Title
COVID-19 Transmission in a Resource Dependent Community with Heterogeneous Populations: An Agent-Based Modeling Approach

Author Information
Aaron D. Wood[1] and Kevin Berry[2]

[1]Department of Economics
John H. Sykes College of Business
The University of Tampa
401 W. Kennedy Blvd., Box O
Tampa, FL USA 33606
ORCID: 0000-0002-3630-4263

[2]Department of Economics
University of Alaska Anchorage
3211 Providence Dr
Anchorage, AK USA 99508
ORCID: 0000-0002-6904-4349

Corresponding Author
Kevin Berry <kberry13@alaska.edu>
Cell: 907-786-4185

Keywords
4 to 6
Agent-based modeling; resource dependent community; fishery; heterogeneous populations; virus transmission; public health; public policy

Acknowledgements
The authors would like to thank Christopher Wallace for helpful feedback on this manuscript.

Statements and Declarations
The authors have no relevant financial or non-financial interests to disclose that are relevant to the content of this article.

Funding
This work was supported by NSF OPP/GEO #2032787 and NSF OPP GEO #1745508 and RISE/GEO #2022876

Abstract
Outbreaks of COVID-19 in crowded work locations led to "superspreading" events during the pandemic that stressed health capacity in rural communities. This led to disparate responses – either isolating and restricting workers to facilities and potentially amplifying spread between them, more intense community wide restrictions, or an acceptance of higher disease spread. An extreme case is the salmon fishery in Bristol Bay, Alaska, where fishermen, factory workers, and residents all interact during the summer fishing season. During the pandemic, policy measures were debated, including community mask mandates, restricting workers to their boats and factories, and even closing the valuable seasonal fishery.

We develop an agent-based SIR model (ABM) to examine COVID-19 transmission in a resource-dependent community populated by distinct subgroups. The model includes a virus spreading within and between three heterogenous populations who interact with other members of their type in their home location, and with different types of agents when out in the community. We simulate various non-pharmaceutical interventions and vaccination rates across these groups. Results demonstrate the efficacy of non-pharmaceutical interventions and vaccinations, as well as tradeoffs between duration and intensity and tradeoffs between groups impacted by the outbreak. This ABM demonstrates the impact of public policy mechanisms on health outcomes in resource-dependent communities with distinct populations.

1. Introduction

" About eighty known died at Naknek. Adult population practically wiped out… Nurses required to handle orphans.
Will advise later what funds are required."

– aerogram to Dr. French, US Commissioner at Dillingham, from Alaska Packers Association Superintendent JF Heinbockel, June 8, 1919

Bristol Bay, Alaska is home to the world's largest wild salmon fishery, and it hosts processing factories and seasonal fishermen that dramatically increase the population in the summer to harvest returning fish. Bristol Bay communities also had a brutal experience with the 1918 Influenza, which disproportionately killed Alaska Natives and punished local communities (deValpine 2015). This experience, and the knowledge and memory held by both the community and industry, lead to dramatic policy actions in response to the 2020 COVID-19 pandemic intended to avoid a repeat of history. Additionally, policymakers were concerned with maintaining a successful sockeye salmon fishery. Fishing is a major part of the economic base of Bristol Bay, and the sockeye fishery is incredibly time sensitive – the majority of the action occurs when the anadromous salmon return to fresh water to spawn over roughly one month in the summer (McKinley Research Group 2021). There is also evidence that COVID-19 imposed a greater burden on remote and rural areas (Armillei et al. 2021).

The fishery directly provides labor income to residents through both commercial fishing and recreational fishing, as well as directly providing food through a subsistence fishery (McKinley Research Group 2021). The extreme seasonality of salmon fishing in Bristol Bay means that an influx of seasonal workers temporarily live in these isolated communities which are disconnected from the wider road system. During their stay in the community, their spending is critical to local businesses focused on serving this workforce. However, all of these out-of-state workers are potential carriers of COVID-19.

The community therefore faces an extreme version of the tradeoff faced by most communities where restrictions and lockdowns early in a pandemic save lives (Amuedo-Dorantes et al. 2021; Cooper et al. 2023; Prakash et al. 2022) but lead to economic costs (Bognanni et al. 2020). Closing the fishery during the pandemic would have protected public health, but it would have caused irreparable economic harm. Keeping the fishery open brings income, and with it disease and additional susceptible people to an isolated and medical resource-constrained community. Potential policy responses, like isolating fishery workers together away from the community in a factory or on a boat also potentially amplify the spread. The effect of similar superspreader events on transmission has been studied previously, and there is evidence that non-pharmaceutical interventions (NPI) may be more necessary and effective when targeted at these events (Althouse et al. 2020; Sneppen et al. 2021). Factory workers live in man camp arrangements where they dine communally and work in close contact with others. There is no way to socially distance on a 32-foot fishing vessel with a crew of 5 people. Previous work has found that similar conditions in meatpacking plants amplified the spread of COVID-19 (Saitone et al. 2021) and that social habits influence the number of cases and deaths (Cristini and Trivin 2022). Work on the impact of weather also suggests that crowded spaces amplify spread (Yakubenko 2021). We also abstract away from the impact on absenteeism in production (Araya 2021). It is necessary to examine the impact of NPI in a scenario where economic interactions influence spread (Murray 2020). Oversimplified models that ignore the differences between economic agents may also miss an important mediating force on pandemic dynamics (Nishi et al. 2020).

Prior work has examined the impact of targeted lockdowns on different age groups to examine the overall economic cost of extreme mitigation (Acemoglu et al. 2020, 2021; Bárcena-Martin et al. 2022). This work finds targeted policies are more effective at reducing the cost of NPI. We also respond to calls to incorporate heterogeneity by occupation and location as well as vaccine treatments instead of NPI (Arias 2021). We expand upon previous work like Bisin and Moro (2020, 2022) to examine the impact of restricted movement in remote rural areas. These areas often face limited healthcare resources and are particularly vulnerable to COVID-19 and other pandemics (Savage et al. 2020).

Our paper explores potential policy responses to the pandemic and details an agent-based model (ABM) we developed to better understand the dynamics of an infectious disease when policies are targeted to various subgroups of the population. In general, an ABM consists of a program, typically written in an object-oriented language, that details an agent type or types, which includes potential traits or decision rules that govern behavior of members of a type. The ABM then uses computational power to create individual agents from a type, simulate their behavior and interactions with other agents of their type, other types, or both, and derive emergent, large-scale outcomes. While ABM has become an increasingly applied tool in several disciplines, much early

work in ABM centered on finance and transportation, and this literature is reviewed by Axelrod and Tesfatsion (2006) and Chen (2012). Related to the present study, other work has used ABM to examine behavior in resource-dependent environments (Wood et al., 2016) or to predict the efficacy of interventions in data rich environments (Tatapudi et al. 2020). These models have the potential to predict the efficacy of interventions (Gans 2022). Other contemporary work on COVID-19 transmission examines the impact of distancing on a population (Silva et al. 2020), or of transmission within a facility (Cuevas 2020).

We incorporate community structural detail and different transmission rates within and between four different locations – fishing vessels, the processing factories, households, and community locations. We distinguish between three types of people – fishermen, factory workers, and household members. We are able to draw these distinctions because of the unique characteristics of the fishery – because it is a time sensitive regulated open access fishery, even residents of Bristol Bay who engage in commercial fishing are distinct from household members, in that they are likely to be fishing for the preponderance of the season. Additionally, we are able to examine policies both before and after the introduction of a vaccine that reduces transmission and find that policies have differential effects before and after introduction (Makris and Toxvaerd 2020).

2. Geographical and Mathematical Models

We develop an agent-based SIR (susceptible, infected, removed) model consisting of three types of agents that interact within a model of distinct locations and heterogenous populations.

2.1 Heterogeneous Populations Community Model

Agents in the heterogenous populations model can be fishermen, factory workers, or household members. Each group has a different home location during the fishing season. Fishing boats are the home location for fishermen, processing factories are the home location for factory workers, and households are the home location for household members. Within each home location, there are subunits that represent individual households, fishing boats, or processing factories. These subunits vary in size by agent type. All agents sometimes visit the community, where they come into contact with members of other groups. The model is represented in Figure 1. We model agent movements on a daily time scale. Every day, agents wake up in their home location. With some probability they then either remain in that location, or they move to the community. After their location for that day is determined, they interact with all other individuals in that location. For example, a fisherman starts the day on their fishing boat. With some probability they either remain there, or they go to the community. If they go to the community they then interact with any other fishermen, factory workers, and household members who also went to the community that day. If they remain on their fishing boat, they only interact with other members of that crew.

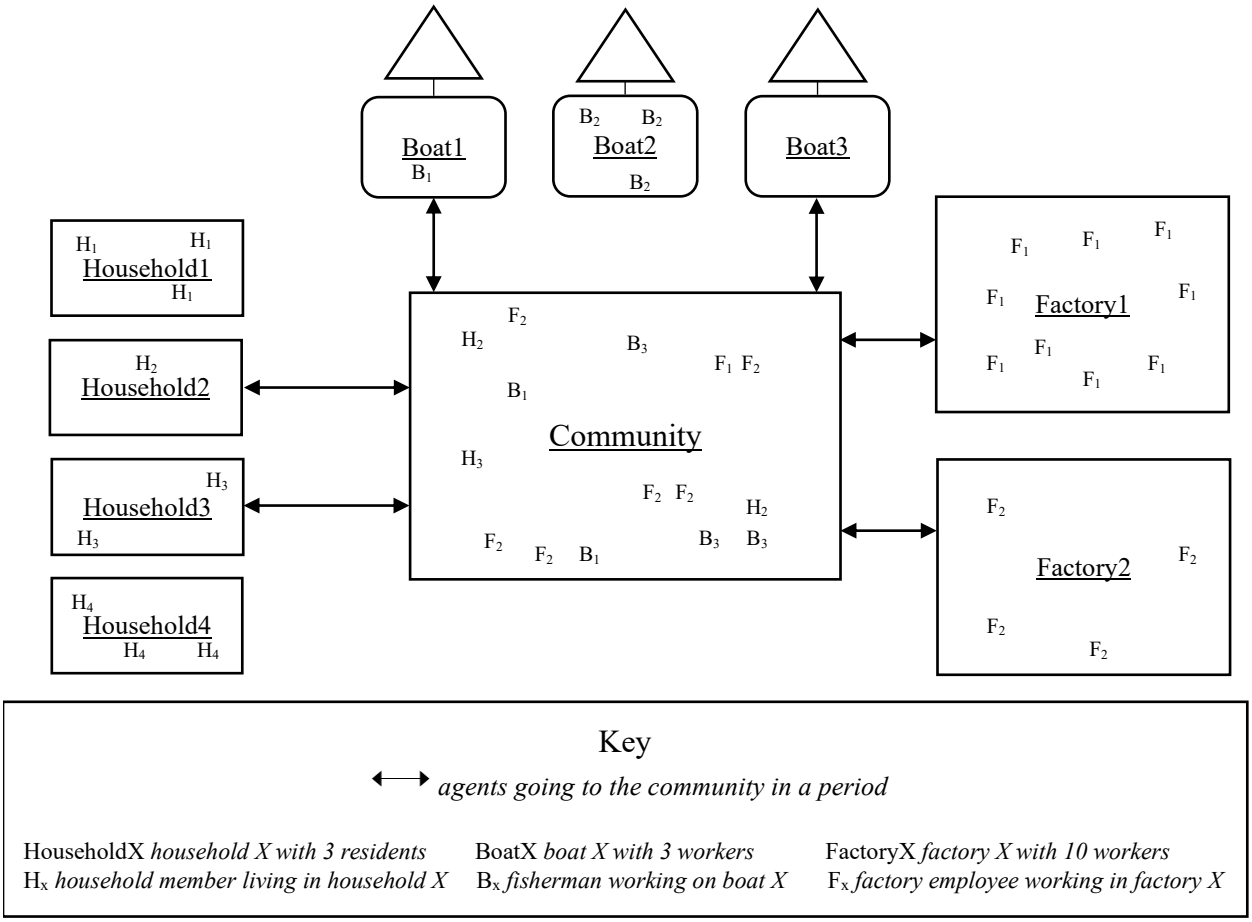

Figure 1: Community model

2.2: SIR Model

Figure 2 provides the details of the infectious disease model. Within any location we develop a daily SIR model where we track the health status of every agent. Once it is determined who is in any given location, we determine if there are any infected individuals in that location. We denote individuals as being susceptible S, infected I, or removed R, which includes both those with post infection resistance and mortality. Individuals are indexed by their type $i \in \{f, p, h\}$ where f denotes fishermen, p denotes processing factory workers, and h denotes household members as well as their location $j \in \{boat, factory, home\}$. Simplified versions of the equations of motion are provided in equations 1-3.

