Tracking Fresher's Flu using an agent based SIR model A case study of Cambridge University undergraduates

Joshua Prince jp993 SF5

June 2025

Contents

1 Abstract

During the start of every academic year, a virus's paradise is created within the University of Cambridge. Large numbers of students from all over the world, carrying various illnesses that other students haven't experienced, before socialising en-mass - resulting in an explosion of infections. This annual phenomenon, colloquially known as "Freshers' Flu," presents a unique challenge for traditional epidemiological modelling. Standard models often rely on static social networks, failing to capture the critical temporal and spatial dynamics that govern real-world interactions: for a disease to spread, individuals must be in the same place at the same time.

This report details the construction and analysis of an agent-based model designed to overcome this limitation. By generating unique, hour-by-hour schedules for 11,172 undergraduate students over the first four weeks of term, this simulation creates a dynamic network of contacts based on students' academic, social, and personal commitments. By overlaying a classic SIR disease model, this report provides a more realistic picture of how an infectious disease propagates through a dense and highly structured community, identifying superspreaders, key locations, and patterns of the spread.

2 Introduction: Fresher's Flu

During the start of every academic year, a virus's paradise is created. Large numbers of students from all over the world, carrying various illnesses that other students haven't experienced, before socialising en-mass - resulting in an explosion of infections. This is known colloquially as Freshers Flu. Within the University of Cambridge, thousands of new and returning students arrive and mix in close quarters within Pubs, Clubs and lecture halls. This high density of social interactions creates ideal conditions for illnesses to spread - and this report seeks to track the spread of different diseases, and observe some of the outcomes of running a simulated model. All students (except the original carriers) can be assumed to initially be susceptible to infection - which makes the early weeks of term a critical period for studying how infections propagate through a student population.

A common framework for modelling the spread of infectious diseases is the SIR model, which divides the population into three states: Susceptible (S), Infected (I), and Recovered (R). Individuals start as susceptible, and upon contact with an infected person, may become infected themselves. After a period of infection, individuals recover and are assumed to be immune, moving into the recovered state. The dynamics of the SIR model are governed by the rates of infection and recovery, and the model can be extended to include more complex features such as varying contact rates, incubation periods, reinfection or vaccinations.

In an SIR model, probability plays a central role in determining whether a susceptible person becomes infected or an infected person recovers at each time step. The transmission rate β is the probability that a susceptible individual will catch the disease from an infected person during contact. This means that if two individuals , A and B, remain in contact for k time steps, the likelihood of the susceptible person A not catching Bs illness is equal to $(1-\beta)^k$ meaning that infection tends to inevitable provided they stay in contact. The recovery rate γ is the probability that an infected person will recover (and become immune) in a given time step - and similarly to infection, can be viewed as inevitable but does not require A and B to stay in contact.

So far during *SF5 Networks, friendship and disease*, the network of students has been assumed to be static, with a disease spreading across an unchanging network. This report and model seeks to change this. For a disease to spread from one person to the next, they must be co-temporal and co-spatial - or in layman terms, the students must occupy the same location at the same time. This requires generating time-resolved networks of student locations, such as the Engineering Tripos Lecture hall at 9am, the College canteen at 12pm with a trip to The Eagle at 8pm. By overlaying the SIR model on these dynamic networks, we can provide a more accurate picture of how Freshers Flu may spread through a student community.

3 Approach

To create the co-spatial and co-temporal model, I focused on the first four weeks of the Michaelmas term, beginning on Wednesday, 1st October. This period has been divided into 672 distinct, hour-long time steps. A key advantage of building this model from the ground up is the ability to define its parameters with precision. From the outset, I decided on a comprehensive approach, choosing to simulate almost the entire undergraduate population, resulting in 11,172 students being modelled.

For each hour of the four-week period, every student is assigned a predetermined location according to a simulated schedule. This creates a dataset with over 7.5 million time-location parings, increasing to 15 million time-location-disease state pairings while running the SIR model. This detailed agent based approach should provide a detailed basis from which to draw conclusions from both general testing, and also by changing parameters. From removing all STEM Students, to closing Clubs once a week, almost everything that is modelled can be altered to see the effect it has on the spread of Freshers Flu.

4 Building the model

This section is designed as a walk-through of the processes which builds the model. The relevant code can be viewed on a public repository hosted on GitHub[1]. The model generation code is spread over 7 Jupyter notebooks, and though the report subsections do not perfectly align with the notebooks, the chronological order is the same. The order is: Group Production, Social nature generation, Timetable_lectures_meals, budget_creation, Calender_finish, Diseasespread and Disease_analysis. I chose this piecewise structure for two practical reasons. First, it broke the complex design process down into manageable logical steps, and second, it makes it simpler to modify the parameters of any single component later on.

