TRADITIONAL KINSHIP STRUCTURES AND EUROPEAN-DERIVED DISEASES AT MISSION SAN DIEGO, CALIFORNIA: A STUDY OF THE 1805-1806 MEASLES EPIDEMIC

A Dissertation

presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

CAROLYN MARIE ORBANN

Dr. Lisa Sattenspiel, Dissertation Supervisor

MAY 2014

The undersigned, appointed by the dean of the Graduate School,

have examined the dissertation entitled

TRADITIONAL KINSHIP STRUCTURES AND EUROPEAN-DERIVED DISEASES AT MISSION SAN DIEGO, CALIFORNIA: A STUDY OF THE 1805-1806 MEASLES EPIDEMIC

Presented by Carolyn Marie Orbann

A candidate for the degree of

Doctor of Philosophy

And hereby certify that, in their opinion, it is worthy of acceptance.

Professor Lisa Sattenspiel

Professor Todd VanPool

Professor Gregory Blomquist

Professor Jeffrey Pasley

ACKNOWLEDGEMENTS

While only my name appears on the title page of this document, its completion would not have been possible without the support and guidance of many, many people. I would like to take a moment to recognize and acknowledge those people to whom I owe my gratitude.

Dr. Lisa Sattenspiel played the biggest role in helping me to get this project off the ground, get it funded and ultimately finished. I sincerely thank her for the support she has given and the example she has provided. She has been a wonderful mentor and a huge source of support over the last eight years.

Thanks also go to the rest of my committee members. Your advice, comments, and continued patience is greatly appreciated. Special thanks to Dr. Greg Blomquist for helping me learn how to use R for the data analysis. Dr. Todd VanPool has always provided a supportive ear and encouragement throughout. Dr. Jeff Pasley pointed me to several valuable historical background sources for this project.

I would like to thank the archivists and librarians at the Kumeyaay archives, the library at San Diego de Alcalá, the San Diego Historical Center and the Huntington Library for their assistance during the research phase of the project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TRADITIONAL KINSHIP STRUCTURES AND EUROPEAN-DERIVED DISEASES AT MISSION SAN DIEGO, CALIFORNIA: A STUDY OF THE 1805-1806 MEASLES EPIDEMIC

Carolyn Orbann

Dr. Lisa Sattenspiel, Dissertation Supervisor

ABSTRACT

European diseases were a significant cause of morbidity and mortality in Native American communities after contact with any European colonizers. Studies of Native Californian communities have documented the effects of European diseases such as smallpox, measles and whooping cough, but only at a few of the 21 Spanish missions established during the Mission Period and primarily late in the historic period. This study used archival and historical records, combined with later ethnographic materials, to investigate the potential impact of a measles epidemic in 1806. This epidemic was documented at other missions and was known to have caused mortality of up to 25% in certain populations, but no direct evidence of illness was recorded at Mission San Diego.

Using an agent-based model designed to reflect both the structure of the historic population and the behavioral patterns of neophyte populations, simulated measles epidemics in the mission population were examined. Simulation results indicate that the pattern of recorded deaths in the winter of 1805/1806 is not consistent with a virgin soil measles epidemic (as produced by the model), but could not rule out measles as a cause of increased deaths.

CHAPTER 1 – INTRODUCTION

The conquest of North American indigenous peoples by European colonizers has been a fertile ground for research in history, anthropology, culture theory, linguistics, archaeology and many other fields. The events of the 15th through 18th centuries led to significant changes to the cultures, populations, and physical environments on both sides of the Atlantic Ocean as colonized populations adjusted to new political systems, intermarried and interbred with European, African and Asian immigrants, and saw centuries-old traditional lifestyles transformed both to fit with and react against the economic expectations of colonizing populations. This time period was also significant for colonizing populations, as national economies incorporated trans-Atlantic trade, social statuses were altered by the inclusion of material wealth and populations from the Americas, and European economies and diets were transformed by raw materials found in the "New World". The European entrance into the Western Hemisphere had significant and long-term impacts on the health of human populations around the world, both positive and negative.

Interest in the Columbian Exchange (as it came to be known) increased during the latter half of the 20th century as the quincentennial anniversary of the arrival of Christopher Columbus in the Caribbean in 1492 was recognized and sometimes celebrated. There are numerous studies on the impacts of colonization on specific Amerind communities, particularly those nations that were large and wealthy, such as the Inca and Aztecs (Borah and Cook 1960; Settipane 1995). Studies looking at the

ecological consequences of contact abound (e.g. Crosby 1972 and Mann 2011), as well as those that seek to broadly describe the consequences of contact by focusing on continental level outcomes (Cook 1976a; Jackson and Castillo 1995). Smaller scale studies contribute to the understanding of overall trends by filling out the details of the impacts of colonization in specific places and times, reactions to individual events or the experiences of a single, small culture (Hackel 2005; Milliken 1995; Stojanowski 2005). This study is one of the puzzle pieces, focusing in on a small population and its experience of a single epidemic. Though the scope is small, the information gained can help to further flesh out knowledge of the results of culture contact in North America and make discussions of the effects of colonization more inclusive of the experiences of both large and small indigenous populations.

In some ways this project is an extension of my master's thesis (Orbann 2006), in which I examined demographic features of four political groups, known as tribelets, during the mission period. The four groups I considered then were from the Sacramento Delta area in north-central California. I used mission record vital statistics to test whether or not these populations appeared to fit the model for demographic stability at the time of conquest. If so, the inference was that pre-contact epidemic disease had likely not reached these groups and caused the kind of mortality that was known in other populations. I quickly discovered that those populations did not resemble the model for demographic stability and in fact appeared to be headed towards extinction through low fertility and high mortality.

In doing that research, I was confounded by the problem that the mission records for those groups were created too late to really capture the phenomenon of interest, the potential early spread of European disease in Alta California (the Spanish colonial term that refers to the present-day state of California). The contact period in Alta California starts by at least the middle of the 18th century with the founding of Mission San Diego. In reality, it probably started much earlier, as knowledge, material goods and possibly infectious disease traveled along trade routes between present-day Southern California, Mexico, and other parts of the U.S. Southwest (which had colonial settlements for almost 200 years before the Spanish expanded into Alta California). If I was interested in the effects of colonization on the peoples of California, I wanted to work with the records of the first mission, Mission San Diego de Alcala.

Another result of that work was the realization that the straightforward application of demographic methods to the mission records was not capturing the complexity of the problem of infectious disease in these mission populations. There are well-known complications with attempting to use parish records for demographic analysis, even when those records are thought to be relatively complete and accurate (Kasakoff and Adams 1995; Preston et al 2001; Smith 1975). Complications include different standards for calculating ages or understanding time, biases in recording events for certain segments of a population, or not adequately capturing the life events of migrant populations. In the case of the records at Mission San Diego, the first set of books was destroyed during an attack and many of the burial records from the mid-19th century were also lost. I needed to do something beyond traditional historical

demography if I wanted to work with these records in a meaningful way. I decided to try to use a computer simulation model, something that had not yet been done for populations in California.

Statement of problem

The Catholic missions in Alta California altered the geographic, social and economic landscape of the region and functioned both as a center for change and a refuge for indigenous people whose traditional lifeways were altered by events beyond their control. Living conditions for neophytes (new converts, in this case of indigenous origin) at the missions are usually portrayed as vastly inferior in terms of health, individual autonomy, and cultural cohesion, in comparison to indigenous populations who avoided conversion and subsequent removal to mission communities (Jackson and Castillo 1995; Larson et al 1994; Powers 1976). This dissertation does not seek to address the psychosocial lives of neophytes at the missions, nor does it investigate how missionized populations used political and economic power to negotiate a place in the colonial context of Alta California differently from populations that never underwent conversion. The goal of this project is to look at the interplay between indigenous social organization, the church-regulated activity patterns of the neophytes and the health of the missionized population. As is discussed in detail in Chapter 6, there is little evidence that epidemic disease played a significant role in the health of Native Californians prior to European exploration and conquest. Because direct documentary evidence of epidemic disease at Mission San Diego is sparse, I chose to develop a computer

simulation model that examines how an infectious disease might spread in a population like that of the neophytes at the mission, who would have organized their behavior around both traditional family structures and newly adopted Catholic routines.

The primary research hypothesis of this project is that people with more kin physically located within the mission community at Mission San Diego would have different levels of disease-related morbidity and mortality during an epidemic of an acute, infectious disease than people with fewer kin. To operationalize this research question several choices were made. First, a disease that was known to have affected mission populations was chosen, in this case measles. Second, a time frame was chosen that the model would be tuned to reflect. In this case, a measles epidemic in 1805-1806 was chosen due to its known severity at other missions, and the presumption (based on increased numbers of deaths seen in the mission death registers) that it impacted the population at Mission San Diego. Third, a reference population was chosen. Mission San Diego was inhabited primarily by people belonging to the Kumeyaay culture groups, making it one of the more culturally homogenous missions in Alta California. A model platform was chosen, and the model developed. The model incorporated a standard model design that was modified to include specific ethnographic and historical information about the study population. The results of the simulations help to provide details about how disease affected the population at Mission San Diego, but can also be more broadly interpreted in the context of a kin-based population with limited access to western medicine.

Organization

This dissertation is organized into ten chapters (including this one) and two appendices. Chapters 2 through 7 are background chapters that contain discussions of literature, both scholarly and historical, that are relevant to the project. Chapter 2 describes agent-based models and specifically the use of agent-based models in anthropology. Chapter 3 is a review of the ethnographic literature on the Kumeyaay, the population of interest for this study. Chapter 4 contains historical background about missionization and the Spanish colonizing population. Chapter 5 is an in-depth examination of the Kumeyaay population at Mission San Diego and the state of the mission in the early 19th century. This chapter describes the landscape, the architecture, what is known about neophyte health, the conversion process, the population structure of missionized populations, and other specific details used to inform the model. Chapter 6 is a discussion of disease in Californian populations both pre- and post-European conquest. Chapter 7 describes the historical and genealogical methods used to reconstruct the mission population. Chapter 8 describes the model that was developed for this project. Chapter 9 is a presentation of the results of the study and a discussion of their implications. Chapter 10 is the concluding chapter to the dissertation. The appendices hold the specific text files that describe the agents and the structures used in the simulations.

Terminology used in this dissertation

Throughout this dissertation I have attempted to use language that is both precise in translation (either from 18th century Spanish or from the Kumeyaay language) and that carries specific meaning in the context of a colonial situation. In attempting to use words or concepts in a sensitive manner, I worked with native speakers of Spanish, consulted with faculty in the Romance Languages Department at the University of Missouri who specialize in historical linguistics, and consulted with the archivist for the Sycuan band of Kumeyaay. It is my hope that the finished project communicates the correct meanings of sometimes antiquated terms and is sensitive to the ways in which the Kumeyaay understand their relationship to this period in their history.

The Spanish priests at Mission San Diego used a variety of terms to describe different aspects of the mission population, including terms for age categories, legitimacy of births, and causes of death. A full list and translation can be found in the user's manual for the Early California Population Project (The Huntington Library 2006). Most relevant to readers of this dissertation include: *recen nacidos*, a term used to indicate a recently born child (often the priest will indicate some estimate of how recently the child was born, in days, weeks or months); *parvulo/a*, a term that today translates as infant, but in the records indicates a young child, perhaps older than an infant but definitely younger than nine years old (the age of majority after which an individual needed to undergo catechism before being baptized into the Catholic Church); *muchacho/a* and *nino/a*, commonly translated as "boy" (or "girl"), again used to indicate a child younger than about nine years old; *adultos*, adults of some age over

nine; *Viejo/a*, a term indicating an older person, though there does not appear to be a consistent age after which a person was thought to enter this category. There is not a clear guide to the ages at which individuals move between these categories, and estimates are based on each priests' understanding of the limits of these categories and competency of guessing the ages of individuals in the mission population.

The concepts of colonization and conquest can be problematic in terms of understanding the dynamic of the mission. Most historians frame the events of the 18th and 19th centuries as Spanish colonization of Alta California. The Kumeyaay and other Native American scholars prefer the term "conquest" to "colonize", as it emphasizes the military nature of the program, as well as the idea that Spanish forces subjugated local populations, as opposed to just settling among them. I think that historians would agree with the notion that the Spanish entrada into the Western Hemisphere represented a subjugating force and that indigenous populations under Spanish rule were not merely accepting Spanish populations, but were politically, economically and militarily controlled by various segments of Spanish society. Therefore, I think it appropriate to use the term conquest in this dissertation. Additionally, the period of Kumeyaay history prior to the Spanish conquest (known as the "prehistoric" period by historians and archaeologists) should be termed the "pre-conquest" period, according to Kumeyaay preference.

Other problematic terminology includes the term "tribe" to describe Kumeyaay political organization. The archivist for the Sycuan band expressed discomfort with the term "tribe", preferring "band" instead. Anthropologist Alfred Kroeber thought that the

term tribe was not the best descriptor for the political organization of native Californians and coined the term “tribelet” (Kroeber 1955), but this is not preferred by many native Californians today. “Band” is used by many of the indigenous groups in California, and thus is used in this dissertation. It is also used by the Bureau of Indian Affairs in official documentation recognizing indigenous groups at the federal level.

I did my best to ensure that my discussion is sensitive to all stakeholders’ perceptions of the historical and political reality as well as consistent with documented historical facts. Any errors in labels, meanings or interpretations are solely my responsibility.

CHAPTER 2 – AGENT-BASED MODELING IN ANTHROPOLOGY

The project described here uses an agent-based computer simulation model to test research questions about the impact of infectious disease in the historical community of Mission San Diego de Alcalá. The model was developed with the goal of addressing specific questions about this particular community that were posed as part of a larger research agenda. Model development was always in service of larger research questions, discussed in chapters that follow this one.

It is important to make the distinction between the model as a tool that was used during this research and the model as a project in itself. Model development came with significant challenges and required training and the acquisition of new skills, but the project described here was not to build a model that functioned as a realistic disease model. The project was to use a disease model in conjunction with other information (including historical and ethnohistorical data) to test research questions. That being said, it is important to understand some features of modeling theory and the application of models to various sub disciplines within anthropology and other, related fields.

In the most general terms, a model is a conception about the way a system functions. Models aim to understand causal relationships and predict outcomes with some level of accuracy. Epstein (2008) asserts that the human mind naturally models, in that it constantly seeks to understand organizing principles behind phenomena (natural and man-made) and wants to use known information to predict the unknown. Most

people don't recognize themselves as creating models, Epstein claims, because they are not creating explicit (written) models.

Numerous strategies have been developed for creating models, including mathematical modeling, statistical modeling, computer simulation, network modeling, and geographic models that use Geographic Information Systems software. All of these techniques have their own strengths and weaknesses, and it is up to the modeler to choose the strategy that is most appropriate to address the question at hand.

This project relies on the use of an agent-based computer simulation model (ABM) that examines the spread of an acute, infectious disease within a human community. As such, it is beneficial to place the model used here into the larger context of modeling within the social and medical sciences.

Computer Simulations and Individual-Based Models

As described above, the term "modeling" refers to a way of approaching research questions involving systems and the ways in which parts of systems behave that result in certain phenomena. Computer models are models that are developed to be run on a computer. They are generally written in a programming language or developed using a consumer-oriented modeling software package. Because of the computing power in modern computers, computer models tend to be more complex than non-computer models. Processors do all the work of solving the equations over the course of a simulation, so developers can put their energy into developing a model that accurately reflects the phenomenon of interest.

While it is hard to identify the first computer simulation model, it likely was designed and implemented during the middle of the twentieth century. As with early computers, the earliest computer simulations were carried out as part of military research (e.g. Anderson and May 1992; Epstein et al 1996), though civil (Jennings and Dickins 1958) and business (Ferber and Schenk 1960) applications soon followed. All of these early applications were used to investigate questions involving the use of physical space, such as targeting of projectiles, particle flow, troop movements or commuters during rush hour. They did not attempt to address heterogeneity in human behaviors, thoughts, or ideas that would lead to complex behaviors or change over time.

As social scientists began to employ computer simulations in the 1950s and 1960s, they used computer simulation models to address psychological processes (Reitman 1960) and to investigate theories of the mind (Hovland 1960). Soon, computer modeling was used by researchers across academia to address questions covering such disparate topics as genetic evolution, business management approaches, and understanding the effects of urban processes on populations of wild animals. Today, computer modeling is an integral part of research in almost any academic field, and the wide variety of computer modeling strategies allow for more precise and more complex models.

Individual-based modeling (sometimes also called agent-based modeling) is a strategy in which individual actors interact within a specified modeling environment. Each individual has certain rules that govern its behavior within the course of a model run. Individual-based models often incorporate some stochasticity (randomness), which

increases the complexity of the model but can also help the model reflect reality more accurately.

Model “individuals” are different from “agents” in important ways. Agents, as defined in computer modeling, are individual entities with the ability to modify their behavior based on information that they gain about the model environment. Agents often have the ability to learn new information about the environment as conditions change. For example, Reynolds’ (1987) model of flocking and schooling behaviors showed how small modifications in individual behaviors, like proximity to and physical orientation of other individuals near to an agent, could lead to large scale phenomena, like flocking or schooling behaviors. In these types of models, agents have a feedback loop of information and have the opportunity to change their own behaviors based on new knowledge. Additionally, agents are able to share information with each other (Gilbert 2008). The individuals in individual-based models do not have the same ability to analyze and react to changes in the model environment. Increasingly, the terms agent and individual are used interchangeably, so that the distinctions outlined above may not always apply (Railsback and Grimm 2011).

A typical individual or agent-based model consists of at least three elements: the “world”, the agents and the rules. The world is the environment in which the simulation is to occur. The shape of the world can vary depending on the needs of the model. Common forms include a rectangular grid, a torus, or an open container. The movement of agents in the world can be limited so that agents only may travel along certain paths, inhabit certain grid points, or must stop when an edge is reached. In ecological models,

the world is often stocked with resources that can then be gathered or consumed by the agents. It is possible to let the world change as a result of agent behavior during the course of a simulation run, for example letting resources regenerate at different rates depending on their use by agents. Some models use GIS or other methods to create advanced worlds that include topography, hydrology or other environmental factors. Theoretical models may use abstract worlds that are simple rectangles or tori with very few other features.

Agents are the active participants in the model. In anthropology, agents tend to represent either individual people or households, but in other disciplines the agents may represent animals, particles or other items. Agents can be either homogenous or heterogeneous, depending on the needs of the researcher. Agents are given rules that govern movement, interaction with the environment, and interaction with other agents.

Once a simulation run begins, these rules govern the outcome of the model. For example, rules governing transmission of a disease that are set within a disease model will determine the likelihood of a transmission event when two agents come into contact with each other. Other rules may include communication of information between agents, background fertility and mortality of agent populations or rules that change agent behavior when certain conditions are met.

Agent-based modeling (ABM) is a relatively recent development in both the fields of anthropological and epidemiological modeling. Agent-based models use individual agents (corresponding to units that have the capability to act independently of other units in the simulation) in the structure of the model. The behavior of those

agents can then be specified to reflect a set of rules that the researcher designs. Social science modelers have embraced the potential that this style of modeling has, and as a result most recent agent-based models have been designed to address social science topics (Heath et al. 2009).

Modeling in Anthropology and Studies of Human Health

Computer simulations have been used to address a variety of topics within anthropology. Probably the earliest use of computer simulation in anthropology was in the study of kinship and other societal structures in small scale populations (Gilbert and Hammel 1966; MacCluer et al. 1971). Gilbert and Hammel (1966) make a strong argument for the use of computer simulation, specifically Monte Carlo simulation, in studies of kinship for the purpose of understanding how population dynamics between and within populations may influence formation of and adherence to marriage rules. MacCluer et al. (1971) used computer simulations to reveal the effects of marriage patterns on the long-term demographics of the Yanomamo. These early attempts were simplistic due to limited technological resources but revealed the potential utility of modeling techniques to anthropologists.

Simulation approaches that originally were used to describe genetic change and evolutionary processes have been applied to the study of cultural transmission (e.g. Mesoudi and Lycett 2009; Mesoudi and O'Brien 2008; Nunn el al. 2006; Reynolds et al. 2000). These studies look at the transmission of information, either from a theoretical standpoint or as it pertains to stylistic features of material goods. They seek to

understand the ways in which information is transmitted between individuals or populations and the relationship between information and evolutionary fitness. These studies all rely on assumptions about how culture relates to the evolutionary fitness of individuals or even groups of individuals and how knowledge may be seen as an asset that allows some to be more competitive in resource poor environments. The notion that culture is a human adaptation that has an effect on human fitness is a way of approaching culture that not all anthropologists may find useful in a methodological sense, but it has utility when trying to understand how human populations interact in various competitive environments.

In archaeology, computer simulations have been used to study site distributions and the interplay between ecological factors and human settlements in prehistory (Johnson et al. 2005; Kohler and Gumerman 2000). For example, Johnson et al. address the ways in which environmental resources constrain the distribution of human settlements on landscapes. Kohler and Gumerman (2000) provide additional examples of the variety of ways in which archaeologists have used computer simulations to address topics of archaeological interest. An agent-based model designed in the same platform as the model used in this dissertation has been used to test assumptions about raw material procurement during the Paleolithic (Brantingham 2003). Additionally, archaeologists interested in site formation processes have used models that include paleoclimatic and other environmental data to compare virtual assemblages to real ones (Clevis et al. 2006; Morrison and Addison 2008).

Computer simulations have been used by physical anthropologists for a wide range of research topics. One of the most common uses of simulation in physical anthropology is to evaluate genetic change over time by modeling mutations or changes in allele frequencies in human or primate populations. Genetic simulations can contribute to the study of primate and human evolution and more recent human population dynamics (Barbujani et al. 1995). Studies of locomotion and biomechanics also benefit from the use of computer models, which allow fine-grained definition in the analysis of physical movement patterns (Nagano et al. 2005).

Computer simulations are also commonly used by anthropologists interested in health and epidemic disease to study how culturally mediated behaviors affect disease transmission and other aspects of health in populations of anthropological interest (for example, Herring and Sattenspiel 2007; Carpenter and Sattenspiel 2009; O'Neill and Sattenspiel 2010).

With the increasing number and scope of agent-based models, research in epidemiology that uses ABM now include research into the effect of containment and vaccination strategies (e.g. Longini et al 2005; Ferguson et al 2005; Eubank et al 2004) and other kinds of public health measures that might affect the damage, both physical (Hotchkiss et al 2005) and economic (Bagni et al 2002), caused by the spread of infectious disease. This study will contribute by introducing an agent-based model specifically designed to deal with the kind of population that anthropologists interested in disease commonly study: small, kin-based, and without access to the benefits of contemporary medicine.

Models as Reliable Tools

Models of all types have proven useful for scientific research across many disciplines. Individual and agent-based models are particularly attractive to ecologists, anthropologists, and social scientists, who work with populations of individuals and are interested in the way that individual behavior impacts group outcomes.

As more modeling platforms are distributed, and computers allow people to build models quickly and independently, it must be noted that models can have significant weaknesses and model results should be evaluated carefully and validated rigorously. Gilbert (2008) advises that those using models verify and validate their models before analyzing the results.

Verification, according to Gilbert, is the process of making sure that the computer program is bug-free and is working as intended. This is more of an issue for those writing model applications directly in a programming language than for those using a graphical user interface (like NetLogo) to develop a model. In many cases, a bug will prevent the model from working properly, throwing errors or preventing the program from compiling. These errors are the easiest to find. More difficult are errors that are not syntax errors, but that cause occasional irregularities in the way the model works. These types of bugs allow the program to run and appear to work normally, but these behavioral irregularities could influence model output and skew the results of the experiment. To assist with verification, a model should be thoroughly tested while in development. In the case of the current model, test parameters were used or stochastic features were turned off to test behaviors and outcomes. For example, while the

disease spread mechanism was in development, the transmission probability was set to 1.0 to ensure that each contact would result in infection. If two individuals contacted each other but did not pass on infection, it was obvious that the process was not working as expected and needed to be fixed. Other verification strategies included: observing the visual output of the model to ensure agents were behaving as expected, writing extensive comments into the code that would allow others to follow the activities in the model, and using a programming environment (Eclipse) that would notify the developer of minor syntax or spelling mistakes that could trigger a failure.

Validation is the second step that should be undertaken once a model is complete, but before the model is used to answer research questions. Validation is the process of analyzing the model output in order to make sure that it is faithfully representing the system is was designed to represent. If a model has been verified, the computer program works; if it has been validated, it can then be used to test hypothesis and answer the research questions for which it was designed. Validation is an important step because it allows the developer to evaluate the impact of each parameter in the model and support an argument that the model is a realistic representation of the study phenomenon. Two of the most common ways to validate a model are via sensitivity analysis and comparisons with known examples.

Sensitivity analysis is a method of testing the relative effects of each parameter against other parameters in order to see the effect on model output. Each parameter is systematically varied and the model output is compared. This process reveals which variables make the most difference in model behavior. For example, sensitivity analyses

for a model closely related the one used for this project revealed that changing the values for the “Latent Period” variable predictably changed the timing and length of an epidemic, but had virtually no impact on the size or characteristic of epidemics. On the other hand, model outcomes were extremely sensitive to variation in the “Transmission Probability” variable, with small fluctuations resulting in very different likelihoods of an epidemic occurring and very different types of epidemics in the model population.

Another way to validate a model is to compare the model output to real-life datasets. This is a useful practice when such datasets exist. However, just because a model does not emulate real-life data does not mean the model is invalid. Real-world datasets often have their own limitations, one of the reasons that models are so attractive to begin with. A data set that perfectly matches the model questions may not exist, as was the problem with the current project. Data may be incomplete or biased. A model may be trying to address a theoretical situation for which it would be impossible to collect data. One must remember that due to the limitations of available data and limitations of the complexity of computer models, it should not be expected that a model will completely emulate an existing data set. Instead, the model output should represent the general pattern (for example, a disease model that generates a realistic-looking epidemic). This indicates that the modeler understood the nature of the relationships underlying the data and properly implemented those relationships in the code.

The biggest hurdle to high-quality, valid models lies with the developers themselves. Several common pitfalls should be avoided during model development,

including circular logic, lack of relevant information on which to base the model, and incorrect assumptions about individual behaviors. One of the advantages to the strategy employed during the development of the current model was the deep understanding of the historical and cultural context of the research questions gained through extensive historical and archival research and family reconstitution. The rules of individual behavior were based on primary source documentation of mission life in the 18th century, ethnographic interviews from the early 20th century, and archaeological data (when available). I tried to avoid assumptions about individual behaviors as much as possible, and thus the model is deeply situated in a particular place and time. I was able to avoid circular logic because the relevant disease data for this population do not exist, which prompted this research project in the first place.

A phrase common among modelers is "Garbage in, garbage out", describing the fact that no matter how well a model might run, if it is based on faulty assumptions or poorly thought out relationships, it will not provide reliable data to test hypotheses. Verification and validation are important tools for ensuring that "garbage" is caught and corrected before becoming the basis for someone's hypothesis testing.

Agent-based computer simulation models have proven to be useful tools for anthropologists from all sub disciplines. They have a wide variety of applications across theoretical perspectives and interests. They are a strong platform for applying the deep and detailed knowledge about small or localized populations that many anthropologists have to questions with broad implications for the understanding of human behavior, human origins or human health. As computing power increases and the availability of

good modeling platforms increases, it is assumed that more anthropologists will use models as part of their research methods toolkit. Though models are useful tools, it should also be remembered that models must be carefully constructed and tested in order to ensure that data generated from a model are appropriate for the questions investigated.

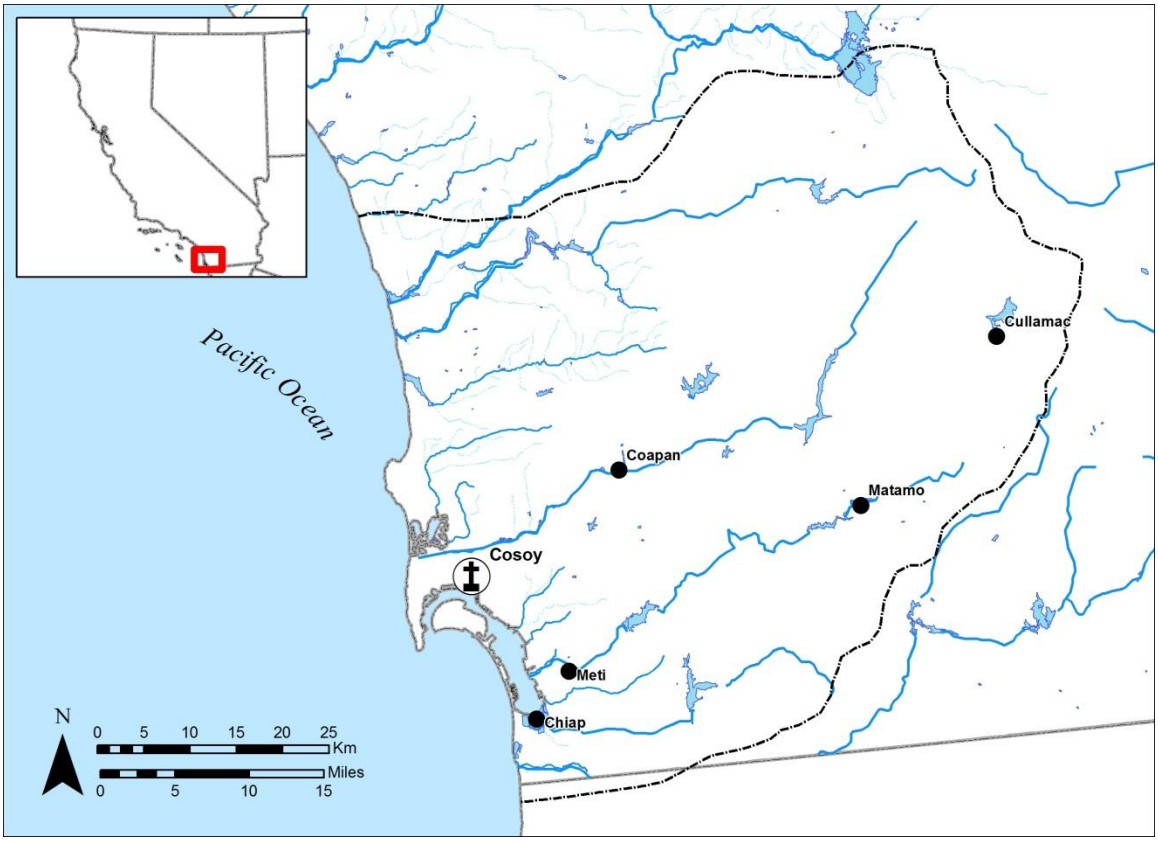
CHAPTER 3 – THE KUMEYAAY: ETHNOGRAPHIC AND HISTORICAL BACKGROUND

The model used for this project was developed using the cultural characteristics of the Native Californian people now known as the Kumeyaay and knowledge about Kumeyaay culture during the Mission Period in California (1769-1833). This nation was the predominant culture group encountered by the Spanish missionaries at San Diego de Alcalá from the time the mission was founded and through all of its existence as a functioning mission. Because they form the historical basis of the model population and the behavior of the agents in the model is based on ethnographic and historical information about the Kumeyaay, it is important to outline the major characteristics of Kumeyaay culture, with emphasis on those features that played a significant role in guiding model development for this project.

Who are the Kumeyaay?

One of the main challenges in understanding the culture of the Kumeyaay comes from the problem of defining them as a culture group in the archaeological and ethnographic record. The term “Kumeyaay” is one of a number of names applied to the culturally and politically allied peoples of extreme southern California. Historically, a variety of names were used to refer to this group, including Kamya, Comeya, Kamia (Shackley 2004), Diegueño, Ipai, Tipai (Shipek 1982), and Quemaya (Gifford 1931). Some of these names result from regional variation in dialects spoken by the Kumeyaay (Shackley 2004) and some were applied by the Spanish and other Europeans (Gifford

1931; Shipek 1982). Hedges (1975) provides a good overview of the history of the name Kumeyaay and the perceptions of this group as a single social unit. Henceforth in this study, the term Kumeyaay will be used to designate indigenous peoples living the area within the dashed line in figure 3.1. The traditional boundaries include an east-west corridor extending from the Pacific Ocean to the Colorado River. The northern extent of Kumeyaay territory is bounded by the Vallecito and Lagunas Mountains, the southern edge extends just beyond the southern border of state of California into the northern portion of Baja California, Mexico (Shackley 2004).



3.1 Traditional Territory of the Kumeyaay (indicated with a dashed line). Mission San Diego indicated by a cross; Kumeyaay villages indicated with dots. Map adapted from an unpublished map held by the Kumeyaay archives by M. Boulanger.

The deserts of southern California consist of several ecozones, with different characteristic plant and animal species and significant differences in climate. Primarily, California deserts are caused by the blockage of ocean air from the west by mountain ranges that run north-south through the center of the state (Hornbeck 1983). The climates tend to be hot and dry with seasonally occurring rains (Kottek, et al. 2006). This geography and climate distinguish the desert regions from the rest of California, which has a greater abundance and diversity of food resources. Thus, the cultures of the Californian deserts are distinct from other native Californian groups as a result of the different challenges their environment presented.

The creation of knowledge about the Kumeyaay

Ethnographers interested in the Kumeyaay first visited the area in the early 20th century (Shackley 2004). At that time, the ethnographic information produced appears to be of the "salvage ethnography" type, in which cultures are described in detail, language is recorded to whatever extent possible, and all ethnographic information is gathered with an eye toward the pre-contact period. This being said, work during this time provides the basis for our understanding of Kumeyaay culture as it may have existed prior to European contact and during the mission period. The problem of culture change and colonization is one that plagues most early ethnographies in North America, as most indigenous peoples had been in contact with European cultures in one way or another for some time prior to the arrival of the first trained ethnographer to their community. This does not mean that ethnographic information is not useful, just that it

must be read with the understanding that significant changes likely occurred over time due to European influence, environmental change, or other unforeseen factors.

The three earliest ethnographers of the Kumeyaay were Thomas Waterman (1910), Leslie Spier (1923) and Edward Gifford (1931), working between 1910 and 1930. Florence Shipek worked with Kumeyaay peoples in the latter half of the 20th century (Shipek 1981; Shipek 1982; Shipek 1985), and currently anthropologist Mike Wilken (Longstreth and Wilken-Robertson 2010; Wilken 2012) continues to work with populations of Kumeyaay living traditional lifestyles in Baja California. The following section provides details on the pre-contact culture of the Kumeyaay to the extent that it played a role in the design and implementation of the disease model used in this project, including family structure, age- and sex-related activities, housing and health practices. For further details regarding other aspects of Kumeyaay culture, the reader is encouraged to explore the ethnographic literature cited above.

Linguistic and Prehistoric Evidence

Kumeyaay dialects belong to the Yuman language family, which consists of related languages in Baja California and along the Colorado River in Arizona (Luomala 1978). According to Luomala, the Kumeyaay variants are somewhat distinct from other languages of the Yuman family to the east and the south, indicating perhaps a more ancient split of the Kumeyaay dialects from the original stock or increased isolation of these cultural groups from each other.

Kroeber (1976) indicated that the Kumeyaay languages could be classified into at least two major dialects, with several variants within each. These dialects were organized by geography, with one occurring in the northern portion of the traditional territory and the other occurring in the southern portion. This linguistic geography meshes well with what is known about how Kumeyaay peoples used the land during the historic period, moving along east-west corridors to take advantage of differing resources from the coast to inland areas.

Archaeological data indicate that people have inhabited the California coastline for at least 12,000 years (Lightfoot and Parrish 2009; Moratto 1984). The first migrants to the area appear to have had material cultures oriented toward exploiting marine and estuary resources, including watercraft, nets, and the knowledge to harvest and process aquatic food sources (Erlandson 2001; Erlandson et al. 2005). Over time, these early peoples also took advantage of inland resources, establishing the earliest populations of inland native Californians by 10,000 years ago (Moratto 1984). Archaeological assemblages of this age are dominated by stone tools, including projectile points, scrapers, and knives, along with hearth features. Ground stone is not known from this ancient time period. The relationship of these early inhabitants to the historically known Kumeyaay is not known, although Warren (2004) indicates that the Kumeyaay may not have entered the southeastern desert region until just prior to the historic era. Present-day Kumeyaay bands are inconsistent, some claiming to be descended from the earliest coastal populations ("Sycuan Band of the Kumeyaay" 2012) while others do not ("Kumeyaay History" 2013).

The next several thousand years of California prehistory provided the time for increasing cultural diversity among native Californian groups. Over time, stone tools increased in complexity and new types appeared in different regions along the coast and inland, indicating diverging cultural traditions (Moratto 1984). The appearance of ground stone artifacts around 5000 B.C. is an important signal of changing diet as these artifacts were used to process acorns, grass seeds, and other vegetal foods. The creation of fired clay pottery during the late prehistoric period distinguishes the California desert region from the rest of California, in which woven basketry was used in place of pottery for food cooking and storage. This is an important distinction, as it connects the culture groups of the desert regions to groups along the Colorado River and the greater American Southwest (Kroeber 1976).

Archaeological data can provide more information on the time depth of Kumeyaay use of this region. Moratto (1984) describes archaeological evidence that suggests that up until about 1000 B.C., the coastal area extending from San Diego to Santa Barbara was inhabited by a single culture group, known by archaeologists as the San Dieguito or the La Jolla culture. Sometime around 1000 B.C. the archaeological data indicate a decline of the La Jolla culture in areas inland from the San Diego coastline with a possible replacement by Shoshonean (desert) peoples. It is during this time that the ancestors of the Kumeyaay are recognized by archaeologists in the archaeological record. It is not clear if they arose *in situ* from the declining La Jolla peoples or if local populations were replaced by Shoshonean peoples, as in the case of the Luisenos to the

north (Moratto 1984). The name for the archaeological horizon most likely representing the forebears of the Kumeyaay is the “Cuyamaca Complex”.

Primarily, archaeological evidence for the late prehistoric period in the San Diego region points to a generalized hunting and gathering lifestyle in which maritime resources were incorporated by those groups with access to the coast. Vegetable food sources were dominated by grass seeds and acorns, like much of the rest of California. Where possible, it appears that horticulture was employed to grow such crops as melons, squashes and gourds.

While the archaeological data are important for understanding the development of cultures in prehistoric California, the assemblages from the southern California coast do not speak to many of the cultural attributes that historical epidemiologists may find most useful in understanding disease in prehistory. Burials are rare and, due to the mortuary tradition of cremation in this area, skeletal remains cannot generally be analyzed for evidence of disease. Housing was ephemeral and left only minute traces in the archaeological record, making population estimates difficult. Diet can be reconstructed to a degree, based upon flaked stone and ground stone artifacts, zooarchaeological data, and on the chemical analyses of residues on tools and isotopes in skeletal material. Most problematic, however, is understanding the culture history of the Kumeyaay. Until archaeologists are better able to reconstruct the history of this population, it will be difficult to know how much of the information gained from the archaeological record can be applied to our understanding of this particular culture. Further hindering efforts to build this culture history is the fact that during the 20th

century this region became highly developed, resulting in the loss of much of its archaeological resources through inadequate or complete lack of recording and excavation.

Kumeyaay Culture and Model Development

The Kumeyaay, like all North American indigenous groups, did not have a written native language, meaning that all written history of this group was produced after they came into contact with Europeans. In the case of the Kumeyaay, contact was initiated by Spanish explorers and missionaries during the late 18th century, followed by American explorers and settlers in the mid-19th century, and finally ethnographers and linguists in the early 20th century. Each of these groups of chroniclers had unique purposes for fostering contact with native peoples, and their writings are, of course, colored by their intentions. Background research for the development of the agent-based model described in this dissertation relied on evidence from all types of sources and was found in archival materials, published articles, memoirs and journals, drawings and paintings, and archaeological reports. The following lays out the features of Kumeyaay culture that played a significant role in guiding development of the model.

Villages and housing

In a narrative of the first overland expedition into Alta California from Baja California in 1769, Miguel Costansó recorded coming into contact with several native villages. He noted that the houses were “pyramidal in shape, covered with earth” and that each village appeared to contain 30-40 families (Costansó et al. 1910). This

description of native villages was corroborated by another chronicler of the Portola expedition (Crespí and Brown 2001; Smith and Teggart 1909). Ethnographic sources indicate that the type of houses described by Costansó were more typical of winter housing and that in the summertime, people sought shelter under trees and did not build any structures (Shackley 2004). It is possible that populations in the milder coastal climate may have built houses during parts of the year that their inland desert neighbors did not. Leslie Spier's informant reported that two brothers and their families would often share a single house (Gifford 1931). Gifford also reported that sometimes a small outbuilding for an elderly or infirm relative could be found adjacent to a main house (Gifford 1931), but there is no mention of this in other ethnographic or historical sources. During the 1700-1800s, the dominant form of housing began to resemble western-style houses: square or rectangular structures with walls and a roof (Cline 2008). During this later period, it was known that houses were located fairly close together and inhabitants mingled between buildings, especially if they were blood relatives (Cline 2008).

3.2 Kumeyaay woman in front of her traditional house at Campo, photo by Edward Curtis (published 1907-1930). Edward S. Curtis [Public domain], via Wikimedia Commons from Wikimedia Commons

Gifford (1931) claimed that the Kumeyaay did not organize into true villages; instead he described settlements as a scattering of houses in a particular area. He made no further claims about how these Kumeyaay settlements differed from a more general definition of a village and, given that these settlements were organized around lineages and acted as independent political units, it seems reasonable to think of them as villages. Spanish explorers clearly interpreted Kumeyaay settlements as villages (*rancherias*, as they continue to be called today). Twenty large rancherias were noted in

the vicinity of what would become the mission and its grounds (F. Weber 1979). Several of those, including the villages of Cosoy, Jamacha, Rincon, and Nipaguay, have been documented as archaeological sites (Smithsonian Trinomials 4SDi239, 4SDi35, 4SDi202 and 4SDi4782). Figure 3.1 shows the locations of Kumeyaay villages and other important places within their ethnographic boundaries. Ethnographic sources report the villages were semi-permanent, with inhabitants planting crops in one location before moving on to gather wild foodstuffs in another (Shackley 2004).

Subsistence

Spier (1923) notes that the Kumeyaay living away from the mission in the middle of the 19th century still participated in seasonal rounds, a mode of living in which entire villages moved locations to take advantage of seasonally available food sources in different parts of their territories. This often involved fissioning or fusioning of populations along lineage lines, depending on carrying capacity of the resource. In cases of a poor harvest or drought, able-bodied individuals were known to leave villages in search of alternative resources, leaving stored food for the young, infirm and elderly (Shipek 1981). Some, but not all, food resources within a territory were owned and defended by lineages. These included water sources, some fruiting trees, and some types of oak groves. Additionally, eagle aeries were owned by clans for the procurement of eaglets needed for ceremonial purposes.

The California desert region has a highly variable set of plant and animal resources. Proximity to seasonal water resources, differing elevations and changing

climate all affect the type of food available to people inhabiting these areas. Because of this variability in available food, humans living in the desert have a wide range of subsistence bases and rely on a diverse set of plant and animal resources. Wild foodstuffs, such as acorn, mesquite beans, piñon nuts, wild plum, and grass seeds, formed the basis of the diet for most Kumeyaay communities (Cline 2008; Lightfoot and Parrish 2009). These items all needed special processing with mortars and pestles or *manos* and *metates*. Women were the primary collectors of vegetable foods (except for acorns, which everyone helped to collect) and each family had a granary in which seeds or acorns could be stored (Cline 2008).

Deer, rabbits, lizards, and rodents were added to the diet when possible. Deer were hunted individually, primarily with a bow and arrow, and the meat was shared with others in the village. Rabbits could also be hunted individually, using a bow and arrow, throwing stick, or spear or they could be harvested *en masse* via a drive. A rabbit drive would include any villager that wanted to participate and each hunter kept his/her own kill. Sometimes, a brush fire was set in order to provoke the rabbits into running a particular direction. The rabbits were driven into nets and then clubbed or speared (Cline 2008; Lightfoot and Parrish 2009; Luomala 1978). Waterfowl and other wild birds were also taken, when and where available (Cline 2008).

Though the Spanish observed an abundance of terrestrial wildlife during their overland expedition, Costansó (1910) noted that native peoples they encountered appeared to rely on fish and shellfish as a major part of their diet. This likely varied across the territory with differing levels of availability of animal resources.

Kumeyaay health and healing

Physical descriptions of the Kumeyaay vary depending on the recorder, with some noting that the Kumeyaay were physically pleasing to look at, others indicating that they were "coarse" and unpleasant looking (Costansó 1910; Jayme and Geiger 1970; F. Weber 1979). These descriptions were surely influenced by the context in which the writer encountered the band. Rarely did the writers of the earliest descriptions mention indicators of either relative health or illness when describing interactions with the Kumeyaay.

Health-related ethnographic information about the pre-conquest Kumeyaay is very sparse and must be gleaned from the 20th century ethnographic record. Ethnographers recorded Kumeyaay strategies for vermin control, including the abandonment and burning of houses, mud and paste applied to the hair to control lice, and hair cutting (Shackley 2004). This indicates that even though most housing was temporary, vermin could be an issue in Kumeyaay villages. Other information recorded includes special activities surrounding pregnancy. Pregnancy was a time of food taboos, including meat and salt (Waterman 1910). Some short rituals were performed just after birth that, in effect, cauterized the navel of the neonate (Waterman 1910). Additionally, a period of semi-isolation for the mother and child was common (Cline 2008). Very little can be said about pre-conquest infant or maternal mortality rates, given the lack of documentary or archaeological evidence.

Traditional Kumeyaay pharmacology was documented in the modern era by Garriga and Weber (1978), and includes remedies related to menstruation, fluxes

(dysentery), diphtheria, malaria, scarlet fever, smallpox, and syphilis in addition to remedies for other common ailments. The inclusion of so many infectious diseases of European origin may tell us something about the disease load borne by Kumeyaay populations in the historic and modern periods.

In the pre-conquest era and in non-missionized villages, medical treatment would have been the responsibility of a shaman. This individual could be either male or female and would have had religious as well as medical responsibilities in the community. For neophyte populations, shamanic healing was discouraged and reliance on the friar or a doctor was encouraged. Shamanic healing practices included the use of herbal remedies (often in the form of a tea or poultice), the laying on of hands, or other ceremonies that included sucking or blowing on the sick individual by the shaman (Kroeber 1976).

Political structure

Each lineage had a chief, recognized by the rest of the village as a leader for ceremonies, an arbiter of disputes and as someone who was generous with his own personal belongings. A chief could be a hereditary position, including blood relatives outside the nuclear family if there were no heirs or the available heirs were thought to be unfit for leadership (Luomala 1978; Shackley 2004). Typically, chiefs were male, though it was possible for a female to serve as a chief. In this case, the woman was most likely a respected elder or perhaps a shaman prior to serving as chief (Cline 2008).

Ethnography of post-mission period Kumeyaay culture describes the political system of a typical village as having a council that included a chief, a shaman and village elders (Cline 2008). A council is not mentioned in earlier ethnographies, though tribal councils exist in all of the currently federally-recognized Kumeyaay bands.

Rites of passage and marriage

Upon the birth of a child, both mother and child would be isolated from the community for about two months, emerging from isolation to participate in a naming ceremony (Cline 2008; Kroeber 1976). This ensured that the child would not come into contact with "wrong spiritual influence" (Cline 2008:64), but also functioned to keep the mother and child away from any communicable disease at a vulnerable time for them both. If a child was left motherless or an orphan, it was usually up to the mother's female relatives to care for it, though in some cases, the father's female relatives could also be looked to for support.

At adolescence, boys participated in a ritual (the *Taakaayp*) involving consumption of jimsonweed, a plant of the genus *Datura* that has a hallucinogenic effect when taken in the proper dosage (Waterman 1910). According to Spier (1923), not all boys underwent this ceremony, as it was dangerous and could cause death. According to the Kumeyaay, ingestion of this substance can give the user shamanistic powers or access to superhuman powers, such as fire walking or witchcraft (Waterman 1910). Dancing followed the initial ingestion of the extract, followed by a sleep that may have included a dream or vision. If the dream included a song, the dreamer took

possession of that song for future use in religious matters. Food was restricted for about one week, with only small amounts of grain mush consumed by the initiates. Each day for three days after ingestion, the participants formed a procession in which they crawled on hands and knees around the village, prompted by older men of the village. Later in the week, the initiates were taught dances and songs, and they participated in footraces. These events taught them knowledge and values they carried with them through adulthood. For some period of time between a week and a month, the boys and their adult male "sponsors" spent time learning sacred mysteries in the form of stories and songs and creating sand paintings (Shackley 2004). The rite ended with a ceremony involving jumping over a representation of a man made from straw, singing and further dancing.