1) $\dot{S}_{i,j} = -\beta_j S_{i,j} \frac{I_{i \forall i,j}}{N}$

2) $\dot{I}_{i,j} = \beta S_{i,j} \frac{I_{i \forall i,j}}{N} - \gamma I_{i \forall i,j}$

3) $\dot{R}_{i,j} = \gamma I_{i \forall i,j}$

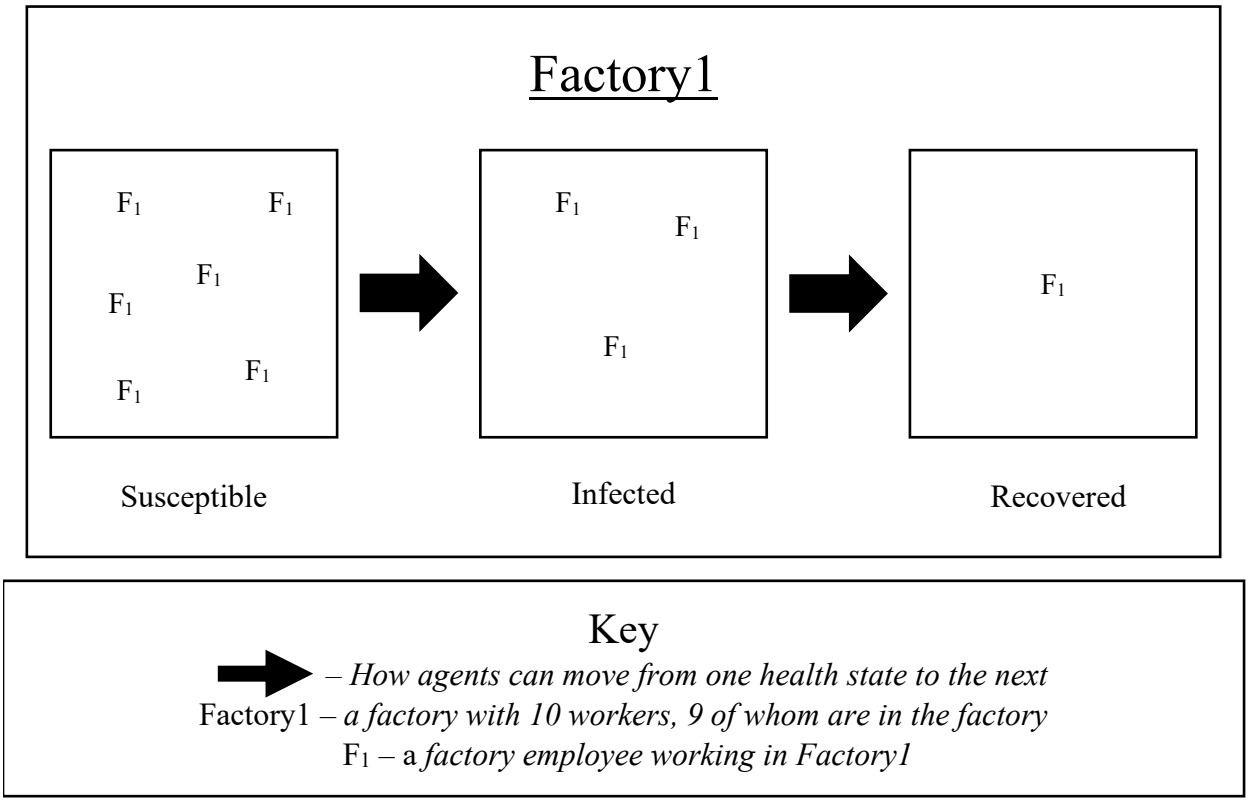

Figure 2: Daily SIR infectious disease model in a factory

Susceptible people in group i become infected at rate β_j subject to coming into contact with infected individuals. The probability of any individual contact being with an infected individual depends upon the proportion of infected individuals in each location, regardless of their type i, or $\frac{I_{i\forall i,j}}{N}$. Individuals then are removed from the infected population $I_{i\forall i,j}$ at the rate γ when they either recover and have immunity or become deceased. The typical season length is short enough that reinfection does not occur.

3. Agent Based Model and Simulation Scenarios

3.1 Agent-Based Model

The agent-based model (ABM) consists of the community and SIR models of agent interaction and public health policy. It provides detailed results of virus transmission in agents within and between heterogeneous populations. The program was written in C++ and tested and debugged in Microsoft Visual Studio 2019 Version 16. Detailed pseudocode is provided in Appendix A to exhibit transparency in program design.

When the program begins, the user is prompted to initialize each parameter and probability relevant to the virus, each agent type, and every public policy mechanism in the ABM. Stay-at-home orders for each type of agent are captured by the probability that an agent of a particular type leaves their home location and goes into the community on a given day. For instance, if the value

for households is 28.5, a household member agent will leave their household and go into the community twice per week in an average week (28.5% of days involve a trip out of the household).

The program also allows initial values specific to each agent type including the number of agents in each population, the size of home locations, the number of initially infected agents, and the number of initially vaccinated agents for each of the fishermen, factory workers, and household members. Infected agents are randomly assigned a number between 1 and γ for how many periods they will remain contagious before they attain removed status.

After parameters are given for each class of agent, the program asks for the number of periods (days) and repetitions to be simulated. On each day, every individual agent of each type is placed either in their home location or out in the community based on the probability that they go out. An SIR model is then simulated in the community and in each factory, boat, and household that is determined by the number of agents in each location, as well as their status as susceptible, infected, or removed. At the end of the period, the status of each agent is updated. The simulation is repeated to provide a distribution of results based on the random transmission and distribution of initial cases.

3.2 Scenarios Considered

For our analysis, we developed a list of plausible scenarios to consider and ran simulations in which we imposed various control measures. These scenarios are listed in Table 1.

Scenario	Name	Description
A	Baseline	Baseline scenario without NPI
B	Stay-at-home orders	Require all agents to stay in their home location the majority of the time
C	Mask mandates in community	Reduced transmission in the community location
D	Mask mandates in community and factories	Reduced transmission in the community and factory locations
E	Vaccinations	Some agents are not susceptible or infected
F	All of the above	Combination of B, D, and E
G	Factory mobile	Encourage the factory workers to leave home location
H	Factory mobile and mask mandates in community	Encourage the factory workers to leave home location and lower transmission rates in the community location
I	Factory mobile and mask mandates in community and factories	Encourage factory workers to leave home locations and lower transmission in the community location and in factories
J	Factory mobile, mask mandates in community and factories, and households lockdown	Encourage factory workers to leave, lower transmission in the community location and in factories, and household members cannot leave home location

Table 1: Scenarios for numerical exercises and brief descriptions

To motivate our numerical experiments, we considered a combination of non-pharmaceutical interventions that were common policy responses in the United States at the time, the actual measures taken in Bristol Bay, and other plausible combinations of control measures. Our interest lay in how the distribution of potential outcomes changed both in average outcomes and variance. We were also interested in potential unintended consequences of different actions.

Our baseline scenario considers non NPI and assumes that behavior does not meaningfully change during the outbreak. This is our base case against which we consider different outcomes. Our first alternative scenario consists of home orders which require everyone to remain in their home location but impose no restrictions on their interactions within that location. This includes both fishermen and factory workers, who have to stay within their boat or factory, respectively. There is precedent for this – during the 2020 season restrictions were passed that limited fishermen to their boats and restricted their movement while in port, and processing factories severely restricted the ability of their workers to access the local community. Our next scenario assumes mask mandates are enforced, but only in the community area. We assume the main effect of mask mandates is to reduce the transmission rates in the location they are used, which in this case is the community where groups mix. We extend that reduction in transmission to factories in scenario D. In Scenario E we introduce vaccinations and assume that inoculated agents are not able to be infected during the simulation period. Scenario F is a combination of vaccination, masks, and stay-at-home orders that restricts the movement between areas and lowers transmission rates.

Scenario G allows only the factory workers to leave their home location. Our motivation here is to examine the potential counterfactual for what actually happened in Bristol Bay. By restricting large quantities of people to a small and crowded working and living area there is the potential of an amplification effect. By bringing large quantities of susceptible people into close contact with infected individuals and not allowing them to distance, outbreaks in factory locations could be worse than otherwise expected. In this case, allowing factory workers to leave their home location could potentially shrink outbreak sizes and shorten outbreaks. We add mask mandates in scenario H, and we impose a universal mask mandate in scenario I. Scenario J extends this one step further and requires household members to remain in their homes and not go to the community location, while still allowing factory workers to distance in the community while masked.

Table 2 has scenario parameter values. Consistent throughout, we assume that non-pharmaceutical measures only impact transmission through the parameter β. In reality, there is likely a change also in the amount of time individuals spend in public and in community locations. There is evidence of both restricted activity and increased time in public.

Each scenario consists of a parameterization that lasts for 60 periods, and each scenario was simulated 1000 times. Table 3 contains details about each heterogeneous group across all scenarios, including the population of each group, the number of agents that reside in each home location, and how many agents are infected at the beginning of the simulation. Table 4 has the specific policy parameters for each scenario.

Agent	Population	Home Location Population	Initially Infected	γ
Fishermen	7000	5	10	14
Factory Workers	5000	200	10	14
Household Members	2226	3	0	14

Table 2: Population data for numerical exercises

	R0: Basic reproduction number in each location				Probability an agent goes into the community in a period			Initially vaccinated agents		
Scenario	Community	Boats	Factories	Households	Fishermen	Factory Workers	Household Members	Fishermen	Factory Workers	Household Members
A	5	7	7	6	20%	20%	20%	0	0	0
B	5	7	7	6	5%	5%	5%	0	0	0
C	1	7	7	6	20%	20%	20%	0	0	0
D	1	7	1.5	6	20%	20%	20%	0	0	0
E	5	7	7	6	20%	20%	20%	4200	3000	1335
F	1	7	1.5	6	5%	5%	5%	4200	3000	1335
G	5	7	7	6	20%	50%	20%	0	0	0
H	1	7	7	6	20%	50%	20%	0	0	0
I	1	7	1.5	6	20%	50%	20%	0	0	0
J	1	7	1.5	6	20%	50%	5%	0	0	0

Table 3: Parameters for numerical exercises

3.3 Analysis

Within the above scenarios we run repeated trials to explore uncertainty around where an outbreak might originate and the stochastic portions of transmission. We then compared average outcomes for variables of interest.

4) $y_h = \beta_0 + \sum_h \beta_h D_h + \epsilon_h$

We use the specification in equation (4) to regress the peak number of individuals infected, the day the number of peak infections occur, the total number of individuals infected over the duration of the outbreak, and the last day an individual is infected on a constant and a dummy variable for treatments other than the baseline. The coefficient β_0 is interpreted as the average value of the variable of interest in Treatment A – Baseline. The coefficient for β_h is the average deviation from the baseline case for all h different treatments. The error term reflects the random component of our simulations. Results are shown in Table 4.

Additionally, we generated a series of box and whisker plots (Figures 3 through 6) to better demonstrate the variability across treatments. These plots demonstrate the potential riskiness of

strategies that may on average reduce the severity of the pandemic (measured in our variables of interest) but can also lead to a greater variance in outcomes.

4. Results

In examining the results in Table 4, Treatment A is used as a baseline against which the other scenarios are compared. Shown in the first row, it denotes the peak number of infected individuals, the day with the peak number of infections, the total number of agents who become infected over the course of the simulation period, and the day at which the outbreak ends when no mitigation efforts are in place. The subsequent rows exhibit how the results for Treatments B through J compare to Treatment A. A negative number in a column about individuals (columns 1 and 3) imply fewer sick agents relative to Treatment A. A negative number in a column about time (columns 2 and 4) imply fewer days relative to Treatment A. For example, a negative number in column 3 relates how many fewer agents became infected on average compared to Treatment A, while a negative number in column 4 conveys how many fewer days on average it took for the outbreak to end. Graphical results of the outcomes in terms of peak infected individuals, day of peak infections, total infected individuals, and end of outbreak are shown in Figures 3, 4, 5, and 6, respectively.

Results

	Dependent variable:			
	Peak Infected Individuals	Day of Peak Infections	Total Removed (previously infected or vaccinated) Individuals	End of Outbreak
	(1)	(2)	(3)	(4)
Treatment A - Baseline	13,624.060***	22.025***	14,226.000***	43.366***
	(50.198)	(0.155)	(59.429)	(0.166)
Treatment B - Stay-at-Home Orders	-3,163.076***	7.789***	-88.950	16.634***
	(70.991)	(0.219)	(84.046)	(0.235)
Treatment C - Mask Mandates in Community	-13,101.620***	-10.732***	-13,669.030***	-17.143***
	(70.991)	(0.219)	(84.046)	(0.235)
Treatment D - Mask Mandates in Community and Factory	-13,575.290***	-16.619***	-14,167.990***	-22.565***
	(70.991)	(0.219)	(84.046)	(0.235)
Treatment E - Vaccinations	-8,444.026***	-0.710***	-0.657	13.039***

	(70.991)	(0.219)	(84.046)	(0.235)
Treatment F - All of the Above	-13,557.840***	-4.748***	-5,589.906***	-10.976***
	(70.991)	(0.219)	(84.046)	(0.235)
Treatment G - Send Factory Out	285.209***	0.510**	-0.000	-0.061
	(70.991)	(0.219)	(84.046)	(0.235)
Treatment H - Send Factory Out Masked	-10,302.630***	-0.357	-10,307.120***	-1.535***
	(70.991)	(0.219)	(84.046)	(0.235)
Treatment 1 - Send Factory Out Masked Everywhere	-13,576.390***	-17.057***	-14,170.020***	-22.659***
	(70.991)	(0.219)	(84.046)	(0.235)
Treatment J - Send Factory Out Masked Everywhere Household Members Home	-13,576.780***	-17.096***	-14,170.530***	-22.771***
	(70.991)	(0.219)	(84.046)	(0.235)
Observations	10,000	10,000	10,000	10,000
R^2	0.923	0.748	0.920	0.876
Adjusted R^2	0.923	0.748	0.920	0.875
Residual Std. Error (df = 9990)	1,587.398	4.892	1,879.324	5.257
F Statistic (df = 9; 9990)	13,387.760***	3,298.647***	12,769.230***	7,805.958***
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 4: Results of ABM simulations presented using a regression model. Treatment A – Baseline is the comparison group. All coefficients in every other treatment are deviations from Treatment A levels for the peak number of infected individuals, the day of peak infections, the total number of previously infected or vaccinated individuals, and the last day with an active case.