4.1 Student and Group Generation

The first stage was to generate 11172 students. My goal was to make these profiles as realistic as possible, so I based their attributes - subject, college, year, and gender - on real university admission statistics[2] [3].

Every effort was made to make this formational step as accurate as possible - even going as far as generating 4th year students for some subjects - in appropriate proportions, while ensuring that 3rd year Language students were excluded during their year abroad. While I was able to match the distribution of subjects across the different colleges, the gender split within them had to be left generic due to a lack of data, with the obvious exceptions of Newnham and Murray Edwards. I decided to exclude mature students and postgraduates from this model, as there is generally little social or academic overlap between undergraduates and postgraduates - especially during the first 4 weeks of the academic year.

The model assumes that each student has a core friendship group of 4-12 people intended for daily meals and social events. The initial Student Group Formation algorithm, detailed in the appendix, deliberately biased group sizes by student subject to reflect real-world trends, creating smaller average groups for STEM students compared to their humanities counterparts as seen in Figure 1. Additionally, groups formed from older students were biased to be more diverse - as during a students time at university, they are more likely to interact with students outside of their immediate social circumstances. While this bias was intentional, the algorithm produced nearly 2000 groups with unrealistic homogeneity: they were 99.8% mono-year, 15% mono-gender, and had a 60% dominant-subject concentration. To fix this, a post-processing switching algorithm was implemented to directly swap students between groups, targeting more realistic distributions while preserving a 5% inter-college mix.

The selection of students for a swap was highly targeted. To correct a group that was, for instance, too skewed towards one year, the algorithm would identify a student in that group and actively search for a swap partner in another group who differed in year but matched on both subject and gender.

This surgical approach ensured that fixing one homogeneity metric did not disrupt the others, and it successfully reduced the overall metrics to my targets of 97% (mono-year), 10% (mono-gender), and 33% (subject concentration). Despite this post processing, groups of all types still remained - with the odd single-subject, single gender group remaining. The final output assigned each of the 11,172 students with a unique StudentID, a GroupID, and a complete set of attributes: Subject, Year, College, and Gender.

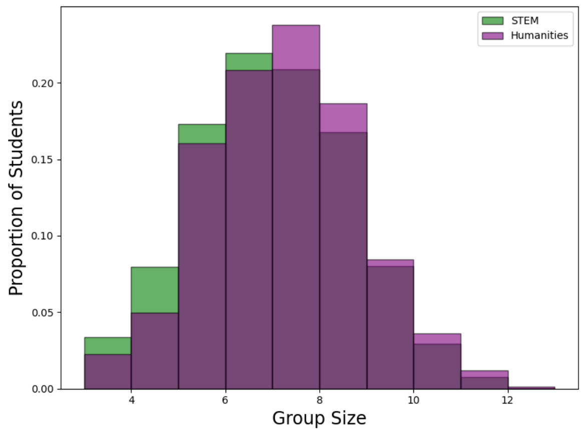

Figure 1: Histogram of Group Sizes - STEM vs Humanities

To model a spectrum of personalities, each student was assigned 'Social' and 'Club' scores - representing the extroversion of the student. These scores were calculated by taking a random base value between 3 and 8 and adding a unique modifier derived from the student's year and group size. Since group size is already linked to a student's subject, this system meant humanities students were, on average, more sociable than their STEM counterparts. However, the random base value ensured a realistic variety of students, allowing for a range of students from a highly social engineer to an introverted geographer.

4.2 Creating the Schedules

4.2.1 Lectures and Meal Slots

To initialise each students hourly timetable for the month, I decided to place their academic contact hours and meal times first- as these are generally immovable. Using a Pandas data frame, an empty timetable with each entry reading 'Free' was formed. Firstly, lectures were assigned in repeating patterns starting from the 8th of October - the beginning of Week 1 in the Cambridge academic calender. To more accurately model academic schedules, lecture allocation was weighed by both subject and year, with STEM students and first-years being assigned more contact hours than humanities students and finalists.

Whether or not a student would go to a canteen depended on both their year - with first years being more likely - and whether the rest of the students social group was attending. Lunch and Dinner could then be allocated to an hours slot at 12:00 or 13:00, and 17:00, 18:00 or 19:00 respectively.

4.2.2 Budget creation

The activities of studying, socialising and exercising were initiated through a budgeted approach. For each activity, a total hourly budget was calculated for the month - which allowed a 'blockwise' allocation algorithm to be used later in the process. The social budget was calculated by assuming that 1st years would socialise for 30 hours in their first week, then 20 hours for the subsequent weeks - with 2nd-4th years socialising for 20 hours in the first week, and then 15 hours per week later on. The budget was then scaled with the relative size of the students combined social and club propensity scores.