Girls transitioning to adulthood had their own rite of passage (also called the *Taakaayp*), which took place whenever there were enough girls around the correct age. They were given a tea of tobacco and then "roasted" in a heated, herb-lined pit for several hours to several days (Waterman 1910). In addition, a ceremonial stone was heated and placed against the girls' abdomens or between their legs. This was thought to help make motherhood easier for them in the future. During this time, the girls had to avoid meat and salt and avoid looking at other people. Waterman noted that this was a relaxing time for girls, who chatted and slept during the ceremony. Older women were in charge of the ceremony and would sing and dance around the roasting pit while the ceremony was taking place. After completing the ritual, a celebration with people from

neighboring villages occurred and the girls received facial tattoos. Slight variations in the ceremony are further described by Waterman (1910).

Both Gifford (1931) and Cline (2008) studied marriage practices of the Kumeyaay; much of the discussion that follows stems from their work. Lineage exogamous marriages were preferred, but not required (Shipek 1982; Spier 1923). Typically, the young man decided when he was ready to marry and began to court a young woman by bringing gifts of food to her house. Her family would signal approval and encourage the girl to marry. There was no special ceremony, but the new couple built a house for themselves, preferably near the groom's family, but sometimes by the bride's family (Cline 2008). Polygyny was accepted, but uncommon, because of the amount of work needed to support two wives. In the case of polygynous marriages, the wives were often sisters or female relatives. Divorce was allowed, and could be initiated by either spouse. Causes for divorce include abuse, infidelity or incompatibility. No compensation was expected by either party, who would return to their natal homes. Remarriage could occur quickly.

Funerals were simple affairs, primarily consisting of the cremation of the deceased along with their belongings (Kroeber 1976). Men would press on the body before cremation and blow on it to tell the spirit to go east (where all spirits were thought to go). Personal artifacts were burned along with the body in order to persuade the spirit that there was nothing left for it in the village. The ashes from the cremation could be gathered up, placed in a jar and stored among rocks or buried away from the village. The Kumeyaay do not appear to have used "graveyards" or areas where many

people's ashes would have been stored. The ashes and any other artifacts of the deceased would have been avoided in order to avoid drawing the attention of their spirit (Kroeber 1976).

At least one year after the death of an individual, the family would be responsible for hosting a memorial ceremony (the *Karuk*). Guests were invited, food prepared and gifts assembled to give away to guests or to burn in the fire at the ceremony. The event took place over several days. The first day was used for mourning the deceased. Later nights were dedicated to singing, dancing, and burning items belonging to the deceased that were specially laid aside after death. Gifts were distributed to guests at this time. Several variations of this ceremony were observed, depending on the number of dead being commemorated and whether or not the deceased had been initiated as a man in the jimsonweed ceremony.

The Kumeyaay today

Currently, the United States Government recognizes two bands of Kumeyaay peoples, the Sycuan and Ewiiaapaayp bands, and eight bands of Diegueño peoples (several of whom self-identify as primarily Kumeyaay) in southern California. According to the US Census bureau, approximately 2500 Kumeyaay and Diegueños live on recognized Indian reservations or in recognized tribal statistical areas as of the year 2010 ("2010 Census Tribal Statistical Areas Program" 2009).

The contemporary bands of Kumeyaay are varied in terms of their land holdings, economic bases and cultural activity. The Sycuan band of Kumeyaay hosts a community

college campus on their reservation lands, the Kumeyaay Community College (www.kumeyaaycommunitycollege.com), which offers a Certificate in Kumeyaay Studies. The certificate program focuses on language, history and culture of the Kumeyaay people. Other reservations have history centers and museums that are open to both band members and the public. Several bands developed casinos on tribal holdings, which provide an economic base for the Kumeyaay populations in San Diego County. Some Kumeyaay people in Baja California live in ways that are more similar to historic and pre-conquest populations (Wilken 2012).

This chapter has briefly outlined what is known about Kumeyaay culture as it existed both before and outside of the mission system. The friars, soldiers and other visitors to the mission did not record much of the Kumeyaay culture at Mission San Diego, so this ethnographic knowledge is used as a basis for understanding the cultural *milieu* of neophytes at the mission. Of course, many of the practices mentioned in the ethnographic record would have been negatively impacted by the strictures of the friars at the mission. Polygamy would not have been allowed, marriage was a more structured relationship, and subsistence was drastically changed, as will be discussed in Chapter 5. The ethnographic baseline provided here allows for the development of a more realistic model that has behaviors that are grounded in a deeper understanding of the Kumeyaay cultural system.

CHAPTER 4 – SPANISH CONQUEST AND CONVERSION IN ALTA CALIFORNIA

The goal of the Spanish Mission system throughout the Americas was to convert the indigenous inhabitants to Catholicism, transform indigenous ways of life to reflect that of European peoples, and ultimately have a population of Spanish citizens in the colonies that would cement Spanish rule in the Western Hemisphere and repel attempts by other colonizing powers to establish colonies of their own (Costansó et al 1910). In reality, the administration of missions in different parts of the Spanish colonies varied dramatically depending on the colonial administrator of the region, which religious house was currently in charge of the missions, the level of sustained agricultural production of individual missions, or even the administrative style of individual friars at particular missions.

The conquest and colonization of Alta California in the mid-18th century took place almost two hundred years after the first Spanish colonial incursions into Mexico and the Caribbean (which started in earnest in the middle of the 16th century). The motivations and justifications behind these efforts had evolved from earlier times, as had the ability of the Catholic Church and the Spanish crown to finance these forays. The path to the establishment of Mission San Diego includes a complex web of interactions between the Spanish, Portuguese, French and Russian governments, the Vatican, the Jesuit and Franciscan religious orders and the numerous indigenous peoples of Mexico and the American Southwest. To understand the establishment of Mission

San Diego and guiding principles of those who ran it, one must first understand the historical context from which it emerged.

Spanish colonization in the Western Hemisphere

Legal justification

The stated justification for Spanish colonialism in the Western Hemisphere comes from a series of three papal bulls issued by Pope Alexander VI in May and September 1493, shortly after the 1492 "discovery" of islands in the Caribbean by Christopher Columbus. These bulls, known as the "Alexander Bulls" or the "Papal Donation" gave exclusive rights to territory, peoples, and resources of all lands in the Western Hemisphere to the Spanish Crown, provided the Crown colonize and convert all peoples in these lands to Catholicism and continue to govern Spain as a Catholic kingdom (Hoffman 1973). Enforcement of these proclamations was almost immediately adjusted via the Treaty of Tordesillas, a treaty between the Spanish and Portuguese governments. This treaty modified the Spanish claims to exclude the far eastern edges of North and South America, including present-day Brazil and Newfoundland, and agreed to Portuguese control of these territories (D. Weber 1992).

Other Catholic colonial powers, including England and France, took a rather dismissive view of the Papal Donation and spent much of the 16th century quietly carrying out explorations in North America while laying out diplomatic and legal cases against the details of the bulls. For example, the French King denied that the Pope had any right to apportion newly discovered lands and that, even if the pope were to have

that authority, the bulls could only apply to those lands that were known and claimed at the time the bulls were written (D. Weber 1992). In 1535, Pope Leo XII supported the French interpretation of the bulls, weakening the Spanish case for its rights to the entirety of the Western Hemisphere (Hoffman 1973). The French and English both argued that territory in the newly discovered lands should be claimed by those asserting "rights of discovery, conquest, and settlement" (Hoffman 1973, p. 162). With the loss of papal support for their claims, the Spanish Crown was then forced to accept the French and English interpretation. This interpretation would shape Spanish strategy during the colonization of Alta California, as they designed a mode of colonization that maximized territory claimed while minimizing the resources and manpower put into that territory by the Spanish crown.

Other challenges to the Papal Bulls came from within the Catholic Church, as Bartolomé de Las Casas, a former hacienda owner and ordained Dominican friar, presented his interpretation of the text of the bulls to critique maltreatment of indigenous peoples by Spanish colonizers (Carman 2010). Las Casas' argument stemmed from a passage that included a directive to "subject" native peoples to conversion, which was sometimes used as a justification for violent action against indigenous populations who resisted Spanish conquest. Las Casas argued that by "subject" Pope Alexander meant rather to convince through force of argument, not through physical force. In 1550-1551, Las Casas participated in an important dialog where he developed his case for the moral and just treatment of all native peoples of the Western Hemisphere (Canteñs 2013). This dialog and the ensuing discussions would have been

known by those involved in later colonizing efforts and probably shaped their attitudes about their activities.

Economic development

The colonization of the Americas was as much an economic enterprise as anything else. The discovery of gold, silver, fur-bearing animals and other valuable trade resources made the Americas attractive to Europeans seeking to profit from colonization. The earliest Spanish settlers in the Americas brought with them the *encomienda* system, an economic system similar to a European feudalism, where *encomenderos* were granted land by the Spanish king in return for military or other special services. *Encomiendas* not only included a land grant, but also rights to the labor of the populace (presumably subjects to the crown) living on the land (D. Weber 1992). *Encomiendas* could be hereditary for several generations, but not for perpetuity. *Encomenderos* would profit off these grants, but were also supposed to provide for the populace, both physically and spiritually.

Almost from the beginning, *encomenderos* were accused of mistreatment of natives. Religious and civilian reformers attempted to bring abuses to light, starting in the early 16th century (Castañeda 1949). The Spanish king, though he was dismayed, was convinced by colonial leaders that the *encomiendas* were necessary to support colonial economies. In 1542, King Carlos V decided that the reported abuses could no longer be tolerated and he issued a proclamation, known as the "New Laws" that sought to dismantle the *encomienda* system and reform the way that native peoples were treated

by colonists (Castañeda 1954). This was only partially successful, and lead to a patchwork approach, whereby land grants were still sought and issued and native peoples were employed more informally. The dismantling of the *encomienda* system did not end abusive practices directed toward indigenous workers (Cuello 1988).

Religious involvement

The Catholic Church played an integral part in Spanish colonial efforts in the Western Hemisphere. As discussed above, it was the Pope who gave spiritual authority to the Spanish Crown's desire to annex the lands, peoples and resources of the New World. Along with the blessing of the Vatican came the participation of numerous missionaries from at least three Catholic monastic orders: the Society of Jesus (Jesuits), the Order of Friars Minor (Franciscans), and the Order of Preachers (Dominicans). Each order played a role in the creation and administration of missions in the new colonies.

Initially, the Jesuits had been in control of the missions of Baja California and looked to play a large role in any activities in Alta California (F. Weber 1979). The Jesuits fell afoul of European politics, however, and in 1767 they were expelled from Spain and all Spanish colonies (having already been expelled from Portugal and France earlier in the same decade) (Roehner 1997). The void left by the Jesuits in the missions of Baja California was filled by Franciscans, who took on the job of administering existing missions until they were assigned to the task of establishing missions in Alta California. Once the Franciscans were asked to move to Alta California, the Dominicans then took charge of all the existing missions in Baja California (James 1913).

By the time that serious thought was given to colonizing Alta California in the mid- to late-18th century, many of the initial conflicts between colonizers had been settled (to some degree), and a basic level of stability was in place in terms of colonial economies, power structures, and international relations (D.Weber 1979). This, of course, would change as power shifted in continental Europe, the Russians began to expand in search of new economic resources, and citizens of the British colonies in North America headed west in search of land and resources for economic expansion.

Spanish colonization of Alta California

Though the Spanish had occupied lands to the south and east for over two centuries, pressure to explore, conquer, and settle among peoples of Alta California did not increase until the Spanish government in Mexico became concerned about the role of foreign governments in supporting trappers and traders along the coast of present-day Washington, Oregon, and Northern California (D. Weber 1992). Additionally, Spanish missions in Baja California needed reorganization after the expulsion of the Jesuits in 1767. These two major factors, along with the political ambitions of colonial leaders in Mexico led to planning for a Spanish expansion into Alta California.

In order to reassert its claim to this portion of the New World, the Spanish government decided to embark upon a strategic settlement of its northern territory. Plans were modest and included establishment of a series of presidios and missions, each about a day's walk from the next, which could provide both Spanish eyes and ears on foreigners in the area and resupply centers for ships headed to or from trans-Pacific

trade with other Spanish colonies (e.g. the Philippines). Of course, the Catholic Church had its own goals in agreeing to participate in the settlement of Alta California, primarily the conversion of any native peoples encountered by the friars.

History of colonization

Davidson et al's (1887) guide to exploration of the California coastline provides a detailed overview of the locations of most of the early Spanish voyages up the coast with maps and original diary entries to guide the reader. The follow discussion is based on information found in that guide, focusing primarily on entries in or around San Diego, though information from other sources is also included.

The first Spanish expedition to enter present-day California was the voyage of 1542 led by Juan Rodríguez Cabrillo (D. Weber 1992). At this time, California had a mythical place in the Spanish psyche. A number of different stories had been told about the land and peoples to the north of Mexico. Included amongst these were tales of large amounts of gold and races of Amazon-like female warrior societies (Bouvier 2001). Spanish explorers of this time were directed by the monarchy to find these societies in order to "civilize" them and get access to their reputed wealth (Bouvier 2001). Additionally, Spanish explorers and later priests were ordered to attempt to convert any native peoples encountered to Catholicism (Bouvier 2001; F. Weber 1979). The diaries of Cabrillo himself reflect the constant search for people and information. Additionally, the diaries of others on his expedition indicate some hope that they might encounter a route west to the Spice Islands of the South Pacific (Kelsey 1986). Ultimately, Cabrillo

died on this voyage, and very little in terms of settlements, maps, or other useful information was obtained.

The reaction of the first native peoples encountered by Cabrillo (generally fearful, often fleeing) indicates to some authors (e.g. Bouvier 2001) that these southerly tribes may have had prior negative contact with overland Spanish explorers, possibly the Cortez or de Alarcón expeditions of the 1530s (Kelsey 1985). It is likely that even if they had not had direct contact, indigenous communities had heard the news of the conflict in Mexico, for trade in far southern California often included northern Mexican peoples and routes (Davis 1961). In fact, due to a number of conflicts between the Spanish and native peoples in Mexico, Cabrillo's expedition was ordered to establish any permanent settlements away from native villages to avoid conflict until the land was pacified (Kelsey 1986). The Spanish seemed to expect antipathy from any native peoples they encountered.

After this expedition, there was no major policy directive aimed at settling California until the 1769 royal proclamation initiated the mission period (Engelhardt 1920). Prior to the official proclamation, plans were in place to begin an expedition north. This expedition, led by Gaspar de Portolá, was to consist of two ocean-faring and two overland parties that were to explore the Pacific Coast and interior routes from Baja California to San Diego Bay (Teggart 1909). In 1768, the ocean-faring parts of the expedition started up the California coast. In 1769, the two overland groups departed from Baja California with orders to meet the ships in San Diego and establish a formal Spanish settlement there (D. Weber 1992). When the overland group arrived in San

Diego they found that the sailors had gotten lost and the ships did not arrive for some time. By the time they finally arrived, many of the sailors were suffering from scurvy and there were not enough able-bodied men to sail the ships any further up the coast, complicating plans to continue the expedition northward to Monterey Bay, which had been identified, but not reliably mapped, by Cabrillo (Kelsey 1986).

Leaving the sickened sailors at San Diego, part of the overland expedition continued north in search of Monterey, eventually ending up near San Francisco Bay instead (Natella 1975; D. Weber 1992). Mission San Diego was formally established on July 16, 1769 by a Franciscan friar, Junipero Serra, later to be known as the father of California's missions (Bowman 1964), who had decided to stay in San Diego to minister to sick sailors and begin his mission.

One of the more substantial accounts of the overland expedition was written by Juan Crespí, a Mallorcan missionary intent on helping to establish missions in Alta California (Crespí and Brown 2001). Other Spanish officials (Serra and Pedro Fages, for example), also kept diaries of their travels and later visitors to the missions wrote accounts of their visits that provide a glimpse of what neophyte life was like under the mission system.

Crespí described the people of California as well behaved, affable, pleasant, and clever, though their conditions are "poverty-stricken and wretched" (Crespí and Brown 2001). This description, according to Bouvier (2001), is characteristic of the early reports of missionaries. This may be due to the optimism of these early trips or the missionary zeal that seemed to be a part of the missionaries' personalities. In fact, most of the early

missionary work in California was carried out by five missionaries from Mallorca who had specifically come to Mexico together with the intent of evangelizing among the natives of the New World (Geiger 1947).

Crespí shows what he considers patience when native people "steal" his possessions, often playing it off as playfulness (Crespí and Brown 2001). His tone is paternalistic throughout the text, and he often speaks of attempting to take opportunities to convert native people who served as guides or trading partners. These attempts are usually cut short by the disappearance of the native subject. Crespí interpreted this as unfortunate coincidence, but perhaps this can be read as active resistance to evangelism. Natives were willing to be paid guides and to trade material goods, but any attempt to force European ways, at least in this religious context, was often a failure due to the complete lack of interest. Perhaps the native subject saw his relationship with the Spanish as a purely material one, and as soon as that material relationship appears to be over, he left.

The Kumeyaay do appear in the descriptions produced by Spanish explorers. The documentary record is not as detailed for this group as for other, larger tribes in California (such as the peoples of the San Francisco Bay), but we do learn a little bit about how the Spanish saw the peoples of the first mission through these reports (summarized in Laylander 2000). Unsurprisingly, the characterizations are often contradictory and portray the Kumeyaay as "clever" and "simple-minded", "business-like" and "thieving, murderous", "bold, arrogant" and "docile, friendly", "attractive" and "ugly features", "well-built" and "shorter", "small", "the most degenerate of all people",

"restless…rugged", and "weak". These characterizations clearly have more to do with the individual recording them and the particular interactions between individuals than they do with describing the entire population of Kumeyaay.

The Spanish expectations of native governmental structures are not explicitly mentioned, but it appears that they did expect a single leader in each village. It is interesting to note that while the Spaniards easily recognized native men in roles of authority (Kelsey 1986), it appears that women in leadership roles may not have been recognized as such (Bouvier 2001).

It is perhaps impossible to fully separate the economic, military and religious goals of Spanish colonizers but, the demographic makeup of the early expeditions give us a clue to their general aims. On Cabrillo's initial voyage, at least one and possibly two priests traveled with sailors (Kelsey 1986). These priests did not appear to have been given instructions any more specific than their overarching program of conversion and civilization, however. In later times, priestly narratives begin to dominate, providing an alternative view of native peoples.

The beginnings of Mission San Diego have their roots in earlier Spanish colonial efforts. The joint military-religious expedition signals the fragile trust between secular and religious authorities when it came to the potential exploitation of resources and people in the New World. The fact that Franciscans were given the task of creating the Alta California mission system is the result of events thousands of miles away on the European continent. The relatively late start of colonizing activities in Alta California can be traced to the economic needs of the Spanish Crown, which placed a priority on

resources (like gold, silver, sugar and cotton) that had not been identified in early forays of the California Coast. It also had to do with the nature of the indigenous societies that early explorers found in California. Spanish conquistadores found it easier to take control of societies with strict hierarchies and defined territories (e.g. the Aztecs and Incas). The fluid group structures of many Californians meant that social hierarchies either did not exist or had very little power. Tribal boundaries were fluid and people could cross territories with little fear of reprisal. Spanish politicians and military leaders thus did not see any benefit to attempting to colonize these territories until it looked like other nations might do so first.

CHAPTER 5 - THE KUMEYAAY AT MISSION SAN DIEGO

As this project focuses on neophyte life at Mission Basilica San Diego de Alcalá, an in depth review of the history of the mission is provided. The historical information presented was crucial when developing the model used in this project. Such information as building attributes, diet, gendered and age-graded activity patterns, and the schedule of native life at the mission provided the base on which agent activity was programmed in the model.

Numerous primary and secondary sources were consulted; however, all the sources were produced by Europeans, including the Spanish and Mexican friars, American, Russian and French visitors to the mission, and later European-American historians. There are no primary sources produced by native Californians that describe life at Mission San Diego. The result of this lack of native voices is that information about indigenous life at the mission was written entirely from the etic perspective, and often displayed the anti-indigenous prejudices common at the time. Additionally, most of the sources are administrative in nature, and mention the indigenous neophytes only in terms of reporting on the success of proselytizing, keeping the peace, or as factors in the local economy. Cultural information was often indirect and had to be inferred from primary source materials; for example, friars complaining about dancing and singing among neophytes could indicate a continued adherence by the neophytes to pre-contact religious beliefs (which were often expressed through these behaviors). Those

caveats aside, historical research did allow for the construction of a model that is highly contextualized and very sensitive to this particular population.

History of the Mission

In May of 1769, two groups of travelers making up the overland arm of the Portolá expedition arrived in present-day San Diego (Teggart 1909). This group consisted of 40 Spanish and Mestizo soldiers, 30 Indian volunteers from the missions in Baja California to serve as translators and muleteers, the "seed" population of the new mission, and the missionaries assigned to found the new mission (including Junípero Serra, now known the as the father of the California Missions) (D. Weber 1979). On June 30, 1769, the ocean-faring portion of the Portola expedition landed at the port of San Diego (Smith and Teggart 1909). The ship had lost most of its crew to scurvy; only 18 of the original 90 landed at the port (Costansó et al. 1910). Many of the sailors who did survive arrived at the port suffering from advanced scurvy and were weakened or dying.

The military goal of the sea bound expedition was to identify ports along the California coast that could be used as stops for Spanish trade ships and warships. The captain was especially interested in re-locating Monterey Bay, which was first identified by the Spanish in the 16th century. The religious men on the journey were tasked with extending the reach of the Catholic Church through the establishment of missions. Eventually, each mission was to be about one days' ride from the next and they were to provide material support (through agricultural output and animal products) to the

military presence and new settlers. The missions were to be the backbone of the Spanish colony in Alta California.

Physical Layout

The mission was originally established in close proximity to the San Diego Presidio and near the Pacific Ocean. After a series of poor harvests and violent interactions between indigenous people and presidio inhabitants, the mission was reestablished inland (east) several miles from the presidio on a small hill in an area now known as “Mission Valley”. At the time of its founding, the relocated mission was located within 10 leagues (probably 30-35 miles) of 20 large native villages (D. Weber 1979). This placement was thought to be ideal for agricultural production as well as for easy access to a large native population.

Construction of new buildings and renovations of old ones was constant through the life of the mission. The church building would have been the first one built, along with quarters for the friars (Bowman 1964). During the early days, only a small proportion of the converts would have lived on the mission grounds. In fact, during the 1780s, it was estimated that only about ¼ of the mission’s neophytes lived at the mission (D. Weber 1979).

By 1805, the mission grounds held not only the church and friars’ quarters but also a native *rancheria* with an unspecified number of households, a number of occupational outbuildings, and several other buildings. These included a large, enclosed women’s dormitory (known as a *monjería*), a granary, kitchen, pantry, two guest rooms,

a tack room, possibly a toilet or outhouse, guard house, iron implement shed and blacksmithing area (Engelhardt 1920). Visitor Otto von Kotzebue noted the native living conditions as extremely unclean and unsanitary (F. Weber 1991), but did not provide further details about this assessment.

Changes to the natural environment around the mission would have occurred as oak scrubland and grasslands were cleared in order to provide fields for planting. Crops included wheat, corn, beans, olives, and fruit trees (Hedder 1915; Wickson 1888). Rangeland was also needed for the many thousands of cattle, sheep, pigs, horses and mules belonging to the mission by the early 19th century (Duhaut-Cilly 1929). Changes to the local environment surely would have impacted the indigenous populations' ability to continue to gather food in these areas. Some of the early conflicts between unconverted Indians and newly arrived Spanish had to do with the killing of cattle or other domesticated animals grazing in areas traditionally used for foraging by native peoples (Kelsey 1985). The Spanish either did not recognize or were not concerned about the extent of their impact on the natural environment and the further impacts on native peoples.

Daily Routine

The Franciscan friars that ran the mission expected that converts live on the mission grounds and participate in the mission community. The schedule (see Table 5.1) was strict and did not allow for much variation in daily activity by neophytes. Accounts of the schedule described below can be found in Clinch (1904) and F. Weber (1991). The

friars thought that structure was important in order to prevent the neophytes from returning to their old lifestyles. The friars also followed the schedule, as they thought it was important for them to provide a living example of what they considered a "good Catholic" life.

Table 5.1 Daily schedule at Mission San Diego (as recorded in historical documents)

Time	**Activity**
Daybreak (~5am)	Awaken, breakfast, prayer
8am-12pm	Work
12-2pm	Lunch
2pm-5pm	Work
5pm-9pm	Dinner, socializing

Breakfast usually consisted of a bean, wheat or corn porridge (*atole*) and was consumed by the family at home. From about 8 am to noon, adults and older children worked. Women and girls carried out food preparation activities, weaving, sewing or other domestic activities in the mission or village areas. Men and older boys worked with stock animals, in the fields or orchards or at skilled trades. Two hours were allocated for lunch and midday rest. Lunch consisted of a meat and vegetable stew (*posole*) that, like breakfast, was also consumed at home. From 2 pm until 5 pm, everyone went back to work. At 5 pm, work was ended and everyone had supper (another serving of *atole*). Supper was followed by several hours of relaxation, games, or social exchange. The women in the *monjería* were locked in for the night at 8 pm.

Everyone else in the village was expected to retire for the night by 9 pm. On Sundays and Feast days, no work took place and prayer and worship took up most of the day.

The Conversion Process and Maintenance of Traditional Customs

As discussed earlier, the primary goal of the Franciscan friars in establishing missions in Alta California was to convert native peoples to Catholicism and teach them to give up native lifeways in favor of a European cultural model. This process was not always as easy as the missionaries thought, however, and often neophyte culture at a mission was syncretic, preserving some elements of indigenous culture while incorporating some behaviors taught by the missionaries. Each mission was unique in this respect, as each had its own pastiche of indigenous neophytes from various cultures and missionaries with their own approaches to conversions. In some cases, the friar in charge of a mission favored a strict approach to conversion, disallowing travel back to native settlements by neophytes and punishing those deemed to be clinging to indigenous practices. In other cases, travel to visit the unconverted could be tolerated and even encouraged by friars (mostly in times of supply shortages when food became scarce) (Shipek 1981).

Conversion

The process of converting newly encountered peoples to the Catholic faith was not as simple as baptizing the unconverted. According to Catholic doctrine, any individual of the age of majority (somewhere between about 7 and 10 years old, according to most sources (Clinch 1904; F. Weber 1991) had to participate in catechism

prior to baptism. Infants and younger children could be baptized without completing the catechism, and very sick individuals could be provisionally baptized in "*periculo mortis*" (danger of death) or "*articulo mortis*" (at the point of death) (The Huntington Library 2006). In order to complete the catechism and receive the sacrament of baptism an individual had to: know the sign of the cross, recite the Our Father, Hail Mary, Apostles' Creed, the Act of Contrition, Act of Faith, Hope, and Charity, Confiteor (for the sacrament of confession), the Ten Commandments, the Precepts of the Church, the seven Sacraments, the six Necessary Points of Faith, and the four Last Things, all in Spanish (Engelhardt 1920). Neophytes attended catechism classes, and often groups of "graduates" would all receive the sacrament of baptism on the same day. All baptisms were recorded in the mission's baptismal register, the official document containing information on all baptized members of the mission community. Often, after catechism was completed and individuals were baptized, native marriages would be "renewed" (*renovaron*) by the friar. Renewal of preexisting marriages served to sanctify those marriages in the Catholic Church. The newly renewed marriages and any new marriages were recorded in the mission marriage registers, providing a second data point to track individuals within the mission system. Confirmations also took place and were recorded in a register, however the registers for San Diego are not yet translated and digitized and therefore were not used for this project.

Maintenance of traditional culture at the mission

Assessing the persistence of traditional cultural behaviors in converted neophytes is extremely difficult, as very little information about their activities at the mission was recorded. Most historical accounts written by either the friars or outsiders do not spend much time detailing information about the native peoples living at the mission. In some cases, instances of gambling were recorded and there is some discussion of traditional religious practices that were discouraged (Jayme and Geiger 1970).

Historical documents from the Mission Period make no mention of the puberty ceremony or memorial ceremonies described in Chapter 3. It is assumed that these types of activities would have been discouraged by the priests at the mission and by any other Spanish citizens. Inferences can be made about the continuation of traditional marriage practices through an examination of the marriage and naming practices of individuals recorded in the baptismal and marriage records. That work is ongoing and results are not yet available.

The missionaries encouraged burial of the dead instead of cremation, setting aside a plot of land adjacent to the church as a cemetery. The death records from Mission San Diego do note that individuals who died away from the mission, often in a native village, were cremated without receiving Catholic mortuary rites.

The documentary evidence that describes the Mission San Diego community contains detailed information on baptisms, marriages and burials in among mission neophytes, as well as providing some supporting description of lifeways at the mission.

It does not, unfortunately, provide much detail on causes of death, illness and population health, or other kinds of background detail on the health of this mission community. Additionally, reports containing information on household structures and censuses carried out in the 18th and 19th century have been lost, further complicating the attempt to reconstruct this population.

CHAPTER 6 – THE 1805-1806 MEASLES EPIDEMIC IN CONTEXT

Like other parts of the Americas, the disease-scape of indigenous Californians was greatly altered as a result of European conquest. Prior to conquest, infectious disease is thought to have played a relatively small role as a cause of mortality among Native Californians. For some time after European conquest, infectious diseases were the main cause of mortality among native peoples and were responsible for elevated mortality rates and reduced fertility rates to the point that native populations declined by 75-90% by the end of the 19th century (Cook 1976a; Cook 1976b; Hackel 2005; Milliken 1995; Walker and Johnson 1994). The reasons for the extreme morbidity and mortality caused by European diseases in California are not well-understood, but the Californian experience is similar to that of indigenous peoples in other regions of both North and South America. Researchers interested in population decline as a result of contact espouse several hypotheses about European disease in the Americas and the reasons for native population decline.

One of the most common explanations for the extreme mortality in the colonial period is that Europeans brought diseases that indigenous Americans previously had no experience with, including measles, smallpox and malaria. These diseases may have caused virgin-soil epidemics in which large portions of the population became sick due to a lack of any previous exposure. In addition to a greater than normal number of sick, these new diseases may have also proven more deadly, killing at a higher rate in these populations than other human populations in which they were endemic. This may be

the result of differences in immune function between European populations and indigenous populations (Ramenofsky 1996) or in differences in childhood exposure to these diseases that yielded milder cases in European populations (Crosby 1972). Additionally the breakdown of networks of care (usually made up of young and middle-aged adults) for the ill and the inability of normal caregivers to take care of the young and old (whether or not they too were sick) could cause higher than normal mortality in those groups during times of epidemic disease (Milliken 1995).

Infectious Disease prior to Spanish conquest

For much of the early part of the 20th century, historians, cultural anthropologists and archaeologists thought of California as a kind of Eden, populated by "affluent foragers" who gathered their food from the naturally abundant flora and fauna found in the region (see the discussion in Raab and Jones 2004). This myth was deconstructed in the last part of the 20th century as a variety of archaeological studies revealed how populations were affected by changes in resource availability and environmental change (Broughton 1994; Eerkens et al. 2002; Glassow 1999; Wohlgemuth 2005) and the resulting intra-group conflict (Andrushko et al. 2005; Walker 1989). It is now known that prehistoric Californians were not passive consumers living in an environment that constantly provided for their needs, but actors who modified the landscape (Anderson et al. 1998) and negotiated for rights to resources as politically independent groups, sometimes leading to violence as highly valued food and trade resources became scarce (Bartelink 2006; Shipek 1981; Walker and Lambert 1989).

Data from the archaeological record indicate intensifying use of certain key resources (fish, large game, and acorns) through the prehistoric period (Broughton 1994; Broughton 2002). Archaeologists have interpreted this as signaling an increase in competition for resources, possibly as a result of population growth (Broughton and Bayham 2003). Accompanying these archaeological changes are skeletal evidence of dietary change and increases in lesions thought to be the result of infectious disease (Lambert 1993; Walker and Lambert 1989), specifically treponemal diseases. Most of this archaeological work focused on the San Francisco Bay area and the Central Coast around Santa Barbara. Bioarchaeological research of this type is less common for the California deserts, likely due to the fact that most southern Californian desert communities cremated their dead, leading to few burials that could be analyzed using bioarchaeological methods.

There is little evidence for non-treponemal diseases among California populations. In general, bioarchaeological studies have yet to reveal significant paleopathological indicators of most types of infectious disease, so identifying the cause of death for individuals dying from measles, influenza, or whooping cough, for example, often rely on accompanying historical documentation of the death or at least of the epidemic. In California, there are no cemeteries known dating the pre-contact period in which large numbers of dead were buried as a result of a single epidemic, making the connection between individual deaths and infectious disease very difficult to prove. The apparent lack of infectious disease among most indigenous Californian communities prior to conquest, whether real or a result of a lack of data, provides a sharp contrast

with the known impact of infectious diseases like measles, dysentery, influenza and smallpox among both mission and non-mission populations of Native Californians in the years after Spanish colonization.

Introduced diseases and their impacts

To many Europeans, the decline of indigenous populations in the Americas was a natural outcome of the colonization program: “As a general rule, savages die out when they come in contact with civilization” (Hittell 1888). To those who were concerned about the population declines caused by disease, the reasons behind the high mortality rates for what were thought to be mild, childhood diseases was somewhat of a mystery (Crosby 1972). Even attempts to treat the sick or control the spread of epidemics were not often successful, especially in the missions of Alta California. It should be noted that healers on the colonial frontier in North America were not always ignorant of advancements in the treatment of disease made in Europe. Surgeons were often part of exploratory parties and were relied upon to treat both the colonists and the neophyte populations, when possible (Lyman 1925). In 1802, an expedition left Spain carrying a load of orphans to serve as live incubators of smallpox in order to complete a vaccination campaign in Spanish colonies in the Caribbean and Mexico (D. Weber 1992). This campaign did not stop in Alta California, but the knowledge gained by local authorities could have been transmitted to the northern colonies, either directly through the movement of new colonists or via the extensive correspondence between military and mission outposts in the north and their supervisors in Mexico.

Disease in California

The extent to which introduced disease impacted populations of native Californians not associated with the mission system is currently unknown. Some have argued that diseases that caused widespread epidemics in colonial Mexico in the 16th century, such as smallpox and measles, most likely spread into pre-conquest California, though archaeological and historical evidence may not be adequate to evaluate this premise (Erlandson and Bartoy 1995; Erlandson and Bartoy 1996; Erlandson et al. 2001). It is thought that the geographic pattern in which people left native villages in order to join a mission population may indicate how far the impacts of contact were being felt at a given time (Hackel 2005; Milliken 1995). Perhaps people were migrating to the missions as their populations became stressed through disease-related population loss, environmental changes and decreases in the yields of traditional foodstuffs, as well as the pressure of settlers in the region. There is little published archaeological evidence of mass grave cemeteries among non-mission populations for this time in Southern California, though some have argued that archaeologists should look at more than skeletal collections to assess this premise (Erlandson and Bartoy 1996). Other lines of archaeological data could include site settlement changes, including territorial shifts, population movements and reductions in resource use, indicating a diminished population. It can be assumed, given the sometimes frequent movement between mission populations and outlying populations at many missions, that epidemics infecting neophytes would make their way into the greater population, though the extent of the damage cannot be known with current information.

Disease in the Missions

Compared to parish records and vital statistics for contemporary European populations, the historic record does not contain much information about the impact of epidemic diseases in the missions of California. Historical sources mention disease in general as a problem at the mission, but there is a lack of detail regarding the timing or specific types of diseases (Cook 1976a) and data quality varies between missions and across time. The burial registers at some missions record specific dates and causes of death for many individuals. The register at San Diego contains very few specific dates of death (recording instead date of burial and/or the date of notification of death) and almost no causes of death were recorded. This variation in recorded detail can complicate attempts to follow an epidemic from mission to mission or to compare the health of populations at different missions.

Documented epidemics at other missions (Table 6.1) include diphtheria, typhoid, pneumonia, smallpox, and measles epidemics occurring between 1796 and 1844 (Hackel 2005; Twitchell 1925; Walker et al. 1989; Walker and Johnson 1994).

Table 6.1 Documented epidemics in California 1769-1840

Year	Disease	Documented Region
1777	Unknown	Santa Clara area
1796	Typhoid	Santa Barbara missions
1796	Pneumonia	Santa Barbara missions
1800-1802	Diphtheria	Los Angeles and Santa Barbara missions
1806	Measles	Pan-California
1822-23	Measles	Santa Barbara missions
1828	Measles	Pan-California
1830-1833	Malaria?	San Francisco and central valley regions
1832	Unidentified (influenza?)	Santa Barbara missions
1838	Smallpox	Pan-California

The measles epidemics of 1806 and 1828 and the 1838 smallpox epidemic are known to have spread throughout almost all mission populations in California (Cook 1976a). Other epidemics were known to affect smaller regions or even a single mission. Sexually transmitted diseases were thought to be a particular problem (Nunis 1994), specifically syphilis and gonorrhea. Spanish friars were especially worried about sexually transmitted diseases because of their connection to what was understood to be immoral behavior (Engelhardt 1920; Hurtado 1999), but these diseases are of special concern too because of their effects on fertility. Spontaneous abortion is a side effect of both syphilis and gonorrhea, and long-term infections of both diseases cause sterility at high rates in both males and females (Nunis 1994). In fact, some link overall population decline in Native Californians to the fertility problems that accompanied high rates of

syphilis in the population during this period (Milliken 1995; Twitchell 1925; Walker and Johnson 1994).

The change from seasonal hunting and gathering to sedentary agriculture may also have played a role in the shifting disease burden for California natives. Malaria, thought to be introduced during the colonial period, could have increased as the environment was modified for agricultural purposes. Changes at Mission San Diego included the construction of two permanent villages in close proximity (the mission community and the presidio), the conversion of many acres of grasslands to grazing range, and the construction of dams and irrigation channels for watering crops. Studies have demonstrated the increased incidence of malaria in populations residing around agricultural areas (Asenso-Okyere 2009; Livingstone 1958), and it is likely that a similar phenomenon occurred in 18th and 19th century California. Additionally, the close proximity of certain segments of the population to domesticated animals could have increased the risk of zoonotic diseases, like glanders, brucellosis, or listeriosis, or parasites that can inhabit multiple host species.

Hygiene and sanitation practices were transformed for many native peoples during the mission period. In pre-conquest times, houses or whole villages could be moved in response to food procurement needs or as a way to distance oneself from structures that had become infested with vermin (fleas, ticks, and a variety of rodents were present in California). With the permanent settlement at the mission, avoiding vermin became more difficult. Writing about the poor living conditions at Mission San Francisco, Otto von Kotzebue stated “The uncleanliness in these barracks baffles

description, and this is perhaps the cause of the great mortality" (quoted in F. Weber 1991 pg. 59). Visitors to other missions noted that native housing was occasionally burned as a sanitary practice and that the populations were often overrun with fleas and suffering from high levels of sexually transmitted disease. After a visit to San Diego in 1803-1805, American Captain William Shaler asserted that living conditions at that mission were inferior to conditions at any other mission he had seen (quoted in F. Weber 1991 pg. 47).

Interestingly, smallpox did not appear to be a problem early in the Mission Period. As early as the 1780s, it appears that quarantine and isolation were common practice in the Spanish colonies as a method for preventing smallpox outbreaks (Valle 1973). Numerous education and variolation campaigns took place in Spanish colonies during the late 18th century. Additionally, at the behest of the Spanish King, a large-scale campaign aimed at education and vaccination in all Spanish colonies was undertaken in 1803 (Franco-Paredes et al. 2005). All these efforts were successful in staving off any serious smallpox epidemics in Baja and Alta California until 1822 (Nunis 1994).

Though a trained medical doctor was assigned to Alta California, his post was at Monterey, over 400 miles from Mission San Diego (Nunis 1994). Most medical care for mission neophytes would have been provided either by friars or by traditional means. This means that, as in other areas of mission administration, quality varied from mission to mission depending on the competency of individual friars. Some had more experience or training than others. Friars also had varying levels of interest in traditional

healing practices, some rejecting all traditional practices outright, other learning about native herbal remedies and implementing them into their own regimes of care (Nunis 1994). Apparently, some friars were trained in performing caesarean sections in order to baptize newborns in the case of maternal death during childbirth. In all of the ten known cases, the mother was dying or dead and the infants also died within a few hours (Nunis 1994).

It should be noted that indigenous populations in California had their own pharmacopeia and their own strategies for treating and healing illnesses common in pre-contact times (Garriga and Weber 1978). Of course, they were not prepared for dealing with imported illness, and traditional techniques such as the use of sweathouses or very close contact between the ill and a healer may have served to exacerbate the spread of infectious disease.

As mentioned above, the incomplete historical record at San Diego means that it is not possible to know the rates of various causes of death. The burial register for the mission has gaps, and even those records that do exist do not have comprehensive information about causes of death. Cause of death was recorded for about 250 of the almost 3000 burial records from Mission San Diego. Of these, "illness" (*enfermedad*) was listed as a cause of death for only 70 individuals. There were no cases in which a specific disease was listed as a cause of death. Other recorded causes of death include murder, execution and accidents, such as snake bites, falls or animal attacks. It appears that the recorders at San Diego were most likely to record a cause of death for those deaths that were sudden, unexpected, or of potential legal interest. Unfortunately, this means that

there is no direct documentary evidence of the measles epidemic investigated as part of this dissertation for Mission San Diego.

Measles as a possible cause of death at Mission San Diego in 1805-1806

So why is measles thought to be at Mission San Diego in 1805-1806? During my initial investigations into the Mission San Diego data, a large spike in numbers of deaths during this time period was identified. As a result, it was hypothesized that an epidemic disease may have caused these deaths, and the model was developed with this hypothesis in mind. Investigations into historical evidence from other missions led to the idea that the excess deaths at San Diego may have reflected the effects of the measles epidemic of 1806, an epidemic described at several other missions. This epidemic has been considered by some to be the worst epidemic to impact the mission populations in Alta California (Cook 1976a). Missions to the north reported the epidemic during the spring and summer of 1806 (Cook 1976a). By looking at the burials at Mission San Diego during this time, I conclude that the epidemic could have spread to this mission during the winter and spring of 1805/1806 (Figure 6.1). Numbers of burials began to rise in October 1805, peaking in January 1806 at 45 deaths. By April 1806, the increase in burials ended.

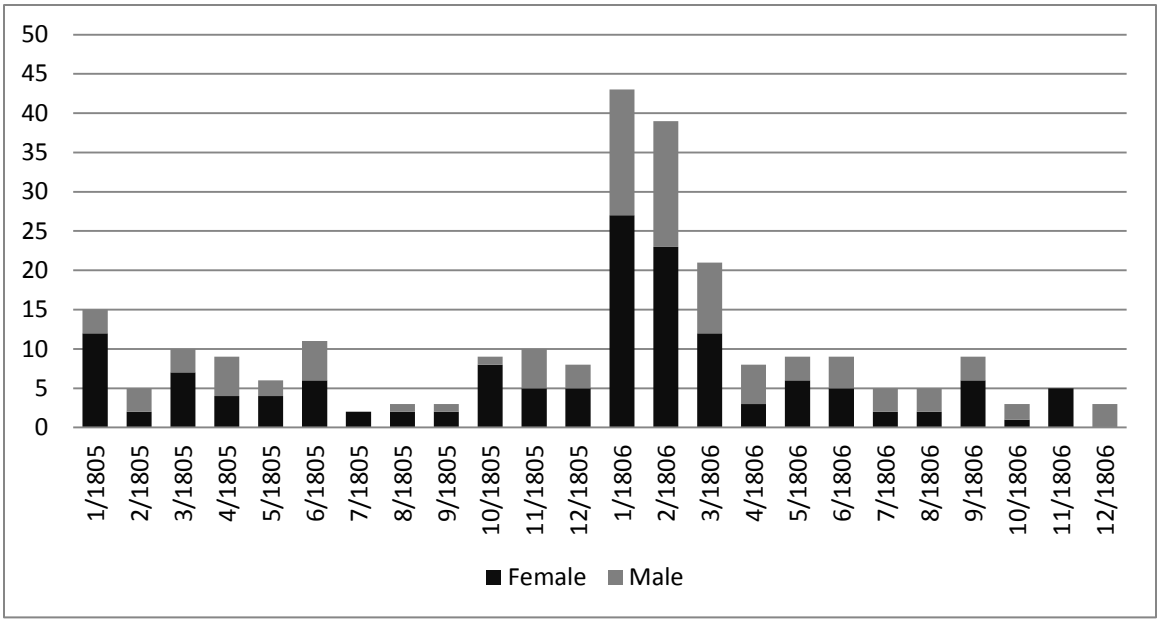


Figure 6.1 Numbers of burial recorded at Mission San Diego by month and year, 1805-1806

The timing identified here fits well with the fact that measles was documented at Mission San Miguel Arcangel (north of present-day San Luis Obispo, about 300 miles north of San Diego) in early March 1806 and Mission Dolores in San Francisco, 500 hundred miles north of San Diego, in April 1806 (Cook 1976a). Though the documentary evidence is lacking to definitely show that measles caused the large numbers of deaths seen during 1805 and 1806, use of a model to simulate the spread of measles through the mission population may be able to help fill in the gaps that exist in the historical record.

To briefly summarize the essential information covered in these background chapters, Chapters 3 through 6 have shown that the conquest of California took place as part of an organized strategy by the Spanish government and the Catholic Church to solidify political, social, and economic power and fend off challenges from European

rivals. Native Californians, unaware of the complexities of European motives, made decisions based on their own needs, material, social or spiritual, in a very heterogeneous way. The long term effects for indigenous communities around Mission San Diego was a loss of autonomy, increases in mortality and morbidity, and an overall population decline. The purpose of this project is to supplement the knowledge obtained through analysis of available documentary evidence by combining it with a computer simulation model that tests hypotheses about disease transmission among the native community at Mission San Diego.

CHAPTER 7 – FAMILY RECONSTITUTION AND LINEAGE RECONSTRUCTIONS

One purpose of this project was to build an agent-based disease model that could be used to test hypotheses about the relationship between social organization and activity patterns in a mission community during an outbreak of epidemic disease. In order to develop the model, it was necessary to use the available vital statistics on the historical population of Mission San Diego (the "historic population") as a starting point for the population to be used in the model (the "model population"). To that end, all available records were used to reconstruct the population as it existed on January 1, 1805. This chapter describes the reconstructed population, including a description of the information that could not be found in existing historical records. For the purposes of the model, gaps in information were filled in using distribution fitting techniques, described at the end of the chapter. The basic population structure was "reconstituted" using the mission registers of births, marriages and deaths. This allowed the identification of family groups and the grouping of some related families into larger lineages based on blood relationships and the application of ethnographically informed kinship rules for agent behavior in the model.

Family reconstitution is a set of methods developed by French historical demographers working in the mid-20th century. Henry and Fleury (1956) published one of the first studies that used parish data to reconstruct family relationships over time in a particular population. Their methods were quickly adopted by English and later

American historical demographers (Wrigley 1966). Traditionally, parish records served as primary data sources for reconstitution, but census data (Levine 1976), vital records or other kinds of records have also proven useful for reconstituting a population.

Reconstitution relies on having longitudinal data for a single population in which individuals can be identified. Usually, this entails having several sources of data (baptismal records, marriage records, and death records, in the case of Mission San Diego) that can be used to link individual life events to a person. The California mission data are ideal for family reconstitution for several reasons. First, the records are remarkably complete. On any given record, there are a number of ways to cross-reference an individual, from the unique identification numbers (baptismal ID) that carry over from baptismal to marriage to burial records, to the inclusion of information about parents or other family members. Second, the records cover a sufficiently long amount of time. Family reconstitution depends on having data from several generations of a population in a place and time. In the San Diego data, it is not uncommon to have records for at least three generations of a family, and in at least one case, a family can be traced for five generations. Finally, this data set includes information that allows one to follow at least a segment of the population that migrated away from the mission. If individuals moved from Mission San Diego to another mission's sphere of influence and received sacraments at the new mission, their records are still available. This helps alleviate a major problem with historical population data, that of the loss of individuals who migrate out of the area (see Kasakoff and Adams 1995 for a discussion of the impact of migration on family reconstitution). In many cases, individuals do still

“disappear” from the data set by leaving the mission to return to native lands or moving to an area outside of the mission system's influence; however, at least some of the losses from San Diego do appear at other missions later in time.

How to reconstitute?

The actual process of carrying out a reconstitution is not complex. The main goal is to use multiple lines of evidence to follow individuals through time and space and to tie related individuals together. The simplest form of reconstitution, and the kind employed in this project, uses parish records to follow individuals from their baptism, through their sacraments (such as first communion, confirmation, and marriage) to their deaths. Additionally, the records of their kin may contain references to them that can be used to gain further information about their lives. The California mission records were largely reconstituted as a part of the Early California Population Project (The Huntington Library 2006). This entailed the translation and digitization of the baptismal, marriage, and death registers for Mission San Diego. The confirmation records for Mission San Diego were not digitized and were not included in this study, though the inclusion of these data is planned in the future.