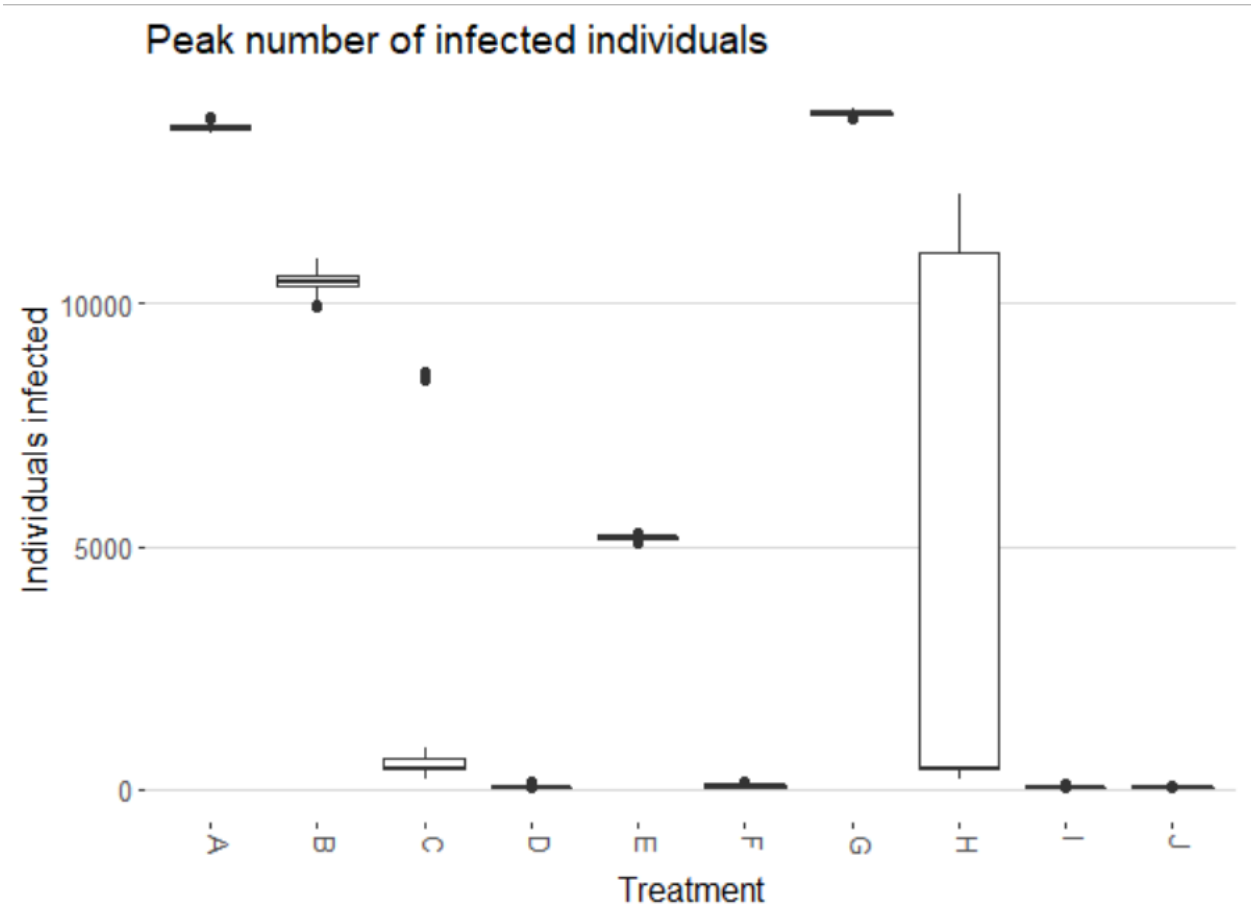

Figure 3: Peak number of infections

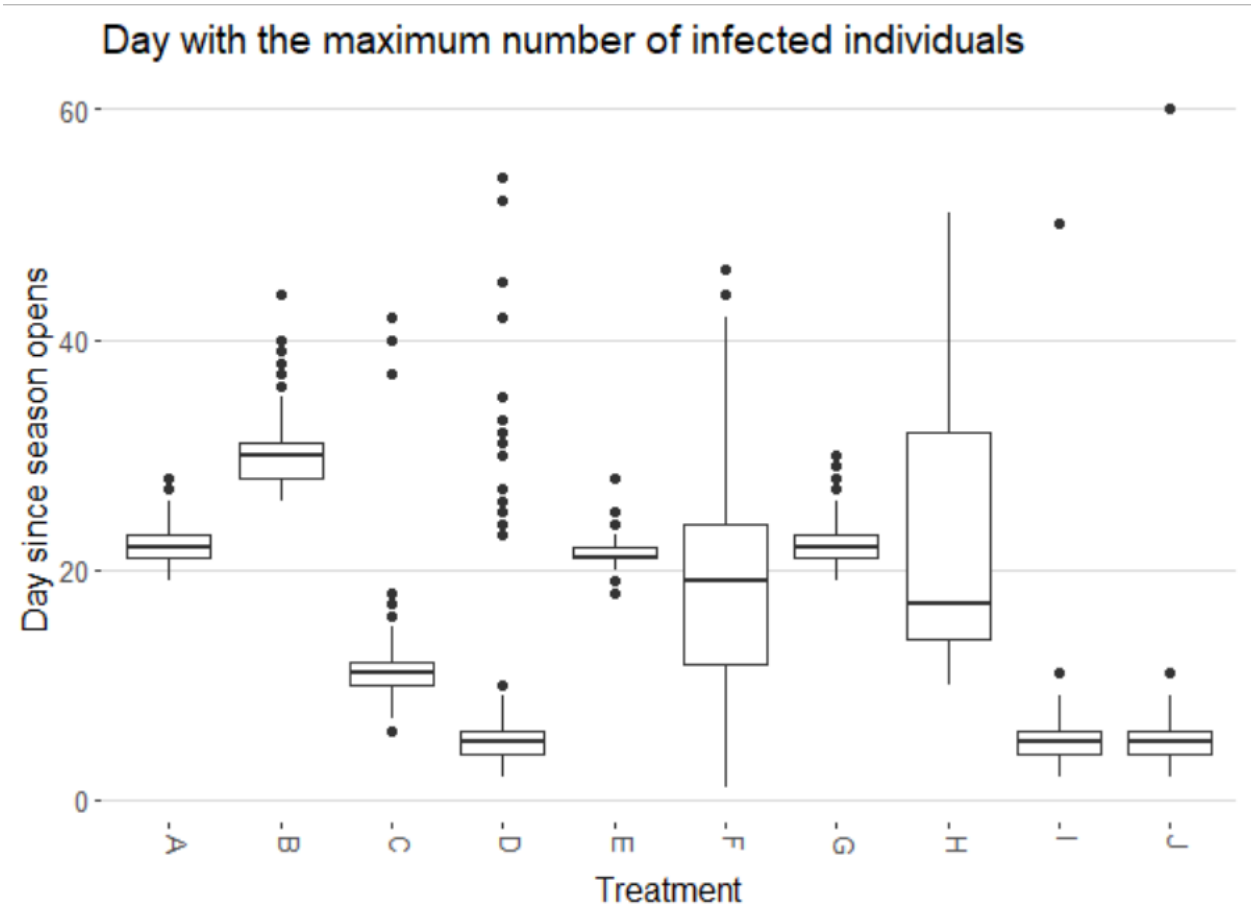

Figure 4: Day of maximum number of infected individuals

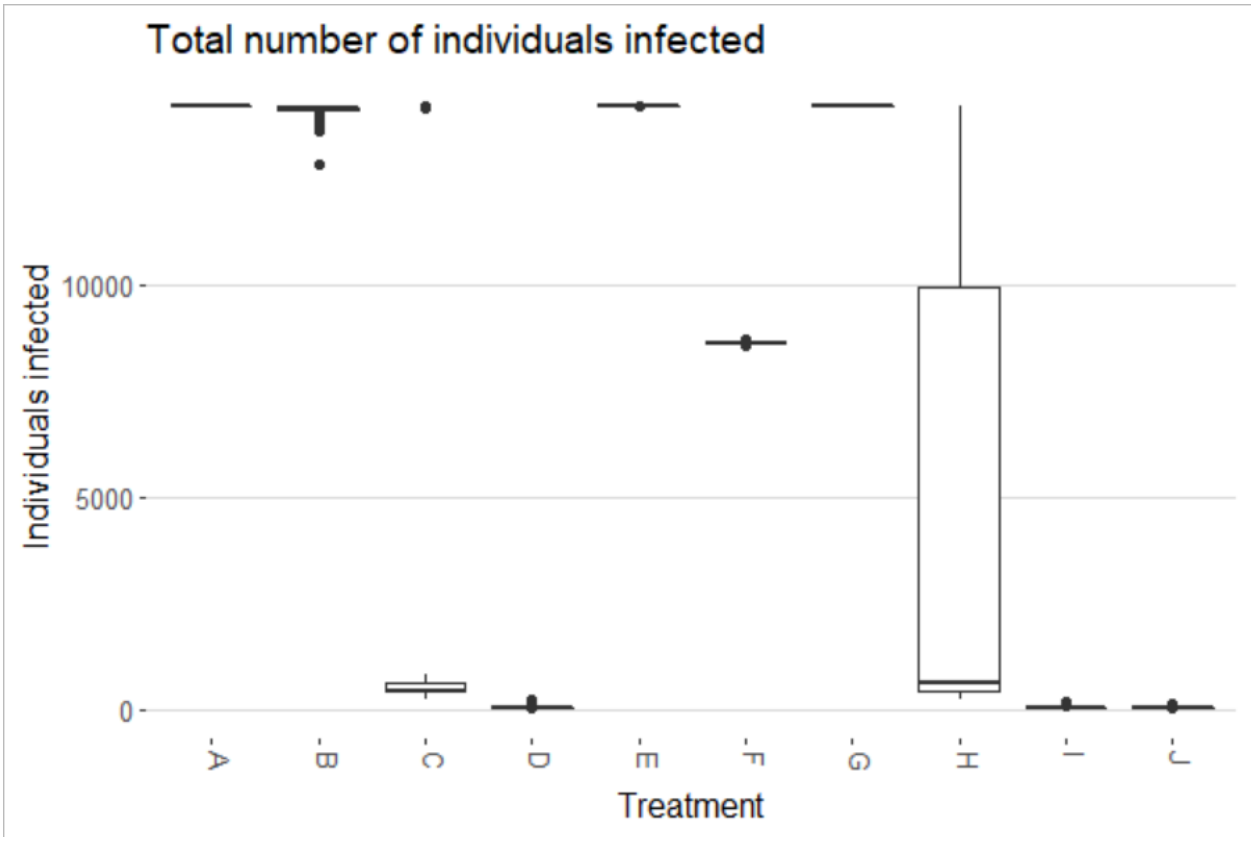

Figure 5: Total removed (previously infected or vaccinated) individuals

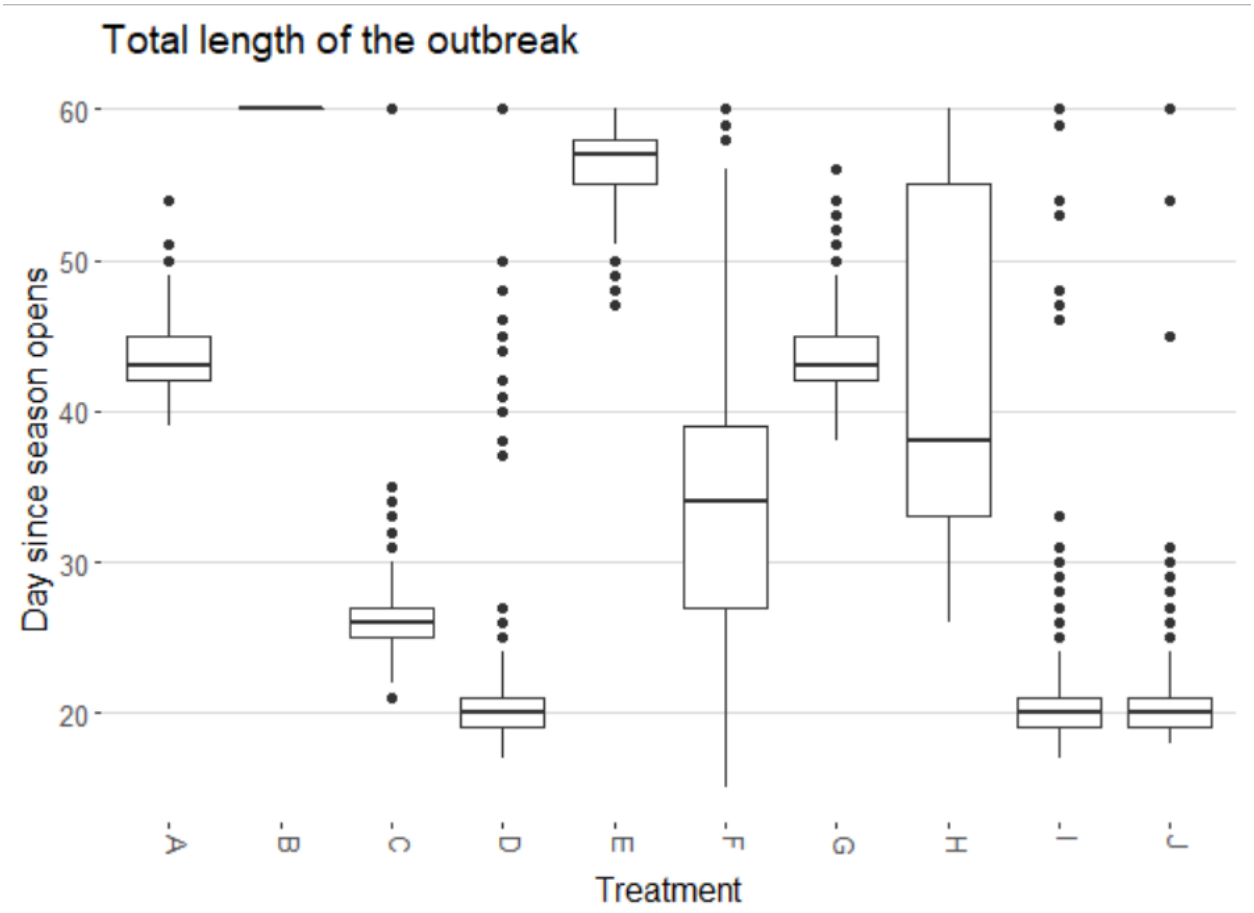

Figure 6: End of the outbreak

In Treatment A, the entire community is rapidly infected, and the outbreak ends when the last individual recovers or dies. In total 13,624 agents are infected at the peak, and this occurs on day 22 of the simulation. All agents become infected during the 60-day simulation, with the end of the outbreak occurring on day 44. This is the result of assumed exponential growth in infected cases and no endogenous behavioral response to the disease. We use this as a baseline to compare other strategies, acknowledging there is an extensive literature on the endogenous response to COVID-19, and that this is an upper bound on actual outcomes.

Treatment B imposes stay-at-home orders, and 3,163 fewer agents are infected at the peak, which occurs 8 days later. While only 88 fewer agents are infected in total, the outbreak lasts 17 more days. The peak and length of the outbreak are delayed. In our simulations the efficacy of stay-at-home orders lies in reducing the peak number of infections and reducing stress on finite resources at the height of the outbreak. The orders are imperfect – while trips are reduced by 75%, individuals still move between their home locations and the community location infrequently allowing the disease to spread. Perfectly enforced stay-at-home orders would reduce total cases further and potentially shorten the outbreak.

Treatment C incorporates the effective imposition of masks in the community location. Masks are assumed effective at reducing the reproductive number in the community location. This location

is where individuals are able to pass the disease between different agent types and home locations. As a result 13,102 fewer agents are infected at the peak, and 13,669 fewer contract the virus over the course of the simulation. Further, the outbreak peaks 11 days sooner and ends 17 days sooner when compared to Treatment A. In total, that yields an outbreak of 557 individuals lasting 26 days, demonstrating the impact of effective and required masking in public spaces during a pandemic. Outbreaks are also more frequently localized to individual factories, boats, or households as the disease is less able to spread between locations.

Treatment D has masking in factories in addition to the community location, and results in fewer peak infections, fewer total infections, fewer days to the peak number of infections, and fewer days to the outbreak than Treatment C. Treatments C and D are associated with many fewer peak and total infections than Treatment A. Additionally, the peak outbreak occurs sooner, helping to avoid fatigue and stress on small medical facilities in a remote, rural area.

Treatment E examines when 60% of each type of agent is vaccinated against the virus. Compared to Treatment A, 8,444 fewer people are infected at a peak that occurs on roughly the same day, and while roughly the same number of agents are eventually resistant to the disease in the "removed" category, 60% of them acquired immunity to the disease from the vaccine before the simulation began. This is important in interpreting the result in Table 4 and Figure 5. That is, while the total number of removed individuals is roughly the same, 8,536 of them are due to vaccination, while almost all unvaccinated individuals are eventually infected with the disease. Overall, the outbreak lasts 13 days longer, however the system does not face the same extreme stress from high peak infection numbers.

Treatment F combines the policy mechanisms of Treatments B, D, and E, and the results exhibit a vast reduction in the number of peak infected individuals, with 13,558 fewer infected at a peak that ends 5 days sooner on average than that of Treatment A. In total, 5,590 fewer agents are removed when stay-at-home orders, mask mandates, and vaccines are combined, and the outbreak ends 11 days sooner. The majority of individuals who do acquire immunity to the disease do so through vaccination, rather than infection. On average, only roughly 100 people are actually infected with the disease.

In Treatment G, factory workers are actively sent out in the community to reduce the contagion occurring in the factories. The results are similar to Treatment A, with slightly more agents infected at the peak of the outbreak.

Treatment H again has the factory workers more likely to be out in the community in a period, but they are now required to engage in effective masking. 10,303 fewer agents are infected at the peak, and 10,307 fewer agents get infected over the course of the outbreak. The peak outbreak occurs on the same day as Treatment A, but the entire outbreak ends around 2 days sooner.

Treatment I has the factory workers masked in the community and in their home factories, and the results have 13,576 fewer infections at a peak that occurs 17 days sooner than Treatment A. In total, 14,170 fewer infections occur, and the outbreak is 23 days shorter.

Lastly, Treatment J has factory workers masked in the community and in their respective factories, while household members are kept on stay-at-home orders. The addition of stay-at-home orders do not change the outcome observed in Treatment I. Factory workers are a large portion of the community, and the outbreaks in this setting predominantly occur within factories. There is not a difference between allowing factory workers to mingle in the community or only in the factory if household members are isolated at home.

5. Conclusions