A study budget was generated for each student to account for time spent on independent work. This was based on a normally distributed weekly average of 45 working hours, ensuring a realistic range of workloads. To calculate the final number of self-study hours, each student's total lecture time was subtracted from their total budget. Similarly, there is a range of athleticism across the student population with each student assigned a weekly budget that is scaled by 4 - with 10% of students doing 1 hour a week of 'Outdoors' (e.g. exercise), 40% doing 4 hours a week, 40% doing 12 hours a week and the remaining 10% of students - no doubt blues athletes or rowers undertaking 20 hours a week. This information is all stored in a CSV file, containing the students information, their sparse timetable, and their activity budgets.

4.2.3 Blockwise allocation

In order to allocate social time from the social budget, three general rules are followed. One: Social sessions tend to occur in time blocks, and are rarely scattered across a day. Two: Students choose to socialise with their group, so should be more likely to socialise if their group is also socialising and Three: Socialising mainly occurs in the evening, but can start as early as 6pm and extend until 3am if the students are going clubbing. To achieve this, the Group-Aware Social Block Allocation algorithm (see Appendix) is followed. The blockwise allocation begins by scheduling social time, with weekly budgets (33% of the monthly budget for Freshers' week, 22% for others) broken into normally distributed 1-6 hour blocks. These blocks are then synchronized within friendship groups, primarily in the evening between 6pm and 3am, to maximize overlap.

Next to be allocated is sleep; Sleep is distributed for each student by first assigning them a personal sleep requirement, which is drawn from a normal distribution centred around 7 hours per night, but clipped between 6 and 10 hours. For every day in the timetable, the algorithm determines the earliest bedtime for each student, typically around 10pm, but with some random variation to reflect real-life habits. If the student is socialising that evening, sleep is scheduled to start after their last social activity of the night. The algorithm then allocates a continuous block of sleep, aiming to meet the student's required hours, and allows for "sleeping in" if there is free time in the morning. If the student's overnight sleep does not meet their requirement, the remaining hours are made up with naps during free periods later on in the day. This process results in a realistic spread of bedtimes and awake times across the student population, with most sleep occurring at night but with enough flexibility to accommodate late social events and individual differences.

Study blocks are then allocated by splitting the study budget into blocks similarly to social slots - though only lasting between 1 and 4 hours. These blocks are then assigned to available 'Free' slots within the students timetable, with the aim of fulfilling as much of their budget as possible - by splitting apart blocks if necessary. In an identical fashion, Exercise time is allocated by splitting a students outdoor budget into 1-3 hour long blocks - and then slotted into the timetable, spreading out exercise across the week. Finally 2-3 trips to the supermarket per week are allocated to each student - with randomised timings between 8am and 10pm - the opening hours of most real world supermarkets.

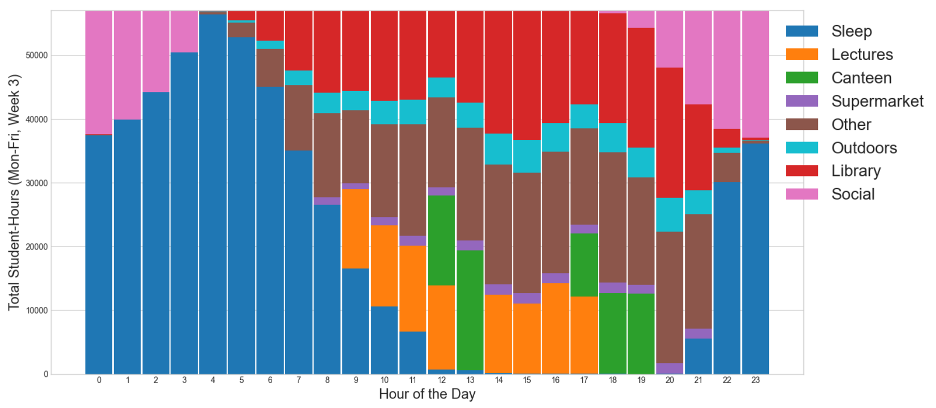

Figure 2: Activity Distribution by Hour (Mon-Fri, Week 3)

With all remaining hours scheduled as 'Other' - meant to represent solo hobbys, admin time or leisure - the average distribution of activity slots can be viewed in Figure 2. Here, it is clear to see that most students sleep at night, with social occasions dedicated to the evenings, and a range of activities filling up schedules during the day. Within the GitHub repository, is a coloured timetable[4] for the first 15 groups - allowing the block and group wise allocation to be viewed easily.

4.3 Specific location allocation

Now that each student has a general timetable, it is time to enter specific locations. Using a database of over 200 locations in Cambridge[5], each with their coordinates, activity locations could be chosen through a function of a students' college, year, subject or previous location.