Much of the initial reconstitution was completed by employees of the Huntington, but it was against their policy to share their entire database with researchers, so the San Diego population was re-reconstituted by the author into a new database. Though this did take a lot of time, it allowed for a deep familiarity with the data in a way that would not otherwise have been attained. Additionally, the re-

reconstitution served to double-check the connections made in the Huntington's database, correcting small errors and suggesting new connections that they did not initially find. All corrections and added connections were reported to the Huntington staff member responsible for the database and, after they were checked, were incorporated into the Huntington's database. Figure 7.1 is a screenshot of a typical baptismal record from the Huntington's webpage.

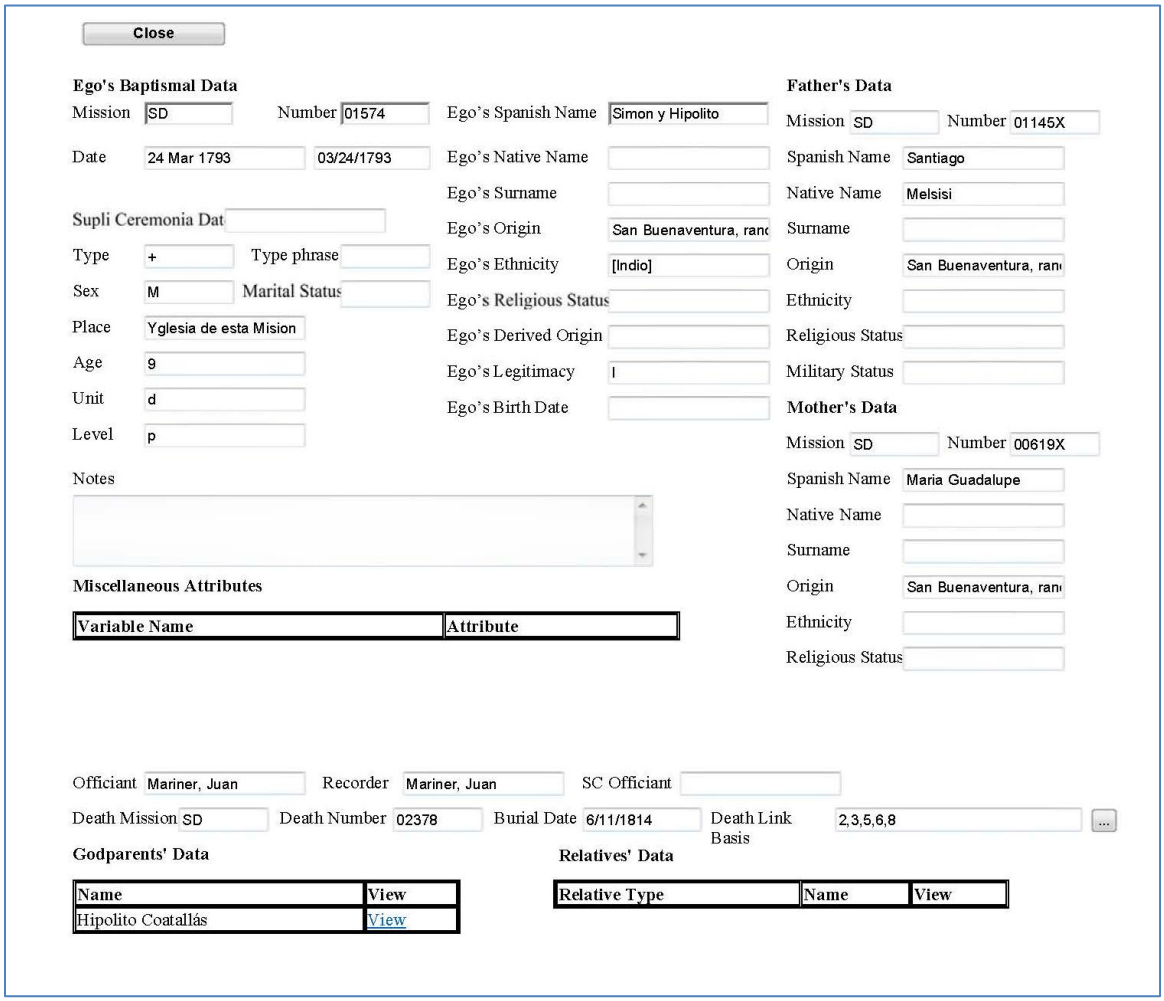


Figure 7.1 Screenshot of a Typical Baptismal Record from Mission San Diego (courtesy of the Huntington Library ECPP)

Reconstructing Mission San Diego

The reconstitution of the Mission San Diego population began by creating records for each individual listed in the baptismal register. The baptismal register was chosen because it contained records for the largest number of individuals, out of the three registers digitized from the mission (baptismal, marriage, burial). Between 1769 and 1845, the friars recorded 7104 baptisms. However, the baptismal records contain information about over 9000 individuals. These include not only individuals of indigenous origin who were baptized at the mission, but also the Spanish and Mexican population at the presidio and in the area and indigenous peoples from Baja California who traveled with Spanish traders or explorers serving as guides and translators. Sometimes information was recorded about unbaptized Kumeyaay who were the parents or other relatives of individuals who were baptized. These individuals were not part of the mission community, but in some cases could be used to understand the relationships between baptized individuals.

Individual information for each baptismal record was entered into Ancestral Quest, a commercial genealogy program. Links between many people in the mission population were established using the information on parents, siblings and other relatives found in the baptismal records.

Information from the marriage and burial registers served as the second and third data points for individuals listed in the baptismal records. For this project all deaths and marriages that occurred before 1/1/1807 were incorporated into the

reconstitution. Additionally, marriage information for those individuals in the 1805 population who married after 1/1/1807 was entered, in order to obtain information about the age at first marriage for the San Diego population, which could be used to estimate ages (described below). Ultimately, incorporation of all the marriage and burial records is planned, in order to pursue research questions that look at overall demographic patterns or events that occurred later in time at the mission.

Confirmation records are also available for this population. These records have not been added to the online database, and in order to use them researchers need to access either copies of the original books or use microfilms available from Family Search, the genealogical library of the Church of Jesus Christ of Latter-day Saints. These records present an additional challenge because they have not yet been translated from Spanish to English. For these reasons, the confirmation data were not included in the reconstitution for this project. However, part of the plan for further research with this population includes incorporating confirmation data, as it may be a valuable source of information about the adolescent years for Kumeyaay at Mission San Diego, including allowing for more precise age reconstruction for individuals lacking other records.

Reconstructing lineages

One goal of this project was to use native naming conventions, ethnographic information and village of origin data to incorporate lineage membership into the model population. This could not be carried out for this project for a number of reasons. Though initial forays into the mission records provided some hope of finding lineage or

clan names recorded in the "native name" portion of the record, the issue of what was actually recorded in that field was complex. The "native name" category appears to contain some combination of actual Kumeyaay names, lineage or clan names (with a variety of spellings), and potentially village or location names. Additionally, the transference of native name from parent to child does not always appear to follow cultural rules, as interpreted from the ethnographic literature, nor do the native names always conform to known lineage names. Further, the "native name" designation sometimes appears to correspond to native village name, indicating that perhaps individuals were identifying themselves by their home village instead of a lineage or clan grouping. Kroeber noted this phenomenon, but claimed that lineage names actually reflected village names (Kroeber 1976). In sum, the problem of lineage reconstruction proved too complex for this project and will have to await future research.

Features of the Historic Population at Mission San Diego

Before discussion of the model population, it is important to examine how much is known about the historic population at Mission San Diego, as of 1/1/1805. Population reconstruction has revealed a population of 1443 individuals of indigenous origin living at the mission at the beginning of 1805. Of those, 723 were male, 720 were female. Age could not be estimated for 127 males and 133 females. The general population numbers by sex revealed by reconstitution is consistent with information recorded in annual reports documenting general population statistics of each mission each year (Sánchez and Barona 1804).

Population Profile

Most individuals for whom age could not be estimated are likely beyond the infant stage. As people were baptized, the friars attempted to estimate ages to the nearest year or half-year. For infants, friars tended to estimate the day, week, or month age level. For individuals without an age estimate, the record indicates a general "level" estimation, usually "*adulto/a*" for adults, "v*iejo/a"* for older adults and "*parvulo/a*" for child. For this population, no *infante* was recorded without some further age estimate. Figure 7.2 shows the age distribution for the population with known ages.

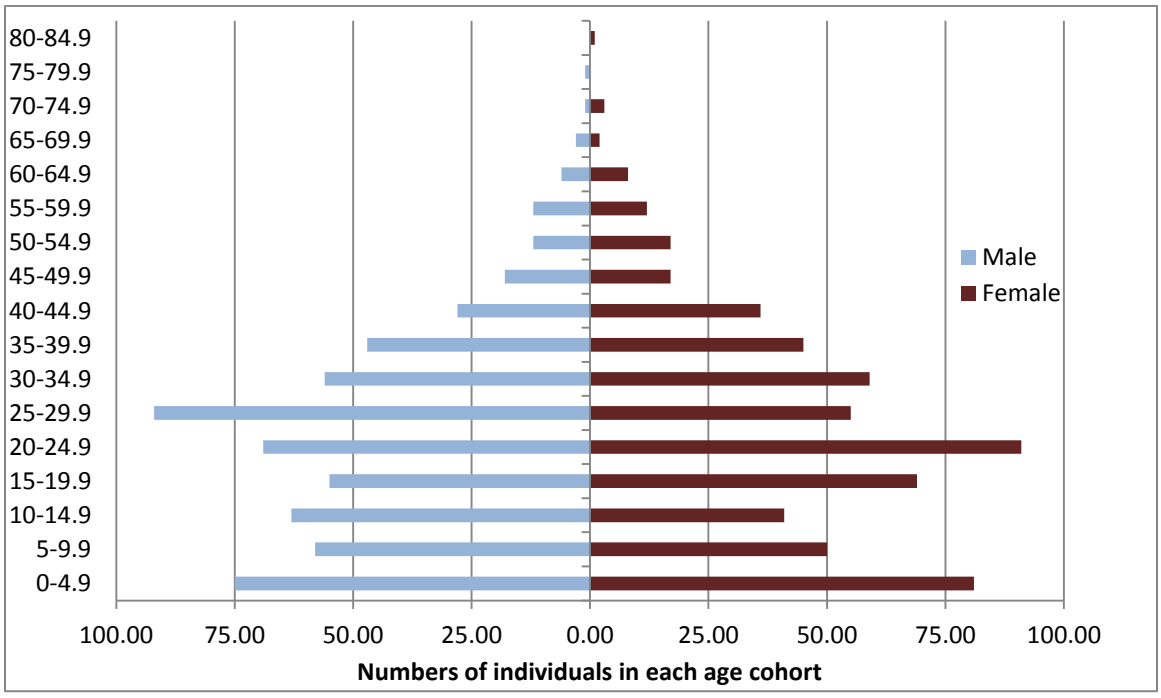


Figure 7.2 Population pyramid of historic population at Mission San Diego, January 1805

There are several interesting things about this population pyramid. First, there is a relatively equal balance between the sexes at most ages. Historical and demographic data from other missions suggested that there might be fewer male than female adults at the mission due to social factors that might have kept males from embracing mission

life (Milliken 1995; Orbann 2006), but this does not appear to be the case at Mission San Diego. There are also fewer children than one might expect from a population in a state of natural fertility (i.e. not employing birth control or family planning to limit births). One would expect a population with little or no birth control to have a wide base, meaning that there are many infants and children. In this case, either high rates of in-migration of young adults or high rates of infant and childhood mortality could be to blame for the fewer than expected numbers of children at the mission. It could also be possible that infectious diseases, including sexually transmitted diseases, played a role in reducing numbers of offspring through infertility and unsuccessful pregnancies.

Age at Baptism

The population living at Mission San Diego in 1805 was composed of people who were born at the mission and people who moved to the mission over time, starting when the mission was established in 1769. Most of the people living at the mission in 1805 had moved to the mission relatively recently, but a sizeable number had been living in the mission's sphere of influence for 25 years or more. Figure 7.3 shows the year of baptism for all individuals with recorded baptismal dates (n=746).

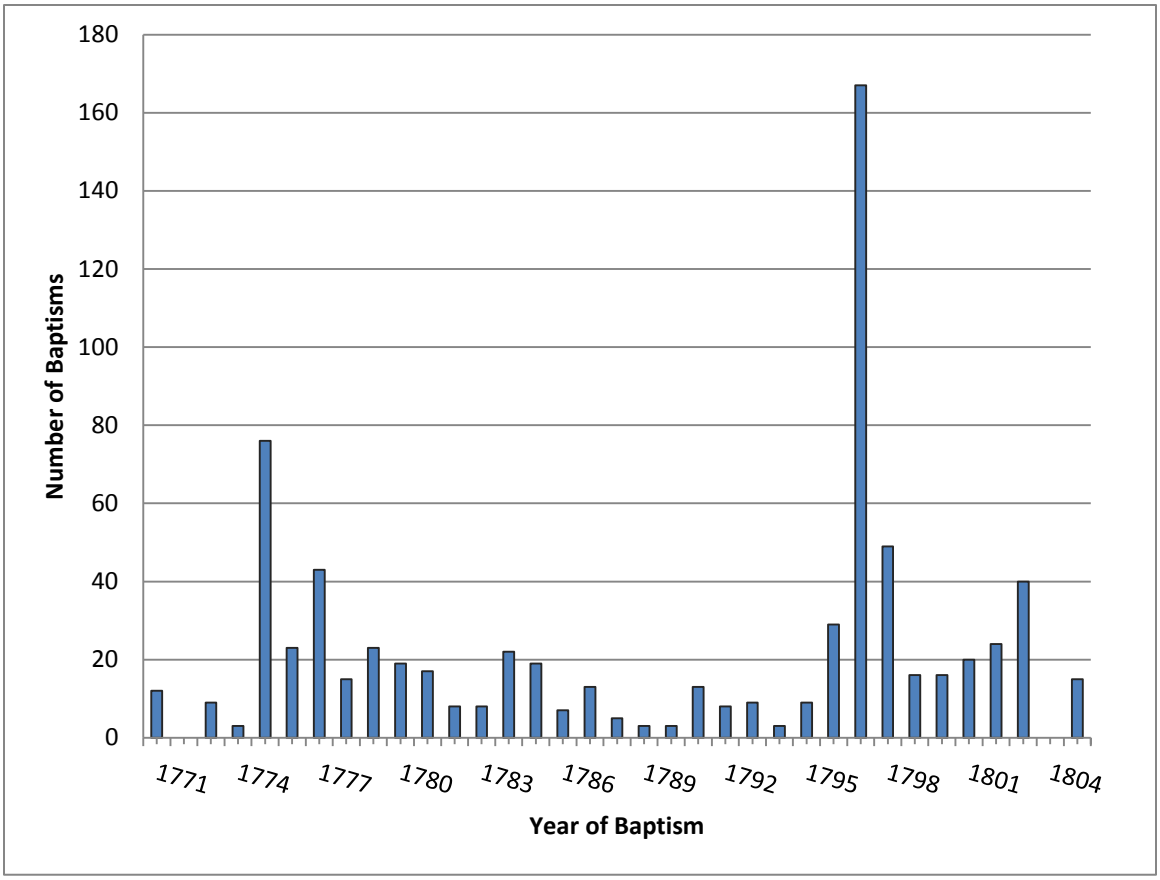


Figure 7.3 Year of baptism for Mission San Diego neophytes alive January 1, 1805

The median age at baptism was 15.5 years old for females and 12.5 years old for males. This may indicate that males living at the mission at this time were more likely to have been born there or brought there during childhood with families, as opposed to migrating there as adults. In fact, most people living at the mission in 1805 were baptized as infants. The distribution of age at baptism can be seen in Figure 7.4. It appears that most people who were not baptized as infants or young children did end up being baptized as fairly young adults.

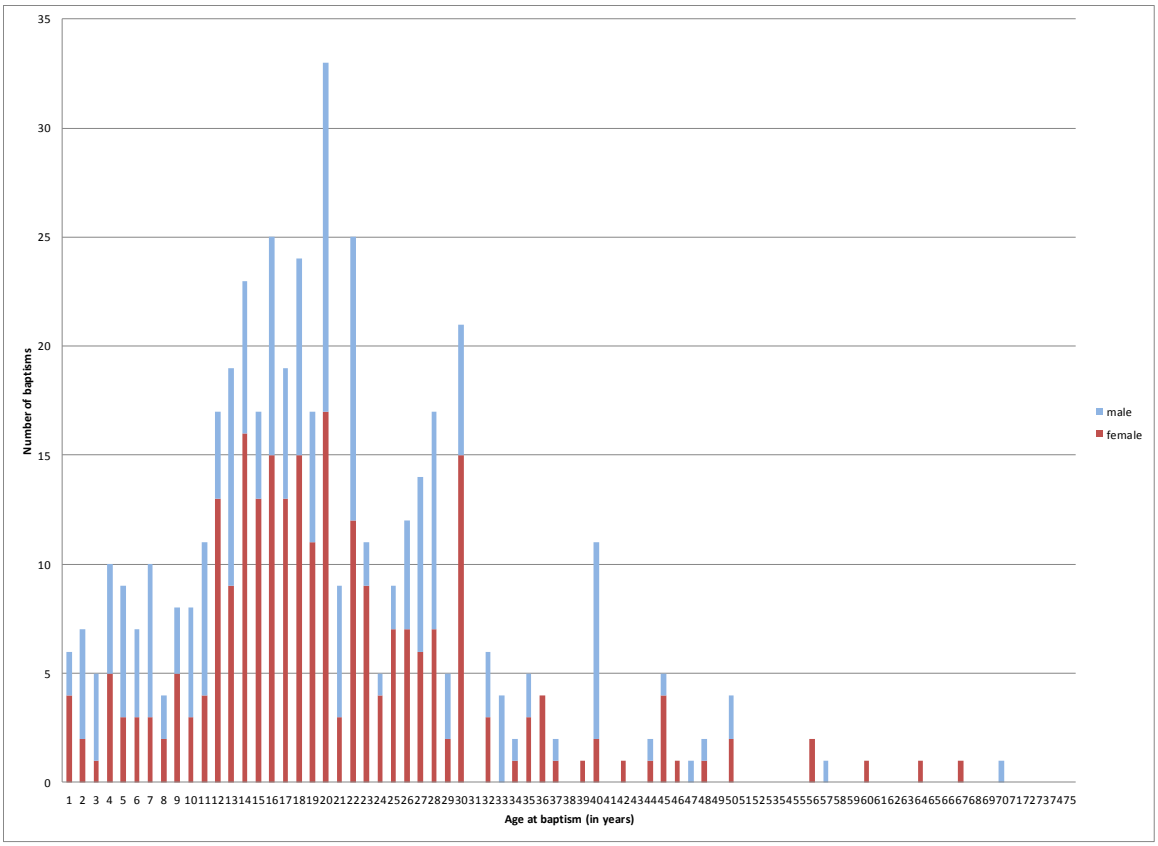


Figure 7.4 Age at baptism for Mission San Diego neophytes alive January 1, 1805. Infants under 1 year old excluded (female = 88, male = 98 baptisms).

The numbers of individuals baptized at ages older than 30 is low. Some of those elderly who were baptized were done so *in periculo mortis,* meaning that they were thought to be sick or dying at the time of their baptism. These data may support the theory that, in California, moving to a mission was something that was an attractive or strategic choice for younger people and that older people were resistant to making that change (Hackel 2005; Milliken 1995).

Age at First Marriage

Age at first marriage was also calculated for the historical population. This was needed in order to support the estimation of ages for the model population. Most of the

individuals who did not have estimated age at baptism did go on to marry and have children while living in the mission community. So, if a distribution of known ages at first marriage could be calculated, those individuals lacking age data, but with supporting dates for marriage and procreation could have ages estimated. Looking at the ages of first marriage for the part of the population with known ages helped in assigning ages to individuals in the model population without known ages. Figure 7.5 shows the distribution of ages at first marriage for males and females at the mission. A reliable age at first marriage was known for 611 individuals. The ages for the rest of the model population had to be estimated.

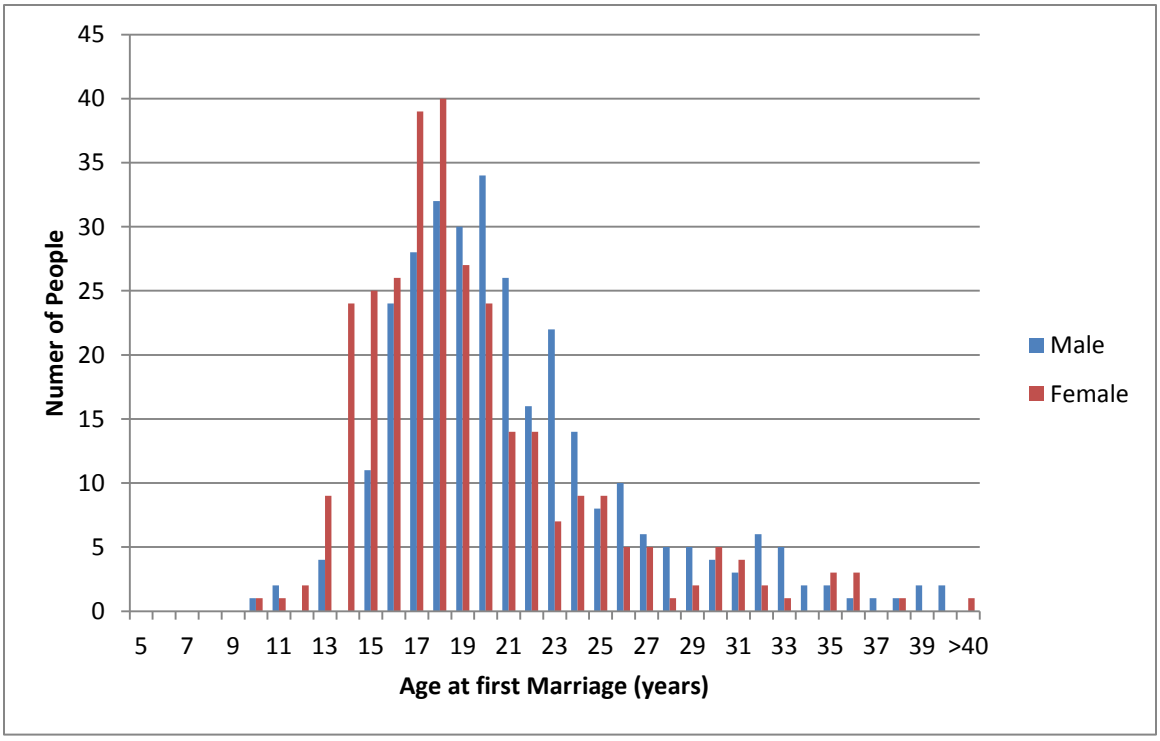


Figure 7.5 Age at first marriage distribution for Mission San Diego, 1805

It is clear from this figure that more females married at younger ages than males, overall. This is not surprising in either a traditional Native Californian or a mission community context. The median age at marriage for males was 20 years old and females was 18 years old, though the late teens and early 20s was the most common time for a person of either sex to marry for the first time. It appears that more males than females married for the first time in their late 20s and early 30s, though people waiting this long do not represent a significant part of either population. The summary statistics for age at first marriage are shown in Table 7.1

Table 7.1 Summary statistics for known age at first marriage

	Male	**Female**
N	307	304
Average	21.25	19.39
Standard Deviation	5.39	5.29
Median	20	18
Mode	20	18

In order to assign missing ages to the model population, it was necessary to test whether the distribution of age at first marriage was different for males and females. This would help in deciding whether or not to use separate probability distributions for the estimated ages in the model population. The male and female distributions of ages at first marriage from the historic population were compared using a two-sample Kolmogorov-Smirnov test (n1=304, n2=307), which found that the distribution of age at

first marriage for males did not differ significantly from the female distribution (at α=0.05). Therefore, it was possible to use a single probability distribution to assign ages to the model population based on marriage dates for both males and females (see the discussion of age estimation in the next section).

The Model Population

The population used in the model is based on the historical population at Mission San Diego in 1805. This means that the historical and ethnographic data were used to build a population as completely as possible and then missing data were filled in using best-guess techniques. The model population looks very similar to the historically known population, but a level of detail regarding household assignments and lineage membership was included in the model that is not possible to know from the historical information. The model population file is available in full in Appendix B.

Age Estimation for the Model Population

Not all of the Mission San Diego population was born at or near the mission, leading to incomplete birth data for many individuals in the records, especially for the years around the time of the establishment of the mission. When possible, the officiant of a baptism would record the birthdate of the baptized individual. In the case of infants, often a specific date or at least a month and year were recorded. In the case of children and adults, the age was usually estimated in whole years by the officiant, though rarely an estimate of half years occurred (e.g. 1.5 years old). In some cases, only an "age level" was recorded. Age levels include the general (and non-exclusive)

categories of *recen nacido* (infant), *parvulo/a* (baby or toddler), *muchacho/a* (boy or girl), *nino/a* (boy or girl), *adulto/a* (adult), *viejo/a* (elderly). These categories are not clearly delimited, so one must be careful not to make too many assumptions based on these labels.

For those individuals with ages estimated by the friar, a year of birth was calculated based on the baptism date and the age listed on the baptismal record. It was sometimes possible to cross-check this calculation with age data listed on death or marriage records, or with supplemental data listed on various other records. For example, a note that an individual had a twin allowed for a verification of age. The individual's age for the model was then calculated based on whole years elapsed between 1/2/year of birth and 1/1/1805. That is, by having the model start on January 1, 1805, it was assumed that no one in the population had yet had his or her birthday that year unless a birthday of January 1 was known.

Age had to be completely estimated for 225 individuals in the model population. As mentioned in the Introduction, the original baptismal book was destroyed in 1775 during an attack on the mission. The records that book contained were reconstructed from memory after the attack by the friars who originally recorded them (excluding Fr. Luis Jayme, who had been killed), but were incomplete. This means that both estimated age at baptism and date of baptism were lost or not recorded at all. In these cases, another strategy was needed to come up with age estimates. Other data could be used to inform age estimates, such as marriage date or date of first childbirth. Known dates of first marriage or first childbirth for individuals without known age or date of baptism

could be compared to a distribution of known ages at first marriage from the San Diego mission population. The age at first childbirth is then assumed to be one year post-marriage, as this was a natural fertility population, and once married most individuals at San Diego had their first child within a year. To estimate ages based on known first marriage or first child dates, it was necessary to look at the individuals with known ages, compile a distribution of ages at first marriage and first childbirth and then use a random number generator to generate ages and first marriage that fit the known distribution so that ages could be assigned to individuals in the model population who were based on historic individuals without age information.

Once it was determined that male and female ages at first marriage in the historic population came from the same distribution, it was necessary to create a fitted distribution of possible age at first marriages that could be used for assigning ages to model agents. Using EasyFit 5.5, a statistical fitting program, the known age at first marriage distribution was evaluated for compatibility with known distributions. In this case, it appears that the data can be described by a Burr distribution (a log-logistic distribution). Random numbers to be used as model "age at first marriages" were generated according to an appropriate Burr distribution and were applied to model agents needing an age value. Age could then be chosen for model agents by calculating the difference between date of marriage (using the "age at first marriage" as the agent's age) and 1/1/1805. In some cases, marriages took place prior to the model start date, in some cases marriages post-dated the model start date, but in either case the same strategy was employed for determining the model agent's age. A total of 133 females

and 127 males had ages estimated based on this method. The population pyramid for the entire model population is shown in Figure 7.6. It looks similar to the historical population shown in Figure 7.2, as it should if the reconstruction was carried out in a way that was sensitive to the original population distribution.

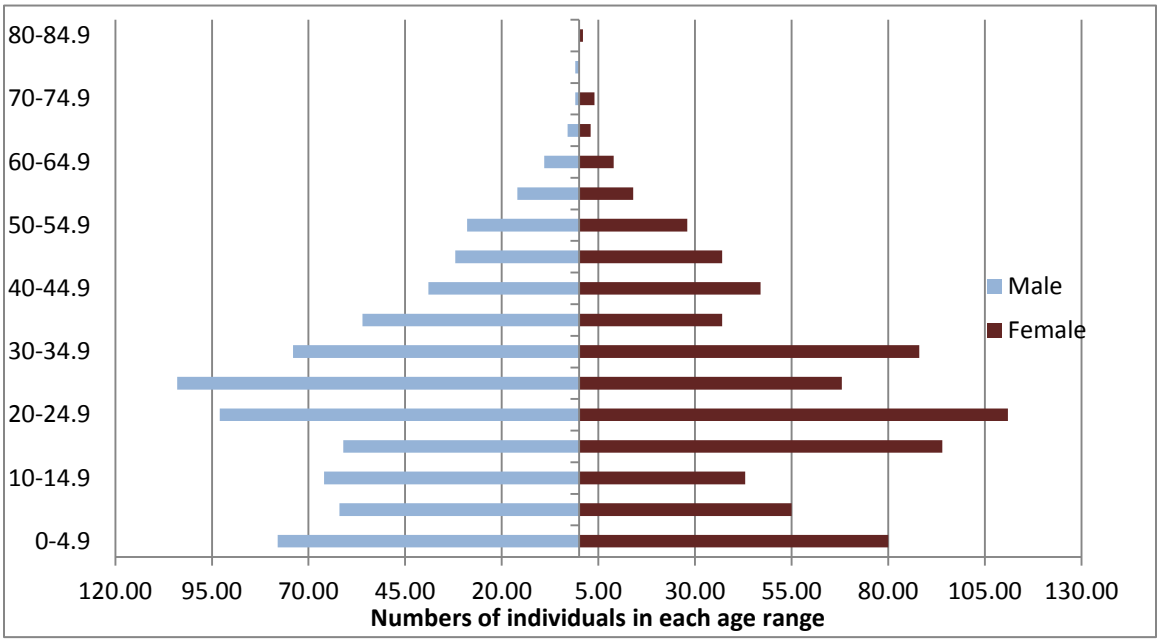


Figure 7.6 Population pyramid for San Diego Mission Model Population

A population pyramid was created that plots the model population overlaid onto the historic population for comparison purposes. Figure 7.7 clearly shows that most of the people who had ages assigned were adults. A few more females than males needed to have ages assigned (n=6), and it appears that many of the females without age estimates may have been younger than males needing estimates.

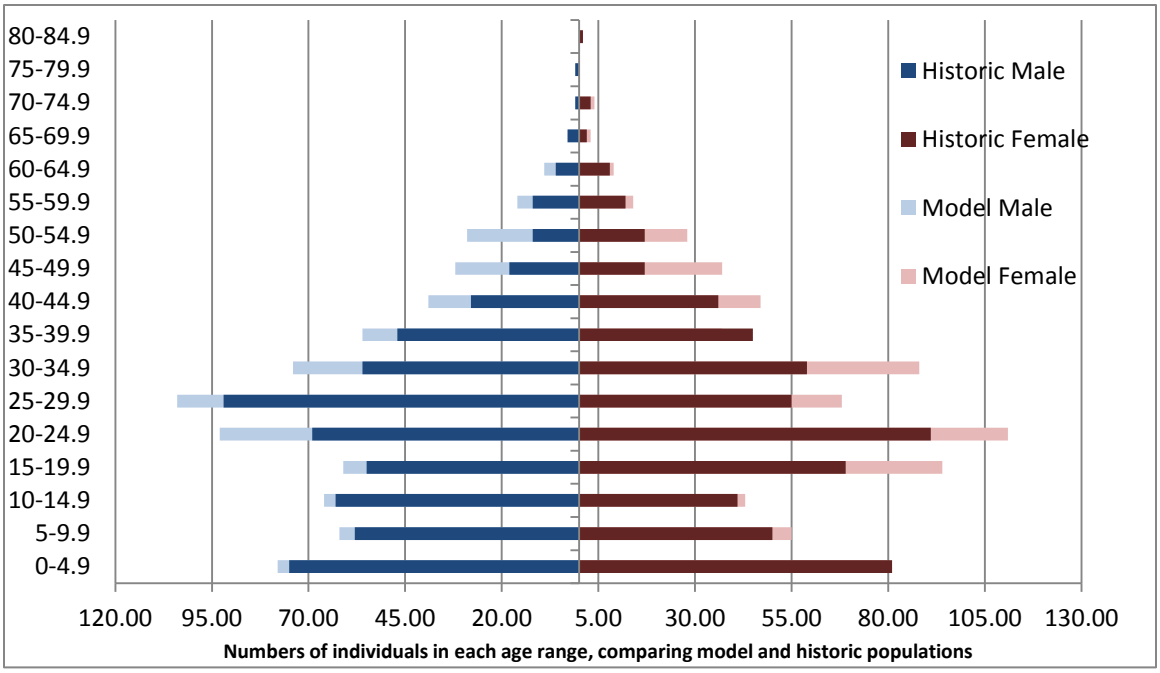


Figure 7.7 Population pyramid showing the San Diego Model population with individuals of estimated ages overlaid onto the historic population that includes only individuals of known age.

Overall, the model population, like the historic population, appears to have fewer than expected children and slightly more males than females. Females are younger, in general, with a median age of 22 years in the historical population and 23 years in the model population; males had a median age of 23 years in the historical population and 25 years in the model population.

Reconstructing Households

Yearly censuses, known as *padrones*, were carried out at each mission. These would have shown the household information for all neophyte families living at San Diego over a number of years during the life of the mission. Unfortunately, the *padrones* for San Diego were lost, and so households had to be reconstructed according to the demographic and cultural rules discovered during historical and ethnographic research.

Ethnographic data indicate a preference for patrilineality and patrilocality in traditional Kumeyaay culture (Shipek 1982), though it appears that there was a considerable amount of flexibility in terms of post-marital residence, reckoning of kin, and living arrangements. It is thought that individuals therefore had numerous options for household assignment at the mission (Shackley 2004).

Households in the model were constructed according to the following rules:

1. A husband, wife and their children lived in a household together.
2. If present, brothers shared a household, though if they had more than two children the brothers were assigned to independent households. Many household were thus formed by a married older brother with several small children and a younger brother, either unmarried or married without children.
3. Aged parents would live with the eldest adult living child, preferring a son over a daughter.
4. Children would live with a parent of either sex in the event of the death of the other parent.
5. Orphaned children were assigned to a patrilineal relative, if possible. Otherwise, a matrilineal relative was acceptable.
6. Single women over the age of 10 years old were assigned to the *monjería*, the dormitory for unmarried women.

A total of 247 households were constructed using the rules described above. The priest was assigned to a separate quarter within the mission building and no one else in the community was allowed to enter his quarters, though he could visit houses in the

community. Additionally, 139 individuals were assigned to the *monjería*. The average household size was 5.4 individuals at the start of the model run. The smallest households contained just two people and the largest household had 13 people. Household size fluctuated during a model run as people (primarily children) were reassigned due to deaths and subsequent household dissolution. A large area of the map was designated as "household space", with households organized on this model space in a semi-random fashion. This reflects archaeological data that indicates a separate space near the mission for the neophytes living at the mission (Smithsonian Trinomial 4SDI202).

The model population for the San Diego model was constructed with the goal of mirroring, as closely as possible, the real historical population at Mission San Diego in 1805. This means that the population may not conform well to other reference populations that are built using historical information from other parts of the world or other time periods. Both the historic and model populations are the result of unique historical processes, including fluctuating birth and death rates, emigration and immigration, and the vagaries implicit in historical records. Despite the complications that stem from trying to base a model population on a historical population (many of which are described in this dissertation), one of the strengths of this approach is that the computer simulation model uses a population that is real rather than hypothetical and that the model developer has evidence that decisions can be based on, in terms of demographic structure and population dynamics. The next chapter describes the San Diego model, which is the environment inhabited by the San Diego model population.

CHAPTER 8 – MODEL DESCRIPTION

This chapter contains detailed information about the model used in this project. Included in this chapter are an overview of the program used in the creation of the model, agent and building information, movement schedules, and a detailed description of disease transmission. In addition, description of data collection methods is also included. The information in this chapter is consistent with the Overview, Design Concepts and Details (ODD) Protocol established by Grimm, et al. (2010), though the format does not follow the recommendations of the protocol.

Model Software

The model was created using Repast Simphony modeling software. The Recursive Porous Agent Simulation Toolkit (Repast) is an open source set of modeling libraries designed for those interested in a high degree of control over the program behind an agent-based model (North et al 2013). It is implemented by installing the RePast toolkit into an Eclipse programming environment. Repast is implementable in a number of different programming languages (Groovy, Python, etc.). The current model was programmed in Java.

The Agents

The agents in this model are based on real people alive at Mission San Diego in the early 18th century, though, as described in Chapter 7, missing data were filled in using distribution fitting methods. A database has been created that gives each agent a

unique ID and attributes based on the mission population. Agent information is fed to the program via a text file. See Table 4 for a description of the variables in the agent text file. The full agent file used in simulation runs is included in Appendix 1.

Table 8.1 Individual agent attributes

Variable	Type	Description
Agent ID	Integer	User-generated unique agent identification number.
Father's Patriline	Integer	A number corresponding to the patriline of ego's father.
Disease Status	Integer	Initialized at 0, changes in accordance with the disease process.
Dwelling	Integer	The building to which the agent is assigned.
Mother's Patriline	Integer	A number corresponding to the patriline of ego's mother.
Extended Family	Integer	The extended family group or lineage of the agent.
Sex	String	"M" or "F" as noted in mission documents.
Age	Double	The age of the agent, in whole years.
Spouse	Integer	The ID number of the agent's spouse.
Health History	Double	A variable that can be used to set the relative health status of the agent. Not used in this model.
Occupation	Integer	Denotes an individual's occupation and used to regulate daily movement around the model space.

The World

The model world is an enclosed space on which the agents move and interact. It consists of a context that contains all the buildings and agents and a grid that organizes them. The world is a 100 by 100 grid with defined borders. It is not a torus, so when

agents hit an edge they must move in another direction to stay on the space. The grid is occupied by building agents (hereafter known as buildings) and person agents (hereafter known as people).

The buildings reflect the social spaces known to have existed at Mission San Diego at the turn of the 18th century. They do not, however, reflect those spaces in a true geographic sense. They do not conform to the "real" dimensions of those places, nor is their distribution over the grid reflective of their real-world locations. There are several reasons for this. First and perhaps most important is that there is no known map of Mission San Diego from the early 19th century. A Historic American Buildings Survey crew attempted to map the mission at some point in the 20th century, but that was after years of disrepair and without knowledge of the location of outbuildings or other structural features of the grounds (Peterson 2012). Line drawings of the mission and mission grounds from mid- to late-19th century exist and are somewhat useful for getting a sense of the physical environment of the mission, but they do not provide enough detail for a geo-referenced map to be created. Additionally, information about the native settlement adjacent to the mission (a village called *Nipaguay*) does not provide information about the size or distribution of the neophyte population during this period. The archaeological records in the area do not contain maps and have no references to the number of potential houses that were in use. The lack of a proper map was not a major problem in the development of this model, as the disease mechanics that this model is built on do not rely on an understanding of pathways or travel times. Agent movement on the space is described in detail below.

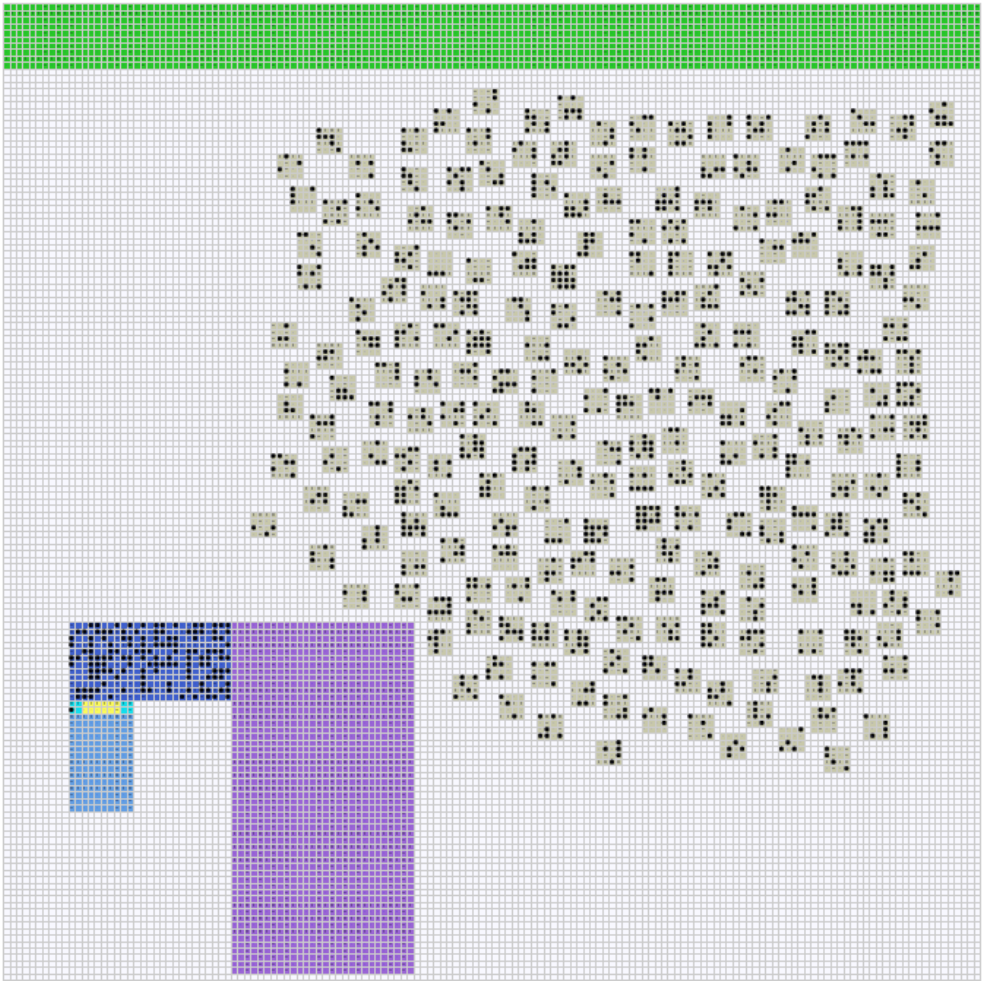


Figure 8.1 A screenshot showing the map of the model with agents (black dots) in their home buildings. Building types are as follows: houses = grey, church = purple, monjería= dark blue, kitchen = blue, priest quarters = aqua, storage area = yellow, men's occupation area = green

Because of some constraints in the way that grids work in Repast, buildings had to be programmed in a slightly convoluted way. The use of a grid as a model space means that traditional Java "container objects", like polygons, cannot be placed on the space for use as buildings in which agents could move. In the San Diego Model, "buildings" are actually groups of individual building agents with shared attributes, each sitting on a single grid point and adjacent to one another on the grid. These are created via a text file (found in Appendix B) in a process similar to that of agent creation.

Buildings have physical location information as well as building type information. This model contains houses, dormitories, a church, occupational locations and a school. Ethnographic information suggests that either nuclear families or the families of two brothers lived in separate houses and single women lived in sex-segregated dormitories on the grounds of the mission. Occupations were also sex-segregated, with men working in agricultural fields, tanning areas or at artisanal crafts (carpentry or candle making). Women worked at food preparation, weaving, and sewing, primarily at their residences (both community and in the *monjería*). Children, when not in catechism, helped in a variety of locations, depending on their age and sex. Priests lived separately, near the church building and had some personal assistants. The personal assistants included both men and women from the non-mission community of San Diego and were not included in this model. Buildings do not move or change status in any way during simulation runs.

Model Initialization

The model is initialized with a population of 1438 agents divided into 247 households. The disease parameters are initialized as shown in Table 5. The disease parameters were set in accordance with published values known for measles from epidemiological and medical literature (Hersh 2004; Kim-Farley 1993).

The value for the transmission probability was calculated according to the following formula: $1-(1-x)^t = ß$ for the value of x. Here *x* is the probability of transmission for each tick, *t* is the time (in ticks) that an agent spends in the "infectious" disease state, and ß is the cumulative transmission probability over the course of an infection. In

this case, published accounts of measles transmission in unvaccinated populations estimate a transmission probability of at least 90% between an infectious and susceptible individual (Aaby et al. 1983; Shanks et al. 2011; Wolfe 1982). Thus, ß is set at 0.9739 for a transmission probability of 0.025 per contact and an infectious period of 144 ticks (6 days at one tick per hour).

The length of the latent period is set at 240 ticks, which, at one tick per hour, corresponds to ten days. Epidemiological literature indicates a ten day latency for measles is typical in most infections.

The length of the infectious period is set at 144 ticks, which, at one tick per hour, corresponds to six days. Infectious periods are usually reported as a range of potentially infectious days. For the model, it was decided to use a fixed value representing an average length of infectious period for all infections.

The run length value is the total length of time the model will run for each simulation. It was set at 3600 ticks (150 days), sufficiently long enough for all epidemics to end by the end of the run.

The per tick probability of death is calculated similarly: $1-(1-d)^t$ = total death rate, where *d* is the probability of death per tick and *t* is the time (in ticks) that an agent is at risk of dying from the disease. The total death rate of 0.25 is based on studies that indicate extreme mortality during the 1806 measles epidemic in California, as high as 25% of the entire mission population at some missions (Cook 1976a; Milliken 1995).

Table 8.2 Initial values of model parameters

Parameter	Value
Transmission Probability (per tick)	.025
Length of Latent Period (ticks)	240
Length of Infectious Period (ticks)	144
Run Length (ticks)	3600
Probability of Death (per tick)	0.001996

Agent Schedule and Movement

During initialization, agents are placed randomly within their assigned home building. Each home building has enough space for an entire family plus several empty spaces. This ensures that the houses are not over-packed (artificially increasing the likelihood of disease transmission) and allows room for visitors. To move, an agent (the "mover") picks random x and y coordinates within the building to which it wants to move. Then, the "mover" disappears from its former location and reappears in the new location. If the randomly chosen coordinates already contain another person, the "mover" then chooses another random set of coordinates within the same desired building. If there is no space within the desired building, the "mover" will most likely just move within the building in which it is presently located. In some cases, alternate movement patterns have been established to deal with overcrowding of houses. There are no movement paths and movement is instantaneous. It is not possible to make

contact with another agent while moving, only once movement is completed. Movement is dependent on age, sex and occupation of the person. The movement schedule can be found in Tables 6 and 7. This schedule was based on historical documentation about life at the mission, described in Chapter 5.

A typical model run starts at midnight Monday and runs for 150 days. The model uses one tick per hour. Each agent determines where to move according to the schedule and then moves. After movement is complete, the disease transmission processes occur (if appropriate), disease statuses change, and needed reassignments occur. These are all discussed below. Immigration and emigration are not included in this model; neither are background demographic processes like births and non-disease related deaths. The model was designed to examine a short epidemic and these other features would not play a large role over this short period.