We introduced a unique ABM program that simulates virus transmission in a small fish processing community in Bristol Bay, Alaska. The model allows the disease to move within and between heterogeneous groups, and we contribute a novel way of examining ABM results using regression analysis to contrast differences more clearly between scenarios. We demonstrate how different non-pharmaceutical methods for slowing the spread of a disease between locations and person types can impact disease outcomes. We focus on social interactions between groups and show that these interactions mediated by nonpharmaceutical interventions determine overall disease burden, consistent with the existing literature (Cristini and Trivin 2022; Cooper et al. 2023). We demonstrate that targeting different interventions leads to tradeoffs between variability in outcomes and disease burden. Additionally, we show the impact of vaccination in changing how people acquire resistance to infectious disease. These insights are important, as these communities are often critically dependent upon a small group of large employers, and the ability of different groups in society to interact allows the community to benefit from the development of their resources.

We find that rural, remote locations benefit not only from isolating different parts of the community, but from effective NPI and vaccination. This is consistent with peripheral areas being more at risk of COVID-19 (Armillei et al. 2021). Vaccination paired with NPI and movement restrictions can almost eliminate the risk of a pandemic. However, if a society places a high value on the ability of its household members to interact with outside workers, it can achieve similar results through vaccination and NPI alone. Our model shows that vaccination and efforts to reduce the reproductive number that we summarize as “masking” can allow local economies in these locations to continue to operate, and allow household members to interact with workers in ways that ensure spillovers to local communities from economic activity.

We find tradeoffs between duration and intensity, and tradeoffs between which groups are impacted by the pandemic. For example, we find that absent additional NPI, isolating processing factory workers from the rest of the community can amplify outbreaks within these factories. This can potentially still lead to stress on public health facilities, while denying the community the economic benefits of local economic activity. This contributes to the literature as it examines inter and intra group contagion and enables the study of the impact of various public policy measures when applied individually or in combination to all or certain groups. This approach can be used prescriptively or predictively to examine future outbreaks and the impact of public policy mechanisms on health outcomes. Additionally, we provide insights that are relevant to policymakers who may need to assuage the fears of different community groups. For instance, a community that is protecting itself from a seasonal influx of people may be more interested in protecting household members while limiting other NPI. Our model highlights how the unintended

consequences of amplifying spread within other groups without additional NPI could still stress healthcare capacity and adversely impact household members.

References

Acemoglu D, Chernozhukov V, Werning I, Whinston MD (2020) A multi-risk SIR model with optimally targeted lockdown. NBER Working Paper Series. Working Paper 27102. https://doi.org/10.3386/w27102.

Acemoglu D, Chernozhukov V, Werning I, Whinston MD (2021) Optimal targeted lockdowns in a multigroup SIR model. Am Econ Rev-Insights 3(4):487-502. https://doi.org/10.1257/aeri.20200590.

Althouse BM, Wenger EA, Miller JC, Scarpino SV, Allard A, Hébert-Dufresne L, Hu H (2020) Superspreading events in the transmission dynamics of SARS-CoV-2: Opportunities for intervention and control. Plos Biol 18(11):e3000897. https://doi.org/10.1371/journal.pbio.3000897.

Amuedo-Dorantes C, Borra C, Rivera-Garrido N, Sevilla A (2021) Early adoption of non-pharmaceutical interventions and COVID-19 mortality. Economics & Human Biology 42:101003. https://doi.org/10.1016/j.ehb.2021.101003.

Araya F (2021) Modeling the spread of COVID-19 on construction workers: An agent-based approach. Safety Sci 133:105022. https://doi.org/10.1016/j.ssci.2020.105022.

Arias J, Fernández-Villaverde J, Ramírez JR, Shin M (2021) The causal effects of lockdown policies on health and macroeconomic outcomes. NBER Working Paper Series. Working Paper 28617. https://doi.org/10.3386/w28617.

Armillei F, Filippucci F, Fletcher T (2021) Did Covid-19 hit harder in peripheral areas? The case of Italian municipalities. Economics & Human Biology 42: 101018. https://doi.org/10.1016/j.ehb.2021.101018.

Axelrod R, Tesfatsion L (2006) A guide for newcomers to agent-based modeling in the social sciences. In: Tesfatsion L, Judd KL (eds) Handbook of computational economics, vol 2. Agent-Based Computational Economics, Amsterdam, pp 1647-1659.

Bárcena-Martín E, Molina J, Muñoz-Fernández A, Pérez-Moreno S (2022) Vulnerability and COVID-19 infection rates: A changing relationship during the first year of the pandemic. Economics & Human Biology 47: 101177. https://doi.org/10.1016/j.ehb.2022.101177.

Bisin A, Moro A (2020) Learning epidemiology by doing: The empirical implications of a spatial-SIR model with behavioral responses. NBER Working Paper Series. Working Paper 27590. https://doi.org/10.3386/w27590.

Bisin A, Moro A (2022) Learning epidemiology by doing: The empirical implications of a spatial-SIR model with behavioral responses. J Urban Econ 127: Article 103368. https://doi.org/10.1016/j.jue.2021.103368.

Bognanni M, Hanley D, Kolliner D, Mitman K (2020) Economics and epidemics: Evidence from an estimated spatial econ-SIR model. Finance and Economics Discussions, Washington: Board of Governors of the Federal Reserve System. Discussion Series 2020-091. https://doi.org/10.17016/FEDS.2020.091.

Chen S-H (2012) Varieties of agents in agent-based computational economics: A historical and an interdisciplinary perspective. J Econ Dyn Control 36(1):1-25. https://doi.org/10.1016/j.jedc.2011.09.003.

Cooper D, Garga V, Luengo-Prado MJ, Tang J (2023) The mitigating effect of masks on the spread of Covid-19. Economics & Human Biology 48: 101195. https://doi.org/10.1016/j.ehb.2022.101195.

Cristini A, Trivin P (2022) Close encounters during a pandemic: Social habits and inter-generational links in the first two waves of COVID-19. Economics & Human Biology 47: 101180. https://doi.org/10.1016/j.ehb.2022.101180.

Cuevas E (2020) An agent-based model to evaluation the COVID-19 transmission risks in facilities. Comput Biol Med 121:103827. https://doi.org/10.1016/j.compbiomed.2020.103827.

deValpine, MG (2015) Influenza in Bristol Bay, 1919: “The saddest repudiation of a benevolent intention”. Sage One 5(1):1-8. https://doi.org/10.1177/2158244015577418.