For each student, every scheduled supermarket trip is assigned a specific location. By default, 80% of students go to "Sainsbury's (Sidney Street)" for their shopping trips, unless they are from Girton or Homerton colleges. Students from these two colleges, as well as the remaining 20% of students from other colleges, are assigned to the supermarket that is geographically closest to their college. This approach ensures that most students use the main city supermarket, while those further out or a minority from other colleges use a more convenient local option. The overall supermarket model proves quite accurate - with an average of 230 students going to Sainsbury's (Sidney Street) each hour during opening hours, which seems like a reasonable number.

Canteen locations are assigned based on a mix of group dining and proximity. When a student is scheduled to eat, there's a 70% chance they'll join their group if other group members are also eating in a canteen at the same time. If the student has just come from a lecture, there's a 50% chance they'll eat at the canteen closest to their subject building and a 50% chance they'll eat at their college canteen. If they haven't just come from a lecture, 90% of the time they'll eat at their college canteen, and 10% of the time they'll pick a random canteen within about 2 km of their college. This approach favours a students current location, college and group location - which should create an accurate mapping of mealtime locations.

Most students spend their study hours either in their college library or their subject library, with a small proportion of third- and fourth-year students (about 10%) using the University Library during the day (9am–7pm). Some students are "room workers" and will do a portion of their library time in their own room instead. Additionally, about 20% of students spend half of their daytime library hours (9am–5pm) in a nearby café instead of a library. Once a student's library preference is set, they tend to use the same library for 90% of their library hours.

Social locations are assigned in a way that reflects both group behaviour and the typical progression of a student night out in Cambridge. For each hour where a student is marked as 'Social', the algorithm first checks which other members of their social group are also socialising at the same time. All group members who are social in a given hour are assigned to the same location, ensuring that friends spend their social time together. The choice of location depends on the students year group, time of night and day of the week.

At the start of the evening, most first-year students (about 75%) begin their night in their college bar, while students in later years are more likely to start in a pub. College bars are chosen based on the majority college of the group, and 95% of the time, the group will use their own college bar; otherwise, they might visit a nearby college bar. For groups starting in a pub, the location is randomly selected from the five pubs closest to the group's college. Once a group has settled in a bar or pub, they generally remain there for the duration of the social block - by enforcing 'stickiness' unless the night progresses late enough for a move to a club.

If the social block extends to 11pm or later, the group may move to a club: 30% of groups make this transition at 11pm, while the rest move at midnight. 50% of the time however, students that could go to a club, choose not to - and go to 'Afters' in a students room. Here, disease can only spread between members of the same group and the students will remain here until they go to sleep. The specific club is chosen based on the day of the week - which reflects the Cambridge clubbing scene, with Wednesday Revs, Friday MASH, Sunday Lola's and occasionally a Thursday Vinyl. This results in a realistic social progression during an evening - with each stage of the night generated clearly.

4.3.1 Accuracy of the schedule

A subjective analysis of the generated location timetables[6] suggests that that they are approximately 95% realistic. While minor inconsistencies appear — such as illogical travel from a nightclub to a college bar at 2 a.m.— most students exhibit well-defined daily structures with plausible variability. However, a key area for future refinement is reducing the variability throughout the day. Achieving a more nuanced student behaviour would require iterative `For` or `While` loops, a method that proved computationally infeasible, with runtimes estimated in hours or days. Consequently, I prioritized performance by using vectorized Numpy operations. This approach reduced execution time to seconds or minutes, but at the cost of the specific logic needed to perfect a students schedule.

4.4 Disease Spreading

We model the spread of diseases by initialising 20 students to have a set of diseases with different transmission and recovery rates, near the start of the timetable. T = 12 is chosen to avoid the irregular timetable that occurs at the start of the model. The code can model any number of diseases - with a disease state associated (0 = S, 1 = I, 2 = R) with each disease assigned to the student. For example if a student has recovered from A and C, is infected with B and is susceptible to diseases D and E, their code will be 21200. Using a simple SIR model, disease spreads throughout the student population. This results in over 15 million entries within the spreadsheet - as each student has a location and disease state for each time step.

For every hour, the model checks where each student is — whether that's a lecture hall, club, canteen, or any other location. If a susceptible and an infected student are in the same place at the same time, there's a chance the disease will be transmitted, based on how infectious the disease is and how many[7] students an infected student comes in close contact with. This contact number is an estimate - and can easily be varied. Recovery is also handled probabilistically: each hour, infected students have a chance to recover, moving to the "recovered" state. All infection events are logged, so that the spread can be analysed in detail. The result is a simulation of how Freshers Flu could spread through a real student population, capturing both the timing and the locations of every possible contact.

4.5 Model Critiques

At its heart, the model is just the outcome of guided random probability - and as such, can be guided to show certain outcomes. This results in the model displaying what it is designed to show - such as STEM students spreading disease less - though this could easily be altered by changing a single parameter. Despite this, if the model could be constructed from real world data of student group and interaction counts, it would provide more reliable analysis. This model has been created based upon my unique perspective of University of Cambridge Undergraduates' social experiences - as well as plenty of estimations, but I believe the model has a strong set of foundations from which it can be improved.