Table 8.3 Weekday schedule of agent activities (Monday-Saturday)

Time	Demographic Category						
	Single Women	**Single Men**	**Married Women**	**Married Men**	**Female Children**	**Male Children**	**Priests**
12:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
1:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
2:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
3:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
4:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
5:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
6:00 am	Monjería	Home	Home	Home	Home	Kitchen	Home
7:00 am	Monjería	Home	Home	Home	Home	Home	Home
8:00 am	Church	Church	Church	Church	Church	Church	Church
9:00 am	Monjería	Fields	Weave	Fields	Weave	Fields or Kitchen	Random
10:00 am	Monjería	Fields	Weave	Fields	Weave	Fields or Kitchen	Random
11:00 am	Monjería	Fields	Weave	Fields	Weave	Kitchen	Random
12:00 pm	Monjería	Home	Home	Home	Home	Home	Home
1:00 pm	Monjería	Home	Home	Home	Church (school)	Church (school)	Church (School)
2:00 pm	Monjería	Fields	Weave	Fields	Weave	Fields or Kitchen	Random
3:00 pm	Monjería	Fields	Weave	Fields	Weave	Fields or Kitchen	Random
4:00 pm	Monjería	Fields	Weave	Fields	Weave	Kitchen	Random

5:00 pm	Monjería	Home	Home	Home	Home	Home	Home
6:00 pm	Monjería	Visit	Visit	Visit	Visit (with parents)	Visit (with parents)	Home
7:00 pm	Monjería	Visit	Visit	Visit	Visit (with parents)	Visit (with parents)	Home
8:00 pm	Sleeping	Visit	Visit	Visit	Visit (with parents)	Visit (with parents)	Home
9:00 pm	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
10:00 pm	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
11:00 pm	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping

Table 8.4 Sunday schedule of agent activities

Time	Demographic Category						
	Single Women	**Single Men**	**Married Women**	**Married Men**	**Female Children**	**Male Children**	**Priests**
12:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
1:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
2:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
3:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
4:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
5:00 am	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
6:00 am	Monjería	Home	Home	Home	Home	Kitchen	Home
7:00 am	Monjería	Home	Home	Home	Home	Home	Home
8:00 am	Church	Church	Church	Church	Church	Church	Church
9:00 am	Church	Church	Church	Church	Church	Church	Church
10:00 am	Church	Church	Church	Church	Church	Church	Church
11:00 am	Church	Church	Church	Church	Church	Kitchen	Church
12:00 pm	Monjería	Home	Home	Home	Home	Home	Home
1:00 pm	Monjería	Home	Home	Home	Home	Home	Home
2:00 pm	Monjería/Village	Visit	Visit	Visit	Visit (with parents)	Visit (with parents)	Random
3:00 pm	Monjería/Village	Visit	Visit	Visit	Visit (with parents)	Visit (with parents)	Random
4:00 pm	Monjería/Village	Visit	Visit	Visit	Visit (with parents)	Kitchen	Random

5:00 pm	Monjería	Home	Home	Home	Home	Home	Home
6:00 pm	Monjería	Visit	Visit	Visit	Visit (with parents)	Visit (with parents)	Home
7:00 pm	Monjería	Visit	Visit	Visit	Visit (with parents)	Visit (with parents)	Home
8:00 pm	Sleeping	Visit	Visit	Visit	Visit (with parents)	Visit (with parents)	Home
9:00 pm	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
10:00 pm	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping
11:00 pm	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping	Sleeping

Disease Process and Transmission

Measles

As this model was operationalized with a measles-like disease, a brief review of the epidemiological literature discussing measles is presented. An explanation of how disease attributes were incorporated into the model follows.

The measles virus affects humans primarily, spreading from person to person via direct contact, droplet transmission or possibly via contaminated items (Kim-Farley 1993). Exposure almost always results in infection of susceptible individuals. The disease incubates for about 10 days before any symptoms are seen. Fever is usually the earliest symptom, with the characteristic rash not appearing until several days after the fever and respiratory symptoms (Hersh 2004). People infected with the disease are known to be infectious during the second half of the incubation period and continue to be infectious for about four days after the appearance of the rash (Kim-Farley 1993). Many deaths during a measles outbreak may be due to secondary bacterial infections leading to pneumonia, diarrhea or encephalitis (Kim-Farley 1993). Measles may also result in brain damage or deafness. Infection with the virus confers lifelong immunity, and infants are given temporary immunity from their mothers for up to 6-9 months after birth (Kim-Farley 1993). Among both historic and contemporary human populations, measles is generally thought of as a disease of childhood. In many large populations, it is endemic and cyclical (Keeling and Grenfell 1997).

For populations in which measles was endemic prior to the invention and administration of a vaccine, the disease was cyclical with a two-year major cycle and a seasonal component (Anderson et al. 1984). In populations where measles was unknown, introduction of the disease was known to cause high mortality across age groups (Panum et al. 1910; Wolfe 1982), with female mortality higher than male mortality in some cases and particularly high mortality among infants and children (Shanks et al. 2011). In some unexposed populations in North America, mortality was also greater than expected among older adults (Cook 1976a; Wolfe 1982). For the purposes of this project, the model population was assumed to have no prior measles exposure; in other words the theorized measles epidemic of 1805-1806 was assumed to be a virgin-soil epidemic.

Model Disease Process

The disease process in this model follows an SEIR epidemic process, which consists of four disease states, susceptible (S), exposed or latent (E), infectious (I), and recovered or dead (R), through which an agent transitions over the course of the run. An agent cannot inhabit multiple disease states at the same time, and in this model, movement between states is irreversible. The SEIR disease process in commonly employed in models of infectious disease in human populations, particularly those diseases that require direct contact for transmission (see also Anderson and May, 1992; Hethcote, 2000; Keeling and Rohani, 2007; Sattenspiel, 2009 for examples and discussions of many of these models).

The disease processes are scheduled to take place after an agent has completed a time step's prescribed movement. An agent who is susceptible will examine each of its four level-1 Von Neumann neighbors (see Figure 8.2) to see if any of them are infectious. In general, Von Neumann neighbors are those inhabiting cells directly to the North, South, East and West of the agent in question. Level-1 neighbors are those who inhabit spots only one position away from the agent in question on the grid. So, Level-1 Von Neumann neighbors consist only of those individuals who are one position to the North, South, East, or West of the agent in question. This means that an agent will have a maximum of four Level-1 Von Neumann neighbors in the San Diego Model.

For each infectious neighbor, a random number between zero and one is drawn by the susceptible agent. If that random number is below the transmission probability threshold, the susceptible agent becomes exposed and will go through the disease process. If the random number is above the threshold, the disease state of the susceptible agent does not change. An agent who is infectious goes through a similar process, except that it examines its neighbors to find susceptible ones. If the infectious agent has susceptible neighbors, the process for determining whether disease transmission occurs is the same. Each agent performs these operations after it moves, so it is possible for a single agent to have multiple sets of contacts each time step, depending on how many other agents move into and out of its Von Neumann neighborhood.

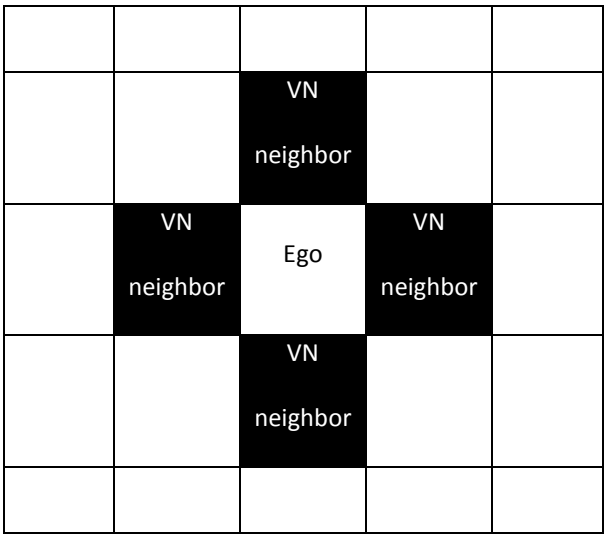


Figure 8.2 Single-level Von Neumann neighborhood

All agents in the model begin each simulation run susceptible to the disease. Disease is introduced just prior to the first step by randomly picking one agent and setting their disease status as “exposed”. The individual with the “first case” designation was recorded for each simulation run for the purposes of comparing the effect of different starting points for the epidemic. There is no variation in susceptibility or partial immunity for the model population. There is also no differential health status factor that influences either the transmission or course of the disease. An agent can only be infected once per run and once it has recovered it has full immunity for the rest of the simulation run.

Disease attributes, such as transmission probability, length of the disease states, and mortality probability are set as parameters prior to running the model. These can be changed to test the model or to simulate a different disease. Extensive sensitivity analyses to determine how variation in all parameters affected model outcomes were completed using the base model from which the San Diego Model was adapted.

Because of the similarity between the San Diego Model and the base model, further sensitivity analyses have not been completed. Instead, reasonably realistic parameter values for the San Diego Mission Model (listed in Table 8.2) were determined as described above.

Mortality and Caregiver Reassignment

The San Diego model includes mortality, and thus also includes mechanisms to provide for the care of individuals left behind by dying agents. The rules for designating new caregivers were informed by ethnographic and historical information about the Kumeyaay and rules for neophytes living at Mission San Diego.

Mortality

In this model, agent death can only happen while an agent is in the infectious state. Each agent is infectious for 6 days (144 ticks), and for each tick the agent is in that state, a random draw is compared against the probability of death variable to determine if the agent dies. Similar to the disease transmission algorithm, if the random number is lower than the probability of death variable, the agent dies.

Upon death, the agent's disease state becomes "dead" and the agent is removed from the model. A GhostPerson agent is created that contains all of the information about the dead agent plus information about the conditions surrounding its death (place, time, etc.). This GhostPerson agent occupies a spot on a special grid reserved for ghost agents and will no longer take up space in any of the model's social spaces. Ghost agents cannot interact with live person agents in any way.

Caregiver Reassignment

In certain cases, the death of an agent will require action on the part of other agents. This includes the death of a husband, or of a parent or other caregiver of children.

When a husband dies, his wife, if she is under 45, will move to the *monjería*. The priests in charge of the mission preferred to have all unmarried women (both never married and widowed) in the *monjería* in order to prevent sex outside of marriage between neophytes and between Spanish/Mexican settlers and native women. When she moves to the *monjería* the new widow brings male children under the age of 4 and all female children with her. During historic times, the widow would leave the *monjería* upon remarriage, which could occur very quickly after the death of a spouse, and form a new household with her new husband. Due to the short time span of the model, remarriage is not included at this time.

When parents die, a process is in place to ensure that children are not left without caregivers. First, the dying parent looks to see if any children belonging to the household are still alive. If so, then the agent looks to see if the mother of the children is alive. If not, the dying agent looks for other adult females in the household. If no adult female is found, any female children and all children under 4 years old must move to another household or the monjería. They first look for households containing kin either of their father or mother's patrilines. If a kin house contains an adult female, the girls and under-4s can move there. If no appropriate kin houses are located, the girls and children under 4 years old are moved to the *monjería* to be cared for. If there are no

adult females alive in the dying agent's household, but there are adult males, male children over 4 years old may stay in their natal home. If no adults are left in the original household, male children will also look for a kin house to move to, but they are not constrained by the need for a female caregiver. If they cannot find a kin house, a random house is chosen and, provided a living adult is assigned there, the boys are assigned to that new house. In the unlikely event that no house with a living adult can be found, the boy will remain at his natal house for the remainder of the simulation.

The rules of reassignment are somewhat more restrictive for girls than boys primarily to reflect the strong attempts to control female behavior in the mission setting. The population at Mission San Diego had more females than males, and missionaries were concerned with the regulation of sexual behaviors and prevention of children born out of wedlock (Bouvier 2001; Hurtado 1999). They were also concerned with preventing fraternization between indigenous neophytes (who they saw as more innocent) and soldiers and colonists of Mexican and Spanish descent (who the missionaries saw as taking advantage of the sexual openness of native peoples or using positions of power to coerce sexual activity)(Engelhardt 1920). The priests also saw the rape of native women as problematic, both in a moral sense and in terms of effectively promoting the conversion of native peoples (Jayme and Geiger 1970), and so the monjería was a way to keep indigenous women away from non-indigenous men and under a watchful eye.

Simulation Data Collection

Data are collected during model runs and output into text files. Currently, the model generates three data files, but more can be generated if the user desires. The present data files are “Final Data”, “Daily Data”, and “Case Data”. Each of these three data sets can be used to answer different questions and they contribute valuable information to the overall research project.

“Final Data” records the total number of people who are in each disease state at the end of the run. These data are useful for understanding the overall size of the epidemic and the mortality in a given run.

The “Daily Data” data set consists of the numbers of individuals in each disease state at each time tick of the simulation run. It allows for the understanding of the progression and timing of each epidemic. It also records the identity of the first case, information that can be useful for testing the impact of starting an epidemic with agents from different demographic categories on the epidemic outcome. Data for this dataset are recorded at each time tick of a simulation run.

The “Case Data” data set records data from each time tick and complied at the end of a run. It includes an extensive amount of information about each transmission event, including who infected whom, when and where the infection took place and when and where an agent’s death occurred (if it died before recovering). Ultimately the case data could be used to reproduce the exact chains of transmission throughout an epidemic, but the quantity of data is such that this would require extensive time and computer power.

Stochasticity and Emergent Phenomena

Stochasticity is built into many of the behaviors of the Simple Person agents. The first case of disease is chosen through a random pick. The order in which agents do their daily movement is random. Male children choose some of their activities according to a probability scale that changes with their age. Visiting and reassignment location choices have an element of randomness incorporated, in that agents are randomly choosing from a list of possible destinations. The disease transmission mechanics rely on agents choosing a random number and comparing it to the parameter established for disease transmission.

Generally, all random choices use Java's random number generator, which is based on a uniform distribution. In the case of the movement of male children, different test variables are generated that increase the likelihood of the child going to the fields as opposed to the kitchen, during work periods. This reflects the assumption that as male children approach adulthood, they would be more likely to take on the tasks of adult males at the mission.

The behaviors and resulting epidemics that emerge from these random fluctuations in choices are the true subject of this dissertation. Briefly, epidemic spread is governed by rules and timing of movement, but there is an element of random chance in both the movement of agents and in the transmission of disease. So, the resulting epidemic emerges from the choices made by each agent at each time step. It is hoped that the patterns found in the resulting epidemics can help us to understand disease

mortality at the historical mission and come to some greater understanding of disease mortality and population decline in 19th century Mission San Diego.

CHAPTER 9 – SIMULATION RESULTS AND DISCUSSION

The model was run 300 times using the parameter values described in Chapter 8. Sensitivity and replication analysis of a related model suggested that at least 300 runs might be needed in order to capture the variability in the model results. With a smaller number of runs, the results determined by averaging all runs could be significantly skewed by atypical runs. Data were analyzed in Excel and R. The model generated three data sets – final, daily, and case data; the results derived from each are described below. A discussion of the results in the context of the project research questions follows.

Final Data Results

The Final Data set is collected at the end of each simulation run and records the agent chosen to be the first case and how many individuals were susceptible, infectious, recovered or dead at the end of a run. In each simulation run, a check was performed to verify that no individuals remained infectious by the end; this was done to make sure that each run lasted long enough to ensure that the epidemic had ended. This data set can be used to see how often an epidemic occurs and to estimate its general severity. An epidemic is defined by the author as an instance when at least five percent of the population (in this case 72 individuals) becomes infected over the course of the simulation. In the 300 runs used for this project, epidemics resulted 291 times (97% of the time). The average number of recovered agents was 1044 and an average of 348

died. This means that not only did the epidemics exceed the 5% threshold required by the definition, but each epidemic came close to affecting the entire population of 1438.

Daily Data Results

The Daily Data set is collected at the end of each time tick of the simulation run. It collects information on how many new infections and deaths occur on each tick and can be used to track these values over the course of an epidemic. Typical results from this data set include information on the peak number of infectious individuals, when the peak occurs, and how long the epidemic lasts (measured by time of last infectious agent). Each run is slightly different because of randomness in the model, so the average of each of these measures was calculated both for all runs and for a subset that excluded non-epidemic runs. Table 9.1 shows the average peak number of infectious agents, peak time and time of last infectious agent for 300 runs of the model.

Table 9.1 Descriptive Statistics for Daily Data

	Peak Number of Infectious agents	Peak Time (in ticks)	Time of Last infectious agent (in ticks)
Mean (all runs)	365.07	1421.61	2152.75
Standard Deviation (all runs)	71.63	293.48	406.29
Mean (epidemics only)	376.33	1463.04	2216.67
Standard Deviation (epidemics only)	32.45	153.41	154.01

Approximately 25% of the population, on average, was infected at the peak of an epidemic. The average epidemic lasted 2216 ticks (about 92 days), after peaking at about 1463 ticks (day 60). To get a sense of general epidemic patterns as well as

diversity in run results, the distribution of infectious individuals for 100 runs plus the average of all epidemics (an “average epidemic run”) are shown in Figure 9.1.

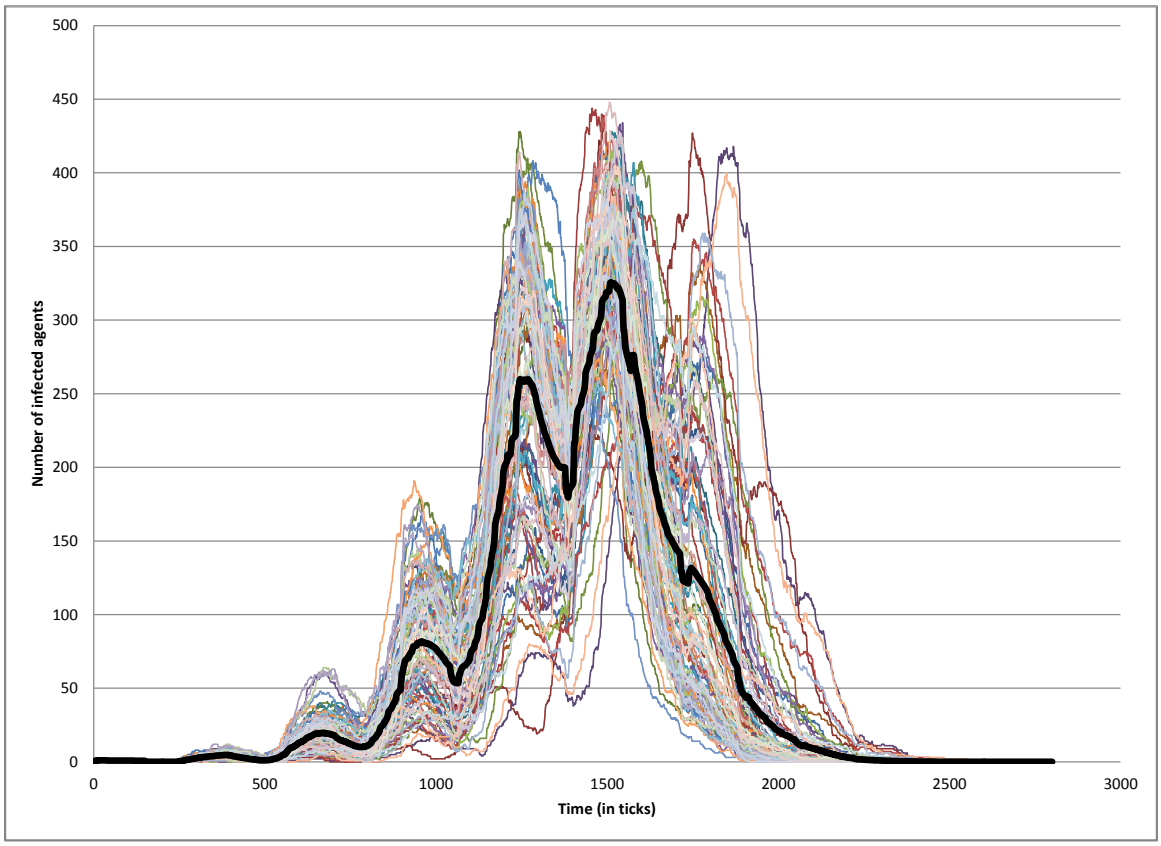


Figure 9.1 One Hundred Daily Runs and the average run (bolded)

Though significant variability is evident, many of the runs displayed in Figure 9.1 follow a similar pattern. The curves have multiple peaks over the course of the run, with the highest peak occurring between 1000 and 2000 ticks (41 to 83 days). Several short peaks appear in early in each run leading to a single high peak. These peaks are roughly 300 ticks apart, which correspond to the latent period (240 ticks) plus infectious period (144 ticks) for the disease process. So, after the initial case is chosen and infected, by about time 300 the peak resulting from the second generation of cases is evident. The

peak in the 600-700 tick range likely reflects the third generation of cases, and so on. The epidemic ends around tick 2200 (92 days), on average.

The last infectious agent variable shows the most variability of any of the variables tracked in this data set, with some epidemics continuing until almost 2800 ticks (116.6 days). The reasons for some epidemics lasting longer than others is not entirely clear, but it likely has to do with some interplay between the length of time individuals spend in the infectious category and the death rate, as deaths can often end infectious chains and cause epidemics to subside.

The distribution of deaths in the average epidemic run is seen in Figure 9.2.

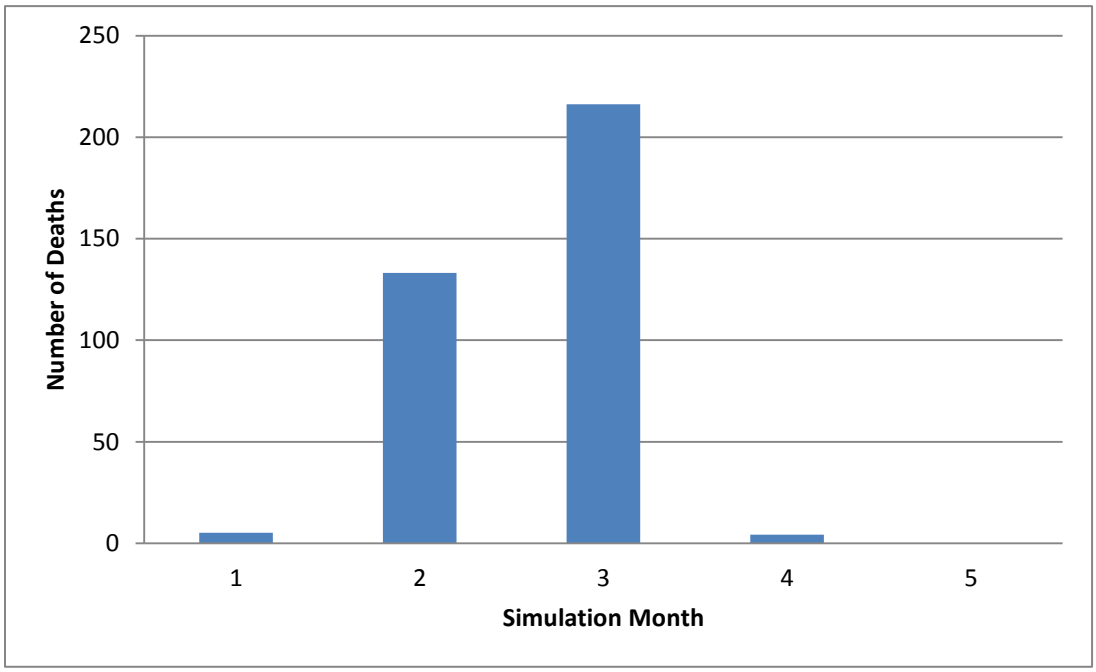


Figure 9.2 Distribution of deaths during the average epidemic run

Individual deaths were compiled into monthly numbers, in order to compare them with historical data. The distribution is unimodal with the highest number of deaths occurring in the third month of the epidemic. The peak is followed by a dramatic

decline in numbers of deaths in the fourth and fifth months. The mortality in the model was set at a fixed rate of 25%, so that any infected individual had a 25% chance of dying at some point during the infectious period. This value was chosen because it conformed to mortality estimates of this particular epidemic at other missions in Alta California and estimates of mortality rates in other historic populations (see Chapter 6).

The mortality seen during a typical model run does not compare well with mortality documented at the mission during 1805-1806 (see Figure 6.1), in which there is abrupt increase in numbers of deaths between December of 1805 (n=8 deaths) and January of 1806 (n=43) followed by decreasing, but still higher than typical, numbers of deaths in February (n=39) and March of 1806 (n=21). Deaths in April of 1806 (n=8) and subsequent months appear to return to a more typical level. The pattern of deaths in the model-generated data shows an increase in numbers of deaths between months one (n=5) and three (n=216) and a return to low numbers in month 4 (n=4). In addition to the mismatch of the patterns, the model generated many more deaths than were recorded in the historic population. There are at least three possible reasons for the overestimation of mortality by the model:

1. The model was parameterized assuming the population had no immunity or previous experience with the measles. A population with prior measles experience would have some immunity among the survivors and thus would not play a role in this epidemic.
2. The disease parameters may be set incorrectly and thus produce epidemics that are too big. The historical literature does not specifically mention measles at

Mission San Diego, so there are no data on how many people were sick, when an epidemic may have begun, or how long it lasted. Disease parameters were set according to contemporary medical knowledge, which may not capture disease behavior well in this historic population.

3. Measles was not the cause of increased deaths in the mission population. It has not been confirmed that the increased numbers of deaths in this population is a result of measles or any other infectious disease. Perhaps another explanation for these deaths is more appropriate.

While all three of these reasons could explain the lack of fit between the model output and the historical records, the first reason seems most likely given the historical circumstances. The assumption of no prior immunity was a conservative decision that recognized that the question of when European diseases first entered California is not settled. A discussion of the evidence for pre-conquest epidemics in California in general is discussed in Chapter 6. If epidemics did enter Alta California prior to the first Spanish expedition, populations in San Diego probably would have been exposed first, due to geographic proximity to other Spanish colonies. Trade between Northern Mexico and Southern California was well established prior to Spanish conquest (Davis 1974), so it is very likely that a disease with a long latent period like measles could have been brought into Southern California communities prior to 1769. If that was the case, then it is likely that the population at the mission community had some number of adults with full immunity due to prior infection. Future research plans include adjusting the model population to add immunity to some adult individuals.

It is not likely that the disease transmission variables were incorrectly parameterized, as measles is quite well known, both in contemporary and historic contexts. The epidemics generated by the model compare well with observed measles epidemics in terms of their shape and progress through the population.

The case for measles at Mission San Diego is a circumstantial one, based on historical knowledge of measles at nearby missions around the time that numbers of deaths at Mission San Diego increased dramatically. It is possible that the increase in deaths over the winter of 1805-1806 at San Diego was due to some other cause. However, the fact that the disease is well-known at other missions in the area, and is known to be the cause of increased mortality at those missions provides support for the idea that measles may be the cause of at least some of the excess mortality at Mission San Diego.

Cases Data Results

This data set has perhaps the most potential for data analysis. At the end of a simulation run, the computer gathers information from each agent that includes whether they became infected, whether they died, who infected them and whether they shared either a matriline or patriline with their infector, when and where they were infected, and when and where they died. In essence, this data set can be used to reconstruct the epidemic from the first to the last cases. The case data for all 300 runs were compiled and compared to get a feel for the behavior of the model.

As described in Chapter 8, to start the epidemic the model randomly chooses an agent to infect and designates that agent as the "first case". Table 9.2 shows how many first cases come from each of the occupational categories.

Table 9.2 Distribution of Runs by First Case Occupation

Occupation Code	Demographic Attribute	Number of Runs as First Case
100	Married Male	76
200	Married Female	107
300	Unmarried Male	31
400	Unmarried Female	21
500	Male Child	24
600	Female Child	12
700	Infant of Either Sex	29
800	Priest	0

Model data were examined to discover any variations that might be due to differences in which agent was the first case that started the epidemic (or non-epidemic). Initially, the thought was that those runs beginning with an unmarried female (occupation code 400) might be less likely to result in an epidemic, due to movement restrictions. This does not appear to be the case. There does not appear to be a relationship between first case and whether or not an epidemic will ensue, though this is difficult to test statistically because there are so few non-epidemics in the sample. Table 9.3 shows the number of non-epidemics for each of the first case occupation codes.

Table 9.3 Distribution of Non-epidemic runs by first case occupation

Occupation Code of First Case	Number of Non-Epidemic Runs
100	2
200	2
300	1
400	1
500	1
600	0
700	2

Across all 300 runs, an average of 476 agents per run was infected by kin; that is other agents who shared either a matriline or patriline designation with the agent. Almost twice that, 958, was infected by agents who were not direct kin, on average. The most common place of infections in all runs was the agent's home dwelling. The second most common place was the church building or, in a few cases, the fields.

These results are intriguing because it was assumed that if individuals were being infected in their homes, then their infectors would be kin, either kin who shared a dwelling (immediate family) or kin who were visiting (extended family). So, for the modal infection location to be one's home and the most common infector to be a non-kin individual, there must be something else at work. One possibility is that women are being infected as they inhabit outdoor space around their homes. These spaces allow

women whose homes are close together to congregate in an informal way, as part of their daily routines of activities around their dwelling.

Another interpretation is that data collection methods do not adequately record kin relationships. The procedure for designating someone "kin" takes into account only whether ego's patriline and matriline match that of the infector. Because of marital exogamy, if a female encounters a relative of her husband's she does not recognize him as kin, unless they happen to have a matching matriline. The same problem would occur for a head of household male and his wife's family. Additionally, children may not recognize matrilineal kin of their mother and father for the same reason. This explanation could be tested by correcting for unrecognized kin in later versions of the model.

The church as a common location of infection is not surprising, as everyone in the community was expected to attend church daily and for some groups of people (children) the church was a center of additional activities. Additionally, during church services, the church is densely packed, which gives infected individuals many opportunities to infect susceptible individuals.

Timing of infections was also analyzed using this dataset. Agents were grouped into two categories, those who were infected by kin (K) and those infected by non-kin (NK). Mann-Whitney U tests were calculated for each run comparing the median time of infection for the group of individuals infected by kin vs. the group of individuals infected by non-kin. Significant results ($p \leq 0.05$) were obtained for 82 of the 291 epidemic runs

(28.1%). Table 9.4 shows the breakdown of results when testing for timing of infections for kin vs. non-kin.

Table 9.4 Mann-Whitney U tests comparing time of infection for those infected by kin vs. non-kin

	N = 300 runs
No epidemic	9
Epidemic with no significant difference in timing of infection	209
Epidemic with significantly later infections when infected by non-kin (NK)($p<0.05$)	4
Epidemic with significantly later infections when infected by kin (K)($p<0.05$)	78

The average difference in timing of infection for those runs where NK infections took place significantly later than K infections was 19 ticks (0.8 days), with a minimum of 13.5 ticks (0.56 days) and a maximum of 21.5 ticks (0.90 days). So, when non-kin infections take place significantly later than kin infections, it is not qualitatively much later (less than 1 day).

The average difference in timing of infection for those runs where K infections took place significantly later than NK infections was 38.2 ticks (1.6 days), with a minimum of 4.5 ticks (0.19 days) and a maximum of 101 ticks (4.2 days). The differences for statistically significant runs of this type were much more variable than for significant runs where NK infections were later, though this may be because of the much larger number of significant runs of this type. Qualitatively, these results point to a slight lag between individuals infected by kin vs. those infected by non-kin, typically about 1.5 days.

It is difficult to read too much into these results due to potential problems that agents have in actually recognizing kin. Given the fact that the case data set includes detailed information on all infection events, it would be possible to go back through the data and find the kinship connections that might have been missed, but it would be extremely time intensive. The results that have been analyzed point to some inconsistencies that indicate the need to reevaluate kin recognition systems in the program. For example, it seems highly unlikely for the modal infection location to be an agent's own home if they are almost twice as likely to be infected by a non-kin member as by a kin member. Agents do interact with non-kin, but they typically do so in public spaces, like church and the fields.

Summary of findings

The data produced by the model thus far do provide some insight into the historical population at Mission San Diego. It seems very unlikely that, if the increased number of deaths in late 1805 do represent measles-related mortality, that the measles epidemic was a virgin soil epidemic. The measles hypothesis should be reevaluated, incorporating the idea that the population probably had experience with the disease and some individuals may be immune at the time of a new epidemic. This is an important finding, as it does provide some evidence that is not available via investigations into the primary historical literature.

The progression of infections during the epidemic is interesting and appears consistent with real-world measles epidemics ("Measles Outbreak" 2000; Torner et al. 2007), especially in regards to how the number of cases increases in a stepped fashion

to the peak, followed by a large decline in number of cases at the end of the epidemic. The timing of the virtual epidemic in comparison to the historically known increase in deaths at Mission San Diego does not compare well. The virtual epidemic is somewhat shorter than the time span during which deaths are increased at the mission. It is possible that due to parameter values in the model that are higher than they should be, the virtual epidemic progresses faster than it should. Another interpretation is that the increase in deaths at the mission is due to multiple factors in addition to measles, which explains why it lasts longer than expected. Other factors possibly contributing to increased number of deaths could include other diseases, both acute infectious and chronic, like dysentery, syphilis, or influenza, all of which were documented at other missions and may have impacted local conditions at San Diego without prompting a regional epidemic that would have been recorded at other missions. Other theoretical explanations for increases in numbers of death could include civil unrest or natural disasters, but nothing like that is known from the historical record. It seemly likely than another attack or conflict would be mentioned, given the mission had already been attacked in 1775. Also, the fact that the increase in deaths happens over a long period of time rules out any single event, like an attack or an earthquake, causing a large number of deaths.

The data that specifically describe patterns of infection within the community and timing of infections seem to be confounded by problems with the program itself. That is, the computer program is not capable of recognizing all the potential relatives that an individual may come into contact with, and thus mis-records information about

whether or not familial contact results in disease spread. These initial results provide some insights into the original questions about the cause of increased deaths at San Diego in 1805 and 1806, but the detailed information regarding epidemic spread in this community needs to be considered cautiously. The results indicate that the disease process in the model is functioning as expected, but the epidemics generated appear to overestimate the amount of mortality, compared to the historical record. It is important to rectify this issue before attempting to look too closely at the networks of transmission and the role of family structures.

Analysis of the Case Data set could not be fully addressed due to current limitations in the computer program, as described above. Initial findings were counterintuitive in terms of place of infection and identity of infector, leading to the discovery that kinship was not always being recognized properly by the program. Thus, some adjustments will have to be made to the data collection methods before any further research is completed. Problems aside, the model is successful at reproducing a very complicated population structure with complex activity schedules. Simulations generate realistic measles epidemics, in comparison to published accounts of measles. Once the corrections are made so that the model records kin properly, information gained from the output could provide some interesting insights into the way that disease is spread in this community.

CHAPTER 10 – CONCLUSIONS

This project consisted of three distinct phases, each of which contributed to answering the questions posed at the outset. The first phase included the data gathering and family reconstitution of the 1805 mission population at Mission San Diego de Alcala for its use as a reference population. The second phase was the development of a computer simulation model to address the effects of an acute infectious disease in a population modeled on the real population at Mission San Diego de Alcala. The third phase was to run the model and analyze the results to see how kinship and age- and gendered-behavior may have affected the spread of measles in the model population. The results of each phase have been discussed in previous chapters, but will be summarized below.

Data Gathering and Family Reconstitution

As described in Chapter 7, the population of individuals alive at Mission San Diego as of January 1, 1805 was reconstructed. This was completed using the baptismal, marriage, and death records created at the mission on the occasion of the administration of each sacrament (baptism, marriage, burial rites). Reconstruction revealed a population of 1443 people at the mission or living within its sphere of influence. This is consistent with general historical documentation from administrative records at the mission.

The population at the mission was slightly older than expected for a natural fertility population. Infants and children likely succumbed at higher rates than adults to

diseases of all types, a problem noted by contemporary writers observing mission life. Though the adults were overrepresented, most adults were baptized (and thus "converted" to mission life) as either infants or young adults. It does not appear that older adults made the transition to mission life as readily as young adults. This is particularly true for males, who have an overall earlier age at baptism indicating it was more likely they were baptized as babies or young children, not as individuals acting with autonomy.

Age at first marriage was also reconstructed in order to assist in age estimation for individuals without recorded ages. In both the Kumeyaay and 18th and 19th century Catholic societies, marriage at an early age (typically by 18 years old) was common, so the low average age at marriage seen at the mission would not have been a big adjustment for converted Kumeyaay people. That being said, not much research has looked into the specifics of spousal choice and the role of traditional mate choice rules at Mission San Diego. This avenue of research could provide some very interesting insight into the real autonomy of Kumeyaay people living within the mission system.

Model Development

The development of an agent-based computer simulation model was completed as a part of this project. The model is structured around a dynamic network that is based on the historically known population of Kumeyaay neophytes at Mission San Diego and their activity patterns. Using information gained during the data gathering phase, a historically contextualized schedule of activities was developed and implemented. The behavior of model agents is based on published primary and

secondary documentation from historic, ethnographic and theological resources. When published resources could not provide specific answers about individual behaviors, behavior was programmed to comply with what seemed most likely given the context.

Future research will be able to use both the reconstructed population and the completed model to answer a number of questions that are currently beyond the scope of this dissertation. Now that the population has been reconstituted, a number of basic demographic questions can be addressed, including change in birth rates and death rates over time, investigations into infant and maternal mortality, and further examination of marriage practices at the mission. The model, once corrected, can be run again to address similar questions of the role of kin in disease spread. Questions about timing of epidemics or the relative importance of the identity of the first case may also provide interesting avenues of research.

Hypothesis testing using the model

The hypotheses posed at the beginning of this project were designed to investigate the relationship between the spread of an infectious disease at the mission and kinship among the mission community. The main findings are as follows:

1. It cannot be confirmed that the increased numbers of deaths seen during the 1805-1806 winter were due to a measles outbreak at the mission. The historically documented increase in deaths is not entirely inconsistent with what the model produced, but the numbers of deaths produced by the model far exceed the actual number of deaths seen during the months in question. Additionally, though the model compares well with real-world measles

epidemics, it does not compare well to the distribution of deaths during the time period in question at San Diego.

2. Based on initial findings, it appears that the model is not currently structured in a way that supports the investigation of the role of kinship in the spread of disease. Contradictory data regarding the source of infections and the place of infections demand further attention be paid to how agents within the model recognize and track their kin.
3. Though the model and its results are not yet adequate to answer the questions posed at the beginning of this project, the model represents a significant achievement in terms of describing the way that an infectious disease moves through a population of individuals who are endowed with complex behaviors.

Further Research

There are avenues for further research in each of the three phases of this project. First, continued demographic analysis of population dynamics at Mission San Diego is planned. Questions regarding changes in birth and death rates over time, population adjusted rates, and fluctuations in migration patterns could all be investigated. As mentioned above, understanding the influences behind spousal choice is of particular interest. This line of questioning would provide significant contribution to historical understanding of population dynamics in the Alta California missions, as much of the research to date has focused on missions along the central and northern coasts whose populations come from very different cultural backgrounds than that of Mission San Diego.

Work with the model will continue. Data gathering methods need to be refined in order to gather the appropriate data to answer the kinds of questions posed by this project. One of the challenges encountered during data analysis was identifying a strategy that allowed for a rigorous, statistical comparison of the results across the 300 model runs. Statistical tests could compare the behavior of groups within a single simulation run (for example the timing of infection for those infected by kin vs. those infected by non-kin), but a strategy for comparing those results across all 300 runs was not identified. For example, Table 9.4 shows the simple counts of statistical significance for one set of tests, but coming up with a way to represent these counts in terms of "typical" or "atypical" model behavior would be very useful. Further investigation into these types of methods will continue in an effort to take advantage of the data this model is able to produce.

Finally, it is my hope to incorporate the experiences of contemporary Kumeyaay in a more meaningful way. There are questions about long-term health implications of mission-era activities that can be addressed, and that could be of interest to the descendants of the mission population. Genealogies for current Kumeyaay band members could link the present-day community to the mission-era community could be possible. In that the case, a number of life-history or demographic studies that look at fertility and mortality, family size, or marriage pattern changes over time would be of great interest. Using ethnographic methods to study the way in which present-day Kumeyaay formulate health-promoting or health-seeking behaviors in the context of their history is something that has potential to make studies of

mission-era communities of real value to indigenous populations today. How does their understanding of what happened at the mission impact their use of social institutions, their identity as Native Californians, and their perceptions of the role of medical institutions like the Indian Health Service or other governmental health agencies?

This project made use of a wide variety of data sources and analytical techniques, including historical demography, ethnohistory, agent-based modeling and statistical analysis, in order to increase detailed knowledge about historical populations. It has provided an in-depth understanding of the population at Mission San Diego at the turn of the 19th century. Though the model itself was of limited use in terms of answering questions posed at the outset of the project, preliminary data revealed that the pattern of deaths at the mission during this time were not consistent with a virgin soil measles epidemic, prompting questions about the pre-conquest disease transfer between European and native Californian populations. This finding alone is significant, as there is some disagreement about the potential for epidemics of European diseases in Alta California prior to 1769. Further research with this model will address this and other questions about the population of Kumeyaay at Mission San Diego and their experiences in a time of drastic culture change.

REFERENCES CITED

2010 Census Tribal Statistical Areas Program: Guidelines for Digital Participants. (2009) In: Bureau USC, editor. Washington, D.C.

Aaby P, Bukh J, Lisse IM, and Smits AJ. 1983. Measles mortality, state of nutrition, and family structure: a community study from Guinea-Bissau. The Journal of infectious diseases 147(4):693-701.

Anderson MK, Barbour MG, and Whitworth V. 1998. A World of Balance and Plenty: Land, Plants, Animals, and Humans in a Pre-European California. In: Gutiérrez R, and Orsi R, editors. Contested Eden: California Before the Gold Rush. Berkeley: University of California Press. p 12-47.

Anderson RM, Grenfell BT, and May RM. 1984. Oscillatory Fluctuations in the Incidence of Infectious Disease and the Impact of Vaccination: Time Series Analysis. The Journal of Hygiene 93(3):587-608.

Anderson RM, and May RM. 1992. Infectious diseases of humans : dynamics and control. Oxford; New York: Oxford University Press.

Andrushko VA, Latham KA, Grady DL, Pastron AG, and Walker PL. 2005. Bioarchaeological evidence for trophy-taking in prehistoric central California. Am J Phys Anthropol 127(4):375-384.

Asenso-Okyere K. 2009. The linkages between agriculture and malaria issues for policy, research, and capacity strengthening. Washington, DC [etc.]. International food policy research institute (IFPRI).

Bagni R, Berchi R, and Cariello P. 2002. A comparison of simulation models applied to epidemics. Journal of Artificial Societies and Social Simulation 5(3).

Barbujani G, Sokal RR, and Oden NL. 1995. Indo-European origins: A computer-simulation test of five hypotheses. American Journal of Physical Anthropology 96(2):109-132.

Bartelink EJ. 2006. Resource Intensification in Pre-Contact Central California: A Bioarchaeological Perspective on Diet and Health Patterns Among Hunter-Gatherers from the Lower Sacramento Valley and San Francisco Bay [Dissertation]. College Station: Texas A&M.

Borah WW, and Cook SF. 1960. The population of central Mexico in 1548; an analysis of the Suma de visitas de pueblos. Berkeley: University of California Press.

Bouvier VM. 2001. Women and the Conquest of California. Tucson: University of Arizona Press.

Bowman JN. 1964. The Birthdays of the California Missions. The Americas 20(3):289-308.

Brantingham PJ. 2003. A Neutral Model of Stone Raw Material Procurement. American Antiquity 68(3):487-509.

Broughton JM. 1994. Declines in Mammalian Foraging Efficiency During the Late Holocene, San Francisco Bay, California. Journal of Anthropological Archaeology 13:371-401.

Broughton JM. 2002. Prey Spatial Structure and Behavior Affect Archaeological Tests of Optimal Foraging Models: examples from the Emeryville Shellmound vertebrate fauna. World Archaeology 34(1):60-83.

Broughton JM, , and Bayham FE. 2003. Showing off, Foraging Models, and the Ascendance of Large-Game Hunting in the California Middle Archaic. American Antiquity 68(4):783-789.

Canteñs B. 2013. The Rights of American Indians. In: Nuccetelli S, Ofelia S, and Otávio B, editors. A Companion to Latin American Philosophy. Chichester, UK: Wiley-Blackwell.

Carman G. 2010. On the Pope's Original Intent: Las Casas Reads the Papal Bulls of 1493. Colonial American Review 7(2):193-204.

Carpenter C, and Sattenspiel L. 2009. The design and use of an agent-based model to simulate the 1918 influenza epidemic at Norway House, Manitoba. American Journal of Human Biology 21(3):290-300.

Castañeda CE. 1949. Fray Juan de Zumárraga and Indian Policy in New Spain. The Americas 5(3):296-310.

Castañeda CE. 1954. Spanish Medieval Institutions in Overseas Administration: The Prevalence of Medieval Concepts. The Americas 11(2):115-129.

Clevis Q, Tucker GE, Lock G, Lancaster ST, Gasparini N, Desitter A, and Bras RL. 2006. Geoarchaeological simulation of meandering river deposits and settlement distributions: A three-dimensional approach. Geoarchaeology 21(8):843-874.

Clinch BJ. 1904. California and its Missions: Their History to the Treaty of Guadalupe Hidalgo.

Cline LL. 2008. Just Before Sunset. San Diego, CA: Sunbelt Publications, Inc.

Cook SF. 1976a. The conflict between the California Indian and white civilization. Berkeley: University of California Press.

Cook SF. 1976b. The population of the California Indians, 1769-1970. Berkeley: University of California Press.

Costansó M, Hemert-Engert Av, Teggart FJ, and Hopkins RC. 1910. The narrative of the Portola expedition of 1769-1770. Berkeley: University of California.

Crespí J, and Brown AK. 2001. A description of distant roads: original journals of the first expedition into California, 1769-1770. San Diego, CA: San Diego State University Press.

Crosby J, Alfred W. 1972. The Columbian Exchange: Biological and Cultural Consequences of 1492. Westport, CT: Greenwood Press.

Cuello J. 1988. The Persistence of Indian Slavery and Encomienda in the Northeast of Colonial Mexico, 1577-1723. Journal of Social History 21(4):683-700.

Davidson G, Ferrelo B, Cabrillo JR, Vizcaíno S, Ulloa Fd, Coast US, and Geodetic S. 1887. Methods and results : voyages of discovery and exploration on the northwest coast of America from 1539 to 1603 : Appendix No. 7--Report for 1886. Washington [D.C.]: Govt. Print. Off.

Davis JT. 1961. Trade routes and economic exchange among the Indians of California. Ramona, Calif.: Ballena Press.

Duhaut-Cilly AB. 1929. Duhaut-Cilly's account of California in the years, 1827-28. San Francisco: California Historical Society.

Eerkens JW, King J, and Wohlgemuth E. 2002. The Prehistoric Development of Intensive Green-Cone Pinon Processing in Eastern California. Journal of Field Archaeology 29(1/2):17-27.

Engelhardt Z. 1920. San Diego mission. San Francisco, Cal.: James H. Barry Co.

Epstein JM. 2008. Why Model? Journal of Artificial Societies and Social Simulation 11(4):12.

Epstein JM, Axtell R, Project, Institution B, Institute SF, and Institute WR. 1996. Growing artificial societies : social science from the bottom up. Washington, D.C.: Brookings Institution Press.

Erlandson JM. 2001. The Archaeology of Aquatic Adaptations: Paradigms for a New Millennium. Journal of Archaeological Research 9(4):287-350.

Erlandson JM, and Bartoy K. 1995. Cabrillo, the Chumash and Old World Diseases. Journal of California and Great Basin Anthropology 17(2):153-173.

Erlandson JM, and Bartoy K. 1996. Protohistoric California: Paradise or Pandemic? Proceedings of the Society for California Archaeology 9:304-309.

Erlandson JM, Braje TJ, Rick TC, and Peterson J. 2005. Beads, Bifaces, and Boats: An Early Maritime Adaptation on the South Coast of San Miguel Island, California. American Anthropologist 107(4):677-683.

Erlandson JM, Torben CR, Kennett DJ, and Walker P. 2001. Dates, Demography and Disease: Cultural Contacts and Possible Evidence for Old World Epidemics Among the Protohistoric Island Chumash. Pacific Coast Archaeological Society Quarterly 37(3):11-26.

Eubank S, Guclu H, Anil Kumar VS, Marathe MV, Srinivasan A, Toroczkai Z, and Wang N. 2004. Modelling disease outbreaks in realistic urban social networks. Nature 429(6988):180-184.

Ferber R, and Schenk OB. 1960. Mathematical Models of Market Simulation. Journal of Marketing 24(4):69-74.

Ferguson NM, Cummings DAT, Cauchemez S, Fraser C, Riley S, Meeyai A, Iamsirithaworn S, and Burke DS. 2005. Strategies for containing an emerging influenza pandemic in Southeast Asia. Nature 437(7056):209-214.

Franco-Paredes C, Lammoglia L, Santos-Preciado JI, Henderson DA, Inglesby JTV, and O'Toole T. 2005. The Spanish Royal Philanthropic Expedition to Bring Smallpox Vaccination to the New World and Asia in the 19th Century. Clinical Infectious Diseases 41(9):1285-1289.

Garriga A, and Weber FJ. 1978. Andrew Garriga's compilation of herbs & remedies used by the Indians & Spanish Californians : together with some remedies of his own experience. S.l.: s.n.].

Geiger M. 1947. The Mallorcan Contribution to Franciscan California. The Americas 4(2):141-150.

Gifford EW. 1931. The Kamia of Imperial Valley. Washington: U.S. G.P.O.

Gilbert JP, and Hammel EA. 1966. Computer Simulation and Analysis of Problems in Kinship and Social Structure. American Anthropologist 68(1):71-93.

Gilbert N. 2008. Agent-Based Models. Los Angeles: Sage Publications.

Glassow MA. 1999. Measurement of Population Growth and Decline During California Prehistory. Journal of California and Great Basin Anthropology 21(1):45-66.

Grimm V, Berger U, DeAngelis DL, Polhill JG, Giske J, and Railsback SF. 2010. The ODD Protocol: A review and first update. Ecological Modeling 221:2760-2768.

Hackel SW. 2005. Children of Coyote, Missionaries of Saint Francis: Indian-Spanish Relations in Colonial California, 1769-1850. Chapel Hill: University of North Carolina Press.

Heath B, Hill R, and Ciarallo F. 2009. A Survey of Agent-Based Modeling Practices (January 1998 to July 2008). Journal of Artificial Societies and Social Simulation 12(4):9.