Gans, J (2022) The economic consequences of $\hat{R} = 1$: Towards a workable behavioral epidemiological model of pandemics. Rev Econ Anal 14(1):3-25. https://doi.org/10.15353/rea.v14i1.4786.

Makris M, Toxvaerd F (2020) Great expectations: Social distancing in anticipation of pharmaceutical innovations. Cambridge Working Papers in Economics. Working Paper 2097. https://doi.org/10.17863/CAM.62310.

McKinley Research Group (2021) The Economic Benefits of Bristol Bay Salmon. https://www.mcdowellgroup.net/wp-content/uploads/2021/03/economic-benefits-of-bristol-bay-salmon.pdf. Accessed 10 October 2022.

Murray EJ (2020) Epidemiology’s time of need: COVID-19 calls for epidemic-related economics. J Econ Perspect 34(4):105-120. https://doi.org/10.1257/jep.34.4.105.

Nishi A, Dewey G, Endo A, Young SD (2020) Network interventions for managing the COVID-19 pandemic and sustaining the economy. PNAS 117(48):30285-30294. https://doi.org/10.1073/pnas.2014297117.

Prakash N, Srivastava B, Singh S, Sharma S Jain, S (2022) Effectiveness of social distancing interventions in containing COVID-19 incidence: International evidence using Kalman filter. Economics & Human Biology 44: 101091. https://doi.org/10.1016/j.ehb.2021.101091.

Saitone TL, Schaefer KA, Scheitrum DP (2021) COVID-19 morbidity and mortality in U.S. meatpacking counties. Food Policy 101:102072. https://doi.org/10.1016/j.foodpol.2021.102072.

Savage DW, Fisher A, Choudhury S, Ohle R, Strasser RP, Orkin A, Mago V (2020) Investigating the implications of COVID-19 for the rural and remote population of Northern Ontario using a mathematical model. medRxiv. https://doi.org/10.1101/2020.09.17.20196949

Silva PCL, Batista PVC, Lima HS, Alves MA, Guimarães FG, Silva RCP (2020) COVID-ABS: An agent-based model of COVID-19 epidemic to simulate health and economic effects of social distancing interventions. Chaos Soliton Fract 139:110088. https://doi.org/10.1016/j.chaos.2020.110088.

Sneppen K, Nielsen BF, Taylor RJ, Simonsen L (2021) Overdispersion in COVID-19 increases the effectiveness of limiting nonrepetitive contacts for transmission control. PNAS 118(14):e2016623118. https://doi.org/10.1073/pnas.2016623118.

Tatapudi H, Das R, Das TK (2020) Impact assessment of full and partial stay-at-home orders, face mask usage, and contact tracing: An agent-based simulation study of COVID-19 for an urban region. Global Epidemiol 2:100036. https://doi.org/10.1016/j.gloepi.2020.100036.

Wood AD, Finnoff DC, Mason CF (2016) OPEC, the Seven Sisters, and oil market dominance: An evolutionary game theory and agent-based modeling approach. J Econ Behav Organ 123(B): 66-78. https://doi.org/10.1016/j.jebo.2016.06.011.

Yakubenko S (2021) Home alone? Effect of weather-induced behaviour on spread of SARS-CoV-2 in Germany. Economics & Human Biology 42: 100998. https://doi.org/10.1016/j.ehb.2021.100998.

Appendix: Pseudocode

The program consists of nine components: a header file and source file for the TownspersonAgent class, a header file and source file for the FisherpersonAgent class, a header file and source file for the FactoryworkerAgent class, a header file and source file for the BristolBayCommunity class, and a TestLab source file that executes the “main” program.

Because the header and source files are the same for each class of individual agent, we below provide pseudocode for five components: a header and source file for a generic agent type, a header and source file for the community, and a source file for the main program.

GenericAgent.h

Define the class GenericAgent

Declare the public components of the class
    Declare the default constructor
    Declare specific constructors
    Declare the destructor

Declare a function that returns a GenericAgent's agent type
Declare a function that returns a GenericAgent's ID number
Declare a function that returns a GenericAgent's household/boat/factory number
Declare a function that returns a GenericAgent's location
Declare a function that returns a GenericAgent's health status
Declare a function that returns a GenericAgent's recovered/removed date
Declare a function that returns a GenericAgent's recovered/removed status

Declare a function that sets a GenericAgent's agent type
Declare a function that sets a GenericAgent's ID number
Declare a function that sets a GenericAgent's household/boat/factory number
Declare a function that sets a GenericAgent's location
Declare a function that sets a GenericAgent's health status
Declare a function that sets a GenericAgent's recovered/removed date
Declare a function that sets a GenericAgent's recovered/removed status

Declare private variables of the class
    Declare a variable that holds a GenericAgent's agent type
    Declare a variable that holds a GenericAgent's ID number
    Declare a variable that holds a GenericAgent's household/boat/factory number
    Declare a variable that holds a GenericAgent's location
    Declare a variable that holds a GenericAgent's health status
    Declare a variable that holds a GenericAgent's recovered/removed date
    Declare a variable that holds a GenericAgent's recovered/removed status

GenericAgent.cpp

Declare inclusions and libraries, including the GenericAgent header file and maths library functionality

Define a specific constructor, allowing it to accept initial parameter values from the user
Define a specific constructor, creating a new GenericAgent by copying an extant class member
Define the destructor

Implement a function that returns a GenericAgent's agent type
Implement a function that returns a GenericAgent's ID number

Implement a function that returns a GenericAgent's household/boat/factory number
Implement a function that returns a GenericAgent's location
Implement a function that returns a GenericAgent's health status
Implement a function that returns a GenericAgent's recovered/removed date
Implement a function that returns a GenericAgent's recovered/removed status

Implement a function that sets a GenericAgent's agent type
    Take in a sensible value (0, 1, or 2) from the user and set that as the agent's type
Implement a function that sets a GenericAgent's ID number
    Take in an integer from the user and set that as the GenericAgent's ID number
Implement a function that sets a GenericAgent's household/boat/factory number
    Take in an integer from the user and set that as the GenericAgent's household/boat/factory number
Implement a function that sets a GenericAgent's location
    Take in a sensible value (0 or 1) from the user and set that as a GenericAgent's location
Implement a function that sets a GenericAgent's health status
    Take in a sensible value (0 or 1) from the user and set that as a GenericAgent's health status
Implement a function that sets a GenericAgent's recovered/removed date
    Take in a sensible value (from 0 to gamma (γ)) and set that as a GenericAgent's recovered/removed date
Implement a function that returns a GenericAgent's recovered/removed status
    Take in a sensible value (0 or 1) from the user and set that as a GenericAgent's recovered/removed status

BristolBay.h

Define the class BristolBay

Define all relevant inclusions including random number, vector, array, and time library functionality
Include the FisherpersonAgent header file
Include the FactoryworkerAgent header file
Include the TownspersonAgent header file

Declare public components of the class
    Declare the default constructor
    Declare the destructor

    Declare the function that prompts the user to input R_0 for the community
    Declare the function that prompts the user to input R_0 for each boat
    Declare the function that prompts the user to input R_0 for each factory
    Declare the function that prompts the user to input R_0 for each household
    Declare the function that prompts the user to input gamma (γ) for the community
    Declare the function that prompts the user to input gamma (γ)for fisherpersons

Declare the function that prompts the user to input gamma (γ) for factory workers
Declare the function that prompts the user to input gamma (γ) for townspeople
Declare the function that prompts the user to input the probability that fisherpersons go into the community in a period
Declare the function that prompts the user to input the probability that factory workers go into the community in a period
Declare the function that prompts the user to input the probability that townspeople go into the community in a period
Declare the function that prompts the user to input the population of fisherpersons
Declare the function that prompts the user to input the number of fisherpersons per boat
Declare the function that prompts the user to input the number of fisherpersons initially sick
Declare the function that prompts the user to input the number of fisherpersons initially vaccinated
Declare the function that prompts the user to input the population of factory workers
Declare the function that prompts the user to input the number of factory workers per factory
Declare the function that prompts the user to input the number of factory workers initially sick
Declare the function that prompts the user to input the number of factory workers initially vaccinated
Declare the function that prompts the user to input the population of townspeople
Declare the function that prompts the user to input the number of townspeople per household
Declare the function that prompts the user to input the number of townspeople initially sick
Declare the function that prompts the user to input the number of townspeople initially vaccinated

Declare the function that establishes an agent as a FisherpersonAgent type
    This function accepts a FisherpersonAgent and an integer as inputs
Declare the function that establishes an ID number for a FisherpersonAgent
    This function accepts a FisherpersonAgent and an integer as inputs
Declare a function that assigns a FisherpersonAgent to a specific boat as their home location
    This function accepts a FisherpersonAgent and an integer as inputs
Declare a function that assigns a FisherpersonAgent to their location in a given period
    This function accepts a FisherpersonAgent and an integer as inputs
Declare a function that establishes the FisherpersonAgent's infected status
    This function accepts a FisherpersonAgent and an integer as inputs
Declare a function that establishes if a FisherpersonAgent is or was infected at one point
    This function accepts a FisherpersonAgent and an integer as inputs
Declare a function that establishes that a newly infected FisherpersonAgent will be sick for gamma (γ) number of periods

This function accepts a FisherpersonAgent and an integer as inputs
Declare a function that reduces the number of remaining periods in which a FisherpersonAgent is infected by one during each period
This function accepts a FisherpersonAgent and an integer as inputs
Declare a function that sets a FisherpersonAgent as recovered/removed once the gamma (γ) period is complete
This function accepts a FisherpersonAgent and an integer as inputs
Declare the function that establishes an agent as a FactoryworkerAgent type
This function accepts a FactoryworkerAgent and an integer as inputs
Declare the function that establishes an ID number for a FactoryworkerAgent
This function accepts a FactoryworkerAgent and an integer as inputs
Declare a function that assigns a FactoryworkerAgent to a specific factory as their home location
This function accepts a FactoryworkerAgent and an integer as inputs
Declare a function that assigns a FactoryworkerAgent to their location in a given period
This function accepts a FactoryworkerAgent and an integer as inputs
Declare a function that establishes the FactoryworkerAgent's infected status
This function accepts a FactoryworkerAgent and an integer as inputs
Declare a function that establishes if a FactoryworkerAgent is or was infected at one point
This function accepts a FactoryworkerAgent and an integer as inputs
Declare a function that establishes that a newly infected FactoryworkerAgent will be sick for gamma (γ) number of periods
This function accepts a FactoryworkerAgent and an integer as inputs
Declare a function that reduces the number of remaining periods in which a FactoryworkerAgent is infected by one during each period
This function accepts a FactoryworkerAgent and an integer as inputs
Declare a function that sets a FactoryworkerAgent as recovered/removed once the gamma (γ) period is complete
This function accepts a FactoryworkerAgent and an integer as inputs
Declare the function that establishes an agent as a TownspersonAgent type
This function accepts a TownspersonAgent and an integer as inputs
Declare the function that establishes an ID number for a TownspersonAgent
This function accepts a TownspersonAgent and an integer as inputs
Declare a function that assigns a TownspersonAgent to a specific household as their home location
This function accepts a TownspersonAgent and an integer as inputs
Declare a function that assigns a TownspersonAgent to their location in a given period
This function accepts a TownspersonAgent and an integer as inputs
Declare a function that establishes the TownspersonAgent's infected status
This function accepts a TownspersonAgent and an integer as inputs
Declare a function that establishes if a TownspersonAgent is or was infected at one point
This function accepts a TownspersonAgent and an integer as inputs
Declare a function that establishes that a newly infected TownspersonAgent will be sick for gamma (γ) number of periods

This function accepts a TownspersonAgent and an integer as inputs
Declare a function that reduces the number of remaining periods in which a TownspersonAgent is infected by one during each period
This function accepts a TownspersonAgent and an integer as inputs
Declare a function that sets a TownspersonAgent as recovered/removed once the gamma (γ) period is complete
This function accepts a TownspersonAgent and an integer as inputs

Declare private variables of the class
Declare a variable that holds the R_0 for the community
Declare a variable that holds the R_0 for each boat
Declare a variable that holds the R_0 for each factory
Declare a variable that holds the R_0 for each household
Declare a variable that holds gamma (γ) for the community
Declare a variable that holds gamma (γ) for fisherpersons
Declare a variable that holds gamma (γ) for factory workers
Declare a variable that holds gamma (γ) for townspeople
Declare a variable that holds the probability that factory workers go into the community in a period
Declare a variable that holds the probability that fisherpersons go into the community in a period
Declare a variable that holds the probability that townspeople go into the community in a period
Declare a variable that holds the population of fisherpersons
Declare a variable that holds the number of fisherpersons per boat
Declare a variable that holds the number of fisherpersons initially sick
Declare a variable that holds the number of fisherpersons initially vaccinated
Declare a variable that holds the population of factory workers
Declare a variable that holds the number of factory workers per factory
Declare a variable that holds the number of factory workers initially sick
Declare a variable that holds the number of factory workers initially vaccinated
Declare a variable that holds the population of townspeople
Declare a variable that holds the number of townspeople per household
Declare a variable that holds the number of townspeople initially sick
Declare a variable that holds the number of townspeople initially vaccinated