Additionally, the model fails to account for how a student will change their behaviour if they are ill. It assumes that a timetable is set in stone, and that the disease state won't vary their future locations. This could be improved by updating the timetable depending on the disease condition - such as reducing the social hours, increasing sleep and removing all exercise hours.

Another major limitation is the assumption that all students have one core social group - in reality, students often span several groups - if a group even exists at all. Furthermore, its questionable to assume that a student will prioritise a social over sleep, and sleep over a library session, as the time allocation order indicates.

5 Analysis of Infection spread

The complete model can now be analysed how ever the user sees fit. A few analytical examples have been delved into within the section, though adjusting base parameters, and observing how the model changes was not investigated. Further investigation and optimisation of the model could easily pursue this line of research.

5.1 Superspreader analysis

The existence of superspreaders was investigated - with a superspreader being a student who infects an abnormally large number of other students. By analysing superspreaders' subject distribution in Figure 3, it can be noted that Natural Science, Engineering and Medicine students are most likely to be superspreaders - more then their larger student population sizes would indicate.

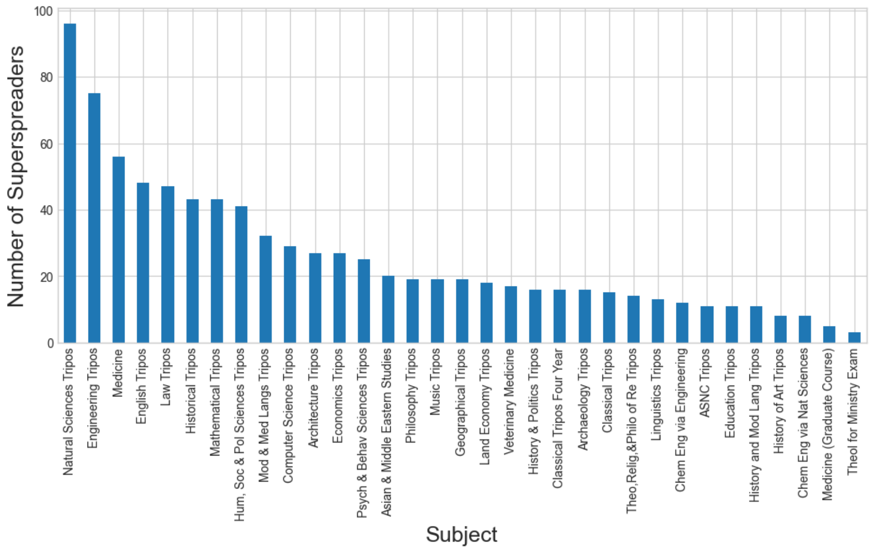

Figure 3: Top 20% Spreaders' Subjects ($\beta = 0.1$, $\gamma = 0.01$)

StudentID	Subject	Year	College	Total Infections	Most prolific location
5279	HSPS	2	King's	13	King's Canteen (6)
5665	History & Politics	1	Lucy Cavendish	13	Lucy C. Library (6)
1736	Engineering	2	Downing	13	Downing Library (9)

Table 1: The most prolific spreaders

The disease tracking aspect of the model allows individual superspreaders to be revealed - with Table 1 indicating the students who infected the highest other students for one simulation with a transmission rate of $\beta = 0.1$, and recovery rate of $\gamma = 0.01$.

5.2 Locations of maximum spread

It was quickly noted that the locations where diseases spread the most were those where people spent the longest. Due to the time allocation of the model, this ended up being college libraries. Plotting the number of infections in a location against the total number of hours spent in this location indicated that the Close contacts database[7] successfully transferred a contagion bias towards night clubs and pubs - while reducing that of Supermarkets and subject libraries as can be seen in Figure 4. Further investigation into the surprising infectiousness of College libraries is required - it likely as a result of a bug in the code.