Hedder J. 1915. Story of California Olive Industry. Overland Monthly and Out West Magazine 66(6):574-574.

Hedges K. 1975. Notes on the Kumeyaay: A Problem of Identification. Journal of California Anthropology 2:71-83.

Henry L, and Fleury M. 1956. Des registres paroissiaux à l'histoire de la population: manuel de dépouillement et d'exploitation de l'état civil ancien. Population (French Edition):142-144.

Herring DA, and Sattenspiel L. 2007. Social contexts, syndemics, and infectious disease in northern Aboriginal populations. Am J Hum Biol 19(2):190-202.

Hersh B. 2004. Measles. In: Heymann DL, editor. Control of Communicable Diseases Manual. 18th ed. Washington, D.C.: American Public Health Association. p 347-354.

Hethcote HW. 2000. The Mathematics of Infectious Diseases. SIAM Review 42:599-653.

Hittell J. 1888. The Doom of the Californian Aborigines. Overland monthly and Out West magazine 11(66):609-613.

Hoffman PE. 1973. Diplomacy and the Papal Donation 1493-1585. The Americas 3(2):151-183.

Hornbeck D, Fuller DL, and Kane PS. 1983. California patterns : a geographical and historical atlas. Palo Alto, Calif.: Mayfield Pub. Co.

Hotchkiss JRMD, Strike DGMD, Simonson DABA, Broccard AFMD, and Crooke PSP. 2005. An agent-based and spatially explicit model of pathogen dissemination in the intensive care unit *. Critical Care Medicine 33(1):168-176.

Hovland CI. 1960. Computer simulation of thinking. American Psychologist 15(11):687-693.

Hurtado AL. 1999. Intimate Frontiers: sex, gender, and culture in old California. Albuquerque: University of New Mexico Press.

Jackson RH, and Castillo ED. 1995. Indians, Franciscans, and Spanish colonization : the impact of the mission system on California Indians. Albuquerque: University of New Mexico Press. vii, 214 p. p.

James GW. 1913. The old franciscan missions of California. Boston: Little, Brown, and company.

Jayme L, and Geiger MJ. 1970. Letter of Luís Jayme, O.F.M., San Diego, October 17, 1772. Los Angeles: Published for San Diego Public Library by Dawson's Book Shop.

Jennings NH, and Dickins JH. 1958. Computer simulation of peak hour operations in a bus terminal. Management Science 5(1):106-120.

Johnson CD, Kohler TA, and Cowan J. 2005. Modeling Historical Ecology, Thinking about Contemporary Systems. American Anthropologist 107(1):96-107.

Kasakoff AB, and Adams JW. 1995. The Effect of Migration on Ages at Vital Events: A Critique of Family Reconstitution in Historical Demography. European Journal of Population 11:199-242.

Keeling MJ, and Grenfell BT. 1997. Disease extinction and community size: modeling the persistence of measles. Science (New York, NY) 275(5296):65-67.

Keeling MJ, and Rohani P. 2007. Modeling infectious diseases in humans and animals. Princeton, N.J. ;Woodstock: Princeton University Press.

Kelsey H. 1985. European Impact on the California Indians 1530-1830. The Americas 41(4):494-511.

Kelsey H. 1986. Juan Rodríguez Cabrillo. San Marino, CA: Huntington Library.

Kim-Farley RJ. 1993. Measles. In: Kiple KF, editor. The Cambridge World History of Human Disease. Cambridge: Cambridge University Press. p 871-875.

Kohler TA. 2000. Putting Social Sciences Together Again: An Introduction to the Volume. In: Kohler TA, and Gumerman GG, editors. Dynamics in Human and Primate Societies. Oxford: Oxford University Press. p 1-18.

Kohler TA, and Gumerman GG, editors. 2000. Dynamics in Human and Primate Societies: Agent-Based Modeling of Social and Spatial Processes. Oxford: Oxford University Press.

Kottek M, Grieser J, rgen, Beck C, Rudolf B, and Rubel F. 2006. World Map of the Koppen-Geiger climate classification updated. Meteorologische Zeitschrift 15(3):259-263.

Kroeber AL. 1955. Nature of the Land-Holding Group. Ethnohistory 2(4):303-314.

Kroeber AL. 1976. Handbook of the Indians of California. New York: Dover Publications, Inc. 995 p.

Kumeyaay History. Kumeyaay: Kumeyaay History (2013). <http://www.kumeyaay.com/kumeyaay-history.html>

Lambert PM. 1993. Health in Prehistoric Populations of the Santa Barbara Channel Islands. American Antiquity 58(3):509-522.

Larson DO, Johnson JR, and Michaelsen JC. 1994. Missionization among the Coastal Chumash of Central California: A Study of Risk Minimization Strategies. American Anthropologist 96(2):263-299.

Laylander D. 2000. Early ethnography of the Californias, 1533-1825. Salinas, CA: Coyote Press.

Levine D. 1976. The Reliability of Parochial Registration and the Representativeness of Family Reconstitution. Population Studies 30(1):107-122.

Lightfoot KG, and Parrish O. 2009. California Indians and Their Environment: an Introduction. Berkeley: University of California Press.

Livingstone FB. 1958. Anthropological Implications of Sickle Cell Gene Distribution in West Africa1. American Anthropologist 60(3):533-562.

Longini IM, Nizam A, Xu S, Ungchusak K, Hanshaoworakul W, Cummings DAT, and Halloran ME. 2005. Containing Pandemic Influenza at the Source. Science 309(5737):1083-1087.

Longstreth GF, and Wilken-Robertson M. 2010. Northern Baja California Indian women's concepts of illness and healing: Implications for public health and clinical practitioners. Global Public Health 5(6):626-638.

Luomala K. 1978. Tipai-Ipai. In: Heizer RF, editor. Handbook of North American Indians, Vol 8 California. Washington, D.C.: Smithsonian Institution. p 592-608.

Lyman GD. 1925. The Beginnings of California Medical History. California and Western Medicine 23(5):561-576.

MacCluer JW, Neel JV, and Chagnon NA. 1971. Demographic Structure of a Primitive Population: A Simulation. American Journal of Physical Anthropology 35(2):193-207.

Mann CC. 2011. 1493: uncovering the new world Columbus created. New York: Knopf.

Measles Outbreak--Netherlands, April 1999-January 2000. MMWR: Morbidity & Mortality Weekly Report 49(14):299.

Mesoudi A, and Lycett SJ. 2009. Random copying, frequency-dependent copying and culture change. Evolution and Human Behavior 30(1):41-48.

Mesoudi A, and O'Brien MJ. 2008. The Cultural Transmission of Great Basin Projectile-Point Technology II: An Agent-Based Computer Simulation. American Antiquity 73(4):627-644.

Milliken R. 1995. A time of little choice : the disintegration of tribal culture in the San Francisco Bay area, 1769-1810. Menlo Park, CA/Novato, CA: Ballena Press ;

Moratto MJ. 1984. California archaeology. Orlando: Academic Press.

Morrison AE, and Addison DJ. 2008. Assessing the role of climate change and human predation on marine resources at the Fatu-ma-Futi site, Tutuila Island, American Samoa: an agent based model. Archaeology in Oceania 43(1):22-34.

Nagano A, Umberger BR, Marzke MW, and Gerritsen KG. 2005. Neuromusculoskeletal computer modeling and simulation of upright, straight-legged, bipedal locomotion of Australopithecus afarensis (A.L. 288-1). American Journal of Physical Anthropology 126(1):2-13.

Natella AA. 1975. The Spanish in America, 1513-1974 : a chronology and fact book. Dobbs Ferry, N.Y.: Oceana Publ.

North M, Collier N, Ozik J, Tatara E, Macal C, Bragen M, and Sydelko P. 2013. Complex adaptive systems modeling with Repast Simphony. Complex Adaptive Systems Modeling 1(1):3.

Nunis J, Doyce B. 1994. Medicine in Spanish California. Southern California Quarterly 76(1):31-57.

Nunn CL, Mulder MB, and Langley S. 2006. Comparative Methods for Studying Cultural Trait Evolution: A Simulation Study. Cross-Cultural Research 40(2):177-209.

O'Neil CA, and Sattenspiel L. 2010. Agent-based modeling of the spread of the 1918–1919 flu in three Canadian fur trading communities. American Journal of Human Biology 22(6):757-767.

Orbann CM. 2006. Inferring precontact demographic trends from the historical record : a case study using four Patwin tribelets: California State University, Chico. 96 p.

Panum PL, Hatcher AS, Petersen JJ, and Dimont J. 1910. Observations made during the epidemic of measles on the Faroe islands in the year 1816. New York: Distributed by the American public health association, Printed by F. H. Newton.

Peterson H. 2012. Mission San Diego de Alcala "Mother of the Missions". In: Survey HAL, editor. Washington, D.C.: US Dept of the Interior.

Powers S. 1976. Tribes of California. Berkeley: University of California Press.

Preston SH, Heuveline P, and Guillot M. 2001. Demography: Measuring and Modeling Population Processes. Malden, MA: Blackwell Publishing

Raab LM, and Jones TL, editors. 2004. Prehistoric California: Archaeology and the Myth of Paradise. Salt Lake City, UT: University of Utah Press.

Railsback SF, and Grimm V. 2011. Agent-based and individual-based modeling: a practical introduction: Princeton University Press.

Ramenofsky AF. 1996. The Problem of Introduced Infectious Diseases in New Mexico: A.D. 1540-1680. Journal of Anthropological Research 52(2):161-184.

Reitman WR. 1960. Heuristic programs, computer simulation, and higher mental processes. Computers in Behavioral Science: Behav. Sci.: Amsterdam, Holland.

Reynolds CW. 1987. Flocks, herds and schools: A distributed behavioral model. SIGGRAPH Comput Graph 21(4):25-34.

Reynolds RG, Whallon R, and Goodhall S. 2000. Transmission Of Cultural Traits By Emulation: An Agent-Based Model Of Group Foraging Behavior. Journal of Memetics - Evolutionary Models of Information Transmission 4(2):34.

Roehner BM. 1997. Jesuits and the State: A Comparative Study of their Expulsions (1590–1990). Religion 27(2):165-182.

Sánchez J, and Barona J. 1804. Informe del estado en que se hallaba la Mision de San Diego, a fin del año de 1802 hasta el fin de 1804. Mission San Diego de Alcalà.

Sattenspiel L, and Lloyd A. 2009. The geographic spread of infectious diseases : models and applications. Princeton: Princeton University Press.

Settipane MD, Guy A., editor. 1995. Columbus and the New World: Medical Implications. Providence: OceanSide Publications, Inc.

Shackley MS, editor. 2004. The early ethnography of the Kumeyaay. Berkeley: Phoebe Hearst Museum of Anthropology.

Shanks GD, Lee S-E, Howard A, and Brundage JF. 2011. Extreme Mortality After First Introduction of Measles Virus to the Polynesian Island of Rotuma, 1911. American Journal of Epidemiology 173(10):1211-1222.

Shipek FC. 1981. A Native American Adaptation to Drought: The Kumeyaay as seen in the San Diego Mission Records 1770-1798. Ethnohistory 28(4):295-312.

Shipek FC. 1982. Kumeyaay Socio-Political Structure. Journal of California and Great Basin Anthropology 4(2):296-303.

Shipek FC. 1985. California Indian Reactions to the Franciscans. The Americas 41(4):480-492.

Smith DE, and Teggart FJ. 1909. Diary of Gaspar de Portola during the California Expedition of 1769-1770. Publications of the Academy of Pacific Coast History 1(3).

Smith DS. 1975. Underregistration and Bias in Probate Records: An Analysis of Data from Eighteenth-Century Hingham, Massachusetts. The William and Mary Quarterly 32(1):100-110.

Spier L. 1923. Southern Diegueño Customs. In: Shackley MS, editor. The Early Ethnography of the Kumeyaay. Berkeley: Phoebe Hearst Museum of Anthropology.

Stojanowski CM. 2005. Spanish colonial effects on Native American mating structure and genetic variability in northern and central Florida: Evidence from Apalachee and western Timucua. American Journal of Physical Anthropology 128(2):273-286.

Teggart FJ, editor. 1909. The official account of the Portola expedition of 1769-1770. Berkeley: University of California.

The Huntington Library, Early California Population Project Database, 2006.

Torner N, Martinez A, Costa J, Mosquera M, Barrabeig I, Rovira A, Rius C, Cayla J, Plasencia E, Parron I et al. . 2007. Measles outbreak in the Barcelona region of Catalonia, Spain, October 2006 to February 2007. Eurosurveillance 12(8):pii=3144.

Twitchell E. 1925. The California Pandemic of 1833. California and Western Medicine 23(5):592-593.

Valle RK. 1973. Prevention of Smallpox in Alta California During the Franciscan Mission Period (1769-1833). California Medicine 119:73-77.

Walker P, and Johnson JR. 1994. The Decline of the Chumash Indian Population. In: Larsen CS, and Milner GR, editors. In the Wake of Contact: Biological Responses to Conquest. New York: Wiley-Liss. p 109-120.

Walker P, Lambert P, and DeNiro MJ. 1989. The Effects of European Contact on the Health of California Indians. In: Thomas DH, editor. Colombian Consequences Vol I: Archaeological and Historical Perspectives on the Spanish Borderlands West. Washington, D.C.: Smithsonian Institution Press. p 349-364.

Walker PL. 1989. Cranial injuries as evidence of violence in prehistoric southern California. Am J Phys Anthropol 80(3):313-323.

Walker PL, and Lambert P. 1989. Skeletal Evidence for stress during a period of cultural change in prehistoric California. In: Capasso L, editor. Advances in Paleopathology. Chieti, Italy: Marino Solfanelli. p 207-212.

Warren CN. 2004. The Desert Region. In: Moratto MJ, editor. California Archaeology. Salinas: Coyote Press. p 339-430.

Waterman TT. 1910. The religious practices of the Diegueño Indians. Berkeley: The University Press.

Weber DJ, editor. 1979. New Spain's far northern frontier. Albuquerque: University of New Mexico Press.

Weber DJ. 1992. The Spanish frontier in North America. New Haven, CT: Yale University Press.

Weber F. 1979. California Catholicity. Los Angeles: Libra Press Ltd.

Weber FJ. 1991. Prominent Visitors to the California Missions, 1786-1842: Dawson's Book Shop.

Wickson EJ. 1888. California Mission Fuits. Overland monthly and Out West magazine 11(65):501-505.

Wilken MA. 2012. An Ethnobotany of Baja California's Kumeyaay Indians. San Diego, CA: San Diego State University.

Wohlgemuth E. 2005. The Medieval Climatic Anomaly in Central Sierran Foothill Prehistory. Proceedings of the Society for California Archaeology 18:307-311.

Wolfe RJ. 1982. Alaska's Great Sickness, 1900: An Epidemic of Measles and Influenza in a Virgin Soil Population. Proceedings of the American Philosophical Society 126(2):91-121.

Wrigley EA. 1966. Family Reconstitution. In: Wrigley EA, editor. An Introduction to English Historical Demography: From the Sixteenth to the Nineteenth Century. London: Weidenfeld and Nicolson. p 96-159.

APPENDIX A – MODEL CODE

This appendix contains the complete code for the San Diego model. It is written in Java and incorporates libraries and methods from RePast Simphony 1.2, a free collection of libraries available for use in the development of agent-based models. To implement this application, the user would need to run the model in a version of Eclipse with RePast preloaded. Alternatively, the user could obtain a version of the model that has a distributable installer file. This would make the model available for use on any desktop computer that contains a current version of Java Runtime Environment.

The full code is split into six files. All are necessary to run the model. Additionally, the model needs user-provided input files that describe the physical layout (number and locations of buildings, for example) and the population attributes. The ones used for this project are contained in Appendix B of this document.

File 1: Context Creator.java

```
package sandiego;
import repast.simphony.context.Context;
import repast.simphony.dataLoader.ContextBuilder;
import repast.simphony.engine.environment.RunEnvironment;
import repast.simphony.parameter.Parameters;
import sandiego.SanDiegoContextCreator;
import sandiego.missioncontext.*;
/**
 *SanDiegoContextCreator is the file that coordinates the other files that make up
 * the model.
 *
 * @author Carolyn Orbann, Lisa Sattenspiel
 * @version January 2012
 */
public class SanDiegoContextCreator implements ContextBuilder<Object> {
        private static Context<Object> mainContext; // used to refer to main context

        /**
         * Builds and returns a context. Building a context consists of filling it
```

```
	 * with agents, adding projects and so forth. When this is called for the
	 * master context, the system will pass in a created context based on
	 * information given in the model.score file. When called for subcontexts,
	 * each subcontext that was added when the master context was built will be
	 * passed in.
	 *
	 * @param context
	 * @return the built context.
	 */

public Context<Object> build(Context<Object> context) {
			SanDiegoContextCreator.mainContext = context;

SanDiegoMissionContext sandiegoMissionContext = new SanDiegoMissionContext();
// instantiates a MissionContext

context.getSubContext(sandiegoMissionContext);/*Adds the new MissionContext to the main context as
a SubContext*/

System.out.println("Getting missionContext");

// Tell the scheduler when to end each run.
Parameters p = RunEnvironment.getInstance().getParameters();
double endAt = (Double) p.getValue("runlength");
RunEnvironment.getInstance().endAt(endAt);

			return context;
	}

public static SanDiegoMissionContext getSanDiegoMissionContext() {return (SanDiegoMissionContext)
mainContext.findContext("sandiegoMissionContext");

	}
}
```

File 2: Mission Plan Style

```
package sandiego.missioncontext;
import java.awt.Color;
import java.awt.Paint;

import repast.simphony.valueLayer.ValueLayer;
import repast.simphony.visualization.visualization2D.style.ValueLayerStyle;
/**
 * @author orbannc
 */
public class MissionPlanStyle implements ValueLayerStyle  {

	protected ValueLayer layer;
	Color darkBlue = new Color(63, 96, 204);
	Color aqua = new Color(19, 223, 236);
```

```
		Color brightBlue = new Color(98, 158, 227);
		Color purple = new Color(154, 101, 213);
		Color mediumGray = new Color(200, 200, 170);
		Color brightGreen = new Color (40, 200, 40);
		Color brightYellow = new Color (250, 250, 90);
		Color lightGray = new Color(248, 248, 255);
public void addValueLayer(ValueLayer layer) {
this.layer = layer;
}
public int getRed(double... coordinates) {return 0;}
public int getGreen(double... coordinates) {return 0;}
public int getBlue(double... coordinates) {return 0;}
public float getCellSize() {return 15.0f;}

/**
 * Return the color based on the value at given coordinates. background = 0; house = 1; monjeria = 2;
priest quarters = 3; kitchen = 4; church = 5; fields = 6; storage room = 7
*/
public Paint getPaint(double... coordinates) {double v = layer.get(coordinates);

/*
 * This switch statement uses the building type value to set the appropriate color for a building that then
appears on the grid display.
 */
switch ((int)v)	{
		case 0: return lightGray;
		case 1: return mediumGray;
		case 2: return darkBlue;
		case 3: return aqua;
		case 4: return brightBlue;
		case 5: return purple;
		case 6: return brightGreen;
		case 7: return brightYellow;
		default: return Color.red;
		}//Close switch
}
}
```

File 3: SanDiegoMissionContext.java

```
package sandiego.missioncontext;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;

import bsh.This;

import repast.simphony.context.Context;
```

```
import repast.simphony.context.space.continuous.ContinuousSpaceFactoryFinder;
import repast.simphony.context.space.grid.GridFactoryFinder;
import repast.simphony.dataLoader.ContextBuilder;
import repast.simphony.engine.watcher.query.ParseException;
import repast.simphony.query.InstanceOfQuery;
import repast.simphony.random.RandomHelper;
import repast.simphony.space.continuous.RandomCartesianAdder;
import repast.simphony.space.grid.Grid;
import repast.simphony.space.grid.GridBuilderParameters;
import repast.simphony.space.grid.RandomGridAdder;
import repast.simphony.util.ContextUtils;
import repast.simphony.valueLayer.GridValueLayer;

/**
 * SanDiegoMissionContext creates buildings and agents and puts them on the
 *map, which is also created in this file. In addition, it includes code to read in files
 * containing initial agent characteristics (SDagents##.txt) and building
 * characteristics (SDbldgs##.txt) and to put the agents into their assigned houses
 * at the beginning of the simulation.
 *
 * @author Carolyn Orbann, Lisa Sattenspiel
 * @version January 2012
 */

public class SanDiegoMissionContext implements ContextBuilder<Object> {
	private int xcoord;
	private int ycoord;
	public int fPat;//refers to an agent's father's and own patriline
	public int diseaseStatus;/**0 = susceptible, 1 = exposed, 2 = infectious,
		3 = recovered, 4 = dead*/
	public int agentHouse;// used to identify individual dwellings
	public int mPat; // agent's mother's patriline
	public int extFam;// used to indicate membership in a particular extended family
	public String agentSex;
	public double agentAge;
	public int spouseID;
	public double healthHistory;// an index to represent the impact of various health
		factors
	public int agentWork;
	public int presentLoc = -1;// used to indicate where an agent is physically located

/*
	 * The following two variables are used for agent x coordinate and y
	 * coordinate; added to agent constructor but initialized here instead of in
	 * the input file
	 */
	public int xLocInit = 0;
	public int yLocInit = 0;

	public int mblistLoc = 0;/*This variable refers to the position of a building and its
		parts on the Master Building List*/
	public int timeToInfectious = -1; //0 is a meaningful value, so this is initialized at
		-1
```

```
public static int numBuildings = 0;
public static int nBldgs;
public static int totPop = 0;
public static int dimX = 150;
public static int dimY = 150;
public int groupSize = 1;
public static int bldgTypeList [][];//Keeps track of how many building of each type
        are in the model
public static int numResidents [][]; //This is the town context version of
        atLocationList. It changes whenever someone moves.
public static int permanentResidents [][];//This is a residents list that keeps track
        of agents permanently assigned to buildings
public static ArrayList<ArrayList<SDSimpleBuilding>> masterBuildings;
public static int houseList [][];
public static int monjeriaList [][];
public static int priestqList [][];
public static int houseLineageList [][];

public int houseIndex = 0;
public int houseLineageIndex = 0;
public int monjeriaIndex = 0;
public int priestqIndex = 0;

public static int numHouse = 0;
public static int numMonjeria= 0;
public static int numPriestQ = 0;
public static int numKitchen = 0;
public static int numChurch = 0;
public static int numFields = 0;
public static int numStorage= 0;

public static boolean firstCaseChosen = false;
public boolean needsNewDwelling = false;


@SuppressWarnings("unchecked")
public Context<Object> build(Context<Object> missionContext) {

System.out.println("Building missionContext");

setNumHouse(0);
setNumChurch(0);
setNumMonjeria(0);
setNumPriestQ(0);
setNumKitchen(0);
setNumFields(0);
setNumStorage(0);

/* Create a new 2D grid to model the San Diego population. The inputs
 * to the GridFactory include the grid name, the context in which to
 * place the grid, and the grid parameters. Grid parameters include the
 * border specification, random adder for populating the grid with
 * agents, boolean for multiple occupancy, and the dimensions of the
```

```
         * grid.
         */

        Grid grid = GridFactoryFinder.createGridFactory(null).createGrid("MissionGrid", missionContext,
        new GridBuilderParameters<Object>( new
        repast.simphony.space.grid.WrapAroundBorders(),new RandomGridAdder<Object>(), true,
        dimX, dimY));

        /*
         * Create a new 2D continuous space to model the physical space on which
         * the buildings sit. The inputs to the Space Factory include the space
         * name, the context in which to place the space, border specification,
         * random adder for populating the grid with agents, and the dimensions
         * of the grid.
         */


        ContinuousSpaceFactoryFinder.createContinuousSpaceFactory(null).createContinuousSpace("Mi
ssionSpace",missionContext,new RandomCartesianAdder<Object>(),new
repast.simphony.space.continuous.WrapAroundBorders(), dimX, dimY, 1);

        /*
         * The Grid Value Layer is for visualizations of the buildings. The
         * definitions can be found in MissionPlanStyle.java
         */
        GridValueLayer vl = new GridValueLayer("MissionPlan", 0.0, true, dimX,dimY);

        missionContext.addValueLayer(vl);

        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildings = new
ArrayList<ArrayList<SDSimpleBuilding>>();

        /*
         * The following lines open up a new reader and the building definitions
         * file we want to read. The code then specifies that the program go
         * through each line in the file, break it up into its component parts
         * (i.e. type, id, etc). All of the processes done to create each
         * building are nested within this for loop. The file path in the
         * openFile statement must be complete and specific to the machine on
         * which the model is running.
         */

        BufferedReader input1 = null;
        input1 = openFile("C:/Users/orbannc/workspace/SanDiego/building_defs/SDbldgs.txt");

        if (input1 == null) {
                System.out.println("BUILDING FAIL!");
                System.exit(0);
                }

        for (String line; (line = readLine(input1)) != null;)
        // Loops through the lines until there are no more
                {
```

```
String s[] = line.split(",");// Splits the line according to the comma delimiter
int bldgID = stoi(s[0]);// This takes an integer as a string and changes
                        // it into integer represented as integer
int type = stoi(s[1]);
int width = stoi(s[2]);
int length = stoi(s[3]);
int initialX = stoi(s[4]);
int initialY = stoi(s[5]);

mblistLoc = numBuildings;
/*This sets mblistLoc to be equal to the current value of numBuildings, which corresponds to the
position of the building being read in
on the master building list*/

/*
 * The next group of statements use building attributes from the parameter file
*to place building objects on SanDiegoMissionContext, on MissionGrid, and on
*the ValueLayer according to their defined characteristics. Each grid
* node over the area encompassed by a building is a "partBuilding"
* object that has the specified building type and id of the entire
* building, but is a single node with a 1x1 dimension.
 */

SDSimpleBuilding building = new SDSimpleBuilding(grid, vl, bldgID, type,
        width, length, initialX, initialY, mblistLoc);

numBuildings ++;

switch (building.type){
        case 1:
                numHouse++;
                break;
        case 2:
                numMonjeria++;
                break;
        case 3:
                numPriestQ++;
                break;
        case 4:
                numKitchen++;
                break;
        case 5:
                numChurch++;
                break;
        case 6:
                numFields++;
                break;
        case 7:
                numStorage++;
                break;
        default:
```

```
                        System.out.println("Unknown building type in building counter. Incrementing
inaccurate.");
                        break;
                }

        ArrayList<SDSimpleBuilding> buildingCells = new ArrayList<SDSimpleBuilding>();
        // create the buildingList, include a unique name for each list tied to building ID

        for (int i = building.initialX; i < building.initialX+ building.width; i++) {
                for (int j = building.initialY; j < building.initialY+ building.length; j++) {
                        SDSimpleBuilding partBuilding = new SDSimpleBuilding(grid, vl,
                                                                bldgID, type, 1, 1, i, j, mblistLoc);
                        missionContext.add(partBuilding);
                        grid.moveTo(partBuilding, i, j);
                        vl.set(building.type, i, j);

                        buildingCells.add(partBuilding);
                                }
                        }

                masterBuildings.add(buildingCells);

                }

        int houseList [][] = new int [numHouse][2];
        int houseLineageList [][] = new int [numHouse][34];
                /*number of columns is equal to the number of possible patrilines plus the number of
                possible matrilines plus 2 (mblLoc and Building ID) for a particular house.*/
        int monjeriaList [][]= new int [numMonjeria][2];
        int priestqList [][]= new int [numPriestQ][2];

        //The following loops initializes the buildingID Lists
        for (int i=0; i<numHouse; i++){
                for (int i2=0; i2<2; i2++){
                houseList[i][i2]=0;
                        }}
        for (int i=0; i<numHouse; i++){
                for (int iLL=0; iLL<2; iLL++){
                houseLineageList[i][iLL]=0;
                        }}
        for (int j=0; j<numMonjeria; j++){
                for (int j2=0; j2<2; j2++){
                monjeriaList[j][j2]=0;
                        }}
        for (int l=0; l<numPriestQ; l++){
                for (int l2=0; l2<2; l2++){
                priestqList[l][l2]=0;
                        }}

        /*The following loop initializes and populates the numResidents, *permanentResidents and
        bldgTypeList arrays. The rows of the numResidents *and permanentResidents arrays are the
        building positions on the *masterBuildingsList.  The first column is the  building ID, and the
        second column *is the number of residents assigned to that building. The SDSimplePerson
```

```
*version of numResidents is called atLocationList, and the contents change *during the course of
the simulation to keep track of the number of agents *present at the building at a given time
step. The contents of numResidents *change whenever atLocationList changes.  The contents of
permanentResidents *(which is initially  identical to numResidents) remain constant unless we
need *to change them (e.g. death, marriage, migration, etc.) so we can keep track of *how many
agents are permanently assigned to the dwelling regardless of where *they currently are during a
simulation. The bldgTypeList array links the building *ID and the type to its position on the
master building list. The first column is the *Building ID and the second column is the building
type.  There is no *SDSimplePerson version of this array.  * */

numResidents = new int [numBuildings][2];
permanentResidents = new int [numBuildings][2];
bldgTypeList = new int [numBuildings][2];
for (int j = 0; j<numBuildings; j++){
        ArrayList<SDSimpleBuilding> structure = new
                ArrayList<SDSimpleBuilding>();
        structure.addAll(masterBuildings.get(j));
        SDSimpleBuilding structureInfo = structure.get(0);
        numResidents [j][0] = structureInfo.bldgID;
        numResidents [j][1] = 0;
        permanentResidents [j][0] = structureInfo.bldgID;
        permanentResidents [j][1] = 0;
        bldgTypeList [j][0] = structureInfo.bldgID;
        bldgTypeList [j][1] = structureInfo.type;
        makeBuildingIDLists (structureInfo, houseList, houseLineageList,
                monjeriaList, priestqList, houseIndex, monjeriaIndex,
                priestqIndex);
        }

nBldgs = numBuildings;

/*These statements guarantee that the following variables are reset
 * during the initialization of a new run*/

setHouseIndex(0);
setMonjeriaIndex(0);
setPriestQIndex(0);
setTotPop(0);
setNumBuildings(0);
setFirstCaseChosen(false);

/*These statements converts the masterBuildings ArrayList and the individual
 * building lists to Arrays found in SDSimplePerson. This must happen in
 * order for all movement methods found in SDSimplePerson to have access
 * to the building information needed for proper functioning.
 */
SDSimplePerson.a = masterBuildings.toArray();
SDSimplePerson.hList = houseList;
SDSimplePerson.hLineList = houseLineageList;
SDSimplePerson.mList = monjeriaList;
SDSimplePerson.pList = priestqList;
SDSimplePerson.atLocationList = numResidents;
```

```
/*
 * The following lines open up another new reader and the agent
 * definitions file we want to read. The code then specifies that the
 * program go through each line in the file, break it up into its
 * component parts (i.e. id, extended family, etc). All of the processes done
 * to create each agent are nested within this for loop. The file path
 * in the openFile statement must be complete and specific to the
 * machine on which the model is running.
 */

BufferedReader input2 = null;
input2 =
        openFile("C:/Users/orbannc/workspace/SanDiego/agent_defs/SDfullpop.
        txt");
if (input2 == null) { System.out.println("AGENT FAIL!");
                        System.exit(0);
        }

for (String line; (line = readLine(input2)) != null;)
// Loops through the lines until there are no more
{String s[] = line.split(",");// Splits the line according to the comma delimiter
 int agentID = stoi(s[0]);//stoi takes the string that is read in and changes it into
        integer
 int fPat = stoi(s[1]);// since this variable is a string, it does not need to be
        changed
int diseaseStatus = stoi(s[2]);
int agentHouse = stoi(s[3]);
int mPat = stoi(s[4]);
int extFam = stoi(s[5]);
String agentSex = s[6];
double agentAge = stod(s[7]); // stod takes the string that is read in and changes
        it into a double
int spouseID = stoi(s[8]);
double healthHistory = stod(s[9]);
int agentWork = stoi(s[10]);

/*
 * Populate the missionContext with the agents, who are given the
 * characteristics read in from the input file. All agents are given a disease status
 *of susceptible; the first case is picked using code in the SDSimplePerson file,
 *and that agent is given the status of exposed. The code below also finds initial
 * home coordinates for each agent and moves them to that cell if it is empty or
 *finds another possible cell within the building's coordinate range (The agent
 *moves to the first available empty cell that is suitable). If no suitable
 *coordinates can be found in the agent's assigned house, a method is called
 *within the initializeHome method that reassigns the agent permanently to a
 *new dwelling.
 * In this version of the model, the reassignDwelling method may not work
 *properly without some modification because it lineage information is not
 *complete at the time each agent is initialized (except for the last agent). At this
 *time, we are setting up agent files so that no houses are full at initialization.
 */
```

```
SDSimplePerson person = new SDSimplePerson(agentID, fPat, diseaseStatus,
        agentHouse, mPat, extFam, agentSex,agentAge, spouseID, healthHistory,
        agentWork, presentLoc, xLocInit, yLocInit);

        totPop++;
        missionContext.add(person);
        initializeHome(agentHouse, person.xLocation, person.yLocation, person,
                masterBuildings, nBldgs);
                }
//This will print the House Lineage List Array.
/*for (int r = 0; r <numHouse; r++){
        System.out.println("Dwelling Row "+ r);
        for(int c = 0; c < houseLineageList[r].length; c++){
                System.out.print(houseLineageList[r][c]+" ");
                        }
                System.out.println();
                        }*/
//System.out.println("The total population size in this simulation is: "+totPop);
        System.out.println("returning missionContext");
        return missionContext;
        }

/*The Reader methods start here*/
/**
 * The openFile method, openKbd and readLine methods are used to deal with
 * the file reading
 */

static BufferedReader openFile(String filename)
// Creates files, returns null if there is an exception
{
        try {
                return new BufferedReader(new FileReader(filename));
                }

        catch (IOException e) {
        return null;

        }// close try/catch
        }// Close openFile

public static BufferedReader openKbd() {

        return new BufferedReader(new InputStreamReader(System.in));
        }

        static String readLine(BufferedReader input)// Reads in the line and returns null if there are no
more lines
        {
                try {
                        return input.readLine();
                }
```

```
                catch (IOException e) {
                        return null;
                }// Close try/catch
        }// Close readLine

/*
 * The stoi, stod, and stob methods are used to transform strings into
 * desired variable types
 */

static int stoi(String s)// Parses string values into integers, and catches exceptions
        {
                try {
                        return Integer.parseInt(s);
                }

                catch (NumberFormatException e) {
                        System.out.println("String format exception stoi");

                        return 0;

                }// Close try/catch
        }// Close stoi

        static double stod(String s)// Parses string values into doubles, and catches exceptions
        {
                try {
                        Double d = new Double(s);
                        return d;
                }

                catch (NumberFormatException e) {
                        System.out.println("String format exception stod at line");

                        return 0;
                }// Close try/catch
        }// Close stod

static Boolean stob(String s) throws ParseException// Parses string values into integers,
        and catches exceptions
        {
        return Boolean.parseBoolean(s);
        }// Close stob

/**Non-reader methods start here*/

/*The makeBuildingIDLists method makes arrays that contain the ID numbers of all buildings of a
particular type.
* For each of these arrays, the first column consists of the locations of each building on *the
masterBuilding list. The second column gives the ID number of that building. (The *ID number is read in as
one of the building's characteristics.)
*/
public void makeBuildingIDLists(SDSimpleBuilding building, int houseIDs [][],int
```

```
        houseLineageIDs [][], int monjeriaIDs [][], int priestqIDs [][], int housei, int monji,
        int priesti){

        switch (building.type){
        case 1:
                houseIDs[housei][0] = building.mblistLoc;
                houseIDs[housei][1]= building.bldgID;
                houseLineageIDs[housei][0] = building.mblistLoc;
                houseLineageIDs[housei][1]= building.bldgID;
                housei++;
                setHouseIndex(housei);
                break;
        case 2:
                monjeriaIDs[monji][0]= building.mblistLoc;
                monjeriaIDs [monji][1] = building.bldgID;
                monji++;
                setMonjeriaIndex (monji);
                break;
        case 3:
                priestqIDs[priesti][0] = building.mblistLoc;
                priestqIDs [priesti][1] = building.bldgID;
                priesti++;
                setPriestQIndex (priesti);
                break;
}
}//close makeBuildingIDLists

/*The initializeHome method moves an agent from the random location on which it was *orginially placed
to its home location as determined by the dwelling variable assigned *to the agent. If there is no space in
the assigned building, reassignHome is called in *order to pick a new (random) building of the same type
as the original assigned *building. This method also populates the numResidents array so that it contains
a *record of how many agents are assigned to each building.  In the present version of the *model, we are
guaranteeing that all agents can find space in their originally assigned *dwelling, so random reassignment
is not a problem. However, if that situation changes *the reassignDwelling method will need to be
changed to reflect proper kinship rules *with regards to structuring of households.
 */

@SuppressWarnings("unchecked")
public void initializeHome(int agentHouse, int xLocat, int yLocat, SDSimplePerson
        person, ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList,int
        numHouses) {

        Context context = ContextUtils.getContext(person);
        Grid patch = (Grid) context.getProjection("MissionGrid");
        int dwellingRow = -1;
        int dwellingType = 0;
        int accessList [][] = null;
        person.foundLocation = false;
        boolean fPatFound = false;
        boolean mPatFound = false;
        int houseLineList [][] = SDSimplePerson.hLineList;
        person.numCells = person.getPossibleCoords(agentHouse, masterBuildingsList,
                patch);
```

```
	for (int index = 0; index <nBldgs; index++) {
		if (bldgTypeList[index][0] == agentHouse) {
			dwellingType = bldgTypeList[index][1];
			break;
		}
	}

	switch (dwellingType){
	case 1:
		accessList = SDSimplePerson.hList;
		break;
	case 2:
		accessList = SDSimplePerson.mList;
		break;
	case 3:
		accessList = SDSimplePerson.pList;
		break;
	default:
		System.out.println("ERROR: THIS AGENT'S DESIGNATED BUILDING TYPE IS NOT A
RESIDENCE.");
	}

	for (int index = 0; index < nBldgs; index++) {
		if (accessList[index][1] == agentHouse) {
			dwellingRow = index;
			person.presentLocation = accessList[index][0];
			break;
		}
		}

	if (numResidents[(accessList[dwellingRow][0])][1] < person.numCells) {
		xLocat = person.possXLoc;
		yLocat = person.possYLoc;

		person.assignLocation(person.dwelling, xLocat, yLocat, person,
			masterBuildingsList, patch);
		numResidents [person.presentLocation][1]++;
		permanentResidents [person.presentLocation][1]++;

/*The following code loops through the columns of the houseLineageList for the calling *agent's dwelling
row. It first checks to see if an agent's paternal and maternal patrilines *have already been recorded.  If
they aren't, that information is added to the first *available position (ie. a 0). It also checks to make sure
that if the values are the same, *they are only added once.  The houseLineageList is used to identify
preferred dwellings *for visiting and any reassignment of agents along kin lines. This lineage list is only
kept *for houses, not for the monjeria or priest quarters, as this information is not needed *for
movement to those buildings.
 */

if (dwellingType == 1){
	for (int col = 2; col < houseLineList[dwellingRow].length; col ++){
	if (houseLineList [dwellingRow][col] == person.fathersPatriline){
		fPatFound = true;
			}
```

```
		if (houseLineList [dwellingRow][col] == person.mothersPatriline){
			mPatFound = true;
				}
			}
		if (fPatFound == false){
			for (int col2 = 2; col2 < houseLineList[dwellingRow].length; col2 ++){
		if (houseLineList[dwellingRow][col2]== 0){
			houseLineList[dwellingRow][col2] = person.fathersPatriline;
		break;
				}
			}
		}
		if (mPatFound == false && (person.fathersPatriline != person.mothersPatriline)){
			for (int col3 = 2; col3 < houseLineList[dwellingRow].length; col3 ++){
		if (houseLineList[dwellingRow][col3]==0){
			houseLineList[dwellingRow][col3] = person.mothersPatriline;
			break;
				}
			}
		}
		}
		}
		else{
			if (person.foundLocation == true) {
				//		System.out.println("Agent ID: "+person.agtID+" I am in a moveHome
loop that I" +" should have skipped.");
			} else {
				this.needsNewDwelling = true;
				person.dwelling = this.reassignDwelling(person,accessList);
			//System.out.println("I am agent: "+person.agtID+" and I have been reassigned to:"
			//+person.dwelling);
agentHouse = person.dwelling;
person.numCells = person.getPossibleCoords(person.dwelling, masterBuildingsList,
		patch);
xLocat = person.possXLoc;
yLocat = person.possYLoc;
person.assignLocation(person.dwelling, xLocat, yLocat, person, masterBuildingsList,
		patch);
numResidents [person.presentLocation][1]++;
permanentResidents [person.presentLocation][1]++;

/*In the following loop, the reassigned house is checked for kinship membership using *the same method
as in checking the original house*/

if (dwellingType == 1){
		for (int col = 2; col < houseLineList[dwellingRow].length; col ++){
		if (houseLineList [dwellingRow][col] == person.fathersPatriline){
			fPatFound = true;
				}
		if (houseLineList [dwellingRow][col] == person.mothersPatriline){
			mPatFound = true;
					}
				}
```

```
                if (fPatFound == false){
                for (int col2 = 2; col2 < houseLineList[dwellingRow].length; col2 ++){
                        if (houseLineList[dwellingRow][col2]==0){
                                houseLineList[dwellingRow][col2] =
                                        Person.fathersPatriline;
        break;
                                        }
                                }
                        }
        if (mPatFound == false){
                for (int col3 = 2; col3 < houseLineList[dwellingRow].length; col3 ++){
                        if (houseLineList[dwellingRow][col3]==0){
                        houseLineList[dwellingRow][col3] = person.mothersPatriline;
                break;
                                        }
                                }
                        }
                        }
                        }
        }
}
/* THIS METHOD HAS BEEN RETAINED FROM THE ST. ANTHONY MODEL, BUT WILL NEED *TO BE ADAPTED
IF IT IS TO BE USED FOR THE SAN DIEGO MODEL.
*
* The present method randomly picks and assigns a new dwelling building for an agent,
* but the ideal would be to pick a dwelling that is assigned to an appropriate kin.
*
* The problem is that if an agent needs to be reassigned and is made early in the
* initialization process, there may no kin on the map yet and so, will not have
* an appropriate place to go.  In the interest of time and resources, this issue
* is being circumvented by making sure that no houses are overpacked at the
* beginning of the model run.
*/

public int reassignDwelling(SDSimplePerson person, int dwellingList[][]) {
        int newDwelling = -1;
        int newDwellPos = RandomHelper.nextIntFromTo(1, numHouse) - 1;
        newDwelling = dwellingList[newDwellPos][1];

        while (numResidents[dwellingList[newDwellPos][0]][1] >= person.numCells) {
                newDwellPos = RandomHelper.nextIntFromTo(1, numHouse) - 1;
                newDwelling = dwellingList[newDwellPos][1];
        //}
        }
        return newDwelling;
}
        public int getInitialX() {
                return xcoord;
        }
        public void setInitialX(int xcoord) {
                this.xcoord = xcoord;
        }
        public int getInitialY() {
```

```
		return ycoord;
}
public void setInitialY(int ycoord) {
		this.ycoord = ycoord;
}
public int getHouseIndex() {
		return houseIndex;
}
public void setHouseIndex(int houseIndex) {
		this.houseIndex = houseIndex;
}
public int getMonjeriaIndex() {
		return monjeriaIndex;
}
public void setMonjeriaIndex(int monjeriaIndex) {
		this.monjeriaIndex = monjeriaIndex;
}
public int getPriestQIndex() {
		return priestqIndex;
}
public void setPriestQIndex(int priestqIndex) {
		this.priestqIndex = priestqIndex;
}
public static int getTotPop() {
		return totPop;
}
public void setTotPop(int totPop) {
		SanDiegoMissionContext.totPop = totPop;
}
public int getNumBuildings() {
		return numBuildings;
}
public void setNumBuildings(int numBuildings) {
		SanDiegoMissionContext.numBuildings = numBuildings;
}
public static boolean isFirstCaseChosen() {
		return firstCaseChosen;
}
public static void setFirstCaseChosen(boolean firstCaseChosen) {
		SanDiegoMissionContext.firstCaseChosen = firstCaseChosen;
}
public static int getNumHouse() {
		return numHouse;
}
public static int getNumMonjeria() {
		return numMonjeria;
}
public static int getNumKitchen() {
		return numKitchen;
}
public static int getNumPriestQ() {
		return numPriestQ;
}
```

```
	public static int getNumChurch() {
		return numChurch;
	}
	public static int getNumFields() {
		return numFields;
	}
	public static int getNumStorage() {
		return numStorage;
	}
	public static void setNumStorage(int numStorage) {
		SanDiegoMissionContext.numStorage = numStorage;
	}
	public static void setNumHouse(int numHouse) {
		SanDiegoMissionContext.numHouse = numHouse;
	}
	public static void setNumMonjeria(int numMonjeria) {
		SanDiegoMissionContext.numMonjeria = numMonjeria;
	}
	public static void setNumKitchen(int numKitchen) {
		SanDiegoMissionContext.numKitchen = numKitchen;
	}
	public static void setNumPriestQ(int numPriestQ) {
		SanDiegoMissionContext.numPriestQ = numPriestQ;
	}
	public static void setNumChurch(int numChurch) {
		SanDiegoMissionContext.numChurch = numChurch;
	}
	public static void setNumFields(int numFields) {
		SanDiegoMissionContext.numFields = numFields;
	}
}
```

File 4: SDGhostPerson

```
/**
 *
 */
package sandiego.missioncontext;

/**
 * @author orbannc
 *
 */
public class SDGhostPerson {
		//SDGhostPerson is a dead clone of a formerly-living SDSimplePerson agent.
		//These variables are defined as in SDSimplePerson
public int agtID;
public int fathersPatriline;
public int disStatus;
```

```
public int dwelling;
public int mothersPatriline;
public int extFamily;
public String sex;
public double age;
public int spouse;
public double hlthHist;
public int occupation;
public int presentLocation;
public int xLocation;
public int yLocation;
public int timeOfDeath = -1;
public int placeOfDeath = -1;
public int occType = 0;

public SDGhostPerson(int agentID, int fPat, int diseaseStatus,
        int agentHouse, int mPat, int extFam, String agentSex, double agentAge,
        int spouseID,double healthHistory, int agentWork, int presentLoc,
        int xLoc, int yLoc, int timeDeath, int placeDeath) {

                this.agtID = agentID;
                this.fathersPatriline = fPat;
                this.disStatus = diseaseStatus;
                this.dwelling = agentHouse;
                this.mothersPatriline = mPat;
                this.extFamily = extFam;
                this.sex = agentSex;
                this.age = agentAge;
                this.spouse = spouseID;
                this.hlthHist = healthHistory;
                this.occupation = agentWork;
                this.presentLocation = presentLoc;
                this.xLocation = xLoc;
                this.yLocation = yLoc;
                this.timeOfDeath = timeDeath;
                this.placeOfDeath = placeDeath;

                this.occType = ((int) (this.occupation / 100)) * 100;

        }
}
```

File 5: SDSimpleBuilding.java

```
package sandiego.missioncontext;

import repast.simphony.space.grid.Grid;
import repast.simphony.valueLayer.GridValueLayer;

/**
 * SimpleBuilding defines characteristics of our building agents.
```

```
 */

public class SDSimpleBuilding {
	public int bldgID = 0, type = 0, initialX = 0, initialY = 0, width = 0, length = 0,

	mblistLoc = 0;
	public Grid<SDSimpleBuilding> grid;
	public GridValueLayer layer;

public SDSimpleBuilding(Grid<SDSimpleBuilding> buildingGrid, GridValueLayer
	buildingLayer, int buildingID,	int buildingType, int buildingWidth, int
	buildingLength, int buildingInitialX, int buildingInitialY, int mblistLocation)	{

		this.grid = buildingGrid;
		this.layer = buildingLayer;
		this.bldgID = buildingID;
		this.type = buildingType;
		this.width = buildingWidth;
		this.length = buildingLength;
		this.initialX = buildingInitialX;
		this.initialY = buildingInitialY;
		this.mblistLoc = mblistLocation;

	}

/*These methods must be included in order for the visualization to work correctly.*/

public int gettype() {
	return type;
	}

public void setType(int type) {
	this.type = type;
	}
}
```