BristolBay.cpp

Declare inclusions and libraries, including maths library functionality
Include the FisherpersonAgent header file
Include the FactoryworkerAgent header file
Include the TownspersonAgent header file

Define the default constructor
Define the destructor

Implement the function that prompts the user to input R_0 for the community
    Print a message that welcomes the user to the program with copyright and authorship information
    Print instructions to the user concerning R_0
    Accept R_0 for the community via keyboard input
    Return R_0 for the community to the program

Implement the function that prompts the user to input R_0 for boats
    Accept R_0 for boats via keyboard input
    Return R_0 for boats to the program

Implement the function that prompts the user to input R_0 for factories
    Accept R_0 for factories via keyboard input
    Return R_0 for factories to the program

Implement the function that prompts the user to input R_0 for households
    Accept R_0 for households via keyboard input
    Return R_0 for households to the program

Implement the function that prompts the user to input gamma (γ) for the community
    Print instructions to the user concerning gamma (γ)
    Accept gamma (γ) for the community via keyboard input
    Return gamma (γ) for the community to the program

Implement the function that prompts the user to input gamma (γ) for boats
    Accept gamma (γ) for boats via keyboard input
    Return gamma (γ) for boats to the program

Implement the function that prompts the user to input gamma (γ) for factories
    Accept gamma (γ) for factories via keyboard input
    Return gamma (γ) for factories to the program

Implement the function that prompts the user to input gamma (γ) for households
    Accept gamma (γ) for households via keyboard input
    Return gamma (γ) for households to the program

Implement the function that prompts the user to input the probability that fisherpersons go into the community in a period
    While loop to ensure the probability is within logical bounds
        Print instructions to the user concerning probabilities
        Accept the probability via keyboard input
    Return the probability to the program

Implement the function that prompts the user to input the probability that factory workers go into the community in a period

While loop to ensure the probability is within logical bounds
Accept the probability via keyboard input
Return the probability to the program

Implement the function that prompts the user to input the probability that townspeople go into the community in a period
While loop to ensure the probability is within logical bounds
Accept the probability via keyboard input
Return the probability to the program

Implement the function that prompts the user to input the population of fisherpersons
While loop to ensure the population is nonnegative and takes integer values
Print instructions to the user concerning this and the next three functions related to fisherpersons: population, boat size, initial sick, and initial vaccinated
Accept the population via keyboard input
Return the population to the program

Implement the function that prompts the user to input the number of fisherpersons per boat
While loop to ensure the boat size is nonnegative and does not exceed the population of fisherpersons
Accept the boat size via keyboard input
Return the boat size to the program

Implement the function that prompts the user to input the number of fisherpersons initially sick
While loop to ensure the number of initially sick is nonnegative and does not exceed the population of fisherpersons
Accept the number of initially sick via keyboard input
Return the number of initially sick to the program

Implement the function that prompts the user to input the number of fisherpersons initially vaccinated
While loop to ensure the number of initially vaccinated is nonnegative and does not exceed the population of fisherpersons
Accept the number of initially vaccinated via keyboard input
Return the number of initially vaccinated to the program

Implement the function that prompts the user to input the population of factory workers
While loop to ensure the population is nonnegative and takes integer values
Print instructions to the user concerning this and the next three functions related to factory workers: population, factory size, initial sick, and initial vaccinated
Accept the population via keyboard input
Return the population to the program

Implement the function that prompts the user to input the number of factory workers per factory
While loop to ensure the factory size is nonnegative and does not exceed the population of factory workers

Accept the factory size via keyboard input
Return the factory size to the program

Implement the function that prompts the user to input the number of factory workers initially sick
While loop to ensure the number of initially sick is nonnegative and does not exceed the population of factory workers
Accept the number of initially sick via keyboard input
Return the number of initially sick to the program

Implement the function that prompts the user to input the number of factory workers initially vaccinated
While loop to ensure the number of initially vaccinated is nonnegative and does not exceed the population of factory workers
Accept the number of initially vaccinated via keyboard input
Return the number of initially vaccinated to the program

Implement the function that prompts the user to input the population of townspeople
While loop to ensure the population is nonnegative and takes integer values
Print instructions to the user concerning this and the next three functions related to townspeople: population, factory size, initial sick, and initial vaccinated
Accept the population via keyboard input
Return the population to the program

Implement the function that prompts the user to input the number of townspeople per household
While loop to ensure the household size is nonnegative and does not exceed the population of townspeople
Accept the household size via keyboard input
Return the household size to the program

Implement the function that prompts the user to input the number of townspeople initially sick
While loop to ensure the number of initially sick is nonnegative and does not exceed the population of townspeople
Accept the number of initially sick via keyboard input
Return the number of initially sick to the program

Implement the function that prompts the user to input the number of townspeople initially vaccinated
While loop to ensure the number of initially vaccinated is nonnegative and does not exceed the population of townspeople
Accept the number of initially vaccinated via keyboard input
Return the number of initially vaccinated to the program

Implement the function that gives a FisherpersonAgent an identifier of their agent type
This function accepts a member of the FisherpersonAgent class and an integer from the TestLab

This function calls the FisherpersonAgent member function that sets agent type

Implement the function that gives a FisherpersonAgent an ID number

This function accepts a member of the FisherpersonAgent class and an integer from the TestLab

The function calls the FishpersonAgent member function that sets ID numbers

Implement the function that assigns a FisherpersonAgent to their boat

This function accepts a member of the FisherpersonAgent class and an integer from the TestLab

The function calls the FishpersonAgent member function that sets boat number

Implement the function that establishes the location of a FisherpersonAgent

This function accepts a member of the FisherpersonAgent class and an integer from the TestLab

The function calls the FishpersonAgent member function that sets their location

Implement the function that establishes the health status of a FisherpersonAgent as infected

This function accepts a member of the FisherpersonAgent class and an integer from the TestLab

The function calls the FishpersonAgent member function that sets their health status

Implement the function that establishes the health status of a FisherpersonAgent as not infected

This function accepts a member of the FisherpersonAgent class and an integer from the TestLab

The function calls the FishpersonAgent member function that sets their health status

Implement the function that establishes the date that a FisherpersonAgent will no longer be infected

This function accepts a member of the FisherpersonAgent class and an integer from the TestLab

The function calls the FishpersonAgent member function that sets when their health is fixed

Implement the function that establishes reduces the time the FisherpersonAgent will be infected by one each period

This function accepts a member of the FisherpersonAgent class and an integer from the TestLab

The function calls the FishpersonAgent member function that sets when their health is fixed

Implement the function that establishes that a FisherpersonAgent is recovered/removed

This function accepts a member of the FisherpersonAgent class and an integer from the TestLab

The function calls the FishpersonAgent member function that sets when they are recovered/removed

Implement the function that gives a FactoryworkerAgent an identifier of their agent type

This function accepts a member of the FactoryworkerAgent class and an integer from the TestLab

This function calls the FactoryworkerAgent member function that sets agent type

Implement the function that gives a FactoryworkerAgent an ID number

This function accepts a member of the FactoryworkerAgent class and an integer from the TestLab

The function calls the FactoryworkerAgent member function that sets ID numbers

Implement the function that assigns a FactoryworkerAgent to their factory

This function accepts a member of the FactoryworkerAgent class and an integer from the TestLab

The function calls the FactoryworkerAgent member function that sets factory number

Implement the function that establishes the location of a FactoryworkerAgent

This function accepts a member of the FactoryworkerAgent class and an integer from the TestLab

The function calls the FactoryworkerAgent member function that sets their location

Implement the function that establishes the health status of a FactoryworkerAgent as infected

This function accepts a member of the FactoryworkerAgent class and an integer from the TestLab

The function calls the FactoryworkerAgent member function that sets their health status

Implement the function that establishes the health status of a FactoryworkerAgent as not infected

This function accepts a member of the FactoryworkerAgent class and an integer from the TestLab

The function calls the FactoryworkerAgent member function that sets their health status

Implement the function that establishes the date that a FactoryworkerAgent will no longer be infected

This function accepts a member of the FactoryworkerAgent class and an integer from the TestLab

The function calls the FactoryworkerAgent member function that sets when their health is fixed

Implement the function that establishes reduces the time the FactoryworkerAgent will be infected by one each period

This function accepts a member of the FactoryworkerAgent class and an integer from the TestLab

The function calls the FactoryworkerAgent member function that sets when their health is fixed

Implement the function that establishes that a FactoryworkerAgent is recovered/removed

This function accepts a member of the FactoryworkerAgent class and an integer from the TestLab
The function calls the FactoryworkerAgent member function that sets when they are recovered/removed

Implement the function that gives a TownspersonAgent an identifier of their agent type
This function accepts a member of the TownspersonAgent class and an integer from the TestLab
This function calls the TownspersonAgent member function that sets agent type

Implement the function that gives a TownspersonAgent an ID number
This function accepts a member of the TownspersonAgent class and an integer from the TestLab
The function calls the TownspersonAgent member function that sets ID numbers

Implement the function that assigns a TownspersonAgent to their household
This function accepts a member of the TownspersonAgent class and an integer from the TestLab
The function calls the TownspersonAgent member function that sets household number

Implement the function that establishes the location of a TownspersonAgent
This function accepts a member of the TownspersonAgent class and an integer from the TestLab
The function calls the TownspersonAgent member function that sets their location

Implement the function that establishes the health status of a TownspersonAgent as infected
This function accepts a member of the TownspersonAgent class and an integer from the TestLab
The function calls the TownspersonAgent member function that sets their health status

Implement the function that establishes the health status of a TownspersonAgent as not infected
This function accepts a member of the TownspersonAgent class and an integer from the TestLab
The function calls the TownspersonAgent member function that sets their health status

Implement the function that establishes the date that a TownspersonAgent will no longer be infected
This function accepts a member of the TownspersonAgent class and an integer from the TestLab
The function calls the TownspersonAgent member function that sets when their health is fixed

Implement the function that establishes reduces the time the TownspersonAgent will be infected by one each period
This function accepts a member of the TownspersonAgent class and an integer from the TestLab

The function calls the TownspersonAgent member function that sets when their health is fixed

Implement the function that establishes that a TownspersonAgent is recovered/removed
This function accepts a member of the TownspersonAgent class and an integer from the TestLab
The function calls the TownspersonAgent member function that sets when they are recovered/removed

TestLab.cpp

Define all relevant inclusions including random number, vector, array, and time library functionality
Include the BristolBay header file
Include the FisherpersonAgent header file
Include the FactoryworkerAgent header file
Include the TownspersonAgent header file

Implement the function that prompts the user to input the number of periods/rounds to be simulated
Declare an integer variable to hold the number of periods to be simulated
While loop to ensure the number of rounds is a nonnegative integer
Print instructions concerning the number of periods to the user
Accept the number of periods via keyboard input
Return the number of periods to be simulated to the program

Implement the function that prompts the user to input the number of repetitions to be simulated
Declare an integer variable to hold the number of repetitions to be simulated
While loop to ensure the number of repetitions is a nonnegative integer
Print instructions concerning the number of iterations to the user
Accept the number of repetitions via keyboard input
Return the number of repetitions to be simulated to the program

Declare main program

Create an object of the BristolBay class

Declare a variable of type double to hold R_0 for the community
Call the BristolBay object's function that prompts the user to input R_0 for the community and assign it to the variable of type double
Declare a variable of type double to hold R_0 for the fisherpersons
Call the BristolBay object's function that prompts the user to input R_0 for the fisherpersons and assign it to the variable of type double
Declare a variable of type double to hold R_0 for the factory workers

Call the BristolBay object's function that prompts the user to input R_0 for the factory workers and assign it to the variable of type double
Declare a variable of type double to hold R_0 for the townspeople
Call the BristolBay object's function that prompts the user to input R_0 for the townspeople and assign it to the variable of type double

Declare a variable of type double to hold gamma (γ) for the community
Call the BristolBay object's function that prompts the user to input gamma (γ) for the community and assign it to the variable of type double
Declare a variable of type double that is equal to $1/\gamma$ used in the modeling of virus transmission in the community
Declare a variable of type double to hold gamma (γ) for the fisherpersons
Call the BristolBay object's function that prompts the user to input gamma (γ) for the fisherpersons and assign it to the variable of type double
Declare a variable of type double that is equal to $1/\gamma$ used in the modeling of virus transmission on boats
Declare a variable of type double to hold gamma (γ) for the factory workers
Call the BristolBay object's function that prompts the user to input gamma (γ) for the community and assign it to the variable of type double
Declare a variable of type double that is equal to $1/\gamma$ used in the modeling of virus transmission in factories
Declare a variable of type double to hold gamma (γ) for the townspeople
Call the BristolBay object's function that prompts the user to input gamma (γ) for the community and assign it to the variable of type double
Declare a variable of type double that is equal to $1/\gamma$ used in the modeling of virus transmission in households