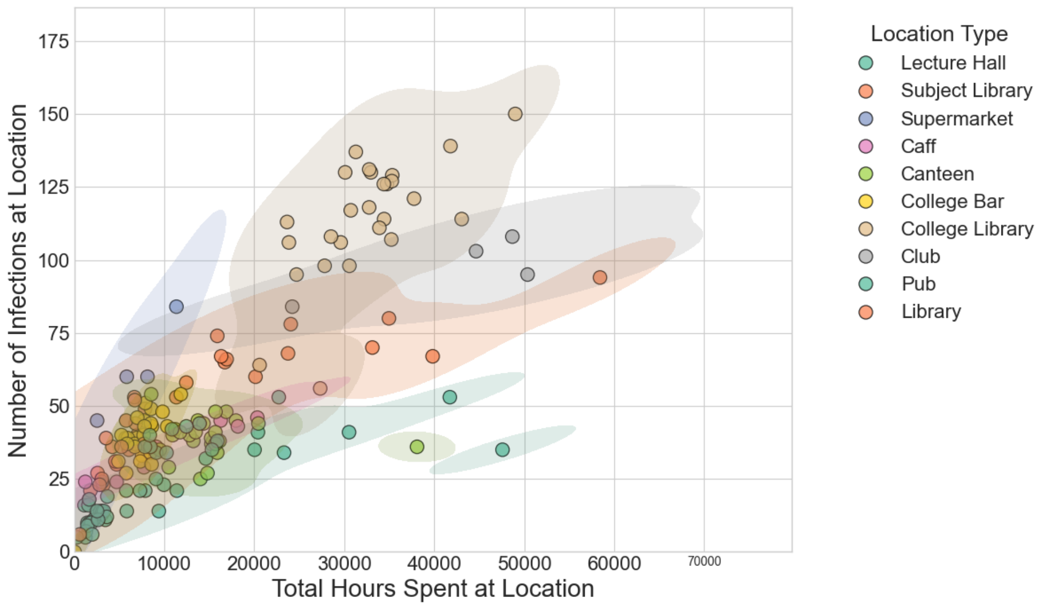

Figure 4: Infections vs. Hours at Location

5.3 Mapping the spread through subjects and colleges

By tracking the spread of Freshers Flu across colleges and subjects, it is possible to track the individual outbreaks which are relatively confined to this local network. For example, in Figure 6, ASNC students are suffering from a major outbreak, while HSPS students remain relatively unscathed. Likewise, in Figure 5 Newnham college is at the peak of its outbreak, just as Homerton's is fading away.

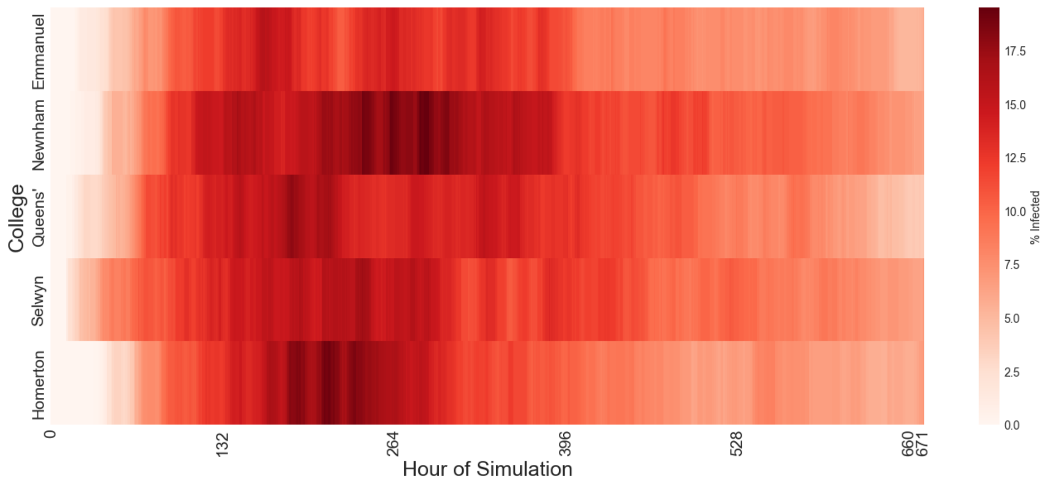

Figure 5: Heatmap of % infected in each college over time ($\beta = 0.1$, $\gamma = 0.01$)

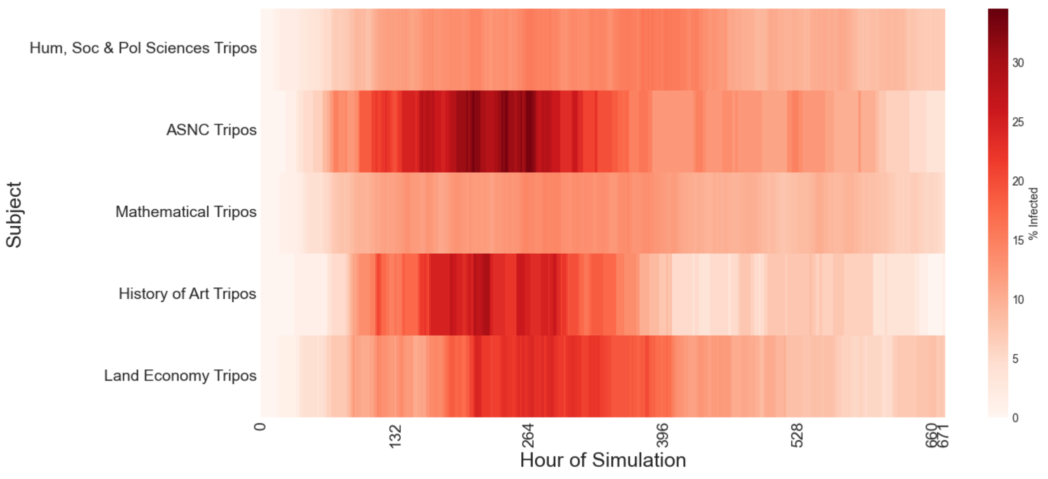

Figure 6: Heatmap of % infected in each Subject over time ($\beta = 0.1$, $\gamma = 0.01$)

