\_timeleft > 0:

self.exposed\_timeleft -= 1

else:

self.status = "I"

self.exposed\_timeleft = self.model.exposed\_duration

if self.status == "I":

if self.infected\_timeleft > 0:

self.infected\_timeleft -= 1

else:

self.status = "R"

self.infected\_timeleft = self.infection\_duration

def step(self):

#step\_no는 1, 2, ... / 홀수반 = 1학년, 짝수반 = 2학년

if self.model.split\_opening and (((self.model.step\_no - 1) // len(self.model.timetable)) // 5) % 2 == 0: #홀수주

open\_schedule = 23

elif self.model.split\_opening and (((self.model.step\_no - 1) // len(self.model.timetable)) // 5) % 2 == 1: #짝수주

open\_schedule = 13

else:

open\_schedule = 123 #격주등교 안함

inschool = open\_schedule == 123 or (open\_schedule == 13 and not self.group\_no in range(2,29,2)) or (open\_schedule == 23 and not self.group\_no in range(1,29,2))

m = self.model.timetable[(self.model.step\_no - 1)% len(self.model.timetable)]

#격주등교시 미등교 학년 비활성

if self.model.split\_opening and self.group\_no in range(2,29,2) and open\_schedule == 13:

self.model.grid.move\_agent(self, (139, 60))

elif self.model.split\_opening and self.group\_no in range(1,29,2) and open\_schedule == 23:

self.model.grid.move\_agent(self, (139, 60))

#등교하는 경우

elif inschool:

if m == 'meal':

if self.model.meal\_distanced == True:

self.meal\_distanced()

self.spread\_infection(self.model.restaurant\_multiplier)

else:

self.meal\_normal()

self.spread\_infection(self.model.restaurant\_multiplier)

elif m == 'meal\_cont':

self.spread\_infection(self.model.restaurant\_multiplier)

elif m == 'recess':

#교실 간 이동 확률

if self.random.random() < self.model.visit\_prob\_per\_person:

if not open\_schedule == 23 and self.group\_no in range(1,29,2): # 등교한 1학년

self.reside\_no = self.random.randrange(1,29,2)

self.move\_to\_group(self.reside\_no)

self.spread\_infection(1)

elif not open\_schedule == 13 and self.group\_no in range(2,29,2): # 등교한 2학년

self.reside\_no = self.random.randrange(2, 29, 2)

self.move\_to\_group(self.reside\_no)

self.spread\_infection(1)

elif self.group\_no in range(29,43): # 3학년

self.reside\_no = self.random.randint(29, 42)

self.move\_to\_group(self.reside\_no)

self.spread\_infection(1)

else: #미등교자 무시

pass

else:

#이동 안하면 자기 반으로 배치

self.reside\_no = self.group\_no

self.move\_to\_group(self.group\_no)

self.spread\_infection(1)

elif m == 'recess\_cont':

self.move\_within\_bound(self.reside\_no)

if inschool:

self.spread\_infection(1)

elif m == 'dinner' and self.dinner == True:

if self.model.meal\_distanced == True and inschool:

self.meal\_distanced()

self.spread\_infection(self.model.restaurant\_multiplier)

elif inschool:

self.meal\_normal()

self.spread\_infection(self.model.restaurant\_multiplier)

else:

pass

elif m == 'dinner' and self.dinner == False:

self.model.grid.move\_agent(self, (139,60))

elif m == 'dinner\_cont':

if self.dinner == True and inschool:

self.spread\_infection(self.model.restaurant\_multiplier)

else:

print("This error shouldn't happen.")

self.recovery\_countdown()

class SchoolModel(Model):

def \_\_init\_\_(self, N, N\_per\_group, width, height, initial\_num\_infected,

infection\_duration, exposed\_duration, infection\_prob\_per\_contact, exposed\_infection\_prob\_per\_contact,

restaurant\_multiplier, visit\_prob\_per\_person,

meal\_random, meal\_distanced, timetable, dinner\_percentage, split\_opening

):

super().\_\_init\_\_()

self.N = N

self.i\_count = 0

self.e\_count = 0

self.s\_count = self.N

self.r\_count = 0

self.step\_no = 0

self.N\_per\_group = N\_per\_group

self.timetable = timetable

self.infection\_duration = infection\_duration \* len(self.timetable)

self.exposed\_duration = exposed\_duration \* len(self.timetable)

self.infection\_prob\_per\_contact = infection\_prob\_per\_contact

(생략)

self.exposed\_infection\_prob\_per\_contact = exposed\_infection\_prob\_per\_contact

order = []

if self.meal\_random:

for i in range(1, self.N + 1):

order.append(i)

shuffle(order)

self.order = order

self.grid = MultiGrid(width, height, False)

self.schedule = RandomActivation(self)

if self.split\_opening:

initial\_infected = [self.random.randrange(14\*25, self.N) for \_ in range(initial\_num\_infected)]

else:

initial\_infected = [self.random.randrange(0, self.N) for \_ in range(initial\_num\_infected)]

for t in range(0, self.N):