File 6: SDSimplePerson.java

```
package sandiego.missioncontext;

import java.util.ArrayList;
import java.util.Collections;

import repast.simphony.annotate.AgentAnnot;
import repast.simphony.context.Context;
import repast.simphony.engine.environment.RunEnvironment;
import repast.simphony.engine.schedule.ScheduledMethod;
import repast.simphony.parameter.Parameters;
import repast.simphony.query.InstanceOfQuery;
import repast.simphony.query.space.grid.VNQuery;
import repast.simphony.random.RandomHelper;
```

```
import repast.simphony.space.grid.Grid;
import repast.simphony.space.grid.GridPoint;
import repast.simphony.util.ContextUtils;

/**SDSimplePerson defines characteristics of our person agents and their
 * behaviors. Extended descriptions of each of the variables in the agent
 * constructor are given in the model_description_SanDiego.txt file included in the
 * SAdirectedmove project folder.
 *
 * @author Carolyn Orbann, Lisa Sattenspiel
 * @version January 2012
 */

@AgentAnnot(displayName = "Person")
public class SDSimplePerson {

	static int dimX = 150; // The x dimension of the physical space
	static int dimY = 150; // The y dimension of the physical space
	public int agtID;
	public int fathersPatriline;//refers to the patriline of an agent's father (also of the
		agent themselves)
	public int disStatus;
	/*
	 * 0 = susceptible, 1 = exposed, 2 = infectious, 3 = recovered, 4 = dead
	 */
	public int dwelling;// used to identify individual dwellings
	public int mothersPatriline; // used to indicate an agent's mother's patriline
	public int extFamily;// used to indicate membership in an extended family
	public String sex;
	public double age;
	public int spouse;
	public double hlthHist;//an index representing impact of various health factors
	public int occupation;
	public int presentLocation;
	public int originalDwelling;//used to keep track of families if residence changes
	public int xLocation;
	public int yLocation;
	public int occType = 0;
	static Object[] a; //SDSimplePerson version of masterBuilding list in array form.
	public ArrayList<ArrayList<SDSimpleBuilding>> spBuildings; //array a converted
		back to array list

	public static int hList[][];// SDSimplePerson version of the house list.
	public static int hLineList[][];// SDSimplePerson version of the house lineage list
	public static int mList[][];// SDSimplePerson version of the monjeria list.
	public static int pList[][];// SDSimplePerson version of the priest quarters list.
	public static int atLocationList[][]; // SDSimplePerson version of numResidents.
	public int kinHouseList [][];// a list of houses that contain kin

	public static int firstCase = 0;
	public static int firstCaseOccCode = 0;
	public int personID = 0;
	public int personOcc = 0;
```

```
	public int personDwelling = 0;
	public int timeInfected = -1;
	public int placeInfected = -1;
	public int infectorID = -1;
	public int infectorOcc = -1;
	public int infectorMatriline = -1;
	public int infectorPatriline = -1;
	public int infectorDwelling = -1;
	public int timeOfDeath = -1;
	public int placeOfDeath = -1;
	public int timeKeeper = 0;

	public static boolean occupancy = false;/* indicates whether a building cell
		location contains an agent*/
	public boolean foundLocation;
	public int numCells = 0;//tracks the total number of cells in a building
	public boolean canVisit = true;

	public boolean childrenAtHome;
	public boolean childrenUnderFour;
	boolean motherFound;
	boolean otherFemaleFound; //older female in the household, usually a
		grandmother, possible caretaker
	boolean maleCaretakerFound;// any adult male over 12 in the household that
		can be a caretaker if needed
	public boolean newlyInfected = false;
	public boolean newlyDead = false;
	public boolean stepCompleted = false;
	public boolean newDwellingFound = false;

	/*The following two variables are used as x and y locations prior to movement.
	 * They refer to "possible" x and y locations.
	 */
	public int possXLoc = 0;
	public int possYLoc = 0;

	//The following 4 variables are building ids used for movement
	public int tempDwelling = 0;
	public int churchID = 175;
	public int fieldsID = 176;
	public int kitchenID = 172;


/* The environment parameters contain the user-editable values that appear
 * in the GUI. Get the parameters (designated p) and then specifically get
 * the ones indicated in the parentheses.
 */
Parameters p = RunEnvironment.getInstance().getParameters();
	public int timeToInfectious = (Integer) p.getValue("lengthoflatentperiod");
	public int timeToRecovery = (Integer) p.getValue("lengthofinfectiousperiod");
	public double deathProb = (Double) p.getValue("probabilityofdeath");
	int numHouseSP = SanDiegoMissionContext.numHouse;
```

```
public SDSimplePerson(int agentID, int fPat, int diseaseStatus, int agentHouse, int mPat,
        int extFam, String agentSex, double agentAge, int spouseID,double
        healthHistory,      int agentWork, int presentLoc, int xLoc, int yLoc) {

        this.agtID = agentID;
        this.fathersPatriline = fPat;
        this.disStatus = diseaseStatus;
        this.dwelling = agentHouse;
        this.mothersPatriline = mPat;
        this.extFamily = extFam;
        this.sex = agentSex;
        this.age = agentAge;
        this.spouse = spouseID;
        this.hlthHist = healthHistory;
        this.occupation = agentWork;
        this.presentLocation = presentLoc;
        this.xLocation = xLoc;
        this.yLocation = yLoc;
        this.originalDwelling = agentHouse;
        this.occType = ((int) (this.occupation / 100)) * 100;
        /* occType is the first digit of the occupation code multiplied by 100. This will be
         * used in the methods to find a particular tick's activities for an individual agent
         * with that occupation code, e.g. 100 is used to find out what an adult male will
         * do on a particular tick in the simulation.
          */
                }

/* This initSim method picks the initial case(s) randomly. It is only run once, and the   *initial number of
cases is determined by the value of the "pick" variable. Newly *infected agents are put into the exposed
class, which they remain in according to the *user-set value for the length of the latent period.
 */

@ScheduledMethod(start = 6, interval = 0, pick = 1)
public int initSim() {
        System.out.println("Agent " + this.agtID + " is the first case and their occupation
                is: "+this.occupation);
        this.timeToInfectious = (Integer) p.getValue("lengthoflatentperiod");
        this.disStatus = 2;
        this.timeInfected = (int) RunEnvironment.getInstance().getCurrentSchedule().getTickCount();
        this.placeInfected = SDSimplePerson.atLocationList[this.presentLocation][0];
        this.infectorID = 0;
        this.infectorOcc = 0;
        this.infectorDwelling = 0;
        firstCase = this.agtID;
        firstCaseOccCode = this.occupation;
        return firstCase;
                }
        /* The method below is an alternate initSim method that is designed to pick an
         * agent of a specific occType to begin the model.  The first "if" statement
         * can be easily modified to pick an agent of any occupation to start the model
         */
/*@ScheduledMethod(start = 0, interval = 0)
public int initSim() {
```

```
		if (this.occType == 100 && SanDiegoMissionContext.firstCaseChosen==false){
		System.out.println("Agent " + this.agtID + " is the first case. Their occupation is:
			"+this.occupation);
		this.timeToInfectious = (Integer) p.getValue("lengthoflatentperiod");
		this.disStatus = 1;
		this.timeInfected = (int) RunEnvironment.getInstance().getCurrentSchedule().getTickCount();
		this.placeInfected = SDSimplePerson.atLocationList[this.presentLocation][0];
		this.infectorID = 0;
		this.infectorOcc = 0;
		this.infectorDwelling = 0;
		firstCase = this.agtID;
		firstCaseOccCode = this.occupation;
		SanDiegoMissionContext.firstCaseChosen=true;
				}
			return firstCase;
			}*/

/* Schedule the step method for agents. The method is scheduled starting at tick one
 * with an interval of 1 tick. Specifically, the step starts at 1, and recurs at 2,3,4,...etc
 */

@SuppressWarnings("unchecked")
@ScheduledMethod(start = 1, interval = 1)
public void step() {
		//System.out.println("Agent "+this.agtID+" has begun the step method");
		ArrayList<ArrayList<SDSimpleBuilding>> spBuildings = new
		ArrayList<ArrayList<SDSimpleBuilding>>();
		for (int i = 0; i < a.length; i++) {
			spBuildings.add((ArrayList<SDSimpleBuilding>) a[i]);
				}
		if (this.timeInfected != (int)RunEnvironment.getInstance().getCurrentSchedule().getTickCount()){
				this.newlyInfected = false;}
		this.newlyDead = false;
		updateDiseaseStatus(spBuildings);

		/*
		* Dead agents (disStatus = 4) do not go through the rest of the step method *because they do
		not participate in any social activities or disease transmission. *We do need to keep them in the
		simulation to facilitate counting of the number *who have died.
		 */
		if (this.disStatus !=4){

			personID = this.agtID;
			personOcc = this.occupation;
			personDwelling = this.dwelling;

/* The program is being set up for twenty-four time ticks per day. Consequently, each
* time tick is 1 hour long. For a base run, we have set the length of the latent period to
* twenty-four ticks (one day) and our infectious period to 120 ticks (5 days). The run
* length is set to 2400 ticks (100 days).
 */

/* The series of if-else statements sets up a time schedule within which agent activities
```

```
 * will occur. A day is divided into 24 time slots. Values of timeKeeper between 1 and 24
 * correspond to Mondays through Saturdays (in this order) 12am through 11pm each
 * day. Values between 25 and 48 correspond to each hour on Sundays.
 */

if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 1) % 24) == 0)
        && (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 145)
        % 168) != 0)) {
                timeKeeper = 1;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 2) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        146) % 168) != 0)) {
                timeKeeper = 2;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 3) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        147) % 168) != 0)) {
                timeKeeper = 3;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 4) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        148) % 168) != 0)) {
                timeKeeper = 4;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 5) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        149) % 168) != 0)) {
                timeKeeper = 5;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 6) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        150) % 168) != 0)) {
                timeKeeper = 6;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 7) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        151) % 168) != 0)) {
                timeKeeper = 7;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 8) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        152) % 168) != 0)) {
                timeKeeper = 8;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 9) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        153) % 168) != 0)) {
                timeKeeper = 9;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 10) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        154) % 168) != 0)) {
```

```
                timeKeeper = 10;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 11) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        155) % 168) != 0)) {
                timeKeeper = 11;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 12) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        156) % 168) != 0)) {
                timeKeeper = 12;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 13) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        157) % 168) != 0)) {
                timeKeeper = 13;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 14) % 24)        == 0)&&
(((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        158) % 168) != 0)) {
                timeKeeper = 14;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 15) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        159) % 168) != 0)) {
                timeKeeper = 15;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 16) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        160) % 168) != 0)) {
                timeKeeper = 16;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 17) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        161) % 168) != 0)) {
                timeKeeper = 17;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 18) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        162) % 168) != 0)) {
                timeKeeper = 18;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 19) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        163) % 168) != 0)) {
                timeKeeper = 19;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 20) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        164) % 168) != 0)) {
                timeKeeper = 20;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 21) % 24)
```

```
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        165) % 168) != 0)) {
                timeKeeper = 21;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 22) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        166) % 168) != 0)) {
                timeKeeper = 22;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 23) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        167) % 168) != 0)) {
                timeKeeper = 23;
                        }
else if ((((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 24) % 24)
        == 0)&& (((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() –
        168) % 168) != 0)) {
                timeKeeper = 24;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 145) %
        168 == 0) {
                timeKeeper = 25;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 146) %
        168 == 0) {
                timeKeeper = 26;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 147) %
        168 == 0) {
                timeKeeper = 27;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 148) %
        168 == 0) {
                timeKeeper = 28;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 149) %
        168 == 0) {
                timeKeeper = 29;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 150) %
        168 == 0) {
                timeKeeper = 30;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 151) %
        168 == 0) {
                timeKeeper = 31;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 152) %
        168 == 0) {
                timeKeeper = 32;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 153) %
        168 == 0) {
```

```
                timeKeeper = 33;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 154) %
        168 == 0) {
                timeKeeper = 34;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 155) %
        168 == 0) {
                timeKeeper = 35;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 156) %
        168 == 0) {
                timeKeeper = 36;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 157) %
        168 == 0) {
                timeKeeper = 37;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 158) %
        168 == 0) {
                timeKeeper = 38;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 159) %
        168 == 0) {
                timeKeeper = 39;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 160) %
        168 == 0) {
                timeKeeper = 40;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 161) %
        168 == 0) {
                timeKeeper = 41;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 162) %
        168 == 0) {
                timeKeeper = 42;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 163) %
        168 == 0) {
                timeKeeper = 43;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 164) %
        168 == 0) {
                timeKeeper = 44;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 165) %
        168 == 0) {
                timeKeeper = 45;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 166) %
        168 == 0) {
                timeKeeper = 46;
```

```
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 167) %
        168 == 0) {
                timeKeeper = 47;
                        }
else if ((RunEnvironment.getInstance().getCurrentSchedule().getTickCount() - 168) %
        168 == 0) {
                timeKeeper = 48;
                        }

findDaysActivities(this.dwelling, this.xLocation, this.yLocation, this, spBuildings,
        numHouseSP, hList, this.mothersPatriline, this.extFamily);

/* The following lines govern the disease transmission process following
 * the movement of agents.
 */

if (this.disStatus == 2) {
        checkSuscNeighbors();
        } else if (this.disStatus == 0) {
                checkInfNeighbors();
                        }

foundLocation = false;
        }
        this.stepCompleted = true;
                }

/* The following methods contain the code to reset specific variables at the end of a
 * given step. The endStepReset() method is called every step for every agent. Both are
 * called at last priority (-1d/0d is the program - or Java-defined constant value needed
 * to set the priority as "last" but we still do not know what exactly it means). endStep()
 * only needs to be called for one agent for those global variables that only need to be
 * reset once per time unit; endStepReset() is called by each agent at each step to reset
 * stepCompleted and any future variables that require resetting for each agent at each
 * time step.
 */
@ScheduledMethod(start = 1, interval = 1,priority = -1d / 0d)
public void endStepReset(){
        stepCompleted = false;
                }

@ScheduledMethod(start = 1, interval = 1, pick = 1, priority = -1d / 0d)
public void endStep() {
//System.out.println("This is the end of step: "+ (int)
RunEnvironment.getInstance().getCurrentSchedule().getTickCount());
                }

/* The following method sends an agent to the proper activities for the specific time tick
 * indicated by the timeKeeper variable. Although there are 24 ticks (cases in the switch)
 * per day, we do not need to include all of them for each day, since no activity is
 * occurring during certain time periods. The simulation run starts on Monday at 12am.
 */
```

```
public void findDaysActivities(int agentHouse, int xLocFDA, int yLocFDA, SDSimplePerson
        person, ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int
        numHouses, int houseList [][], int mPat, int extFam) {

        switch (timeKeeper) {
                /*cases 1-6 refer to midnight through 5am, people are sleeping in houses
                and thus not moving or calling methods.*/
        case 7:
                do6amActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
                numHouses);
                break;
        case 8:
                do7amActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
                numHouses);
                break;
        case 9:
                do8amActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
                numHouses);
                break;
        case 10:
                doMSat9amActs(agentHouse, xLocFDA, yLocFDA, person,
                masterBuildingsList, numHouses);
                break;
        case 11:
                doMSat10amActs(agentHouse, xLocFDA, yLocFDA, person,
                masterBuildingsList, numHouses);
                break;
        case 12:
                doMSat11amActs(agentHouse, xLocFDA, yLocFDA, person,
                masterBuildingsList, numHouses);
                break;
        case 13:
                doNoonActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
                numHouses);
                break;
        case 14:
                doMonSat1pmActs(agentHouse, xLocFDA, yLocFDA, person,
                masterBuildingsList, numHouses);
                break;
        case 15:
                doMonSat2pmActs(agentHouse, xLocFDA, yLocFDA, person,
                masterBuildingsList, numHouses);
                break;
        case 16:
                doMonSat3pmActs(agentHouse, xLocFDA, yLocFDA, person,
                masterBuildingsList, numHouses);
                break;
        case 17:
                doMonSat4pmActs(agentHouse, xLocFDA, yLocFDA, person,
                masterBuildingsList, numHouses);
                break;
        case 18:
```

```
            do5pmActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
            numHouses);
            break;
case 19:
            do6pmActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
            numHouses, houseList, mPat, extFam);
            break;
case 20:
            do7pmActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
            numHouses, houseList, mPat, extFam);
            break;
case 21:
            do8pmActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
            numHouses, houseList, mPat, extFam);
            break;
case 22:
            do9pmActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
            numHouses);
            break;
//Cases 23 through 30 refer to times between 10pm and 5am in which nothing is happening
case 31:
            do6amActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
            numHouses);
            break;
case 32:
            do7amActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
            numHouses);
            break;
case 33:
            do8amActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
            numHouses);
            break;
case 34:
            doSun9amActs(agentHouse, xLocFDA, yLocFDA,
            person,masterBuildingsList, numHouses);
            break;
case 35:
            doSun10amActs(agentHouse, xLocFDA, yLocFDA,
            person,masterBuildingsList, numHouses);
            break;
case 36:
            doSun11amActs(agentHouse, xLocFDA, yLocFDA, person,
            masterBuildingsList, numHouses);
            break;
case 37:
            doNoonActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
            numHouses);
            break;
case 38:
            doSun1pmActs(agentHouse, xLocFDA, yLocFDA, person,
            masterBuildingsList, numHouses);
            break;
case 39:
```

```
                doSun2pmActs(agentHouse, xLocFDA, yLocFDA, person,
                masterBuildingsList, numHouses, houseList, mPat, extFam);
                break;
        case 40:
                doSun3pmActs(agentHouse, xLocFDA, yLocFDA, person,
                masterBuildingsList, numHouses);
                break;
        case 41:
                doSun4pmActs(agentHouse, xLocFDA, yLocFDA, person,
                masterBuildingsList, numHouses);
                break;
        case 42:
                do5pmActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
                numHouses);
                break;
        case 43:
                do6pmActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
                numHouses, houseList, mPat, extFam);
                break;
        case 44:
                do7pmActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
                numHouses, houseList, mPat, extFam);
                break;
        case 45:
                do8pmActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
                numHouses, houseList, mPat, extFam);
                break;
        case 46:
                do9pmActs(agentHouse, xLocFDA, yLocFDA, person, masterBuildingsList,
                numHouses);
                break;
        //Cases 47 and 48 refer to evening hours of 10pm and 11pm on Sunday in which
        nothing is happening.
                        }
                }

/** The Daily Activity Methods start here */
/*
 * In the following methods, the atLocationList array is decremented at the
 * building that holds the agent calling the step() method. The agent then
 * moves wherever appropriate, and then the atLocationList array is
 * incremented at the new building's location.
 *
 * Most methods for a particular time are divided into M-Sat and Sunday, but when the
 * activities are the same everday (eg. noon), only one method is needed.
 *
 * See document called San Diego Schedules.xls (available on
 * Carolyn's computer) for details on the timing of different activities.
 */

public void do6amActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses) {
```

```
		SDSimplePerson.atLocationList[this.presentLocation][1]--;
		switch (occType) {
			case 500:
				moveToBldg(xLoc, yLoc, kitchenID, person, masterBuildingsList);
				break;
			default:
				moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
				numHouses);
				break;
				}
		SDSimplePerson.atLocationList[this.presentLocation][1]++;
			}

public void do7amActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
		ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses) {

		SDSimplePerson.atLocationList[this.presentLocation][1]--;
		moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList, numHouses);

		SDSimplePerson.atLocationList[this.presentLocation][1]++;
			}

public void do8amActs(int agentHouse, int xLoc, int yLoc,SDSimplePerson person,
		ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses) {

		SDSimplePerson.atLocationList[this.presentLocation][1]--;
		moveToBldg(xLoc, yLoc, churchID, person, masterBuildingsList);
		SDSimplePerson.atLocationList[this.presentLocation][1]++;
			}

public void doMSat9amActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
		ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses) {

		double boyCutoff = 0;
		double boyProb = 0;
		SDSimplePerson.atLocationList[this.presentLocation][1]--;

		switch (occType) {
			case 100:
				moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
				break;
			case 200:
				moveWeave(xLoc, yLoc, agentHouse, person,
					masterBuildingsList);
				break;
			case 300:
				moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
				break;
			case 400:
				moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
					numHouses);
				break;
			case 500:
```

```
                    /* this function was chosen so that the cutoff probability for boys to
                     * move to the fields would range approximately from 0.1 to 0.75 and
                     * increase with the boy's age. So, the older a boy is, the more likely
                     * he is to travel to the fields.
                     */
                            boyCutoff = (this.age*0.08)-0.22;
                            boyProb = RandomHelper.nextDoubleFromTo(0, 1);
                            if (boyProb < boyCutoff){
                                    moveToBldg(xLoc, yLoc, fieldsID, person,
                                            masterBuildingsList);
                                    }
                                    else moveToBldg(xLoc, yLoc, kitchenID, person,
                                            masterBuildingsList);

                                    break;
                    case 600:
                            moveWeave(xLoc, yLoc, agentHouse, person,
                                    masterBuildingsList);
                            break;
                    case 700:
                            if (this.dwelling != 171){
                            moveWeave(xLoc, yLoc, agentHouse, person,
                                    masterBuildingsList);}
                            else moveHome(agentHouse, xLoc, yLoc, person,
                                    masterBuildingsList, numHouses);
                            break;
                    case 800:
                            int priestChoice = RandomHelper.nextIntFromTo(1,
                                    SanDiegoMissionContext.numBuildings);
                            moveToBldg(xLoc, yLoc, priestChoice, person,
                                    masterBuildingsList);
                            break;
                            }
                    SDSimplePerson.atLocationList[this.presentLocation][1]++;
                    }

public void doSun9amActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
            ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses) {

            SDSimplePerson.atLocationList[this.presentLocation][1]--;
            moveToBldg(xLoc, yLoc, churchID, person, masterBuildingsList);
            SDSimplePerson.atLocationList[this.presentLocation][1]++;
                    }

public void doMSat10amActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
            ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses) {

            double boyCutoff = 0;
            double boyProb = 0;

            SDSimplePerson.atLocationList[this.presentLocation][1]--;

switch (occType) {
```

```
		case 100:
			moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
			break;
		case 200:
			moveWeave(xLoc, yLoc,agentHouse, person, masterBuildingsList);
			break;
		case 300:
			moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
			break;
		case 400:
			moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
				numHouses);
			break;
		case 500:
			/* this function was chosen so that the cutoff probability for boys to
			 * move to the fields would range approximately from 0.1 to 0.75 and
			 * increase with the boy's age. So, the older a boy is, the more likely
			 * he is to travel to the fields.
			 */
			boyCutoff = (this.age*0.08)-0.22;
			boyProb = RandomHelper.nextDoubleFromTo(0, 1);
			if (boyProb < boyCutoff){
			moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
				}
			else moveToBldg(xLoc, yLoc, kitchenID, person, masterBuildingsList);
			break;
		case 600:
			moveWeave(xLoc, yLoc,agentHouse, person, masterBuildingsList);
			break;
		case 700:
			if (this.dwelling != 171){
				moveWeave(xLoc, yLoc, agentHouse, person,
					masterBuildingsList);}
			else moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
					numHouses);
			break;
		case 800:
			int priestChoice = RandomHelper.nextIntFromTo(1,
				SanDiegoMissionContext.numBuildings);
			moveToBldg(xLoc, yLoc, priestChoice, person, masterBuildingsList);
			break;
				}
SDSimplePerson.atLocationList[this.presentLocation][1]++;
			}

public void doSun10amActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
		ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses) {

		SDSimplePerson.atLocationList[this.presentLocation][1]--;
		moveToBldg(xLoc, yLoc, churchID, person, masterBuildingsList);
		SDSimplePerson.atLocationList[this.presentLocation][1]++;
			}
```

```
public void doMSat11amActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses) {

        SDSimplePerson.atLocationList[this.presentLocation][1]--;
        switch (occType) {
                case 100:
                        moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
                        break;
                case 200:
                        moveWeave(xLoc, yLoc, agentHouse, person,
                                masterBuildingsList);
                        break;
                case 300:
                        moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
                        break;
                case 400:
                        moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
                                numHouses);
                        break;
                case 500:
                        moveToBldg(xLoc, yLoc, kitchenID, person, masterBuildingsList);
                        break;
                case 600:
                        moveWeave(xLoc, yLoc, agentHouse, person,
                                masterBuildingsList);
                        break;
                case 700:
                        if (this.dwelling != 171){
                                moveWeave(xLoc, yLoc, agentHouse, person,
                                        masterBuildingsList);}
                        else moveHome(agentHouse, xLoc, yLoc, person,
                                masterBuildingsList, numHouses);
                        break;
                case 800:
                        int priestChoice = RandomHelper.nextIntFromTo(1,
                                SanDiegoMissionContext.numBuildings);
                        moveToBldg(xLoc, yLoc, priestChoice, person,
                                masterBuildingsList);
                        break;
                        }
SDSimplePerson.atLocationList[this.presentLocation][1]++;
                }

public void doSun11amActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses) {

        SDSimplePerson.atLocationList[this.presentLocation][1]--;
        switch (occType) {
                case 500:
                        moveToBldg(xLoc, yLoc, kitchenID, person, masterBuildingsList);
                        break;
                default:
                        moveToBldg(xLoc, yLoc, churchID, person, masterBuildingsList);
```

```
                        break;
                        }
SDSimplePerson.atLocationList[this.presentLocation][1]++;
                }

public void doNoonActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses) {

        SDSimplePerson.atLocationList[this.presentLocation][1]--;
        moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList, numHouses);
        SDSimplePerson.atLocationList[this.presentLocation][1]++;
                }

public void doMonSat1pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson
        person, ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int
        numHouses) {

         SDSimplePerson.atLocationList[this.presentLocation][1]--;
         switch (occType) {
                case 500:
                        moveToBldg(xLoc, yLoc, churchID, person, masterBuildingsList);
                        break;
                case 600:
                        moveToBldg(xLoc, yLoc, churchID, person, masterBuildingsList);
                        break;
                case 800:
                        moveToBldg(xLoc, yLoc, churchID, person, masterBuildingsList);
                        break;
                default:
                        moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
                                numHouses);
                        break;
                        }
SDSimplePerson.atLocationList[this.presentLocation][1]++;
                }

public void doSun1pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses) {

        SDSimplePerson.atLocationList[this.presentLocation][1]--;
        moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList, numHouses);
        SDSimplePerson.atLocationList[this.presentLocation][1]++;
                }

public void doMonSat2pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson
        person, ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int
        numHouses){

         double boyCutoff = 0;
         double boyProb = 0;
         SDSimplePerson.atLocationList[this.presentLocation][1]--;
         switch (occType) {
                case 100:
```

```
                    moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
                    break;
                case 200:
                    moveWeave(xLoc, yLoc,agentHouse, person, masterBuildingsList);
                    break;
                case 300:
                    moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
                    break;
                case 400:
                    moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
                        numHouses);
                    break;
                case 500:
                /* this function was chosen so that the cutoff probability for boys to
                *move to the fields would range approximately from 0.1 to 0.75 and
                *increase with the boy's age. So, the older a boy is, the more likely he is
                *to travel to the fields. */

                    boyCutoff = (this.age*0.08)-0.22;
                    boyProb = RandomHelper.nextDoubleFromTo(0, 1);
                    if (boyProb < boyCutoff){
                    moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
                        }
                    else moveToBldg(xLoc, yLoc, kitchenID, person,
                        masterBuildingsList);
                    break;
                case 600:
                    moveWeave(xLoc, yLoc,agentHouse, person, masterBuildingsList);
                    break;
                case 700:
                    if (this.dwelling != 171){
                        moveWeave(xLoc, yLoc, agentHouse, person,
                            masterBuildingsList);}
                    else moveHome(agentHouse, xLoc, yLoc, person,
                        masterBuildingsList, numHouses);
                    break;
                case 800:
                    int priestChoice = RandomHelper.nextIntFromTo(1,
                        SanDiegoMissionContext.numBuildings);
                    moveToBldg(xLoc, yLoc, priestChoice, person,
                        masterBuildingsList);
                    break;
                    }
            SDSimplePerson.atLocationList[this.presentLocation][1]++;
                }

public void doSun2pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses, int
        houseList[][], int mPat, int extFam){

         SDSimplePerson.atLocationList[this.presentLocation][1]--;
         switch (occType) {
                case 100:
```

```
                        moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
                                houseList, mPat, extFam);
                        break;
                case 200:
                        moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
                                houseList, mPat, extFam);
                        break;
                case 300:
                        moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
                                houseList, mPat, extFam);
                        break;
                case 400:
                        moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
                                houseList, mPat, extFam);
                        break;
                case 800:
                        int priestChoice = RandomHelper.nextIntFromTo(1,
                                SanDiegoMissionContext.numBuildings);
                        moveToBldg(xLoc, yLoc, priestChoice, person,
                                masterBuildingsList);
                        break;
                        }
         SDSimplePerson.atLocationList[this.presentLocation][1]++;
                        }

public void doMonSat3pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson
        person, ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int
        numHouses){

        double boyCutoff = 0;
        double boyProb = 0;

        SDSimplePerson.atLocationList[this.presentLocation][1]--;
        switch (occType) {
                case 100:
                        moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
                        break;
                case 200:
                        moveWeave(xLoc, yLoc, agentHouse, person,
                                masterBuildingsList);
                        break;
                case 300:
                        moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
                        break;
                case 400:
                        moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
                                numHouses);
                        break;
                case 500:
                /* this function was chosen so that the cutoff probability for boys to
                 * move to the fields would range approximately from 0.1 to 0.75 and
                 * increase with the boy's age. So, the older a boy is, the more likely
                 * he is to travel to the fields.
```

```
                */
                        boyCutoff = (this.age*0.08)-0.22;
                        boyProb = RandomHelper.nextDoubleFromTo(0, 1);
                        if (boyProb < boyCutoff){
                                moveToBldg(xLoc, yLoc, fieldsID, person,
                                masterBuildingsList);
                        }
                        else moveToBldg(xLoc, yLoc, kitchenID, person,
                                masterBuildingsList);
                        break;
                case 600:
                        moveWeave(xLoc, yLoc,agentHouse, person, masterBuildingsList);
                        break;
                case 700:
                        if (this.dwelling != 171){
                        moveWeave(xLoc, yLoc, agentHouse, person,
                                masterBuildingsList);}
                        else moveHome(agentHouse, xLoc, yLoc, person,
                                masterBuildingsList, numHouses);
                        break;
                case 800:
                        int priestChoice = RandomHelper.nextIntFromTo(1,
                                SanDiegoMissionContext.numBuildings);
                        moveToBldg(xLoc, yLoc, priestChoice, person,
                                masterBuildingsList);
                        break;
                        }
         SDSimplePerson.atLocationList[this.presentLocation][1]++;

                }

public void doSun3pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses){

        SDSimplePerson.atLocationList[this.presentLocation][1]--;
        switch (occType) {
                case 800:
                        int priestChoice = RandomHelper.nextIntFromTo(1,
                                SanDiegoMissionContext.numBuildings);
                        moveToBldg(xLoc, yLoc, priestChoice, person,
                                masterBuildingsList);
                        break;
                default:
                        moveToBldg(xLoc, yLoc,
                                SDSimplePerson.atLocationList[person.presentLocation][0]
                                , person, masterBuildingsList);
                        break;
                        }
SDSimplePerson.atLocationList[this.presentLocation][1]++;
                }

public void doMonSat4pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson
        person, ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int
```

```
        numHouses){

        SDSimplePerson.atLocationList[this.presentLocation][1]--;
        switch (occType) {
                case 100:
                        moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
                        break;
                case 200:
                        moveWeave(xLoc, yLoc,agentHouse, person, masterBuildingsList);
                        break;
                case 300:
                        moveToBldg(xLoc, yLoc, fieldsID, person, masterBuildingsList);
                        break;
                case 400:
                        moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
                                numHouses);
                        break;
                case 500:
                        moveToBldg(xLoc, yLoc, kitchenID, person, masterBuildingsList);
                        break;
                case 600:
                        moveWeave(xLoc, yLoc,agentHouse, person, masterBuildingsList);
                        break;
                case 700:
                        if (this.dwelling != 171){
                                moveWeave(xLoc, yLoc, agentHouse, person,
                                        masterBuildingsList);}
                                else moveHome(agentHouse, xLoc, yLoc, person,
                                        masterBuildingsList, numHouses);
                                break;
                case 800:
                        int priestChoice = RandomHelper.nextIntFromTo(1,
                                        SanDiegoMissionContext.numBuildings);
                        moveToBldg(xLoc, yLoc, priestChoice, person,
                                        masterBuildingsList);
                        break;
                        }
SDSimplePerson.atLocationList[this.presentLocation][1]++;
                }

public void doSun4pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses){

        SDSimplePerson.atLocationList[this.presentLocation][1]--;
        switch (occType) {
                case 800:
                        int priestChoice = RandomHelper.nextIntFromTo(1,
                                SanDiegoMissionContext.numBuildings);
                        moveToBldg(xLoc, yLoc, priestChoice, person,
                                masterBuildingsList);
                        break;
                default:
                        moveToBldg(xLoc, yLoc,
```

```
                                SDSimplePerson.atLocationList[person.presentLocation][0]
                                , person, masterBuildingsList);
                        break;
                        }
SDSimplePerson.atLocationList[this.presentLocation][1]++;
                }

public void do5pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses){

        SDSimplePerson.atLocationList[this.presentLocation][1]--;
        moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList, numHouses);
        this.canVisit = true; //needed to reset the variable for visiting in the next time
                step on Sundays

         SDSimplePerson.atLocationList[this.presentLocation][1]++;

                }

public void do6pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses, int
        houseList[][], int mPat, int extFam){

        SDSimplePerson.atLocationList[this.presentLocation][1]--;
        switch (occType) {
                case 100:
                        moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
                                houseList, mPat, extFam);
                        break;
                case 200:
                        moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
                                houseList, mPat, extFam);
                        break;
                case 300:
                        moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
                                houseList, mPat, extFam);
                        break;
                case 400:
                        moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
                                numHouses);
                        break;
                case 800:
                        moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
                                numHouses);
                        break;
                        }
SDSimplePerson.atLocationList[this.presentLocation][1]++;
                }

public void do7pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses, int
        houseList[][], int mPat, int extFam){
```

```
		SDSimplePerson.atLocationList[this.presentLocation][1]--;
		switch (occType) {
			case 100:
				moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
					houseList, mPat, extFam);
				break;
			case 200:
				moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
					houseList, mPat, extFam);
				break;
			case 300:
				moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
					houseList, mPat, extFam);
				break;
			case 400:
				moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
					numHouses);
				break;
			case 800:
				moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
					numHouses);
				break;
				}
SDSimplePerson.atLocationList[this.presentLocation][1]++;
			}

public void do8pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
		ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses, int
			houseList[][], int mPat, int extFam){

		 SDSimplePerson.atLocationList[this.presentLocation][1]--;
		 switch (occType) {
			case 100:
				moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
					houseList, mPat, extFam);
				break;
			case 200:
				moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
					houseList, mPat, extFam);
				break;
			case 300:
				moveVisit(xLoc, yLoc, person, masterBuildingsList, numHouses,
					houseList, mPat, extFam);
				break;
			case 800:
				moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
					numHouses);
				break;
				}
		 SDSimplePerson.atLocationList[this.presentLocation][1]++;

			}
```

```
public void do9pmActs(int agentHouse, int xLoc, int yLoc, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int numHouses){

        SDSimplePerson.atLocationList[this.presentLocation][1]--;
        switch (occType) {
                case 400:
                        break;
                case 800:
                        break;
                default:
                        moveHome(agentHouse, xLoc, yLoc, person, masterBuildingsList,
                                numHouses);
                                break;
                        }
        this.canVisit = true;//needed to reset here for visiting the next day
        SDSimplePerson.atLocationList[this.presentLocation][1]++;
                }

/**The Move Methods Start Here*/
/*
 * The moveHome method moves agents to or within their assigned dwellings at
 * any time after model initialization. If an agent attempts to move within a full house,
 * it stays where they are. If an an agent tries to enter a filled home, it picks a different
 * dwelling to go to, but is not reassigned permanently.
 *
 * This method is used by all agents, including women in the monjeria and the priests.
 */

@SuppressWarnings("unchecked")
public void moveHome(int agentHouse, int xLocMH, int yLocMH, SDSimplePerson
        person, ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList, int
        numHouses) {

        Context context = ContextUtils.getContext(person);
        Grid patch = (Grid) context.getProjection("MissionGrid");
        int grpNum = 1;
        /*In the present model, only individuals call moveHome, so grpNum is set at
        * one.    If, in the future, we have groups traveling home together, we need to set
        * grpNum to the the size of the travel group*/

        person.foundLocation = false;
        numCells = person.getPossibleCoords(person.dwelling, masterBuildingsList,
                patch);
        if (SDSimplePerson.atLocationList[person.presentLocation][0]==person.dwelling
                &&SDSimplePerson.atLocationList[person.mblLocator(agentHouse,
                masterBuildingsList)][1] == numCells-1){
         //If the individual is already home and the building is full, do nothing
                }
         else if (SDSimplePerson.atLocationList[person.mblLocator(agentHouse,
                masterBuildingsList)][1] < numCells) {
         //if there is space in the dwelling, individual moves within that building
                xLocMH = person.possXLoc;
                yLocMH = person.possYLoc;
```

```
                assignLocation(person.dwelling, xLocMH, yLocMH, person,
                        masterBuildingsList, patch);
                }
        else{
                if (person.foundLocation == true) {
                        System.out.println("Agent ID: "+person.agtID+" I am in a
                        moveHome loop that I" +" should have skipped.");
                                        }
                else{
                        person.tempDwelling = person.findVisitDwelling(person,
                                numHouses,hLineList, grpNum, masterBuildingsList);
                        numCells = person.getPossibleCoords(person.tempDwelling,
                                masterBuildingsList, patch);
                        xLocMH = person.possXLoc;
                        yLocMH = person.possYLoc;
                        assignLocation(person.tempDwelling, xLocMH, yLocMH, person,
                                masterBuildingsList, patch);
                                                }
                        }
                }

/* The moveToBldg method sends agents to the building specified in the call. If no space
 * is available, alternate movement strategies are specified in the basicMove method.
 */
@SuppressWarnings("unchecked")
public void moveToBldg(int xLocMB, int yLocMB, int buildingID, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList) {

        Context context = ContextUtils.getContext(person);
        Grid patch = (Grid) context.getProjection("MissionGrid");

        numCells = person.getPossibleCoords(buildingID, masterBuildingsList, patch);
        basicMove(buildingID, numCells, xLocMB, yLocMB, person, masterBuildingsList,
                patch);
                }

/* The moveWeave method allows agents to move in or around their dwellings during
 * weaving times.  We used a level 2 Moore neighborhood with a test to determine
 * whether there is a building cell with a different building ID on a chosen spot. If that is
 * the case, the agent will not be allowed to move to that location and will choose
 * another.
 */

@SuppressWarnings("unchecked")
public void moveWeave(int xLocMW, int yLocMW, int agentHouse, SDSimplePerson
        person,ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList) {

        Context context = ContextUtils.getContext(person);
        Grid patch = (Grid) context.getProjection("MissionGrid");
        numCells = person.getPossibleCoords(person.dwelling, masterBuildingsList,
                patch);

         xLocMW = RandomHelper.nextIntFromTo(this.possXLoc-2, this.possXLoc+2);
```

```
yLocMW = RandomHelper.nextIntFromTo(this.possYLoc-2, this.possYLoc+2);

person.foundLocation = false;

for (int i = 0; i < numCells*100; i++) {
        /*upper bound is large enough to ensure an agent finds
        *an empty cell if one exists.*/

        occupancy = person.testLocation(xLocMW, yLocMW, patch);
                /*
                 * If a cell is occupied by a person agent, the calling agent must
                 * choose a new cell within the building by calling
                 * getPossibleCoords and resetting posXLoc and posYLoc. This
                 * loop continues as long as the chosen coordinates are already
                 * occupied.
                 *
                 * If a cell is not occupied, the agent moves to that location.
                 *
                 * If an agent attempts to find coordinates the maximum number
                 * of times and is still unable to find an empty cell, foundlocation
                 * is returned as false to the appropriate move method (e.g.
                 * moveHome) and the agent's next strategy is determined.
                 */
        if (occupancy == true && person.foundLocation == false) {
        person.getPossibleCoords(person.dwelling, masterBuildingsList, patch);

        xLocMW = RandomHelper.nextIntFromTo(this.possXLoc-2,
                this.possXLoc+2);
        yLocMW = RandomHelper.nextIntFromTo(this.possYLoc-2,
                this.possYLoc+2);

        occupancy = person.testLocation(xLocMW, yLocMW, patch);
                        } else if (occupancy == false) {
                        person.xLocation = xLocMW;
                        person.yLocation = yLocMW;
                        patch.moveTo(person, person.xLocation,
                                person.yLocation);
                        int destinationRow = 0;
                for (int index = 0; index < SDSimplePerson.atLocationList.length;
                                index++) {
                if (SDSimplePerson.atLocationList[index][0] == agentHouse) {
                        destinationRow = index;
                                                }
                                                }
                        person.presentLocation = destinationRow;
                                        }
                        break;
                }
                        person.foundLocation = true;
        }

/*
 * The agent calling the moveVisit method makes a traveling group composed of live
```

```
 * agents in the current building.  A visit dwelling is determined and the group
 * moves there.
 */
public void moveVisit(int xLocMV, int yLocMV, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList,
        int numHouses, int houseList[][], int mPat, int extFam){

        double prob = 0;
        prob = RandomHelper.nextDoubleFromTo(0, 1);
        ArrayList<SDSimplePerson> travList = new ArrayList<SDSimplePerson>();
                travList.clear();
        travList = this.makeTravGrp(xLocMV, yLocMV, person, masterBuildingsList,
                numHouses, hList);
        int visitBuilding = this.dwelling;

        if ((prob <= 1.0) && (this.canVisit == true)) {
                visitBuilding = this.findVisitDwelling(person, numHouses, hLineList,
                        travList.size(), masterBuildingsList);
                        }
                moveGroup(xLocMV, yLocMV, person, masterBuildingsList, travList,
                        visitBuilding);
                }

/*
 * The moveGroup method contains code to allow a group to visit another place.
 * This is called by any agent directing the movement of a group, or in certain cases
 * directing their own movement. It requires the ID of the building to be
 * visited and the traveling group, formed prior to the method call.
 * The group then moves to that building.
 */
@SuppressWarnings("unchecked")
public void moveGroup(int xLocMG, int yLocMG, SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingsList,
        ArrayList <SDSimplePerson> grpList, int visitBldg) {

        Context context = ContextUtils.getContext(person);
        Grid patch = (Grid) context.getProjection("MissionGrid");

/* Within the do...Activities method, the atLocationList is decremented to
 * remove the calling agent from the occupancy count of their present location
 * before they actually move and after they move it is incremented to add them
 * to the occupancy count of the new location.  When they call moveGroup, a
 * similar process occurs for the entire traveling group, including the agent
 * calling the method.  Thus, without some compensation, the agent calling the
 * method will be subtracted twice from the list before they move and added twice
 * after they move.  To avoid this, we add them back to the atLocationList for
 * the original location before subtracting everyone in the traveling group.
 * For similar reasons, we subtract the calling agent after the for loop below,
 * so when they add themselves in the do..activities method, the count will be correct.
 * This rather convoluted process is necessary because many of the movements do
 * not involve making groups, and moving them together, so therefore it is actually
 * better to do the fix here than change the whole method.
 */
```

```
		SDSimplePerson.atLocationList [person.presentLocation][1]++;
		numCells = person.getPossibleCoords(visitBldg, masterBuildingsList, patch);

		for (int i= 0; i <grpList.size(); i++){
			SDSimplePerson agentI = grpList.get(i);
			SDSimplePerson.atLocationList [agentI.presentLocation][1]--;

			agentI.basicMove(visitBldg, numCells, agentI.xLocation, agentI.yLocation,
				agentI, masterBuildingsList, patch);
			SDSimplePerson.atLocationList [agentI.presentLocation][1]++;
				}
			SDSimplePerson.atLocationList [person.presentLocation][1]--;
			ArrayList<SDSimpleBuilding> ptBldgs = new
				ArrayList<SDSimpleBuilding>();

			ptBldgs.addAll(masterBuildingsList.get(person.mblLocator(visitBldg,
				masterBuildingsList)));
			for (int j = 0; j < ptBldgs.size(); j++) {
				SDSimpleBuilding bldgCell = ptBldgs.get(j);
				GridPoint point = patch.getLocation(bldgCell);
				int x = point.getX();
				int y = point.getY();
				for (Object agentIn : patch.getObjectsAt(x, y)) {
					if (agentIn instanceof SDSimplePerson) {
						((SDSimplePerson) agentIn).canVisit = false;
						}
					}
				}
			}

/*
 * The basicMove method contains code that is common to all of our specific
 * moveMethods and is called for all of them except moveHome. The moveHome code