5.4 Effect of transmission rate

The ability for the model to cope with a seemingly unlimited number of diseases at any time allows for easy analysis on the effect of transmission and recovery rates on infection numbers as seen in Figure 7. As expected, higher transmission rates result in higher numbers of infected students - but interestingly, the transmission rate β does not influence the number of students infected exactly 4 weeks after initial infection with the blue & orange, and purple & brown lines crossing at this point. Extending the model in time could provide insight into this being a crossover point, or a convergence to shared values. I expect this is a crossover, as the more contagious diseases are starting to be impacted by Herd immunity within the student population whereas the less contagious ones are not. An interesting difference between these plots, and regular SIR models, is the cyclic nature of these infections - where infection rates rise at different times during the day, and fall during others. This promotes investigating the effect of closing night clubs - key disease spreading hotspots - to extend the overnight decay in infections, thus reducing the extent of Freshers Flu.

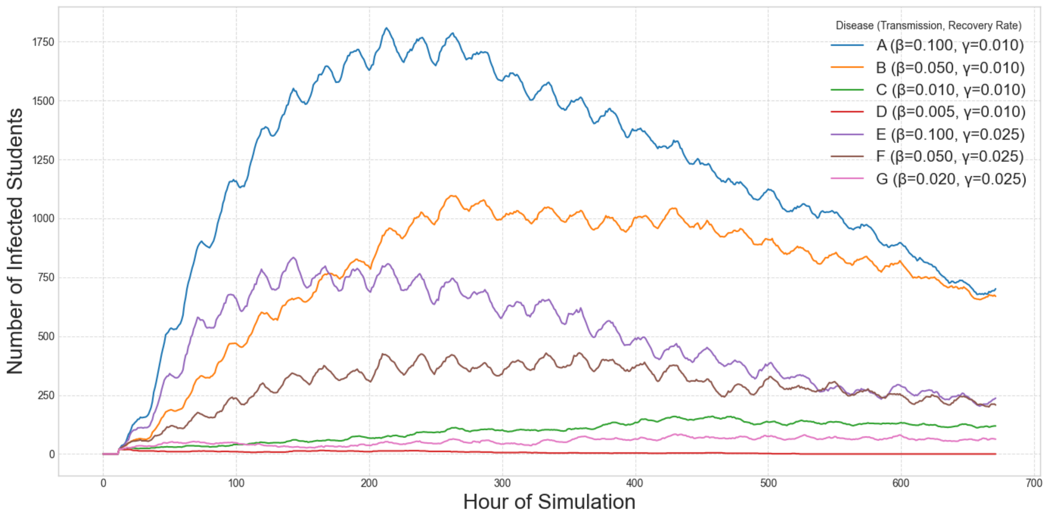

Figure 7: Number of Infected Students Over Time

6 Model Expansions

As the model has been built from the base up, and every parameter is variable, there is great scope for expanding the model. As just suggested, policy actions such as club closing, encouraging mask wearing (thus reducing the number of 'close contacts' in each location) or group isolation could easily be investigated. On top of the standard SIR model, the next step would be to include decaying immunity, which increases the population size susceptible to infection. Additionally, vaccination strategies could be tested, by either vaccinating a random distribution of students (which sets their disease status to '2 = R') or selectively vaccinating specific groups.

A further expansion of this model could include modelling lingering infection through surfaces though this adds an additional layer of complexity. This could be modelled by assigning locations with a lingering disease state - which increases if infected students were present, and can therefore infect future students. The expansion that would make the greatest impact to the model would be dynamically updating a students location schedule based upon their disease state - this would dramatically reduce the spread of diseases, as sick students would be less inclined to socialise.

7 Conclusions

This project successfully built a high-resolution agent-based model by assigning detailed hourly schedules to over 11,000 students to simulate disease spread. The model demonstrated that superspreading was linked to a student's subject, that transmission hotspots were primarily determined by the total time spent in a location, and that daily infection cycles emerged naturally from the population's timetables.

The primary limitations are the use of static social groups and the assumption that sick students do not change their behaviour. However, the model provides a robust foundation for testing practical "what-if" scenarios, such as the impact of closing specific venues or implementing targeted vaccination strategies.

References

[1] J. Prince. *Freshers Flu GitHub repository.* `https://github.com/JPrince321/FreshersFluCambridge`. [Online; accessed 10-June-2025]. 2025.