# (self, unique\_id, group\_no, status, infection\_duration, model)

group\_no = t // N\_per\_group + 1

x1 = ((group\_no - 1) % 14) \* 10 + 1

x2 = x1 + 8

y1 = ((group\_no - 1) // 14) \* 10 + 1

y2 = y1 + 8

a = Student(t + 1, group\_no, "S", self.infection\_duration, self, False)

if t in initial\_infected:

a.status = "I"

if self.random.random() < self.dinner\_percentage:

a.dinner = True

self.schedule.add(a)

self.grid.place\_agent(a, (self.random.randint(x1, x2), self.random.randint(y1, y2)))

self.datacollector = DataCollector(

model\_reporters={"Susceptible": s\_count, "Infected": i\_count, "Recovered": r\_count, "Exposed": e\_count},

agent\_reporters={}

)

self.affected\_collector = DataCollector(

model\_reporters={"Affected": affected},

agent\_reporters={}

)

def step(self):

""" Advance the model by one step."""

self.step\_no += 1

self.i\_count = i\_count(self)

self.s\_count = s\_count(self)

self.e\_count = e\_count(self)

self.r\_count = r\_count(self)

self.datacollector.collect(self)

self.schedule.step()

**do\_experiment 함수 구현**

목적: 모델 실행 한번 한번의 출력값을 적절한 이름의 csv 파일으로 출력해준다.

timetable\_normal = [ (생략) ]

timetable\_extra = timetable\_normal + [ (생략) ]

RESULT\_DIR = "/ (생략) /result/" # 출력 폴더. "/"로 끝나야 한다.

STUDENT\_NO = 1050 # 총 학생 수

def do\_experiment(EXPERIMENT\_NAME, model):

""" RESULT\_DIR/csv/EXPERIMENT\_NAME라는 폴더에 실험 실행 결과의 csv 파일을 저장한다."""

(생략)

**결과 해석용 함수 정의**

**average\_IER:** RESULT\_DIR/csv에서 인자로 받은 EXPERIMENT\_NAME라는 이름을 가진 폴더의 csv 파일들의 평균을 구하여 RESULT\_DIR/average에 저장한다.  
**combine\_same\_hypothesis()**: 여러번 수행한 실험 결과를 한 그래프에 겹쳐서 그린다.  
**combine\_averages():** 여러 가설들의 그래프를 한 그래프로 합쳐서 그린다.

(생략)

**3. 모델 실행**

**가. 마스크의 효과 확인**

no\_of\_experiment = 50

**1) β값이 0.1일 때**

for t in range(no\_of\_experiment):

print(str(t+1) + '번째 실험')

do\_experiment('0.1', SchoolModel(N=STUDENT\_NO, N\_per\_group=25, width=140, height=61, initial\_num\_infected=1,

infection\_duration=7, exposed\_duration=5,

infection\_prob\_per\_contact=0.1, exposed\_infection\_prob\_per\_contact=0.05,

restaurant\_multiplier=3, visit\_prob\_per\_person=0.1,

meal\_random=True, meal\_distanced=False, dinner\_percentage=0.3,

timetable=timetable\_extra, split\_opening=False))

**2) β값이 0.2일 때**

(아래 실험의 대조군과 동일)

**2) β값이 0.3일 때**

(생략)

**나. 거리두기 정책 비교**

no\_of\_experiment = 50

**1) 대조군**

(생략)

**2) 1,2학년 교차 등교를 한다면? -** 가설 1번

(생략)

timetable=timetable\_extra, **split\_opening=True**))

**3) 보충과 야자를 하지 않는다면? -** 가설 2번

(생략)

,**timetable=timetable\_normal,** split\_opening=False))

**4) 교실 간 이동을 금지한다면? -** 가설 3번

(생략)

restaurant\_multiplier=3, **visit\_prob\_per\_person=0,**

**5) 급식 순서를 반별로 먹는다면?**

가설 4번

(생략)

**meal\_random=False**, meal\_distanced=False, dinner\_percentage=0.3,

**4. 결과 분석**

combine\_same\_hypothesis('0.1')

combine\_same\_hypothesis('0.3')

combine\_same\_hypothesis('대조군')

combine\_same\_hypothesis('1번')

combine\_same\_hypothesis('2번')

combine\_same\_hypothesis('3번')

combine\_same\_hypothesis('4번')

average\_IER('0.1', 37) # 37은 하루가 37스텝임을 의미

average\_IER('0.3', 37) # 37은 하루가 37스텝임을 의미

average\_IER('대조군', 37) # 37은 하루가 37스텝임을 의미

average\_IER('1번', 37) # 37은 하루가 37스텝임을 의미

average\_IER('2번', 25) # 25은 하루가 25스텝임을 의미

average\_IER('3번', 37) # 37은 하루가 37스텝임을 의미

average\_IER('4번', 37) # 37은 하루가 37스텝임을 의미

combine\_averages()