 * is a little different because it involved some additional features
 * not found in the other methods (e.g. picking a temporary dwelling if
 * the assigned one was not available when the agent was created).
 */
@SuppressWarnings("unchecked")
public void basicMove(int moveToID, int possibleCells, int xLocBM, int yLocBM,
		SDSimplePerson person, ArrayList<ArrayList<SDSimpleBuilding>>
		masterBuildings, Grid grid) {

		if (SDSimplePerson.atLocationList[person.presentLocation][0]== moveToID &&
			SDSimplePerson.atLocationList[person.mblLocator(moveToID,
			masterBuildings)][1] == possibleCells-1){
				}
		 else if (SDSimplePerson.atLocationList[mblLocator(moveToID,
			masterBuildings)][1] < possibleCells) {
			xLocBM = person.possXLoc;
			yLocBM = person.possYLoc;
			assignLocation(moveToID, xLocBM, yLocBM, person, masterBuildings,
				grid);
```

```
                    if ((foundLocation == false)) {
                            GridPoint point = grid.getLocation(person);
                            person.xLocation = point.getX();// X-coord of ptBldg
                            person.yLocation = point.getY();// Y-coord of ptBldg
// System.out.println("I am agent "+person.agtID+". I could not find a space, so I have"+" remained where
I was.");
                                    }
                            }
                    }

/* The mblLocator method uses a building ID to find that building's position on the
 * master building list.
 */

public int mblLocator (int moveToID, ArrayList<ArrayList<SDSimpleBuilding>>
        masterBuildingList){

        int location=0;
        boolean match = false;
        ArrayList<SDSimpleBuilding> mblBldgs = new ArrayList<SDSimpleBuilding>();

        while (match == false && location < masterBuildingList.size()){
                mblBldgs.addAll(masterBuildingList.get(location));
                SDSimpleBuilding mblBldgIn = mblBldgs.get(0);

                if (mblBldgIn.bldgID == moveToID){
                        match= true;
                                }
                location++;
                mblBldgs.clear();
                        }
                location--;
                return location;
                }

/* The findVisitDwelling method picks a new dwelling for temporary purposes, like
 * visiting. Agents first attempt to find a kin dwelling to visit (mother or father's kin). If
 * that is unsuccessful, they can attempt to visit a random dwelling.  If that doesn't work,
 * they stay home. Females assigned to the monjeria are only allowed to visit kin
 * dwellings.
 *
 * Note: visitDwellPos refers to positions of buildings in the building array lists,
 * visitDwelling refers to the building ID.
 */

@SuppressWarnings("unchecked")
public int findVisitDwelling(SDSimplePerson person, int numHouses, int houseList[][], int
        groupNum, ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingList) {

        int element = 0;
        ArrayList<Integer> houseChoice = new ArrayList<Integer>();
        houseChoice.clear();
```

```
for (int i = 0; i < numHouses; i++){
		houseChoice.add(element);
		element++;
			}
Collections.shuffle(houseChoice);
int posHouse = 0;
int counter = 0;
int visitDwellPos = RandomHelper.nextIntFromTo(1, numHouses) - 1;
int visitDwelling = this.dwelling;
boolean visitDwellingFound = false;
Context context = ContextUtils.getContext(person);
Grid patch = (Grid) context.getProjection("MissionGrid");

makeKinHouseList(numHouseSP, houseList);

for (int j = 0; j<numHouses; j++){
		posHouse = houseChoice.get(j);//posHouse = mblLoc and row on
			kinHouseList
		numCells = person.getPossibleCoords(posHouse, masterBuildingList,
			patch);
		if (this.kinHouseList[posHouse][0] != -1){

		if (((SDSimplePerson.atLocationList[this.kinHouseList[posHouse][0]][1] <=
			numCells - groupNum)&&
			(SDSimplePerson.atLocationList[this.kinHouseList[posHouse][0]][1] > 0)&&
			(SDSimplePerson.atLocationList[this.kinHouseList[posHouse][0]][0] !=
			this.dwelling))){
				visitDwellPos =kinHouseList[posHouse][0];
				visitDwelling = kinHouseList[posHouse][1];
				visitDwellingFound = true;
				break;
				}
			}
			}

if (visitDwellingFound == false && this.dwelling != 171){
		while ((counter
			<=(houseList.length*10))&&((SDSimplePerson.atLocationList[hous
			eList[visitDwellPos][0]][1] > person.numCells- groupNum)	||
			(SDSimplePerson.atLocationList[houseList[visitDwellPos][0]][1] ==
			0)||
			(SDSimplePerson.atLocationList[houseList[visitDwellPos][0]][0] ==
			this.dwelling))){

			visitDwellPos = RandomHelper.nextIntFromTo(1, numHouses) - 1;
			visitDwelling = houseList[visitDwellPos][1];
			numCells = person.getPossibleCoords(visitDwelling,
				masterBuildingList, patch);
			counter++;
				}

 if(counter>(houseList.length*10)){
		visitDwelling = this.dwelling;
```

```
                numCells = person.getPossibleCoords(visitDwelling, masterBuildingList,
                        patch);
                                }
                visitDwellingFound = true;
                        }
            if(this.dwelling == 171){
            //monjeria girls will not have reset their visitDwelling if they didn't find a kin
           house to visit, so they will return to the monjeria
                        }
                        return visitDwelling;
                }

/* findNewDwelling picks a new dwelling for agents who must move when someone
 * else dies. It is called by children who must move when no caretaker is available in
 * their dwelling.
 *
 * All children first look for a kin house.  Girls and infants require at least 1 adult female
 * be assigned to the kin house. If a kin house is not available for them, they move to the
 * monjeria.  Boys will move to a random house if a kin house is not available. In
 * extreme cases where no other house is available, they stay where they are.
 *
 * Note: newDwellPos refers to positions of buildings in
 * the building array lists, newDwelling refers to the building ID.
 *
 */

@SuppressWarnings("unchecked")
public int findNewDwelling(SDSimplePerson person, int numHouses, int houseList[][], int
        groupNum, ArrayList<ArrayList<SDSimpleBuilding>> masterBuildingList) {

        int element = 0;
        this.newDwellingFound = false;
        Context context = ContextUtils.getContext(person);
        Grid patch = (Grid) context.getProjection("MissionGrid");
        Iterable listAgents = null;
        ArrayList adultAgentsFND = new ArrayList <SDSimplePerson>();
        adultAgentsFND.clear();
        InstanceOfQuery <SDSimplePerson> allAgents = new
                InstanceOfQuery<SDSimplePerson>(context, SDSimplePerson.class);

        //listAgents is a list of all SimplePersonAgents in the context, including both dead and alive
        listAgents = allAgents.query();

        /*adultAgents is a list populated by only living adult agents except priests, who
         * cannot serve as caregivers for orphans
         */
        for (Object agentin : listAgents){
                if (((SDSimplePerson)agentin).disStatus !=4
                        &&((SDSimplePerson)agentin).occupation<500 ) {
                        adultAgentsFND.add((SDSimplePerson)agentin);
                        }}

/*House choice is a list of numbers between 1 and the number of houses.
```

```
 * it is used by agents to randomly pick a new house for reassignment testing.
 */
	ArrayList<Integer> houseChoice = new ArrayList<Integer>();
	houseChoice.clear();
	for (int i = 0; i < numHouses; i++){
		houseChoice.add(element);
		element++;
			}
	Collections.shuffle(houseChoice);
	int posHouse = 0;
	int counter = 0;
	int newDwellPos = RandomHelper.nextIntFromTo(1, numHouses) - 1;
	int newDwelling = this.dwelling;

	//makes a list of kin houses to choose from for reassignment to new dwelling.
	makeKinHouseList(numHouseSP, houseList);

	for (int j = 0; j<numHouses; j++){
	posHouse = houseChoice.get(j);
//System.out.println("0. Agent "+ this.agtID+" wants to move to "+this.kinHouseList[posHouse][1]+" I am
currently at "+this.presentLocation);

	numCells = person.getPossibleCoords(this.kinHouseList[posHouse][1],
		masterBuildingList, patch);
	 if (this.kinHouseList[posHouse][1] != -1){
		if((SanDiegoMissionContext.permanentResidents[this.kinHouseList[posH
			ouse][0]][1] <= numCells - groupNum)&&
			(SanDiegoMissionContext.permanentResidents[this.kinHouseList[
			posHouse][0]][1]> 0) && (this.kinHouseList[posHouse][1] !=
			this.dwelling)){
				if (this.occupation == 500){ this.newDwellingFound=true;
					}
				else if (this.occupation==600||this.occupation==700){
					for (Object adultFemale : adultAgentsFND){
						if
						(((SDSimplePerson)adultFemale).sex.equals(
						"F")&&((SDSimplePerson)adultFemale).dwe
						lling==this.kinHouseList[posHouse][1]){
//System.out.println("Found Female: "+((SDSimplePerson)adultFemale).agtID);

	this.newDwellingFound = true;
	break;
		}}}}

	if (this.newDwellingFound == true){

		SanDiegoMissionContext.permanentResidents[this.mblLocator(this.dwelli
			ng, masterBuildingList)][1]--;
		this.dwelling = kinHouseList[posHouse][1];

		SanDiegoMissionContext.permanentResidents[this.mblLocator(this.dwelli
			ng, masterBuildingList)][1]++;
//System.out.println("1. I am agent "+this.agtID+", I want to be reassigned to kin
```

```
dwelling "+this.dwelling);
                        break;
                                        }
                        }
                        }
/*If no suitable kin houses are found, girls and infants are assigned to the monjeria and
 * boys are assigned to random households.
 */

        if (this.newDwellingFound == false && this.dwelling != 171){
                        SanDiegoMissionContext.permanentResidents[this.mblLocator(this.dwelling,
                masterBuildingList)][1]--;

         if (this.occupation==600||this.occupation==700){
                this.dwelling=171;
                if (this.occupation == 600){
                        this.occupation = 400;
                        this.occType = 400;
                                }

        SanDiegoMissionContext.permanentResidents[this.mblLocator(this.dwelling,
                masterBuildingList)][1]++; numCells =
                this.getPossibleCoords(this.dwelling, masterBuildingList, patch);
        if(numCells < SanDiegoMissionContext.permanentResidents [this.mblLocator
                (this.dwelling, masterBuildingList)][1]){

System.out.println("TOO MANY AGENTS assigned to monjeria, need a new strategy");
}//This error comment will print if the monjeria is overloaded, but there is no code to actually deal with
the problem for now

        this.newDwellingFound = true;
                }else if (this.occupation==500){

 while ((counter <=(houseList.length*10)) &&((SDSimplePerson.atLocationList
        [houseList[newDwellPos][0]][1] > person.numCells - groupNum)          ||
        (SDSimplePerson.atLocationList[houseList[newDwellPos][0]][1] == 0)
        || (SDSimplePerson.atLocationList[houseList[newDwellPos][0]][0] ==
        this.dwelling))){
                newDwellPos = RandomHelper.nextIntFromTo(1, numHouses) - 1;
                newDwelling = houseList[newDwellPos][1];
                numCells = person.getPossibleCoords(posHouse, masterBuildingList,
                        patch);
                counter++;

//System.out.println("2.I am agent "+this.agtID+", I want to be reassigned random dwelling
"+newDwelling+" after "+counter+" tries.");
                        }
        if (counter <= (houseList.length)*10){
                this.dwelling = houseList[newDwellPos][1];
                this.newDwellingFound = true;

        SanDiegoMissionContext.permanentResidents[this.mblLocator(this.dwelling,
                masterBuildingList)][1]++;
```

```
                                        }
            if(counter>(houseList.length*10)){
                        newDwelling = this.dwelling;
            //numCells = person.getPossibleCoords(this.dwelling, masterBuildingList, patch);

            SanDiegoMissionContext.permanentResidents[this.mblLocator(this.dwelling,
                        masterBuildingList)][1]++;
//System.out.println("3. I am agent "+this.agtID+", I am staying in my old place
            "+newDwelling+" after "+counter+" tries.");
                                        }

            this.newDwellingFound = true;
                                    } }
            return this.dwelling;
                        }

/* The getPossibleCoords method finds the set of cells corresponding to a
 * an agent's destination and makes those available to the agent so that the
 * agent can choose a particular cell that it might be able to move to. The
 * method also returns the total number of cells that make up the building
 * specified in the method call.
 */
@SuppressWarnings("unchecked")
public int getPossibleCoords(int destinationID, ArrayList<ArrayList<SDSimpleBuilding>>
            masterBuildings, Grid grid) {

            ArrayList<SDSimpleBuilding> ptBldgs = new ArrayList<SDSimpleBuilding>();
            int mblLocation = mblLocator(destinationID, masterBuildings);
            ptBldgs.addAll(masterBuildings.get(mblLocation));

/*Gets all cells corresponding to the specified building from the masterBuilding list and
 * adds those cells to a new arrayList called ptBldgs*/

int numCells = ptBldgs.size();
Collections.shuffle(ptBldgs);// randomly shuffles the ptBldgs list

SDSimpleBuilding bldgIn = ptBldgs.get(0);

/*
* Since ptBldgs is shuffled every time the method is called, the cell in the index position *will(could) be
different for each call.
*/

GridPoint point = grid.getLocation(bldgIn);// gets the location for the cell chosen above

/*
 * These two statements assign the possible coordinates for agent movement.  Actual m
 * movement happens in the assignLocation method.
 */
            this.possXLoc = point.getX();
            this.possYLoc = point.getY();

            return numCells;
```

```
		}

/*
 * The testLocation method tests to see whether a chosen set of coordinates
 * already contains a SDSimple Person agent. If it does, the boolean variable
 * "occupancy" is set to true. The value of the occupancy variable is returned.
 */
@SuppressWarnings("unchecked")
public boolean testLocation(int x, int y, Grid grid) {
	occupancy = false;

	for (Object agentIn : grid.getObjectsAt(x, y)) {
		if (agentIn instanceof SDSimplePerson) {
		occupancy = true;
					}
				}
		return occupancy;
		}

@SuppressWarnings("unchecked")
public boolean assignLocation(int destinationID, int xLocAL,int yLocAL, SDSimplePerson
	person, ArrayList<ArrayList<SDSimpleBuilding>> masterBuildings, Grid grid) {

	 person.foundLocation = false;
	for (int i = 0; i < numCells*100; i++) {
	/*upper bound is large enough to ensure an agent finds an empty cell if one
		exists.*/

	occupancy = person.testLocation(xLocAL, yLocAL, grid);
	/*
	 * If a cell is occupied by a person agent, the calling agent must
	 * choose a new cell within the building by calling getPossibleCoords and
	 * resetting posXLoc and posYLoc. This loop continues as long as the
	 * chosen coordinates are already occupied.
	 *
	 * If a cell is not occupied, the agent moves to that location.
	 *
	 * If an agent attempts to find coordinates the maximum number of
	 * times and is still unable to find an empty cell, foundlocation is
	 * returned as false to the appropriate move method (e.g. moveHome)
	 * and the agent's next strategy is determined.
	 */
	if (occupancy == true && person.foundLocation == false) {
		person.getPossibleCoords(destinationID, masterBuildings, grid);

		xLocAL= this.possXLoc;
		yLocAL = this.possYLoc;
		occupancy = person.testLocation(xLocAL, yLocAL, grid);

		} else if (occupancy == false) {
			person.xLocation = xLocAL;
			person.yLocation = yLocAL;
			grid.moveTo(person, person.xLocation, person.yLocation);
```

```
                        SDSimpleBuilding destination = null;
                        int destinationRow = 0;
                        for (Object destIn : grid.getObjectsAt(person.xLocation,
                                person.yLocation)) {
                        if (destIn instanceof SDSimpleBuilding) {
                                destination = (SDSimpleBuilding) destIn;
                                for (int index = 0; index <
                                        SDSimplePerson.atLocationList.length; index++) {
                                if (SDSimplePerson.atLocationList[index][0] == destination.bldgID) {
                                                                                destinationRow =
                        index;
                                                                        }
                                                                }
                                person.presentLocation = destinationRow;
                                                        }
                                        }
                                        person.foundLocation = true;
                                        break;
                                }
                        }
                        return person.foundLocation;
                }

/* This method makes a list of agents currently in the same building as
 * the calling agent to be used for group movement.  Monjeria girls
 * do not add other monjeria residents to their group and move only themselves.
 *
 */
@SuppressWarnings("unchecked")
public ArrayList<SDSimplePerson> makeTravGrp(int xLocation, int yLocation,
        SDSimplePerson person, ArrayList<ArrayList<SDSimpleBuilding>>
        masterBuildingsList,int numHouses, int houseList[][]) {

        ArrayList<SDSimplePerson> travGrpList = new ArrayList<SDSimplePerson>();
        travGrpList.clear();

        Context context = ContextUtils.getContext(person);
        Grid patch = (Grid) context.getProjection("MissionGrid");
        if (this.dwelling == 171 && this.originalDwelling ==171){
                travGrpList.add(this);
                        }
        else{
                ArrayList<SDSimpleBuilding> ptBldgs = new ArrayList<SDSimpleBuilding>();
                ptBldgs.addAll(masterBuildingsList.get(this.presentLocation));
                for (int i = 0; i < ptBldgs.size(); i++) {
                        SDSimpleBuilding bldgCell = ptBldgs.get(i);
                        GridPoint point = patch.getLocation(bldgCell);
                        int x = point.getX();
                        int y = point.getY();

                        for (Object agentIn : patch.getObjectsAt(x, y)) {
                                if (agentIn instanceof SDSimplePerson) {
                                        if (this.dwelling != 171||(this.dwelling == 171 &&
```

```
                                                this.originalDwelling != 171 &&
                                                ((SDSimplePerson) agentIn).originalDwelling
                                                == this.originalDwelling)){
                                                travGrpList.add((SDSimplePerson) agentIn);
                                                }
/*
 * There may be GhostPerson agents in the building, but they will not be picked up in
 * this loop because they are not SimplePerson agents. The dead SimplePerson agents
 * move to the cemetery at the time they die, so they should no longer be in the
 * building.
 */
                                        }
                                        }
                                }
                        }
                                return travGrpList;
                        }

/* This method makes a list of agents currently in and around the same building who
 * share the same dwelling assignment as the calling agent (a mother) this list then
 * filtered in other methods to facilitate movement of the appropriate group, such as
 * movement to the monjeria upon the death of the calling agent's husband.
 */
@SuppressWarnings("unchecked")
public ArrayList<SDSimplePerson> makeMothersGrp(int xLocation, int yLocation,
        SDSimplePerson person, ArrayList<ArrayList<SDSimpleBuilding>>
        masterBuildingsList,int numHouses, int houseList[][]) {

                int bldgWidth = 0;
                int bldgLength = 0;
                ArrayList<SDSimplePerson> mothersGrpList = new
                ArrayList<SDSimplePerson>();
                mothersGrpList.clear();

                Context context = ContextUtils.getContext(person);
                Grid patch = (Grid) context.getProjection("MissionGrid");

                ArrayList<SDSimpleBuilding> mothersBldg = new
                ArrayList<SDSimpleBuilding>();
                        mothersBldg.addAll(masterBuildingsList.get(this.presentLocation));
        SDSimpleBuilding mothersBldgInfo = mothersBldg.get(0);

/*The following switch hardwires in typical dimensions. It assumes all buildings of a type
 * are of equal size. For better programming practice, another method of determining
 * these variables should be developed.  This could also be accomplished by referencing
 * the text file read in at the time of model initialization.
 */
        switch (mothersBldgInfo.type){
                case 1: //houses
                        bldgWidth = 4;
                        bldgLength = 4;
                        break;
                case 2: //monjeria
```

```
                        bldgWidth = 25;
                        bldgLength = 12;
                        break;

                case 5: //church
                        bldgWidth = 20;
                        bldgLength = 50;
                        break;

                default:
                        System.out.println("A mother is located in a building type where
                                she shouldn't be in makeMothersGrp");
                        break;

                                }
                int xMin = mothersBldgInfo.initialX-2;
                int xMax = mothersBldgInfo.initialX+(bldgWidth+2);
                int yMin = mothersBldgInfo.initialY-2;
                int yMax = mothersBldgInfo.initialY+(bldgLength+2);

                for (int x = xMin; x < xMax; x++ ){
                        for (int y = yMin; y < yMax; y++){
                                for (Object agentIn : patch.getObjectsAt(x, y)) {
                                        if (agentIn instanceof SDSimplePerson) {
                                                if (((SDSimplePerson) agentIn).dwelling ==
                                                        this.dwelling){
                                mothersGrpList.add((SDSimplePerson) agentIn);
                                                }
/*
 * There may be GhostPerson agents in the building, but they will not be picked up in
 * this loop because they are not SimplePerson agents. The dead SimplePerson agents
 * move to the cemetery at the time they die, so they should no longer be in the
 * building.
 */
                                        }
                                        }
                                        }
                        }
                return mothersGrpList;
                }

/* This method makes a list of all houses that contain members of father's
 * or mother's patrilines.
 *
 * This is used to determine first priority houses for visiting and reassignments.
 */
public void makeKinHouseList(int numHouses, int houseLineList[][]) {

        kinHouseList = new int [numHouses][2];
        for(int nrow=0; nrow<numHouses; nrow++){
        for (int ncol=0; ncol<2; ncol++){
        kinHouseList[nrow][ncol]= -1;
                                }
```

```
                        }

        for (int hRow=0; hRow<numHouses; hRow++){
        for (int mfp = 2; mfp<houseLineList[hRow].length; mfp++ ){
                if((houseLineList[hRow][mfp] == this.fathersPatriline)
                ||(houseLineList[hRow][mfp]== this.mothersPatriline)){
                        kinHouseList [hRow][0] = houseLineList[hRow][0];
                        kinHouseList [hRow][1] = houseLineList [hRow][1];
                        break;
                                }
                        }
                                }
                }

/** The Disease Transmission methods start here */

/*
 * We are assuming a Von Neumann neighborhood rather than the Moore
 * Neighborhood. The Moore neighborhood, with its eight neighbors, causes
 * disease transmission to be unrealistically rapid. In thinking about the
 * process, it seems that a limit of four contacts at a time reflects how
 * many individuals people interact with simultaneously, even in relatively
 * crowded situations. Only infectious agents call checkSuscNeighbors, only
 * susceptible agents call checkInfNeighbors.
 */

@SuppressWarnings("unchecked")
public void checkSuscNeighbors() {
        Iterable list = null; // clear out list before using again
        Iterable callerList = null;
        Iterable neighborList = null;
        int callerBldgID = 0;
        int neighborBldgID =0;
        Context context = ContextUtils.getContext(this);
        Grid patch = (Grid) context.getProjection("MissionGrid");
/*
 * The callerList and neighborList get all agents of any kind (person and building)
 * at the location of the agent calling checkSuscNeighbors or the agent returned
 * by the patch search.  If a building agent is on the node of the person agent, the
 * method sets the variable callerBldgID or neighborBldgID to be the building ID
 * of that building object.  If the agent is not physically "inside" a building
 * the variable stays set at 0.
 *
 * The location of the calling agent (infector) is then compared to the neighbor's
 * (suscAgent) to make sure that transmission doesn't occur through walls.
 *
 * The VNQuery list that is generated by the query includes all objects in the
 * neighborhood, i.e. SimplePerson, SDSimpleBuilding and GhostPerson agents. The loop
 * goes through the generated list, checks to see if the object in question is a
 * SDSimplePerson and, if so, the locations are compared to make sure both agents are
 * in the building or outside. If so TransmissionTo method is called, which then
 * determines if the any of the infectious agent's neighbors are susceptible and, if so,
 * whether transmission occurs. Note: the agent calling this method is known to be
```

```
 * infectious, so transmission is a possibility for any susceptible agent on the list.
 */

callerList = patch.getObjectsAt(this.xLocation, this.yLocation);

for (Object possBuild : callerList) {
		if (possBuild instanceof SDSimpleBuilding){
			callerBldgID = ((SDSimpleBuilding)possBuild).bldgID;
					}
				}
		VNQuery vnq = new VNQuery(patch, this, 1, 1);
		list = vnq.query();// makes an iterable list of the Von Neumann neighbors
		for (Object agentin : list) {//cycles through the new list
			if (agentin instanceof SDSimplePerson) {
				neighborBldgID = 0;
				neighborList = patch.getObjectsAt(((SDSimplePerson)
					agentin).xLocation, ((SDSimplePerson) agentin).yLocation);
				for (Object possBldg : neighborList){
					if (possBldg instanceof SDSimpleBuilding){
						neighborBldgID =
							((SDSimpleBuilding)possBldg).bldgID;
					}
					}
				if (callerBldgID == neighborBldgID){
//if both are outside, both variables will be 0 and transmission is possible
				transmissionTo((SDSimplePerson) agentin);
					}
				}
				neighborList = null;
			}
		}

@SuppressWarnings("unchecked")
public void checkInfNeighbors() {
		Iterable list2 = null;
		Iterable callerList2 = null;
		Iterable neighborList2 = null;
		int callerBldgID2 = 0;
		int neighborBldgID2 = 0;
		Context context = ContextUtils.getContext(this);
		Grid patch = (Grid) context.getProjection("MissionGrid");
/*
 * The callerList and neighborList get all agents of any kind (person and building)
 * at the location of the agent calling checkSuscNeighbors or the agent returned
 * by the patch search.  If a building agent is on the node of the person agent, the
 * method sets the variable callerBldgID or neighborBldgID to be the building ID
 * of that building object.  If the agent is not physically "inside" a building
 * the variable stays set at 0.
 *
 * The location of the calling agent (infector) is then compared to the neighbor's
 * (suscAgent) to make sure that transmission doesn't occur through walls.
 *
 * The VNQuery list that is generated by the query includes all objects in the
```

```
 * neighborhood, i.e. SimplePerson, SDSimpleBuilding and GhostPerson agents. The loop
 * goes through the generated list, checks to see if the object in question is a
 * SDSimplePerson and, if so, the locations are compared to make sure both agents are
 * in the building or outside. If so TransmissionTo method is called, which then
 * determines if the any of the infectious agent's neighbors are susceptible and, if so,
 * whether transmission occurs. Note: the agent calling this method is known to be
 * infectious, so transmission is a possibility for any susceptible agent on the list.
 */
        callerList2 = patch.getObjectsAt(this.xLocation, this.yLocation);
                for (Object possBuild : callerList2) {
                        if (possBuild instanceof SDSimpleBuilding){
                        callerBldgID2 = ((SDSimpleBuilding)possBuild).bldgID;
                                }
                        }

        VNQuery vnq = new VNQuery(patch, this, 1, 1);
        list2 = vnq.query();// makes an iterable list of the Von Neumann neighbors
                for (Object agentin : list2) {//cycles through the new list
                        if (agentin instanceof SDSimplePerson) {
                        neighborBldgID2 = 0;
                        neighborList2 = patch.getObjectsAt(((SDSimplePerson)
                                agentin).xLocation, ((SDSimplePerson) agentin).yLocation);
                        for (Object possBldg2 : neighborList2){
                                if (possBldg2 instanceof SDSimpleBuilding){
                                neighborBldgID2 = ((SDSimpleBuilding)possBldg2).bldgID;
                                        }
                                        }
                                if (callerBldgID2 == neighborBldgID2){
                                transmissionFrom((SDSimplePerson) agentin);
                                        }
                                }
                                neighborList2 = null;
                        }
                }
/*
 * The above method is analogous to the checkSuscNeighbors. In this case, the
 * loop goes through the generated list, checks to see if the object in
 * question is a SDSimplePerson and, if so, calls the TransmissionFrom method,
 * which then determines if the any of the susceptible agent's neighbors are
 * infectious and, if so, whether transmission occurs. Note: the agent calling
 * this method is known to be susceptible.
 */

/*
 * updateDiseaseStatus updates the numerical value for agent's disStatus and
 * establishes or changes the length of the period corresponding to that
 * status.
 */
private void updateDiseaseStatus(ArrayList<ArrayList<SDSimpleBuilding>>
        masterBuildings) {

        Parameters p = RunEnvironment.getInstance().getParameters();
        switch (disStatus) {
```

```
                case 0:
                        break;
                case 1:
                        if (this.timeToInfectious == 0) {
                                this.disStatus = 2;
                                this.timeToRecovery = (Integer)
                                        p.getValue("lengthofinfectiousperiod")-1;
                        } else {
                                this.timeToInfectious--;
                        }
                        break;
                case 2:
                        double deathThreshold = RandomHelper.nextDoubleFromTo(0,
                                1);
                        if (deathThreshold < this.deathProb){
//If the randomly chosen number is below the assigned death prob,the agent dies
                                this.disStatus = 4;
                                this.timeOfDeath = (int)
                                        RunEnvironment.getInstance().getCurrentSchedule
                                        ().getTickCount();
                                this.placeOfDeath = SDSimplePerson.atLocationList
                                        [this.presentLocation][0];
                                this.createGhostAgent(this.agtID, this.fathersPatriline,
                                        this.disStatus, this.dwelling,this.mothersPatriline,
                                        this.extFamily,this.sex, this.age,
                                        this.spouse,this.hlthHist,this.occupation,
                                        this.presentLocation,this.xLocation, this.yLocation,
                                        this.timeOfDeath, this.placeOfDeath,
                                        masterBuildings);
                                this.familyChangesUponDeath(this, masterBuildings);
                                } else if (this.timeToRecovery == 0) {
                                        this.disStatus = 3;
                                } else {
                                        this.timeToRecovery--;
                                }
                                break;
                case 3:
                        this.disStatus = 3;
                        break;
                case 4:
                        break;
                }
        }

/*
 * transmissionTo and transmissionFrom assigns a random probability of
 * transmission to a contact between an infectious and a susceptible agent.
 * The agents must either be in the same building both outside of buildings in
 * order to complete a contact. von Neumann neighbors in different buildings
 * do not have the opportunity to transmit disease to each other.
 * When such a contact occurs, the method compares the parameter value of
 * the transmission probability to the assigned probability. If the transmission
 * probability is greater than or equal to the assigned probability, then disease
```

```
 * transmission occurs and the susceptible agent moves into the exposed
 * state. The length of time in the exposed class is determined by the
 * parameter "lengthoflatentperiod". If the transmission probability parameter is
 * less than the assigned probability, the susceptible agent does not change
 * disease status.
 *
 */

public void transmissionTo(SDSimplePerson suscAgent) {

            Parameters p = RunEnvironment.getInstance().getParameters();
            double beta = (Double) p.getValue("transmissionprobability");
            int latent = (Integer) p.getValue("lengthoflatentperiod");
             double prob = 0;
             prob = RandomHelper.nextDoubleFromTo(0, 1);
             if (suscAgent.disStatus == 0 && prob <= beta) {
                        suscAgent.disStatus = 1;
                        suscAgent.timeToInfectious = latent;
                        suscAgent.timeInfected = (int) RunEnvironment.getInstance()
                                    .getCurrentSchedule().getTickCount();
                        suscAgent.placeInfected = SDSimplePerson.atLocationList
                                    [suscAgent.presentLocation][0];
                        suscAgent.infectorID = this.agtID;
                        suscAgent.infectorOcc = this.occupation;
                        suscAgent.infectorPatriline = this.fathersPatriline;
                        suscAgent.infectorMatriline = this.mothersPatriline;
                        suscAgent.infectorDwelling = this.dwelling;
                        suscAgent.newlyInfected = true;
                        if(suscAgent.stepCompleted == true){suscAgent.timeToInfectious--;
                                                }
                                                }
                        }

private void transmissionFrom(SDSimplePerson infAgent) {

            Parameters p = RunEnvironment.getInstance().getParameters();
             double beta = (Double) p.getValue("transmissionprobability");
             int latent = (Integer) p.getValue("lengthoflatentperiod");
            double prob = 0;
             prob = RandomHelper.nextDoubleFromTo(0, 1);
                        if (infAgent.disStatus == 2 && prob<= beta && this.newlyInfected ==
                        false) {
                                    this.disStatus = 1;
                                    this.timeToInfectious = latent - 1;
                                    this.timeInfected = (int) RunEnvironment.getInstance()
                                                .getCurrentSchedule().getTickCount();
                                    this.placeInfected = SDSimplePerson.atLocationList
                                                [this.presentLocation][0];
                                    this.infectorID = infAgent.agtID;
                                    this.infectorOcc = infAgent.occupation;
                                    this.infectorPatriline = infAgent.fathersPatriline;
                                    this.infectorMatriline = infAgent.mothersPatriline;
                                    this.infectorDwelling = infAgent.dwelling;
```

```
                        this.newlyInfected = true;
                                }
                }

@SuppressWarnings("unchecked")
/* The createGhostAgent method is called by a SimplePerson Agent who has died.  It
 * creates a new GhostPerson Agent with  the same attributes as the calling agent.  The
 * calling agent is then moved to the cemetery  (coordinates of 0, 0). The GhostPerson
 * agent is added to the context at the location where the SimplePerson agent died.
 */

public void createGhostAgent(int agentID, int fPat, int diseaseStatus, int agentHouse, int
        mPat, int extFam, String agentSex, double agentAge, int spouseID,double
        healthHistory, int agentWork, int presentLoc, int xLoc, int yLoc, int timeDeath, int
        placeDeath, ArrayList<ArrayList<SDSimpleBuilding>> buildingList){

         SDGhostPerson ghost = new SDGhostPerson(agentID, fPat, diseaseStatus,
                agentHouse, mPat, extFam, agentSex,agentAge, spouseID, healthHistory,
                agentWork, presentLoc, xLoc, yLoc, timeDeath, placeDeath);

                Context context = ContextUtils.getContext(this);
                Grid patch = (Grid) context.getProjection("MissionGrid");
                context.add(ghost);
                patch.moveTo(ghost, ghost.xLocation, ghost.yLocation);

//System.out.println("Agent: "+this.agtID+" Ghost: "+ghost.agtID+" is now a ghost");
        SDSimplePerson.atLocationList[this.presentLocation][1]--;
                        SanDiegoMissionContext.permanentResidents[this.mblLocator(this.dwelling,
                buildingList)][1]--;

         this.newlyDead = true;
         this.xLocation = 0;
         this.yLocation = 0;
         patch.moveTo(this, this.xLocation,this.yLocation);

                }
/* The familyChangesUponDeath method is called by any dying agent.
 * If the dying agent is an adult, it makes sure that any children
 * dependent on it are dealt with appropriately (see below for specifics).
 *
 * If the dying agent is a married male, he makes sure his widow and young and/or
 * female children are moved to the monjeria.
 */
@SuppressWarnings({ "unchecked"})
public void familyChangesUponDeath(SDSimplePerson person,
        ArrayList<ArrayList<SDSimpleBuilding>> buildingList){

        int counter = 1;
        Context context = ContextUtils.getContext(this);
        Iterable listAgentsFC = null;
        ArrayList liveAgentsFC = new ArrayList <SDSimplePerson>();
        ArrayList residentList = new ArrayList <SDSimplePerson>();
        ArrayList toMonList = new ArrayList <SDSimplePerson>();
```

```
		liveAgentsFC.clear();
		residentList.clear();
		toMonList.clear();
		InstanceOfQuery <SDSimplePerson> allAgentsFC = new
			InstanceOfQuery<SDSimplePerson>(context, SDSimplePerson.class);
//listAgents is a list of all SimplePersonAgents in the context, including both dead and alive
		listAgentsFC = allAgentsFC.query();
//liveAgents is a list populated by only living agents
		for (Object agentin : listAgentsFC){
			if (((SDSimplePerson)agentin).disStatus !=4) {
				liveAgentsFC.add((SDSimplePerson)agentin);
			}}
		Collections.shuffle(liveAgentsFC);
//System.out.println("Dying agent: "+this.agtID+" occupation: "+this.occupation+" tick: "+(int)
RunEnvironment.getInstance().getCurrentSchedule().getTickCount()+" Dying agent's dwelling:
"+this.dwelling);
/* Upon the death of an adult, the following loop determines whether they have live
 * children in the household and if so, it identifies possible caretakers for those children.
 */
		if (this.occupation<400){
			this.findLiveChildren(liveAgentsFC);
			this.caretakerTest(liveAgentsFC, buildingList);
			}
/* If the dying agent is a married male, the following loop determines whether
 * they have widow present, and depending on that assessment, the reassignment of
 * children is decided (taking into account their age and sex, etc).
 */
		if (this.occupation == 100){
			for (Object widow : liveAgentsFC){
				if (((SDSimplePerson)widow).agtID == this.spouse) {
//System.out.println("Agent "+((SDSimplePerson)widow).agtID+" is the spouse of agent "+this.agtID+".
Dwelling: "+((SDSimplePerson)widow).dwelling+", Present Location:
"+((SDSimplePerson)widow).atLocationList[((SDSimplePerson)widow).presentLocation][0]);
		if(this.motherFound==false && ((SDSimplePerson)widow).age<45){
                                                        residentList.add((SDSimplePerson)widow);
//System.out.println("Young widow with no kids who needs to go to monjeria");
		break;
                                        }
		else if (this.motherFound==true) {
			if (SDSimplePerson.atLocationList
				[((SDSimplePerson)widow).presentLocation][0] ==
				((SDSimplePerson)widow).dwelling) {
                                                residentList = ((SDSimplePerson)widow).
                                                        makeMothersGrp(xLocation, yLocation, person,
                                                        buildingList, numHouseSP, hList);
                                                                }
                                                else residentList = ((SDSimplePerson)widow).
                                                        makeTravGrp(xLocation, yLocation, person,
                                                        buildingList, numHouseSP, hList);
		//System.out.println("widow with kids");
                                                                break;
                                        }}}
		for (Object newMonRes : residentList){
```

```
                    if ((((SDSimplePerson)newMonRes).occupation == 600
                            ||((SDSimplePerson)newMonRes).occupation == 700
                            ||((SDSimplePerson)newMonRes).agtID == this.spouse)
                            &&((SDSimplePerson)newMonRes).dwelling==this.dwelling){
                                    toMonList.add((SDSimplePerson)newMonRes);
                                            }
                                    }
            for (Object moverToMon : toMonList){
                    SanDiegoMissionContext.permanentResidents[
                            ((SDSimplePerson)moverToMon).mblLocator(((SDSimplePerson)m
                            overToMon).dwelling, buildingList)][1]--;

                    ((SDSimplePerson)moverToMon).dwelling = 171;
                    if (((SDSimplePerson)moverToMon).age>3) {

                    ((SDSimplePerson)moverToMon).occupation = 400;
                    ((SDSimplePerson)moverToMon).occType = 400;
                                            }
//System.out.println("Agent: "+ ((SDSimplePerson)moverToMon).agtID+" has been reassigned to the
monjeria. Its occupation is now: "+((SDSimplePerson)moverToMon). occupation+" counter: "+counter);

counter++;

SanDiegoMissionContext.permanentResidents[((SDSimplePerson)moverToMon).mblLoc
            ator(((SDSimplePerson)moverToMon).dwelling, buildingList)][1]++;
                                    }
                    this.findLiveChildren(liveAgentsFC);
                    this.caretakerTest(liveAgentsFC, buildingList);
                    }
                    }

/*findLiveChildren determines whether there are any living children in a household and
 * if so whether any of those children are under 4 years old
 */
public void findLiveChildren(ArrayList<SDSimplePerson> possAgents){
            this.childrenAtHome=false;
            this.childrenUnderFour=false;

            for (Object possChild : possAgents){
                    if (((SDSimplePerson)possChild).dwelling == this.dwelling
                            &&(((SDSimplePerson)possChild).occupation==700)){

                    this.childrenAtHome=true;
                    this.childrenUnderFour=true;
                            }
                    else if (((SDSimplePerson)possChild).dwelling == this.dwelling
                            &&((((SDSimplePerson)possChild).occupation==500)
                            ||(((SDSimplePerson)possChild).occupation==600))){
                                    this.childrenAtHome=true;
                            }
                            }
//Leave on for testing purposes
/*System.out.println("Calling Agent: "+this.agtID+" Household: "+this.dwelling+"
```

```
 * Children at Home: "+this.childrenAtHome+
 * " Children Under Four: "+this.childrenUnderFour);*/
		}

/* caretakerTest looks for potential replacement caretakers when an agent dies.  It first
 * looks for females, then possible males.  If no caretaker is available, any live children
 * are sent to the orphanage.
*/
public void caretakerTest(ArrayList<SDSimplePerson> possAgents,
		ArrayList<ArrayList<SDSimpleBuilding>> buildingList){

		if (this.childrenAtHome == true){
			this.findPossFemCarer(possAgents);

		if (this.motherFound==false && this.otherFemaleFound == false){
			this.reassignOrphans(possAgents, buildingList);
			}
			}
		}

/* findPossFemCarer first checks to see if the mother of the household is present
 * (OccType 200, age <45). She will be known as the "mother". If a "mother" is not
 * found, another female (by definition over 45), will be tapped to be the caretaker.
 */

public void findPossFemCarer(ArrayList<SDSimplePerson> possAgents){
		this.motherFound = false;
		this.otherFemaleFound = false;

		for (Object femalePossCaretaker : possAgents){
			if (((SDSimplePerson)femalePossCaretaker).dwelling == this.dwelling
				&&((SDSimplePerson)femalePossCaretaker).occupation==200){

				if (((SDSimplePerson)femalePossCaretaker).age<45){
					this.motherFound = true;
				}
				else{
					this.otherFemaleFound = true;
				}
			}
			}
		}

/* The findPossMaleCarer method is called only by boys and identifies any adult males
 * in a household who may be able to step in as caretakers (for boys) if no female
 * caretakers are available.
*/

public void findPossMaleCarer(ArrayList<SDSimplePerson> possAgents){
		this.maleCaretakerFound = false;

		for (Object malePossCaretaker : possAgents){
			if (((SDSimplePerson)malePossCaretaker).dwelling == this.dwelling
```

```
                        &&(((SDSimplePerson)malePossCaretaker).occupation==100||

                        ((SDSimplePerson)malePossCaretaker).occupation==300)) {
                this.maleCaretakerFound = true;
                        }
                }
        }

/* Reassigns orphaned children to a kin house, if possible, or to another suitable
 * dwelling, as determined by the findNewDwelling method. Called by a dying agent to
 * deal with the need to move children out of the house. If appropriate adults are still
 * assigned to the house, nothing needs to be done.
 */

public void reassignOrphans(ArrayList<SDSimplePerson> possAgents,
        ArrayList<ArrayList<SDSimpleBuilding>> buildingList){

        int groupNum = 1;
        for (Object child : possAgents){
                if (((SDSimplePerson)child).dwelling == this.dwelling){
//leave for testing
//      System.out.println ("Household " + this.dwelling + " has no potential caretakers.
        All kids must be reassigned.");

        if (((SDSimplePerson)child).occupation==500){
                this.findPossMaleCarer(possAgents);
                if (this.maleCaretakerFound == true){
//Do nothing
//System.out.println("1.A male caretaker has been found in the house: "+this.dwelling+"
        Agent: "+((SDSimplePerson)child).agtID+" is still in dwelling"
        +((SDSimplePerson)child).dwelling);
                        } else ((SDSimplePerson)child).findNewDwelling(((SDSimplePerson)child),
                        numHouseSP, hLineList, groupNum, buildingList);
//System.out.println("2. The agent: "+((SDSimplePerson)child).agtID+" has been
        reassigned to a new dwelling: "+((SDSimplePerson)child).dwelling);
                                }
        if (((SDSimplePerson)child).occupation==600||((SDSimplePerson)child)
                .occupation==700){

        ((SDSimplePerson)child).findNewDwelling(((SDSimplePerson)child),
                numHouseSP, hLineList, groupNum, buildingList);

//System.out.println("2. The agent: "+((SDSimplePerson)child).agtID+" has been
        reassigned to a new dwelling: "+((SDSimplePerson)child).dwelling);
                                }

        SDSimplePerson.atLocationList[((SDSimplePerson)child).presentLocation][1]--;
        ((SDSimplePerson)child).moveHome(((SDSimplePerson)child).dwelling,
                ((SDSimplePerson)child).xLocation, ((SDSimplePerson)child).yLocation,
                ((SDSimplePerson)child), buildingList, numHouseSP);

        SDSimplePerson.atLocationList[((SDSimplePerson)child).presentLocation][1]++;
                        }
```

```
                    }
                    if (this.maleCaretakerFound==true){
                              this.childrenAtHome = true;
                    }else {
                    this.childrenAtHome = false;
                    }
//This indicates that all children assiged to the calling agent's house have been sent to
          new homes
                    }

/** Data Recording methods start here */
/*
 * These check methods are called by whatever code (built-in) sets up the data sets
 * when the model is run. They are called after the end of each step, and we have
 * determined that the methods cycle through each agent in numerical order. Some of
 * the methods collect particular daily or total values of agents that we have deemed
 * important for our analyses. The code indicates what is being collected.
 *
 * The methods following "checkEpiOver" record the values of parameters during
 * simulations and data related to when, where and by whom agents are
 * infected. These methods allow us to display the parameter values when
 * running batch simulations, which makes it easier both for keeping track
 * of each type of simulation and for subsequent analyses of the data.
 *
 * None of the methods need to be called within the step method or elsewhere
 * in the Java files; they are called during the model run itself.
 */

public int checkSusc() {
          int numberSusc = 0;
          if (disStatus == 0)
          numberSusc++;
          return numberSusc;
                    }
public int checkDailyInfections() {
          int numberNewInfection = 0;
          if (newlyInfected == true)
          numberNewInfection++;
          return numberNewInfection;
                    }

public int checkExp() {
          int numberExp = 0;
          if (disStatus == 1)
                    numberExp++;
                    return numberExp;
                    }

public int checkInf() {
          int numberInf = 0;
          if (disStatus == 2)
                    numberInf++;
                    return numberInf;
```

```
		}

public int checkRec() {
		int numberRec = 0;
		if (disStatus == 3)
			numberRec++;
			return numberRec;
			}

public int checkDailyDead(){
		int numberNewDead = 0;
		if (newlyDead == true){
			numberNewDead++;
			}
			return numberNewDead;
			}
public int checkTotDead(){
		int totNumDead = 0;
		if (disStatus == 4){
		totNumDead++;}
		return totNumDead;
			}
public int checkEpiSize() {
		int epiSize = 0;
		if (disStatus == 3 || disStatus == 4)
			epiSize++;
			return epiSize;
			}
public int checkEpiOver() {
		int epiOver = 0;
		if (disStatus == 0 || disStatus == 3 || disStatus == 4)
			epiOver++;
			return epiOver;
			}
public double checkBeta() {
		Parameters p = RunEnvironment.getInstance().getParameters();
		double trans = (Double) p.getValue("transmissionprobability");
			return trans;
			}
public int checkInfPeriod() {
		Parameters p = RunEnvironment.getInstance().getParameters();
		int infPd = (Integer) p.getValue("lengthofinfectiousperiod");
			return infPd;
			}
public int checkLatPeriod() {
		Parameters p = RunEnvironment.getInstance().getParameters();
		int latPd = (Integer) p.getValue("lengthoflatentperiod");
			return latPd;
			}
public double checkMortProbability() {
		Parameters p = RunEnvironment.getInstance().getParameters();
		double mortProb = (Double) p.getValue("probabilityofdeath");
			return mortProb;
```

```
		}
public static int checkTotalPop() {
	int totalPop = SanDiegoMissionContext.getTotPop();
		return totalPop;
		}
public int checkFirstCaseID() {
	int firstCaseID = getFirstCase();
		return firstCaseID;
		}
public int checkFirstCaseOcc() {
	int firstCaseOcc = getFirstCaseOccCode();
		return firstCaseOcc;
		}
public int checkPersonIDNo() {
	int personIDNo = getPersonID();
		return personIDNo;
		}
public int checkPersonSpouse() {
	int personSpouse = getSpouse();
		return personSpouse;
		}
public int checkPersonMatriline() {
	int personMatriline = getMothersPatriline();
		return personMatriline;
		}
public int checkPersonPatriline() {
	int personPatriline = getFathersPatriline();
		return personPatriline;
		}
public int checkPersonOccCode() {
	int personOccCode = getPersonOcc();
		return personOccCode;
		}
public int checkPersonDwellingNo() {
	int personDwellingNo = getPersonDwelling();
		return personDwellingNo;
		}
public int checkTimeInf() {
	int timeInf = getTimeInfected();
		return timeInf;
		}
public int checkPlaceInf() {
	int placeInf = getPlaceInfected();
		return placeInf;
		}
public int checkInfector() {
	int infector = getInfectorID();
		return infector;
		}
public int checkInfectorOccCode() {
	 int infectorOccCode = getInfectorOcc();
		return infectorOccCode;
		}
```

```
public int checkInfectorMatriline() {
		int infectorsMatriline = getInfectorMatriline();
			return infectorsMatriline;
			}
public int checkInfectorPatriline() {
		int infectorsPatriline = getInfectorPatriline();
			return infectorsPatriline;
			}
public int checkInfectorDwellingNo() {
		int infectorDwellingNo = getInfectorDwelling();
			return infectorDwellingNo;
			}
public int checkTimeDeath() {
		int timeDeath = getTimeOfDeath();
			return timeDeath;
			}
public int checkPlaceDeath() {
		int placeDeath = getPlaceOfDeath();
			return placeDeath;
			}

/**Get and Set methods are below*/

public int getAgtID() {
		return agtID;
			}
public void setAgtID(int agtID) {
		this.agtID = agtID;
			}
public int getFathersPatriline() {
		return fathersPatriline;
			}
public void setFathersPatrline(int fathersPatriline) {
		this.fathersPatriline = fathersPatriline;
			}
public int getdiseaseStatus() {
		return disStatus;
			}
public void setdiseaseStatus(int disStatus) {
		this.disStatus = disStatus;
			}
public int getDwelling() {
		return dwelling;
			}
public void setDwelling(int dwelling) {
		this.dwelling = dwelling;
			}
public int getMothersPatriline() {
		return mothersPatriline;
			}
public void setMothersPatriline(int mothersPatriline) {
		this.mothersPatriline = mothersPatriline;
			}
```

```
public int getExtFamily() {
            return extFamily;
                        }
public void setExtFamily(int extFamily) {
            this.extFamily = extFamily;
                        }
public String getSex() {
            return sex;
                        }
public void setSex(String sex) {
            this.sex = sex;
                        }
public double getAge() {
            return age;
                        }
public void setAge(double age) {
            this.age = age;
                        }
public int getSpouse() {
            return spouse;
                        }
public void setSpouse(int spouse) {
            this.spouse = spouse;
                        }
public double getHlthHist() {
            return hlthHist;
                        }
public void setHlthHist(double hlthHist) {
            this.hlthHist = hlthHist;
                        }
public int getOccupation() {
                        return occupation;
                        }
public void setOccupation(int occupation) {
                        this.occupation = occupation;
                        }
public int getPresentLocation() {
                        return presentLocation;
                        }
public void setPresentLocation(int presentLocation) {
                        this.presentLocation = presentLocation;
                        }
public static int getFirstCase() {
                        return firstCase;
                        }
public static void setFirstCase(int firstCase) {
                        SDSimplePerson.firstCase = firstCase;
                        }
public static int getFirstCaseOccCode() {
                        return firstCaseOccCode;
                        }
public static void setFirstCaseOccCode(int firstCaseOccCode) {
                        SDSimplePerson.firstCaseOccCode = firstCaseOccCode;
```

```
                    }
public int getPersonID() {
                    return personID;
                    }
public void setPersonID(int personID) {
                    this.personID = personID;
                    }
public int getPersonOcc() {
                    return personOcc;
                    }
public void setPersonOcc(int personOcc) {
                    this.personOcc = personOcc;
                    }
public int getPersonDwelling() {
                    return personDwelling;
                    }
public void setPersonDwelling(int personDwelling) {
                    this.personDwelling = personDwelling;
                    }
public int getTimeInfected() {
          return timeInfected;
                    }
public void setTimeInfected(int timeInfected) {
          this.timeInfected = timeInfected;
                    }
public int getPlaceInfected() {
          return placeInfected;
                    }
public void setPlaceInfected(int placeInfected) {
          this.placeInfected = placeInfected;
                    }
public int getInfectorID() {
          return infectorID;
                    }
public void setInfectorID(int infectorID) {
          this.infectorID = infectorID;
                    }
public int getInfectorOcc() {
          return infectorOcc;
                    }
public void setInfectorOcc(int infectorOcc) {
          this.infectorOcc = infectorOcc;
                    }
public int getInfectorMatriline() {
          return infectorMatriline;
                    }
public void setInfectorMatriline(int infectorMatriline) {
          this.infectorMatriline = infectorMatriline;
                    }
public int getInfectorPatriline() {
          return infectorPatriline;
                    }
public void setInfectorPatriline(int infectorPatriline) {
```

```
            this.infectorPatriline = infectorPatriline;
                        }
public int getInfectorDwelling() {
            return infectorDwelling;
                        }
public void setInfectorDwelling(int infectorDwelling) {
             this.infectorDwelling = infectorDwelling;
                        }
public int getTimeOfDeath() {
             return timeOfDeath;
                        }
public int getPlaceOfDeath() {
             return placeOfDeath;
                        }
public void setTimeOfDeath(int timeOfDeath) {
             this.timeOfDeath = timeOfDeath;
                        }
public void setPlaceOfDeath(int placeOfDeath) {
             this.placeOfDeath = placeOfDeath;
                        }
            }
```