Declare a variable of type double to hold β for the community
Calculate β as R_0 / $1/\gamma$
Declare a variable of type double to hold β for boats
Calculate β as R_0 / $1/\gamma$
Declare a variable of type double to hold β for factories
Calculate β as R_0 / $1/\gamma$
Declare a variable of type double to hold β for households
Calculate β as R_0 / $1/\gamma$

Declare a variable of type double to hold the probability that a fisherperson goes into the community in a period
Call the BristolBay object's function that prompts the user to input the probability that a fisherperson goes into the community in a period and assign it the variable of type double
Declare a variable of type double to hold the probability that a factory worker goes into the community in a period
Call the BristolBay object's function that prompts the user to input the probability that a factory worker goes into the community in a period and assign it the variable of type double
Declare a variable of type double to hold the probability that a townsperson goes into the community in a period

Call the BristolBay object's function that prompts the user to input the probability that a townsperson goes into the community in a period and assign it the variable of type double

Declare a variable of type double to hold the population of fisherpersons
Declare a variable of type double to hold the crew size of boats
Declare a variable of type double to hold the number of initially infected fisherpersons
Declare a variable of type double to hold the number of initially vaccinated fisherpersons
Call the BristolBay object's function that prompts the user to input the population of fisherpersons and assign it to the variable of type double
Call the BristolBay object's function that prompts the user to input the crew size of boats and assign it to the variable of type double
Call the BristolBay object's function that prompts the user to input the number of initially sick fisherpersons and assign it to the variable of type double
Call the BristolBay object's function that prompts the user to input the number of initially vaccinated fisherpersons and assign it to the variable of type double

Declare a variable of type double to hold the population of factory workers
Declare a variable of type double to hold the size of factories
Declare a variable of type double to hold the number of initially infected factory workers
Declare a variable of type double to hold the number of initially vaccinated factory workers
Call the BristolBay object's function that prompts the user to input the population of factory workers and assign it to the variable of type double
Call the BristolBay object's function that prompts the user to input the size of factories and assign it to the variable of type double
Call the BristolBay object's function that prompts the user to input the number of initially sick factory workers and assign it to the variable of type double
Call the BristolBay object's function that prompts the user to input the number of initially vaccinated factory workers and assign it to the variable of type double

Declare a variable of type double to hold the population of townspeople
Declare a variable of type double to hold the size of households
Declare a variable of type double to hold the number of initially infected townspeople
Declare a variable of type double to hold the number of initially vaccinated townspeople
Call the BristolBay object's function that prompts the user to input the population of townspeople and assign it to the variable of type double
Call the BristolBay object's function that prompts the user to input the size of households and assign it to the variable of type double
Call the BristolBay object's function that prompts the user to input the number of initially sick townspeople and assign it to the variable of type double
Call the BristolBay object's function that prompts the user to input the number of initially vaccinated townspeople and assign it to the variable of type double

Declare a variable of type int to hold the number of periods to be simulated
Call the function that prompts the user to input the number of periods to be simulated and assign it to the variable of type int
Declare a variable of type int to hold the number of repetitions to be simulated

Call the function that prompts the user to input the number of repetitions to be simulated and assign it to the variable of type int

Print a parameter report to the user to ensure the accuracy of all variables
Output to the screen all parameter values as input by the user with respect to the virus, each agent type, and the number of periods and repetitions to be simulated

Declare a variable of type char to ask the user for their approval of the parameter report
Print a confirmation question asking the user to input a y or Y if they would like to proceed
Accept the user's response via keyboard input and assign it to the variable of type char
If the user inputs y or Y, proceed and print that the program is running agent-based simulations
Else exit the program

For loop that repeats until the number of repetitions is satisfied

Create a vector of FisherpersonAgent objects of size dictated by the user
Create a vector of FactoryworkerAgent objects of size dictated by the user
Create a vector of TownspersonAgent objects of size dictated by the user

For loop that repeats over the size of the vector of FisherpersonAgent objects

Call the BristolBay object's function that assigns an ID number to a FisherpersonAgent object
Call the BristolBay object's function that assigns a FisherpersonAgent object to a boat

For loop that repeats over the size of the vector of FactoryworkerAgent objects

Call the BristolBay object's function that assigns an ID number to a FactoryworkerAgent object
Call the BristolBay object's function that assigns a FactoryworkerAgent object to a factory

For loop that repeats over the size of the vector of TownspersonAgent objects

Call the BristolBay object's function that assigns an ID number to a TownspersonAgent object
Call the BristolBay object's function that assigns a TownspersonAgent object to a household

Declare a variable of type double, initialized at 0, that counts the number of sick fisherpersons
Declare a variable of type double that holds the probability that a fisherperson is initially sick, initialized at the number of initially sick fisherpersons divided by the population of fisherpersons and multiplied by 100
Declare a variable of type double, initialized at 0, that counts the number of sick factory workers
Declare a variable of type double that holds the probability that a factory worker is initially sick, initialized at the number of initially sick factory workers divided by the population of factory workers and multiplied by 100

Declare a variable of type double, initialized at 0, that counts the number of sick townspeople
Declare a variable of type double that holds the probability that a townsperson is initially sick, initialized at the number of initially sick townspersons divided by the population of townspersons and multiplied by 100

For loop that repeats over the size of the vector of FisherpersonAgent objects
    If the count of sick fisherpersons is less than the number of initially sick fisherpersons
        If the probability that a fisherperson is initially sick exceeds a random number between 0 and 100
            Call the BristolBay object's function that establishes a fisherperson as sick
            Declare a variable of type int to hold how long the fisherperson will remain infected, is initialized at a random number between 1 and γ
            Call the BristolBay object's function that establishes how long the fisherperson will remain infected and assign it the int between 1 and γ
            Increase the count of sick fisherpersons by 1
        Else do not change the count of sick fisherpersons
    Else do not change the count of sick fisherpersons

For loop that repeats over the size of the vector of FactoryworkerAgent objects
    If the count of sick factory workers is less than the number of initially sick factory workers
        If the probability that a factory worker is initially sick exceeds a random number between 0 and 100
            Call the BristolBay object's function that establishes a factory worker as sick
            Declare a variable of type int to hold how long the factory worker will remain infected, is initialized at a random number between 1 and γ
            Call the BristolBay object's function that establishes how long the factory worker will remain infected and assign it the int between 1 and γ
            Increase the count of sick factory workers by 1
        Else do not change the count of sick factory workers
    Else do not change the count of sick factory workers

For loop that repeats over the size of the vector of TownspersonAgent objects
    If the count of sick townspeople is less than the number of initially sick townspeople
        If the probability that a townspeople is initially sick exceeds a random number between 0 and 100

Call the BristolBay object's function that establishes a townspeople as sick
Declare a variable of type int to hold how long the townspeople will remain infected, is initialized at a random number between 1 and γ
Call the BristolBay object's function that establishes how long the townspeople will remain infected and assign it the int between 1 and γ
Increase the count of sick townspeople by 1
Else do not change the count of sick townspeople
Else do not change the count of sick townspeople

Declare a variable of type double, initialized at 0, that counts the number of vaccinated fisherpersons
Declare a variable of type double that holds the probability that a fisherperson is initially vaccinated, initialized at the number of initially vaccinated fisherpersons divided by the population of fisherpersons and multiplied by 100
Declare a variable of type double, initialized at 0, that counts the number of vaccinated factory workers
Declare a variable of type double that holds the probability that a factory worker is initially vaccinated, initialized at the number of initially vaccinated factory workers divided by the population of factory workers and multiplied by 100
Declare a variable of type double, initialized at 0, that counts the number of vaccinated townspeople
Declare a variable of type double that holds the probability that a townsperson is initially vaccinated, initialized at the number of initially vaccinated townspersons divided by the population of townspersons and multiplied by 100

For loop that repeats over the size of the vector of FisherpersonAgent objects
If the count of vaccinated fisherpersons is less than the number of initially vaccinated fisherpersons
If, when calling the FisherAgent object that gets the fisherperson's health status, the agent is healthy
If the probability that a fisherperson is initially vaccinated exceeds a random number between 0 and 100
Call the BristolBay object's function that establishes a fisherperson as vaccinated
Increase the count of vaccinated fisherpersons by 1
Else do not change the count of vaccinated fisherpersons
Else do not change the count of vaccinated fisherpersons
Else do not change the count of vaccinated fisherpersons

For loop that repeats over the size of the vector of FactoryworkerAgent objects
If the count of vaccinated factory workers is less than the number of initially vaccinated factory workers

If, when calling the FactoryworkerAgent object that gets the factory worker's health status, the agent is healthy
If the probability that a factory worker is initially vaccinated exceeds a random number between 0 and 100
Call the BristolBay object's function that establishes a factory worker as vaccinated
Increase the count of vaccinated factory workers by 1
Else do not change the count of vaccinated factory workers
Else do not change the count of vaccinated factory workers
Else do not change the count of vaccinated factory workers

For loop that repeats over the size of the vector of TownspersonAgent objects
If the count of vaccinated townspeople is less than the number of initially vaccinated townspeople
If, when calling the TownspersonAgent object that gets the townsperson's health status, the agent is healthy
If the probability that a townsperson is initially vaccinated exceeds a random number between 0 and 100
Call the BristolBay object's function that establishes a townsperson as vaccinated
Increase the count of vaccinated townspeople by 1
Else do not change the count of vaccinated townspeople
Else do not change the count of vaccinated townspeople
Else do not change the count of vaccinated townspeople

Determine how many of each agent type, as well as the entire population, is susceptible, infected, and removed
Create an object that will generate Excel files of various formats to record the results of each repetition of the program
Detail the particulars of each Excel file, including format and recorded content that includes column titles for all agent types and the entire population of agents, as well as their status as susceptible, infected, and removed
Write all data from period 0 into the first row of the Excel output file for each repetition of the simulation

For loop that repeats until the number of periods is satisfied
Declare a variable of type double to hold the number of sick fisherpersons in the period
Declare a variable of type double to hold the number of sick factory workers in the period
Declare a variable of type double to hold the number of sick townspeople in the period

For loop that repeats over the size of the vector of FisherpersonAgent objects
If, when calling the FisherpersonAgent object that gets a fisherperson agent's health status, the agent is sick

Increase the count of sick fisherpersons
Else do not change the count of sick fisherpersons
For loop that repeats over the size of the vector of FactoryworkerAgent objects
If, when calling the FactoryworkerAgent object that gets a factory worker agent's health status, the agent is sick
Increase the count of sick factory workers
Else do not change the count of sick factory workers
For loop that repeats over the size of the vector of TownspersonAgent objects
If, when calling the TownspersonAgent object that gets a townsperson agent's health status, the agent is sick
Increase the count of sick townspeople
Else do not change the count of sick townspeople

Declare a variable of type double to hold the location of a fisherperson in the period
Declare a variable of type double to count the number of fisherpersons in the community in the period
Declare a variable of type double to hold the location of a factory worker in the period
Declare a variable of type double to count the number of factory workers in the community in the period
Declare a variable of type double to hold the location of a townsperson in the period
Declare a variable of type double to count the number of townspeople in the community in the period

For loop that repeats over the size of the vector of FisherpersonAgent objects
If the probability a fisherperson goes out in a period exceeds a random number between 0 and 100
Set the fisherperson's location variable to out in the community
Update the count of fisherperson agents in the community in the period
Else set the fisherperson's location variable to their boat
Call the BristolBay object's function that sets the fisherperson's location

For loop that repeats over the size of the vector of FactoryworkerAgent objects
If the probability a factory worker goes out in a period exceeds a random number between 0 and 100
Set the factory worker's location variable to out in the community
Update the count of factory worker agents in the community in the period
Else set the factory worker's location variable to their factory
Call the BristolBay object's function that sets the factory worker's location

For loop that repeats over the size of the vector of TownspersonAgent objects

If the probability a townsperson goes out in a period exceeds a random number between 0 and 100
Set the townsperson's location variable to out in the community
Update the count of townsperson agents in the community in the period
Else set the townsperson's location variable to their household
Call the BristolBay object's function that sets the townsperson's location

Declare a variable of type double to hold the number of fisherpersons out in the community who are infected and initialize it at zero
Declare a variable of type double to hold the number of fisherpersons out in the community who are susceptible and initialize it at zero
Declare a variable of type double to hold the number of fisherpersons out in the community who are removed and initialize it at zero
Declare a variable of type double to hold the number of factory workers out in the community who are infected and initialize it at zero
Declare a variable of type double to hold the number of factory workers out in the community who are susceptible and initialize it at zero
Declare a variable of type double to hold the number of factory workers out in the community who are removed and initialize it at zero
Declare a variable of type double to hold the number of townspeople out in the community who are infected and initialize it at zero
Declare a variable of type double to hold the number of townspeople out in the community who are susceptible and initialize it at zero
Declare a variable of type double to hold the number of townspeople out in the community who are removed and initialize it at zero

Declare a variable of type double to hold the total number of agents out in the community who are susceptible and initialize it at zero
Declare a variable of type double to hold the total number of agents out in the community who are infected and initialize it at zero
Declare a variable of type double to hold the total number of agents out in the community who are removed and initialize it at zero