[2] University of Cambridge. *Undergraduate Admissions Statistics 2022 cycle.* URL: `https://www.undergraduate.study.cam.ac.uk/files/publications/undergrad_admissions_statistics_2022_cycle.pdf`. [Accessed: 08 June 2025]. June 2023.

[3] Harrington Howard. *Number of students enrolling at a particular subject at a Cambridge College for both undergraduate and graduate.* WhatDoTheyKnow. Freedom of Information request to the University of Cambridge. Available at: `https://www.whatdotheyknow.com/request/number_of_students_enrolling_at` (Accessed: 08 June 2025). 2022.

[4] J. Prince. *Freshers Flu Cambridge Simulation Data: Example Coloured Timetable.* `https://github.com/JPrince321/FreshersFluCambridge`. [Online; accessed 10-June-2025]. 2025.

[5] J. Prince. *Cambridge Locations data.* `https://github.com/JPrince321/FreshersFluCambridge/blob/main/Location_location.csv`. [Online; accessed 10-June-2025]. 2025.

[6] J. Prince. *Freshers Flu Cambridge Simulation Data: Christ's College Timetable with Specific Locations.* `https://github.com/JPrince321/FreshersFluCambridge/blob/main/ExampleTimetable-Christs-specific-loc.xlsx`. [Online; accessed 10-June-2025]. 2025.

[7] J. Prince. *Freshers Flu Cambridge Simulation Data: Close Contacts per hour database.* `https://github.com/JPrince321/FreshersFluCambridge/blob/main/Location_popularity.csv`. [Online; accessed 10-June-2025]. 2025.

8 Appendix

Algorithm 1 Student Group Formation

```
1: Input: CSV file with student metadata: Subject, College, Year, Gender, Adapted Group Size
2: Load student data into students_df; initialize Assigned column to False
3: Initialize groups = [], start_time = current_time()
4: while there are unassigned students do
5:     Select first unassigned student as base_student
6:     Set target_group_size = max(3, round(base_student.AdaptedGroupSize))
7:     Initialize group_members = {base_student_index}
8:     Extract base_student's attributes (Subject, College, Year, Gender)
9:     Compute current unassigned pool and subgroup sizes by year, college, and subject
10:    for Tier in {1: same year & college, 2: same year, 3: same college} do
11:        if group is full then
12:            break
13:        end if
14:        Filter candidates based on tier rules and unassigned status
15:        For each candidate:
             Compute probability based on alignment with base student attributes
             Apply diversity boost/penalty if group is mono-subject
16:        Sort candidates by descending probability
17:        Select top candidates to fill group up to target_group_size
18:        Add selected candidates to group_members
19:    end for
20:    if group still not full then
21:        Randomly select from all remaining unassigned students
22:        Add up to required number to reach target group size
23:    end if
24:    Set final_group_size = max(3, len(group_members))
25:    Mark members as assigned in students_df
26:    Assign Group ID and Group Size to each member
27:    Append group info to groups list
28: end while
29: Save students_df to base_group_formations.csv
```

Algorithm 2 Group-Aware Social Block Allocation

```
Input: List of students with Group IDs, total social budgets, and empty timetables
Input: Weekly allocation fractions (e.g., [1/3, 2/9, 2/9, 2/9])
                                      ▷ Phase 1: Generate individual social block requirements
for each student do
    Split total_social_budget into weekly_budgets using fractions
    Initialize student.social_blocks = {}          ▷ A dict to hold blocks for each week
    for each week from 1 to 4 do
        budget_for_week = student.weekly_budgets[week]
        student.social_blocks[week] = []
        while budget_for_week  0 do
            block_length = normal_integer(1, 6)
            Append block_length to student.social_blocks[week]
            budget_for_week -= block_length
        end while
    end for
end for
                                      ▷ Phase 2: Schedule blocks by prioritizing group overlap
for each group do
    for each week do
        Initialize group_weekly_schedule = []          ▷ Tracks scheduled slots for scoring
        for each student in group do
            for each block_length in student.social_blocks[week] do
                Generate a set of candidate (day, start_hour) slots for the block
                best_slot = null, max_score = -1
                for each candidate_slot do
                    score = calculate_group_overlap(candidate_slot, group_weekly_schedule)
                    if score > max_score then
                        max_score = score, best_slot = candidate_slot
                    end if
                end for
                Store best_slot for this specific block for the student
                Add the hours covered by best_slot to group_weekly_schedule
            end for
        end for
    end for
end for
                                      ▷ Phase 3: Finalize timetables and save
for each student do
    for each scheduled_block with its stored (day, start_hour) do
        Mark the corresponding hours in the student's timetable DataFrame as "Social"
    end for
end for
Save updated student timetables to 'students_social_filled.csv'
```