For loop that repeats over the size of the vector of FisherpersonAgent objects
If the fisherperson is infected and out in the community and not recovered
Update the count of fisherpersons infected in the community by one
Else do not increase the count of fisherpersons infected in the community
If the fisherperson is not infected and out in the community and not recovered
Update the count of fisherpersons susceptible in the community by one
Else do not increase the count of fisherpersons susceptible in the community

If the fisherperson is not infected and out in the community and recovered
Update the count of fisherpersons removed in the community by one
Else do not increase the count of fisherpersons removed in the community

For loop that repeats over the size of the vector of FactoryworkerAgent objects
If the factory worker is infected and out in the community and not recovered
Update the count of factory workers infected in the community by one
Else do not increase the count of factory workers infected in the community
If the factory worker is not infected and out in the community and not recovered
Update the count of factory workers susceptible in the community by one
Else do not increase the count of factory workers susceptible in the community
If the factory worker is not infected and out in the community and recovered
Update the count of factory workers removed in the community by one
Else do not increase the count of factory workers removed in the community

For loop that repeats over the size of the vector of TownspersonAgent objects
If the townsperson is infected and out in the community and not recovered
Update the count of townspeople infected in the community by one
Else do not increase the count of townspeople infected in the community
If the townsperson is not infected and out in the community and not recovered
Update the count of townspeople susceptible in the community by one
Else do not increase the count of townspeople susceptible in the community
If the townsperson is not infected and out in the community and recovered
Update the count of townspeople removed in the community by one
Else do not increase the count of townspeople removed in the community

Calculate the total number infected agents out in the community as the sum of all three infected types out in the community in the period
Calculate the total number susceptible agents out in the community as the sum of all three susceptible types out in the community in the period
Calculate the total number removed agents out in the community as the sum of all three removed types out in the community in the period

Declare a variable of type double that is equal to β for the community multiplied by (the total infected agents out in the community divided by the total susceptible agents out in the community)

For loop that repeats over the size of the vector of FisherpersonAgent objects
    If the fisherperson is out and not infected and not recovered
        If β for the community multiplied by (the total infected agents out in the community divided by the total susceptible agents out in the community) exceeds a random number between 0 and 100
            Call the BristolBay object's function that sets the fisherperson's health status to infected
            Call the BristolBay object's function that sets a fisherperson as infected for γ number of periods
            Increase the count of infected fisherpersons by one
            Increase the count of infected fisherpersons out in the community by one
        Else do not update the count of infected fisherpersons
    Else if the fisherperson is out and infected and not recovered
        Declare a variable of type int and call the FisherpersonAgent object's function that gets the date that the fisherperson will cease to be infected
        Call the BristolBay object's function that reduces the fisherperson agent's remaining infected time by one period
    Else do not update the count of infected fisherpersons

For loop that repeats over the size of the vector of FactoryworkerAgent objects
    If the factory worker is out and not infected and not recovered
        If β for the community multiplied by (the total infected agents out in the community divided by the total susceptible agents out in the community) exceeds a random number between 0 and 100
            Call the BristolBay object's function that sets the factory worker's health status to infected
            Call the BristolBay object's function that sets a factory worker as infected for γ number of periods
            Increase the count of infected factory workers by one
            Increase the count of infected factory workers out in the community by one
        Else do not update the count of infected factory workers
    Else if the factory worker is out and infected and not recovered
        Declare a variable of type int and call the FactoryworkerAgent object's function that gets the date that the factory worker will cease to be infected
        Call the BristolBay object's function that reduces the factory worker agent's remaining infected time by one period
    Else do not update the count of infected factory workers

For loop that repeats over the size of the vector of TownspersonAgent objects
    If the townsperson is out and not infected and not recovered
        If β for the community multiplied by (the total infected agents out in the community divided by the total susceptible agents out in the community) exceeds a random number between 0 and 100
            Call the BristolBay object's function that sets the townsperson's health status to infected
            Call the BristolBay object's function that sets a townsperson as infected for γ number of periods
            Increase the count of infected townspeople by one
            Increase the count of infected townspeople out in the community by one
        Else do not update the count of infected townspeople
    Else if the townsperson is out and infected and not recovered
        Declare a variable of type int and call the TownspersonAgent object's function that gets the date that the townsperson will cease to be infected
        Call the BristolBay object's function that reduces the townsperson agent's remaining infected time by one period
    Else do not update the count of infected townspeople

Declare a variable of type int to hold the number of boats, calculated as the population of fisherpersons divided by the crew size of boats
Declare a variable of type into to hold the number of factories, calculated as the population of factory workers divided by the size of factories
Declare a variable of type int to hold the number of households, calculated as the population of townspeople divided by the size of households

For loop that repeats over the number of boats
    Declare a variable of type double to hold the number of agents on a boat who are susceptible and initialize it at zero
    Declare a variable of type double to hold the number of agents on a boat who are susceptible and initialize it at zero
    Declare a variable of type double to hold the number of agents on a boat who are removed and initialize it at zero
    For loop that repeats over the size of the vector of FisherpersonAgent objects
        If the fisherperson lives on this boat and is infected and is on the boat
            Update the count of fisherpersons infected on this boat by one
        If the fisherperson lives on this boat and is not infected and is on the boat and is not removed
            Update the count of fisherpersons susceptible on this boat by one

If the fisherperson lives on this boat and is not infected and is on the boat and is removed

Update the count of fisherpersons removed on this boat by one

Declare a variable of type double that is equal to β for a boat multiplied by (the total infected agents on this boat divided by the total susceptible agents on this boat)

For loop that repeats over the size of the vector of FisherpersonAgent objects

If the fisherperson lives on this boat and is not infected and is on the boat and is not removed

If β for a boat multiplied by (the total infected agents on this boat divided by the total susceptible agents on this boat exceeds a random number between 0 and 100

Call the BristolBay object's function that sets the fisherperson's health status to infected

Call the BristolBay object's function that sets a fisherperson as infected for γ number of periods

Increase the count of infected fisherpersons on this boat by one

Increase the count of infected fisherpersons by one

Else do not update the count of infected fisherpersons

Else if the fisherperson lives on this boat and is infected and is on the boat and is not removed

Declare a variable of type int and call the FisherpersonAgent object's function that gets the date that the fisherperson will cease to be infected

Call the BristolBay object's function that reduces the fisherperson agent's remaining infected time by one period

Else do not update the count of infected fisherpersons

For loop that repeats over the number of factories

Declare a variable of type double to hold the number of agents in a factory who are susceptible and initialize it at zero

Declare a variable of type double to hold the number of agents in a factory who are susceptible and initialize it at zero

Declare a variable of type double to hold the number of agents in a fact who are removed and initialize it at zero

For loop that repeats over the size of the vector of FactoryworkerAgent objects

If the factory worker lives in this factory and is infected and is in the factory

Update the count of factory workers infected in this factory by one

If the factory worker lives in this factory and is not infected and is in the factory and is not removed

Update the count of factory worker susceptible in this factory by one

If the factory worker lives in this factory and is not infected and is in the factory and is removed

Update the count of factory workers removed in this factory by one

Declare a variable of type double that is equal to β for a factory multiplied by (the total infected agents in this factory divided by the total susceptible agents in this factory)

For loop that repeats over the size of the vector of FactoryworkerAgent objects

If the factory worker lives in this factory and is not infected and is in the factory and is not removed

If β for a factory multiplied by (the total infected agents in this factory divided by the total susceptible agents in this factory exceeds a random number between 0 and 100

Call the BristolBay object's function that sets the factory worker's health status to infected

Call the BristolBay object's function that sets a factory worker as infected for γ number of periods

Increase the count of infected factory workers in this factory by one

Increase the count of infected factory workers by one

Else do not update the count of infected factory workers

Else if the factory worker lives in this factory and is infected and is in the factory and is not removed

Declare a variable of type int and call the FactoryworkerAgent object's function that gets the date that the factory worker will cease to be infected

Call the BristolBay object's function that reduces the factory worker agent's remaining infected time by one period

Else do not update the count of infected factory workers

For loop that repeats over the number of households

Declare a variable of type double to hold the number of agents in a household who are susceptible and initialize it at zero

Declare a variable of type double to hold the number of agents in a household who are susceptible and initialize it at zero

Declare a variable of type double to hold the number of agents in a household who are removed and initialize it at zero

For loop that repeats over the size of the vector of TownspersonAgent objects

If the townsperson lives in this household and is infected and is in the household

Update the count of townspeople infected in this household by one

If the townsperson lives in this household and is not infected and is in the household and is not removed

Update the count of townspeople susceptible in this household by one

If the townsperson lives in this household and is not infected and is in the household and is removed

Update the count of townspeople removed in this household by one

Declare a variable of type double that is equal to β for a household multiplied by (the total infected agents in this household divided by the total susceptible agents in this household)

For loop that repeats over the size of the vector of TownspersonAgent objects

If the townsperson lives in this household and is not infected and is in the household and is not removed

If β for a household multiplied by (the total infected agents in this household divided by the total susceptible agents in this household exceeds a random number between 0 and 100

Call the BristolBay object's function that sets the townsperson's health status to infected

Call the BristolBay object's function that sets a townsperson as infected for γ number of periods

Increase the count of infected townspeople in this household by one

Increase the count of infected townspeople by one

Else do not update the count of infected townspeople

Else if the townsperson lives in this household and is infected and is in the household and is not removed

Declare a variable of type int and call the TownspersonAgent object's function that gets the date that the townsperson will cease to be infected

Call the BristolBay object's function that reduces the townsperson agent's remaining infected time by one period

Else do not update the count of infected townspeople

Declare a variable of type int to hold the number of susceptible fisherpersons at the end of the period

Declare a variable of type int to hold the number of infected fisherpersons at the end of the period

Declare a variable of type int to hold the number of removed fisherpersons at the end of the period

Declare a variable of type int to hold the number of susceptible factory workers at the end of the period

Declare a variable of type int to hold the number of infected factory workers at the end of the period
Declare a variable of type int to hold the number of removed factory workers at the end of the period
Declare a variable of type int to hold the number of susceptible townspeople at the end of the period
Declare a variable of type int to hold the number of infected townspeople at the end of the period
Declare a variable of type int to hold the number of removed townspeople at the end of the period

For loop that repeats over the size of the vector of FisherpersonAgent objects
    If the fisherperson is not infected and not removed
        Increase the count of end of period susceptible fisherpersons by one
    If the fisherperson is infected and not removed
        Increase the count of end of period infected fisherpersons by one
    If the fisherperson is not infected and is removed
        Increase the count of end of period removed fisherpersons by one

For loop that repeats over the size of the vector of FactoryworkerAgent objects
    If the factory worker is not infected and not removed
        Increase the count of end of period susceptible factory workers by one
    If the factory worker is infected and not removed
        Increase the count of end of period infected factory workers by one
    If the factory worker is not infected and is removed
        Increase the count of end of period removed factory workers by one

For loop that repeats over the size of the vector of TownspersonAgent objects
    If the townsperson is not infected and not removed
        Increase the count of end of period susceptible fisherpersons by one
    If the townsperson is infected and not removed
        Increase the count of end of period infected townspeople by one
    If the townsperson is not infected and is removed
        Increase the count of end of period removed townspeople by one

Calculate the total number of end of period susceptible agents as the sum of all three susceptible types
Calculate the total number of end of period infected agents as the sum of all three infected types
Calculate the total number of end of period removed agents as the sum of all three removed types
Write all data for the current period into the next row of the Excel output file

For loop that repeats over the size of the vector of FisherpersonAgent objects
If the fisherperson is infected and this is the final period they are infected
Declare a variable of type int and call the FisherpersonAgent object's function that gets the date that the fisherperson will cease to be infected
If this is the final period in which a fisherperson is infected
Call the BristolBay object's function that reduces the infected agent's remaining infected time by one period to zero
Call the BristolBay object's function that sets the fisherperson's health status to not infected
Call the BristolBay object's function that sets the fisherperson's removed status to removed

For loop that repeats over the size of the vector of FactoryworkerAgent objects
If the factory worker is infected and this is the final period they are infected
Declare a variable of type int and call the FactoryworkerAgent object's function that gets the date that the factory worker will cease to be infected
If this is the final period in which a factory worker is infected
Call the BristolBay object's function that reduces the infected agent's remaining infected time by one period to zero
Call the BristolBay object's function that sets the factory worker's health status to not infected
Call the BristolBay object's function that sets the factory worker's removed status to removed

For loop that repeats over the size of the vector of TownspersonAgent objects
If the townsperson is infected and this is the final period they are infected
Declare a variable of type int and call the TownspersonAgent object's function that gets the date that the townsperson will cease to be infected
If this is the final period in which a townsperson is infected
Call the BristolBay object's function that reduces the infected agent's remaining infected time by one period to zero
Call the BristolBay object's function that sets the townsperson's health status to not infected
Call the BristolBay object's function that sets the townsperson's removed status to removed

Close the Excel output file objects

Print a message to the user that the agent-based simulations are complete.
Return control to close the main program