APPENDIX B – MODEL INPUT FILES

The two files contained in this appendix are model input files that describe the physical layout and population attributes necessary to customize the model. The files below were used in the model and described in the text of this dissertation. However, users wanting to model disease in a different population or different setting can create their own files and read them into the model. The files are read in as .txt files, in order to conform to expectations laid out in the code for the model.

Building Definition File

The Building Defs file contains information about all the buildings that are to be located on the model's physical space. This includes building ID number, dimensions and building type. The construction of this file is described in Chapter 8.

Building ID	Building Type	Length	Width	Starting X Coordinate	Starting Y Coordinate
1	1	4	4	82	37
2	1	4	4	60	57
3	1	4	4	65	77
4	1	4	4	87	62
5	1	4	4	68	121
6	1	4	4	67	64
7	1	4	4	126	94
8	1	4	4	66	71
9	1	4	4	76	52
10	1	4	4	71	58
11	1	4	4	106	103
12	1	4	4	61	68
13	1	4	4	71	127
14	1	4	4	138	62
15	1	4	4	107	51
16	1	4	4	84	83
17	1	4	4	105	87
18	1	4	4	117	116
19	1	4	4	72	85
20	1	4	4	121	96
21	1	4	4	81	120
22	1	4	4	107	56
23	1	4	4	129	50
24	1	4	4	75	68
25	1	4	4	61	127
26	1	4	4	126	102
27	1	4	4	123	118
28	1	4	4	132	88
29	1	4	4	65	50
30	1	4	4	65	108
31	1	4	4	108	129
32	1	4	4	116	67
33	1	4	4	73	74
34	1	4	4	74	115
35	1	4	4	77	101
36	1	4	4	112	115
37	1	4	4	138	71
38	1	4	4	91	79
39	1	4	4	89	55
40	1	4	4	84	100
41	1	4	4	94	52

42	1	4	4	90	74
43	1	4	4	91	118
44	1	4	4	100	118
45	1	4	4	80	72
46	1	4	4	61	121
47	1	4	4	107	74
48	1	4	4	112	123
49	1	4	4	62	115
50	1	4	4	77	58
51	1	4	4	133	106
52	1	4	4	111	68
53	1	4	4	113	105
54	1	4	4	121	69
55	1	4	4	91	102
56	1	4	4	69	102
57	1	4	4	86	117
58	1	4	4	103	69
59	1	4	4	90	123
60	1	4	4	108	108
61	1	4	4	86	93
62	1	4	4	122	50
63	1	4	4	71	53
64	1	4	4	96	124
65	1	4	4	100	118
66	1	4	4	110	79
67	1	4	4	113	92
68	1	4	4	123	129
69	1	4	4	96	113
70	1	4	4	117	85
71	1	4	4	89	67
72	1	4	4	102	108
73	1	4	4	135	98
74	1	4	4	128	81
75	1	4	4	78	108
76	1	4	4	106	97
77	1	4	4	66	97
78	1	4	4	85	76
79	1	4	4	96	129
80	1	4	4	94	86
81	1	4	4	81	51
82	1	4	4	128	115
83	1	4	4	137	93
84	1	4	4	106	62

85	1	4	4	122	82
86	1	4	4	121	63
87	1	4	4	99	87
88	1	4	4	79	113
89	1	4	4	96	100
90	1	4	4	103	92
91	1	4	4	119	124
92	1	4	4	88	111
93	1	4	4	96	108
94	1	4	4	132	74
95	1	4	4	120	77
96	1	4	4	110	85
97	1	4	4	83	67
98	1	4	4	130	56
99	1	4	4	101	81
100	1	4	4	84	56
101	1	4	4	112	97
102	1	4	4	99	58
103	1	4	4	86	50
104	1	4	4	75	90
105	1	4	4	132	67
106	1	4	4	136	129
107	1	4	4	60	78
108	1	4	4	62	84
109	1	4	4	100	52
110	1	4	4	116	110
111	1	4	4	133	83
112	1	4	4	102	128
113	1	4	4	97	95
114	1	4	4	80	95
115	1	4	4	82	61
116	1	4	4	133	61
117	1	4	4	113	55
118	1	4	4	63	90
119	1	4	4	90	128
120	1	4	4	133	120
121	1	4	4	75	63
122	1	4	4	89	87
123	1	4	4	101	113
124	1	4	4	101	102
125	1	4	4	138	83
126	1	4	4	70	80
127	1	4	4	135	56

128	1	4	4	129	125
129	1	4	4	134	51
130	1	4	4	133	114
131	1	4	4	137	77
132	1	4	4	68	114
133	1	4	4	71	96
134	1	4	4	60	73
135	1	4	4	69	91
136	1	4	4	113	50
137	1	4	4	67	85
138	1	4	4	120	102
139	1	4	4	113	60
140	1	4	4	92	92
141	1	4	4	64	103
142	1	4	4	102	76
143	1	4	4	128	108
144	1	4	4	96	75
145	1	4	4	127	74
146	1	4	4	121	58
147	1	4	4	137	88
148	1	4	4	65	55
149	1	4	4	96	80
150	1	4	4	126	68
151	1	4	4	97	69
152	1	4	4	84	106
153	1	4	4	78	78
154	1	4	4	84	125
155	1	4	4	131	93
156	1	4	4	116	72
157	1	4	4	115	80
158	1	4	4	71	107
159	1	4	4	106	117
160	1	4	4	127	62
161	1	4	4	126	87
162	1	4	4	100	64
163	1	4	4	121	111
164	1	4	4	61	62
165	1	4	4	107	123
166	1	4	4	73	122
167	1	4	4	118	90
168	1	4	4	114	129
169	1	4	4	93	61
170	1	4	4	79	85

171	2	25	12	10	43
172	4	10	15	10	26
173	3	2	2	10	41
174	3	2	2	18	41
175	5	28	54	35	1
176	6	150	10	0	140
177	7	6	2	12	41
178	1	4	4	105	37
179	1	4	4	69	43
180	1	4	4	74	46
181	1	4	4	52	57
182	1	4	4	92	47
183	1	4	4	108	42
184	1	4	4	115	44
185	1	4	4	52	71
186	1	4	4	55	79
187	1	4	4	60	97
188	1	4	4	60	109
189	1	4	4	81	45
190	1	4	4	98	46
191	1	4	4	123	43
192	1	4	4	135	46
193	1	4	4	140	53
194	1	4	4	143	59
195	1	4	4	138	103
196	1	4	4	139	109
197	1	4	4	124	123
198	1	4	4	142	125
199	1	4	4	142	131
200	1	4	4	55	66
201	1	4	4	56	85
202	1	4	4	57	91
203	1	4	4	58	104
204	1	4	4	78	125
205	1	4	4	80	130
206	1	4	4	87	42
207	1	4	4	128	44
208	1	4	4	85	132
209	1	4	4	130	130
210	1	4	4	81	90
211	1	4	4	139	119
212	1	4	4	140	114
213	1	4	4	76	40

214	1	4	4	98	38
215	1	4	4	103	44
216	1	4	4	114	39
217	1	4	4	132	37
218	1	4	4	53	101
219	1	4	4	54	117
220	1	4	4	66	130
221	1	4	4	54	111
222	1	4	4	54	96
223	1	4	4	50	89
224	1	4	4	48	127
225	1	4	4	44	118
226	1	4	4	45	106
227	1	4	4	72	133
228	1	4	4	47	63
229	1	4	4	46	72
230	1	4	4	91	33
231	1	4	4	92	40
232	1	4	4	110	34
233	1	4	4	119	35
234	1	4	4	124	38
235	1	4	4	49	78
236	1	4	4	53	123
237	1	4	4	45	111
238	1	4	4	42	123
239	1	4	4	41	97
240	1	4	4	43	91
241	1	4	4	42	86
242	1	4	4	47	83
243	1	4	4	38	68
244	1	4	4	126	32
245	1	4	4	48	94
246	1	4	4	49	116
247	1	4	4	41	77

Agent Definition File

The Agent Defs file contains all necessary information about the agents that are to populate the model. This includes ID, family and lineage information, age and sex

information, and variables that can be used to designate differences in health status between individuals. The construction of this file is described in Chapter 8.

Agent ID	Father's Patriline	Disease Status	Dwelling	Mother's Patriline	Extended Family	Sex	Age	Spouse	Hlth Hist	Occupation
89	13	0	1	14	0	M	29	90	0	100
90	5	0	1	13	0	F	27	89	0	200
87	13	0	1	5	0	M	6	0	0	500
6722	13	0	1	5	0	M	3	0	0	700
93	5	0	2	2	0	M	51	8381	0	100
8381	13	0	2	15	0	F	44	93	0	200
6594	5	0	2	13	0	M	34	6595	0	100
7677	5	0	2	20	0	F	8	0	0	600
6595	20	0	2	5	0	F	19	6594	0	200
95	13	0	3	4	0	M	27	7491	0	100
7491	5	0	3	10	0	F	20	95	0	200
132	4	0	4	9	0	M	29	133	0	100
133	7	0	4	6	0	F	24	132	0	200
6167	4	0	4	7	0	F	0	0	0	700
150	11	0	5	3	0	F	41	0	0	200
8659	20	0	5	11	0	M	18	0	0	300
8051	20	0	5	11	0	M	13	0	0	300
7906	20	0	5	11	0	F	10	0	0	600
6992	20	0	5	11	0	M	6	0	0	500
148	20	0	6	11	0	F	23	147	0	200
147	6	0	6	7	0	M	27	148	0	100
6776	6	0	6	20	0	M	4	0	0	500
3084	6	0	6	20	0	F	0	0	0	700
167	19	0	7	11	0	M	42	168	0	100
168	12	0	7	20	0	F	24	167	0	200
170	20	0	7	11	0	F	60	0	0	200
9143	5	0	7	20	0	M	35	0	0	300
9144	5	0	7	20	0	M	33	0	0	300
6365	5	0	7	20	0	M	37	6364	0	100
6364	17	0	7	2	0	F	34	6365	0	200
8172	5	0	7	17	0	M	15	0	0	300
1976	17	0	8	20	0	M	51	0	0	300
8727	17	0	8	2	0	M	18	0	0	300
5245	15	0	9	9	0	M	43	6854	0	100
6854	17	0	9	2	0	F	31	5245	0	200
7957	15	0	171	17	0	F	11	0	0	400
5195	17	0	8	2	0	M	26	6517	0	100

6517	14	0	8	2	0	F	24	5195	0	200
6518	17	0	8	14	0	F	2	0	0	700
6516	14	0	10	6	0	M	47	7	0	100
6662	2	0	10	8	0	F	48	6516	0	200
8571	14	0	10	2	0	M	19	0	0	300
5054	6	0	171	3	0	F	19	0	0	400
2082	15	0	9	3	0	M	55	2083	0	100
2083	9	0	9	8	0	F	53	2082	0	200
8456	12	0	9	3	0	F	57	0	0	200
8688	17	0	9	12	0	M	17	0	0	300
3318	7	0	11	9	0	M	34	3317	0	100
3317	15	0	11	9	0	F	30	3318	0	200
3319	7	0	11	15	0	M	11	0	0	500
176	17	0	12	15	0	M	31	6231	0	100
6231	6	0	12	3	0	F	25	176	0	200
174	17	0	12	6	0	M	9	0	0	500
219	17	0	16	6	0	F	50	0	0	200
2547	5	0	13	16	0	M	33	2549	0	100
2549	6	0	13	18	0	F	29	2547	0	200
7716	5	0	13	6	0	M	8	0	0	500
2546	5	0	13	6	0	M	4	0	0	500
2550	17	0	12	12	0	M	63	0	0	300
8461	5	0	14	17	0	F	31	6141	0	200
6141	11	0	14	19	0	M	35	8461	0	100
8699	11	0	14	12	0	M	24	0	0	300
6143	11	0	14	12	0	M	20	6144	0	100
6144	12	0	14	17	0	F	19	6143	0	200
6145	11	0	14	12	0	M	0	0	0	700
217	5	0	15	17	0	F	27	216	0	200
216	13	0	15	16	0	M	34	217	0	100
214	13	0	15	5	0	M	6	0	0	500
8550	5	0	16	17	0	M	18	8551	0	100
8551	8	0	16	11	0	F	17	8550	0	200
6240	1	0	171	15	0	F	20	0	0	400
6241	11	0	15	15	0	M	0	0	0	700
5381	18	0	17	1	0	M	57	6238	0	100
6238	9	0	17	13	0	F	52	5381	0	200
5383	18	0	17	7	0	M	13	0	0	300
8174	18	0	17	14	0	M	34	8175	0	100
8175	10	0	17	11	0	F	34	8174	0	200
264	1	0	18	14	0	M	35	0	0	100
6796	1	0	18	13	0	M	4	0	0	500
263	1	0	18	13	0	M	12	0	0	300

6511	8	0	171	5	0	F	16	0	0	400
278	2	0	10	20	0	M	14	0	0	300
283	8	0	171	13	0	F	21	0	0	400
280	5	0	19	8	0	F	3	0	0	700
285	8	0	19	6	0	M	47	6892	0	100
6892	13	0	19	19	0	F	36	285	0	200
306	15	0	20	19	0	M	58	307	0	100
307	12	0	20	5	0	F	58	306	0	200
8580	15	0	20	12	0	M	17	0	0	300
305	15	0	20	12	0	M	31	308	0	100
308	18	0	20	3	0	F	27	305	0	200
304	15	0	20	18	0	F	5	0	0	600
7793	15	0	20	12	0	M	33	8491	0	100
8491	1	0	20	2	0	F	26	7793	0	200
5634	2	0	21	14	0	F	70	8515	0	200
8515	19	0	21	14	0	M	45	5634	0	100
7993	1	0	21	2	0	M	12	0	0	500
5636	1	0	21	2	0	M	21	0	0	300
5638	9	0	171	1	0	F	15	0	0	400
5567	9	0	22	18	0	F	20	2008	0	200
2008	16	0	22	15	0	M	33	5567	0	100
6578	2	0	23	11	0	F	46	0	0	200
6817	9	0	23	2	0	M	5	0	0	500
6580	3	0	23	2	0	M	28	6581	0	100
6581	4	0	23	17	0	F	28	6580	0	200
6777	3	0	23	4	0	M	4	0	0	500
8453	4	0	24	17	0	M	34	0	0	300
6197	4	0	24	17	0	M	28	6198	0	100
6198	11	0	24	18	0	F	28	6197	0	200
451	11	0	25	8	0	M	51	8067	0	100
453	18	0	25	11	0	F	67	451	0	200
450	11	0	25	18	0	M	32	454	0	100
454	16	0	25	13	0	F	33	450	0	200
6780	11	0	25	16	0	F	9	0	0	600
449	11	0	171	16	0	F	12	0	0	400
4802	19	0	26	11	0	M	44	4803	0	100
4803	10	0	26	16	0	F	49	4802	0	200
6950	19	0	26	10	0	M	13	0	0	300
6769	19	0	26	10	0	M	9	0	0	500
6757	19	0	26	10	0	M	4	0	0	500
4804	19	0	26	10	0	M	12	0	0	500
375	19	0	26	10	0	M	45	376	0	100
376	18	0	26	4	0	F	19	375	0	200

6218	19	0	26	18	0	M	0	0	0	700
378	10	0	27	6	0	F	74	0	0	200
377	12	0	27	10	0	M	44	379	0	100
379	17	0	27	7	0	F	36	377	0	200
6727	12	0	27	17	0	M	3	0	0	700
929	1	0	28	20	0	M	35	6683	0	100
6683	11	0	28	3	0	F	33	929	0	200
7790	1	0	28	11	0	M	9	0	0	500
6684	1	0	28	11	0	M	3	0	0	700
434	10	0	29	12	0	M	56	0	0	300
433	10	0	29	13	0	M	35	436	0	100
436	18	0	29	9	0	F	27	433	0	200
8584	10	0	171	2	0	F	18	0	0	400
432	10	0	29	18	0	M	7	0	0	500
440	6	0	30	11	0	M	24	441	0	100
441	8	0	30	17	0	F	15	440	0	200
439	6	0	30	8	0	F	2	0	0	700
438	9	0	31	11	0	F	52	0	0	200
8643	18	0	31	9	0	M	17	0	0	300
6808	18	0	31	9	0	M	31	6807	0	100
6807	2	0	31	13	0	F	27	6808	0	200
6809	18	0	31	2	0	M	4	0	0	500
5650	13	0	32	12	0	F	65	0	0	200
5651	2	0	32	13	0	F	30	5652	0	200
5652	5	0	32	20	0	M	29	5651	0	100
1449	20	0	32	2	0	F	47	0	0	200
1448	5	0	32	20	0	M	31	0	0	300
447	1	0	33	17	0	M	59	0	0	300
446	1	0	33	10	0	M	23	0	0	300
7890	1	0	33	10	0	M	33	7891	0	100
7891	14	0	33	4	0	F	32	7890	0	200
990	5	0	34	10	0	M	36	991	0	100
991	1	0	34	10	0	F	40	990	0	200
8139	5	0	34	8	0	M	14	0	0	300
6691	5	0	34	1	0	F	3	0	0	700
6522	5	0	34	1	0	M	2	0	0	700
4732	1	0	33	10	0	M	27	4735	0	100
4735	16	0	33	17	0	F	22	4732	0	200
6486	1	0	33	16	0	F	2	0	0	700
4734	16	0	35	4	0	M	50	4733	0	100
4733	17	0	35	12	0	F	50	4734	0	200
455	9	0	36	11	0	M	54	0	0	300
6914	9	0	36	14	0	F	5	0	0	600

6576	20	0	36	2	0	M	28	6575	0	100
6575	9	0	36	14	0	F	22	6576	0	200
484	4	0	141	15	0	M	24	0	0	300
511	3	0	37	7	0	M	42	512	0	100
512	16	0	37	18	0	F	36	511	0	200
510	3	0	37	16	0	M	15	0	0	300
7717	3	0	37	16	0	F	8	0	0	600
6676	3	0	37	16	0	F	3	0	0	700
6191	3	0	37	16	0	M	0	0	0	700
514	5	0	38	14	0	M	50	515	0	100
515	6	0	38	2	0	F	39	514	0	200
513	5	0	171	6	0	F	17	0	0	400
6355	5	0	38	6	0	M	1	0	0	700
517	6	0	38	20	0	M	60	0	0	300
9122	6	0	171	2	0	F	35	0	0	400
527	16	0	39	10	0	F	59	0	0	200
4823	3	0	39	16	0	M	31	4824	0	100
4824	2	0	39	9	0	F	27	4823	0	200
525	3	0	171	16	0	F	43	0	0	400
523	3	0	39	8	0	F	11	0	0	400
6873	3	0	39	8	0	F	5	0	0	600
2337	3	0	39	8	0	F	2	0	0	700
553	2	0	40	8	0	M	11	0	0	500
554	2	0	40	14	0	M	33	0	0	300
574	18	0	40	14	0	M	20	575	0	100
575	1	0	40	13	0	F	21	574	0	200
6556	18	0	40	1	0	M	2	0	0	700
6015	18	0	40	1	0	M	1	0	0	700
578	16	0	41	14	0	F	29	5360	0	200
5360	10	0	41	13	0	M	27	578	0	100
7010	16	0	41	10	0	M	11	0	0	500
6694	16	0	41	10	0	F	3	0	0	700
611	17	0	42	11	0	M	30	612	0	100
612	8	0	42	15	0	F	24	611	0	200
6802	17	0	42	8	0	F	4	0	0	600
610	17	0	42	8	0	M	3	0	0	700
619	3	0	171	4	0	F	19	0	0	400
618	10	0	41	3	0	M	4	0	0	500
2715	4	0	43	7	0	M	20	620	0	100
620	9	0	43	19	0	F	17	2715	0	200
632	4	0	45	15	0	M	23	0	0	300
936	1	0	44	18	0	M	23	937	0	100
937	16	0	44	6	0	F	23	936	0	200

6436	1	0	44	16	0	F	1	0	0	700
988	4	0	45	18	0	M	53	987	0	100
987	10	0	45	17	0	F	53	988	0	200
986	4	0	45	10	0	M	20	0	0	300
6446	17	0	46	19	0	F	22	6445	0	200
6445	20	0	46	6	0	M	23	6446	0	100
6447	20	0	46	17	0	M	2	0	0	700
6444	20	0	46	17	0	M	45	6443	0	100
6443	6	0	46	10	0	F	41	6444	0	200
1023	3	0	47	17	0	M	57	1024	0	100
1024	15	0	47	7	0	F	47	1023	0	200
1022	3	0	47	15	0	M	11	0	0	500
7520	3	0	47	15	0	M	8	0	0	500
1035	11	0	47	15	0	M	20	0	0	300
1051	1	0	48	5	0	M	32	1052	0	100
1052	16	0	48	18	0	F	32	1051	0	200
7639	1	0	48	16	0	F	9	0	0	600
7596	1	0	48	16	0	F	9	0	0	600
3477	1	0	48	16	0	F	4	0	0	600
1050	1	0	48	16	0	M	1	0	0	700
1056	19	0	49	8	0	M	67	0	0	300
1055	19	0	49	13	0	M	44	1057	0	100
1057	7	0	49	8	0	F	47	1055	0	200
8906	19	0	171	7	0	F	22	0	0	400
1054	19	0	49	7	0	M	26	1058	0	100
1058	4	0	49	3	0	F	19	1054	0	200
1059	4	0	50	1	0	M	39	1060	0	100
1060	3	0	50	12	0	F	38	1059	0	200
7009	4	0	50	3	0	M	6	0	0	500
6537	4	0	50	3	0	M	2	0	0	700
1082	12	0	51	2	0	M	30	1083	0	100
1083	8	0	51	20	0	F	32	1082	0	200
1382	12	0	51	8	0	M	32	0	0	300
7305	12	0	51	8	0	M	11	0	0	500
3126	12	0	51	8	0	M	9	0	0	500
1081	12	0	51	8	0	M	2	0	0	700
1138	6	0	171	13	0	F	20	0	0	400
1144	4	0	52	19	0	M	53	6454	0	100
6454	15	0	52	8	0	F	43	1144	0	200
6697	4	0	52	15	0	M	3	0	0	700
1143	4	0	52	8	0	M	28	1146	0	100
1146	12	0	52	9	0	F	27	1143	0	200
6601	4	0	52	12	0	M	3	0	0	700

6484	4	0	54	8	0	M	25	6483	0	100
6483	3	0	54	18	0	F	24	6484	0	200
1452	19	0	53	1	0	M	21	6159	0	100
6159	4	0	53	8	0	F	22	1452	0	200
6160	19	0	53	4	0	F	0	0	0	700
1454	4	0	54	14	0	M	38	1457	0	100
1457	20	0	54	15	0	F	38	1454	0	200
1453	4	0	54	20	0	F	6	0	0	600
6236	4	0	54	20	0	M	0	0	0	700
8322	5	0	55	2	0	M	43	8323	0	100
8323	4	0	55	14	0	F	45	8322	0	200
8566	7	0	56	16	0	M	20	6947	0	100
6947	4	0	56	14	0	F	18	8566	0	200
6949	7	0	56	4	0	M	8	0	0	500
8489	4	0	55	14	0	M	29	7248	0	100
7248	9	0	55	6	0	F	35	8489	0	200
1294	10	0	56	14	0	M	48	5498	0	100
5498	9	0	56	19	0	F	36	1294	0	200
1293	10	0	56	5	0	M	13	0	0	300
6889	10	0	56	5	0	M	5	0	0	500
6630	10	0	56	5	0	F	3	0	0	700
6219	10	0	56	5	0	F	0	0	0	700
1383	9	0	171	13	0	F	17	0	0	400
1386	11	0	57	14	0	M	45	1387	0	100
1387	13	0	57	14	0	F	50	1386	0	200
6572	11	0	57	13	0	F	2	0	0	700
1385	11	0	57	13	0	M	7	0	0	500
8081	11	0	57	13	0	M	13	0	0	300
8615	11	0	57	13	0	M	16	0	0	300
7298	11	0	57	13	0	M	26	8518	0	100
8518	1	0	57	12	0	F	31	7298	0	200
3839	11	0	58	13	0	F	22	3839	0	200
3838	3	0	58	2	0	M	25	3838	0	100
3836	2	0	58	11	0	F	57	0	0	200
8672	3	0	171	2	0	F	16	0	0	400
7760	3	0	58	2	0	M	31	7448	0	100
7448	13	0	58	7	0	F	43	7760	0	200
6210	7	0	58	5	0	F	70	0	0	200
1394	8	0	59	20	0	M	21	6930	0	100
6930	11	0	59	3	0	F	24	1394	0	200
1421	8	0	59	4	0	M	21	0	0	300
1425	16	0	60	20	0	F	68	0	0	200
1424	15	0	60	16	0	M	33	1426	0	100

1426	1	0	60	7	0	F	30	1424	0	200
7978	15	0	60	1	0	M	11	0	0	500
3689	15	0	60	1	0	F	5	0	0	600
6139	15	0	60	1	0	F	0	0	0	700
8527	1	0	10	7	0	F	17	6571	0	200
6571	14	0	10	8	0	M	22	8527	0	100
1466	3	0	203	6	0	M	27	0	0	300
1507	19	0	61	12	0	F	50	0	0	200
1504	12	0	61	7	0	M	26	1508	0	100
1508	18	0	61	7	0	F	25	1504	0	200
1535	2	0	182	13	0	M	20	1537	0	100
1537	20	0	182	17	0	F	21	1535	0	200
1655	10	0	171	1	0	F	17	0	0	400
1658	6	0	55	2	0	M	17	0	0	300
1666	11	0	62	8	0	M	35	1669	0	100
1669	5	0	62	18	0	F	48	1666	0	200
8160	11	0	171	5	0	F	14	0	0	400
6890	11	0	62	5	0	F	5	0	0	600
6230	11	0	62	5	0	F	0	0	0	700
1670	5	0	63	8	0	M	51	0	0	300
9073	5	0	63	18	0	M	28	0	0	300
5209	5	0	63	18	0	M	42	6106	0	100
6106	17	0	63	10	0	F	20	5209	0	200
1689	6	0	64	8	0	M	40	1691	0	100
1691	19	0	64	13	0	F	30	1689	0	200
1688	6	0	64	19	0	M	8	0	0	500
1693	18	0	64	19	0	M	20	1694	0	100
1694	4	0	64	11	0	F	18	1693	0	200
1773	12	0	65	4	0	M	35	1774	0	100
1774	1	0	65	5	0	F	30	1773	0	200
1772	12	0	65	1	0	M	2	0	0	700
6354	12	0	171	16	0	F	15	0	0	400
7233	12	0	65	16	0	M	11	0	0	500
7400	11	0	65	1	0	F	10	0	0	600
6729	11	0	65	1	0	F	3	0	0	700
1819	19	0	66	3	0	M	44	8393	0	100
8393	12	0	66	15	0	F	41	1819	0	200
1818	19	0	66	4	0	M	18	0	0	300
8505	19	0	67	12	0	F	21	2249	0	200
2249	3	0	67	11	0	M	26	8505	0	100
2250	3	0	67	16	0	M	54	2252	0	100
2252	11	0	67	19	0	F	49	2250	0	200
8082	3	0	171	11	0	F	13	0	0	400

8463	3	0	171	11	0	F	28	0	0	400
3154	3	0	68	11	0	M	27	3157	0	100
3157	9	0	68	16	0	F	20	3154	0	200
3153	3	0	68	9	0	M	9	0	0	500
6646	3	0	68	9	0	M	3	0	0	700
3156	16	0	68	7	0	F	53	0	0	200
8953	9	0	68	16	0	M	23	0	0	300
8754	9	0	171	16	0	F	19	0	0	400
1821	16	0	171	18	0	F	15	0	0	400
1828	2	0	198	12	0	M	36	0	0	300
1827	2	0	198	15	0	M	10	0	0	500
1924	9	0	247	17	0	M	8	0	0	500
1934	9	0	69	10	0	F	49	0	0	200
9113	2	0	171	9	0	F	30	0	0	400
1932	2	0	69	9	0	M	23	0	0	300
1973	19	0	171	20	0	F	39	0	0	400
1970	20	0	70	1	0	M	20	1971	0	100
1971	7	0	70	19	0	F	16	1970	0	200
1969	20	0	70	7	0	M	0	0	0	700
2079	15	0	71	1	0	M	59	6424	0	100
6424	18	0	71	8	0	F	29	2079	0	200
7627	9	0	71	18	0	M	11	0	0	500
7297	9	0	71	18	0	F	7	0	0	600
6751	9	0	71	18	0	M	6	0	0	500
6422	9	0	71	18	0	F	1	0	0	700
2080	16	0	71	9	0	F	54	0	0	200
2078	15	0	71	16	0	M	36	7744	0	100
7744	9	0	71	10	0	F	48	2078	0	200
2077	15	0	71	9	0	M	14	0	0	300
7724	15	0	71	9	0	M	12	0	0	500
4204	15	0	72	16	0	M	23	4205	0	100
4205	19	0	72	10	0	F	21	4204	0	200
6183	15	0	72	19	0	M	0	0	0	700
6487	15	0	72	19	0	F	2	0	0	700
4203	15	0	72	19	0	M	7	0	0	500
2209	7	0	73	1	0	M	51	6608	0	100
6608	8	0	73	20	0	F	40	2209	0	200
7518	7	0	73	1	0	M	13	0	0	300
7508	7	0	171	8	0	F	12	0	0	400
6610	7	0	73	8	0	F	3	0	0	700
2207	7	0	73	1	0	M	19	8517	0	100
8517	15	0	73	12	0	F	29	2207	0	200
2293	15	0	74	16	0	M	30	2294	0	100

2294	4	0	74	17	0	F	27	2293	0	200
6367	15	0	74	4	0	M	0	0	0	700
2300	19	0	75	16	0	M	51	5571	0	100
5571	5	0	75	13	0	F	39	2300	0	200
6861	19	0	75	5	0	M	5	0	0	500
7936	19	0	75	5	0	M	11	0	0	500
5572	19	0	171	5	0	F	13	0	0	400
8600	19	0	171	5	0	F	16	0	0	400
2299	19	0	75	5	0	M	27	0	0	300
2422	13	0	76	18	0	M	29	6888	0	100
6888	6	0	76	3	0	F	35	2422	0	200
6899	13	0	76	6	0	F	9	0	0	600
2441	11	0	77	10	0	M	50	0	0	300
7839	11	0	171	2	0	F	23	0	0	400
8085	11	0	171	13	0	F	13	0	0	400
8789	11	0	77	13	0	M	19	0	0	300
5570	11	0	77	13	0	M	24	0	0	300
2440	4	0	78	14	0	M	28	6207	0	100
6207	19	0	78	5	0	F	24	2440	0	200
6979	4	0	78	9	0	M	6	0	0	500
6208	4	0	78	19	0	M	0	0	0	700
2447	13	0	79	2	0	M	28	2448	0	100
2448	2	0	79	13	0	F	35	2447	0	200
6685	13	0	79	2	0	M	3	0	0	700
6192	13	0	79	2	0	M	0	0	0	700
2455	3	0	80	9	0	M	35	2456	0	100
2456	16	0	80	3	0	F	31	2455	0	200
7331	3	0	80	16	0	M	11	0	0	500
2454	3	0	80	16	0	M	5	0	0	500
8130	16	0	81	14	0	F	48	0	0	200
2529	5	0	81	17	0	M	29	2532	0	100
2532	3	0	81	12	0	F	28	2529	0	200
6709	5	0	81	3	0	M	3	0	0	700
2528	5	0	81	3	0	M	8	0	0	500
4508	5	0	81	17	0	M	33	8093	0	100
8093	15	0	81	7	0	F	34	4508	0	200
2581	17	0	244	8	0	M	40	2582	0	100
2582	12	0	244	5	0	F	39	2581	0	200
2580	17	0	244	12	0	M	5	0	0	500
2599	2	0	82	11	0	M	51	7930	0	100
7930	3	0	82	14	0	F	44	2599	0	200
7931	2	0	82	3	0	M	11	0	0	500
8697	2	0	82	12	0	M	18	0	0	300

2598	2	0	82	12	0	M	24	0	0	300
2632	16	0	240	15	0	M	21	0	0	300
7617	8	0	83	13	0	F	71	0	0	200
6119	4	0	83	8	0	F	33	6845	0	200
6845	10	0	83	15	0	M	53	6119	0	100
7424	10	0	83	4	0	F	7	0	0	600
6118	10	0	83	4	0	M	21	6120	0	100
6120	3	0	83	16	0	F	26	6118	0	200
2650	10	0	84	16	0	M	32	2652	0	100
2652	3	0	84	1	0	F	37	2650	0	200
6669	10	0	84	3	0	F	3	0	0	700
2649	10	0	84	3	0	M	13	0	0	300
2675	17	0	85	15	0	M	31	6527	0	100
6527	15	0	85	10	0	F	24	2675	0	200
6359	17	0	85	8	0	M	1	0	0	700
2674	17	0	85	8	0	M	6	0	0	500
6187	8	0	171	12	0	F	16	0	0	400
2799	10	0	86	5	0	M	53	2796	0	100
2796	17	0	86	16	0	F	45	2799	0	200
2795	10	0	86	17	0	M	15	5644	0	100
5644	12	0	86	13	0	F	17	2795	0	200
5643	12	0	87	16	0	M	35	5642	0	100
5642	13	0	87	10	0	F	35	5643	0	200
8108	12	0	87	13	0	M	14	0	0	300
7822	12	0	87	13	0	M	9	0	0	500
7819	14	0	88	19	0	M	45	2901	0	100
2901	17	0	88	14	0	F	44	7819	0	200
9087	14	0	88	4	0	M	28	0	0	300
2902	14	0	171	17	0	F	20	0	0	400
6391	9	0	88	6	0	F	60	0	0	200
7883	14	0	171	9	0	F	35	0	0	400
6911	14	0	88	9	0	F	39	6912	0	200
6912	18	0	88	15	0	M	50	6911	0	100
2900	14	0	89	17	0	F	20	4152	0	200
4152	11	0	89	3	0	M	25	2900	0	100
6634	14	0	89	11	0	M	3	0	0	700
6206	14	0	89	11	0	M	0	0	0	700
2963	16	0	90	7	0	F	61	0	0	200
8156	17	0	90	16	0	M	14	0	0	300
2961	17	0	90	16	0	F	24	2960	0	200
2960	4	0	90	3	0	M	27	2961	0	100
6432	17	0	90	4	0	F	1	0	0	700
3128	7	0	91	17	0	M	27	3127	0	100

3127	3	0	91	2	0	F	28	3128	0	200
7146	7	0	171	3	0	F	11	0	0	400
3129	7	0	91	3	0	F	2	0	0	700
3145	14	0	159	8	0	M	12	0	0	300
3159	9	0	92	12	0	M	43	3160	0	100
3160	12	0	92	9	0	F	46	3159	0	200
4693	9	0	92	12	0	M	18	0	0	300
8047	9	0	92	12	0	M	13	0	0	300
6509	9	0	92	12	0	F	2	0	0	700
7865	11	0	92	15	0	M	38	7866	0	100
7866	12	0	92	9	0	F	33	7865	0	200
3172	3	0	171	17	0	F	22	0	0	400
6147	18	0	171	3	0	M	0	0	0	700
3191	6	0	93	3	0	M	29	3192	0	100
3192	2	0	93	20	0	F	30	3191	0	200
6795	6	0	93	2	0	F	4	0	0	600
6358	6	0	93	2	0	M	1	0	0	700
3385	20	0	94	1	0	F	27	3384	0	200
3384	4	0	94	20	0	M	33	3385	0	100
6800	4	0	94	20	0	M	7	0	0	500
3383	4	0	94	20	0	F	3	0	0	700
3401	6	0	247	10	0	M	23	0	0	300
3441	10	0	242	5	0	M	38	4809	0	100
4809	12	0	242	2	0	F	26	3441	0	200
6849	10	0	242	12	0	M	5	0	0	500
3473	13	0	95	15	0	M	67	5910	0	100
5910	1	0	95	17	0	F	55	3473	0	200
3471	13	0	171	15	0	F	23	0	0	400
5911	13	0	95	1	0	M	15	0	0	300
6721	13	0	95	1	0	F	3	0	0	700
8739	13	0	95	1	0	M	19	0	0	300
6652	13	0	95	15	0	M	28	6657	0	100
6657	11	0	95	14	0	F	21	6652	0	200
6656	14	0	171	6	0	F	42	0	0	400
7851	10	0	96	14	0	M	17	0	0	300
7850	10	0	96	14	0	M	15	0	0	300
3480	14	0	97	16	0	M	35	6024	0	100
6024	8	0	97	3	0	F	20	3480	0	200
6844	14	0	97	8	0	M	5	0	0	500
3479	14	0	97	8	0	M	9	0	0	500
3486	8	0	98	18	0	M	25	3488	0	100
3488	14	0	98	16	0	F	28	3486	0	200
6633	6	0	98	18	0	F	20	3520	0	200

3520	20	0	98	6	0	M	28	6633	0	100
4779	15	0	74	8	0	M	21	3688	0	100
3688	10	0	74	15	0	F	32	4779	0	200
3754	8	0	143	18	0	M	19	7495	0	100
7495	5	0	143	7	0	F	21	3754	0	200
3755	5	0	171	19	0	F	29	0	0	400
3811	7	0	99	19	0	F	51	0	0	200
5323	2	0	171	7	0	F	11	0	0	400
3814	2	0	99	7	0	M	20	0	0	300
3815	7	0	171	14	0	F	23	0	0	400
6607	2	0	99	7	0	F	3	0	0	700
3816	7	0	100	6	0	M	41	3817	0	100
3817	14	0	100	18	0	F	53	3816	0	200
3826	9	0	101	4	0	M	34	3827	0	100
3827	5	0	101	8	0	F	27	3826	0	200
8036	9	0	171	5	0	F	12	0	0	400
7777	9	0	101	5	0	F	8	0	0	600
3830	9	0	101	5	0	M	4	0	0	500
6388	9	0	101	5	0	F	1	0	0	700
3864	20	0	246	12	0	M	20	0	0	300
3865	14	0	43	1	0	M	24	8512	0	100
8512	4	0	43	19	0	F	23	3865	0	200
8385	4	0	102	11	0	M	47	8384	0	100
8384	19	0	102	11	0	F	54	8385	0	200
6783	13	0	102	2	0	M	30	6784	0	100
6784	19	0	102	11	0	F	31	6783	0	200
6402	19	0	102	11	0	M	35	6599	0	100
6599	1	0	102	4	0	F	23	6402	0	200
6600	11	0	103	1	0	F	3	0	0	700
3875	14	0	103	6	0	M	45	5890	0	100
5890	11	0	103	14	0	F	46	3875	0	200
7589	11	0	103	14	0	M	11	0	0	500
7530	11	0	103	14	0	F	13	0	0	400
6906	11	0	103	14	0	M	5	0	0	500
5891	11	0	103	2	0	M	14	5894	0	100
5894	12	0	103	10	0	F	16	5891	0	200
6168	11	0	103	12	0	F	0	0	0	700
5893	12	0	104	3	0	M	47	5892	0	100
5892	17	0	104	5	0	F	42	5893	0	200
8594	12	0	104	17	0	M	24	0	0	300
6794	12	0	104	17	0	F	4	0	0	600
8530	12	0	104	17	0	M	20	6994	0	100
6994	6	0	104	13	0	F	19	8530	0	200

3921	20	0	105	19	0	F	26	3922	0	200
3922	16	0	105	18	0	M	28	3921	0	100
6839	16	0	105	20	0	F	5	0	0	600
6375	16	0	105	20	0	M	1	0	0	700
4063	3	0	171	17	0	F	19	0	0	400
4134	12	0	104	14	0	M	20	5414	0	100
5414	1	0	104	6	0	F	18	4134	0	200
4139	19	0	75	3	0	M	8	0	0	500
4185	8	0	105	12	0	M	32	4630	0	100
4630	12	0	105	8	0	F	29	4185	0	200
6833	8	0	105	12	0	F	5	0	0	600
6401	8	0	105	12	0	M	1	0	0	700
4678	17	0	232	15	0	M	21	4202	0	100
4202	19	0	232	8	0	F	19	4678	0	200
4206	14	0	231	11	0	M	33	4207	0	100
4207	4	0	231	15	0	F	20	4206	0	200
4225	9	0	106	8	0	M	20	0	0	300
4342	9	0	106	11	0	M	61	4343	0	100
4343	11	0	106	10	0	F	53	4342	0	200
8135	9	0	106	11	0	M	14	0	0	300
4341	9	0	106	11	0	M	25	4344	0	100
4344	14	0	106	16	0	F	24	4341	0	200
4353	9	0	107	8	0	M	32	4354	0	100
4354	7	0	107	19	0	F	32	4353	0	200
6871	9	0	107	7	0	M	5	0	0	500
6526	9	0	107	7	0	M	2	0	0	700
4376	13	0	108	10	0	F	27	4377	0	200
4377	10	0	108	16	0	M	37	4376	0	100
6908	10	0	108	13	0	M	5	0	0	500
6221	10	0	108	13	0	F	0	0	0	700
4418	5	0	109	19	0	F	26	4418	0	200
4417	7	0	109	15	0	M	29	4418	0	100
6910	7	0	109	15	0	M	5	0	0	500
6530	7	0	109	15	0	F	2	0	0	700
6176	7	0	109	15	0	F	0	0	0	700
4486	20	0	110	2	0	M	28	6193	0	100
6193	1	0	110	3	0	F	30	4486	0	200
6194	20	0	110	1	0	M	0	0	0	700
4507	7	0	111	1	0	F	40	0	0	200
4504	15	0	111	7	0	M	14	0	0	300
6398	15	0	111	7	0	M	24	6399	0	100
6399	6	0	111	3	0	F	29	6398	0	200
6400	15	0	111	6	0	M	1	0	0	700

7466	10	0	112	19	0	M	75	7467	0	100
7467	2	0	112	8	0	F	49	7466	0	200
7472	10	0	112	2	0	M	10	0	0	500
7468	10	0	112	2	0	F	8	0	0	600
4509	10	0	112	2	0	F	51	0	0	200
4578	17	0	207	1	0	M	27	5575	0	100
5575	6	0	207	11	0	F	27	4578	0	200
6426	17	0	207	6	0	F	1	0	0	700
8573	17	0	207	6	0	F	4	0	0	600
8075	12	0	113	13	0	F	45	0	0	200
8076	20	0	113	12	0	M	13	0	0	300
8788	20	0	113	12	0	M	19	0	0	300
4611	20	0	113	12	0	M	26	6917	0	100
6917	13	0	113	11	0	F	31	4611	0	200
4659	3	0	245	13	0	M	29	5669	0	100
5669	20	0	245	16	0	F	27	4659	0	200
6356	3	0	245	20	0	M	1	0	0	700
4665	15	0	128	19	0	M	28	5083	0	100
5083	8	0	128	20	0	F	23	4665	0	200
6708	15	0	128	8	0	F	1	0	0	700
4700	16	0	171	13	0	F	30	0	0	400
4711	13	0	114	4	0	M	45	0	0	300
8926	13	0	171	12	0	F	23	0	0	400
4713	13	0	114	12	0	M	19	4718	0	100
4718	6	0	114	10	0	F	19	4713	0	200
4716	2	0	118	6	0	F	46	0	0	200
9128	2	0	118	6	0	M	50	0	0	300
8559	3	0	115	20	0	M	47	8561	0	100
8561	15	0	115	19	0	F	38	8559	0	200
8560	3	0	115	15	0	F	18	4727	0	200
4727	15	0	115	18	0	M	18	8560	0	100
4772	14	0	239	5	0	M	35	6116	0	100
6116	12	0	239	8	0	F	33	4772	0	200
4795	20	0	116	18	0	M	37	4794	0	100
4794	14	0	116	7	0	F	37	4795	0	200
7607	7	0	116	14	0	M	9	0	0	500
7606	7	0	116	14	0	F	8	0	0	600
6677	7	0	116	14	0	M	3	0	0	700
6395	7	0	116	14	0	F	1	0	0	700
4816	20	0	117	15	0	M	30	4815	0	100
4815	13	0	117	8	0	F	20	4816	0	200
6533	20	0	117	13	0	F	2	0	0	700
6200	20	0	117	13	0	F	0	0	0	700

4930	2	0	118	20	0	M	27	4931	0	100
4931	11	0	118	16	0	F	29	4930	0	200
6902	2	0	118	11	0	M	5	0	0	500
6068	7	0	121	20	0	M	36	4933	0	100
4933	11	0	121	16	0	F	36	6068	0	200
4954	8	0	119	14	0	M	26	4955	0	100
4955	2	0	119	11	0	F	37	4954	0	200
6631	8	0	119	2	0	F	3	0	0	700
5009	14	0	120	1	0	M	31	5008	0	100
5008	20	0	120	17	0	F	31	5009	0	200
7517	14	0	120	20	0	M	9	0	0	500
6983	14	0	120	20	0	M	6	0	0	500
6693	14	0	120	20	0	M	3	0	0	700
6211	14	0	120	20	0	F	0	0	0	700
5011	11	0	205	8	0	M	19	5012	0	100
5012	19	0	205	10	0	F	18	5011	0	200
5014	2	0	179	4	0	M	29	6972	0	100
6972	16	0	179	8	0	F	25	5014	0	200
5017	19	0	121	6	0	F	30	5018	0	200
5018	7	0	121	17	0	M	30	5017	0	100
6190	19	0	121	7	0	M	0	0	0	700
5044	6	0	122	18	0	M	64	0	0	300
7522	6	0	122	8	0	M	12	0	0	500
5043	6	0	171	8	0	F	16	0	0	400
6535	6	0	122	8	0	F	21	5045	0	200
5045	10	0	122	8	0	M	25	6535	0	100
5055	14	0	120	4	0	M	45	0	0	300
8577	7	0	171	8	0	F	19	0	0	400
5897	12	0	123	2	0	M	33	5081	0	100
5081	6	0	123	4	0	F	30	5897	0	200
7278	12	0	123	6	0	F	9	0	0	600
7396	12	0	123	6	0	F	8	0	0	600
6811	12	0	123	6	0	F	5	0	0	600
6460	12	0	123	6	0	M	2	0	0	700
5158	15	0	15	17	0	M	27	5157	0	100
5157	10	0	15	18	0	F	25	5158	0	200
6670	15	0	15	10	0	M	3	0	0	700
5161	1	0	124	8	0	F	55	0	0	200
8923	12	0	171	1	0	F	23	0	0	400
9078	12	0	124	1	0	M	27	0	0	300
6841	12	0	124	1	0	M	30	6840	0	100
6840	7	0	124	5	0	F	35	6841	0	200
8012	12	0	171	7	0	F	12	0	0	400

6842	12	0	124	7	0	M	5	0	0	500
5457	7	0	125	10	0	M	54	5974	0	100
5974	18	0	125	10	0	F	33	5457	0	200
8579	7	0	125	18	0	M	20	0	0	300
5458	7	0	126	18	0	M	28	5461	0	100
5461	8	0	126	5	0	F	28	5458	0	200
6734	5	0	126	8	0	F	4	0	0	600
5459	5	0	126	14	0	F	61	0	0	200
5163	19	0	124	3	0	M	36	0	0	300
5164	8	0	124	5	0	M	19	0	0	300
5186	9	0	171	16	0	F	21	0	0	400
5190	18	0	140	14	0	M	33	7837	0	100
7837	12	0	140	7	0	F	30	5190	0	200
6819	9	0	22	2	0	M	46	0	0	300
6820	9	0	22	1	0	M	27	5196	0	100
5196	14	0	22	3	0	F	24	6820	0	200
5235	13	0	125	19	0	M	53	8307	0	100
8307	5	0	125	7	0	F	48	5235	0	200
8513	6	0	22	5	0	F	23	8514	0	200
8514	16	0	22	3	0	M	23	8513	0	100
8525	13	0	125	16	0	F	23	7127	0	200
7127	19	0	125	12	0	M	24	8525	0	100
5237	2	0	126	12	0	F	34	6719	0	200
6719	13	0	126	3	0	M	39	5237	0	100
6720	2	0	126	13	0	F	3	0	0	700
6718	13	0	126	4	0	M	61	8377	0	100
8377	16	0	126	20	0	F	53	6718	0	200
9163	13	0	126	16	0	M	53	0	0	300
8521	10	0	29	1	0	M	20	7729	0	100
7729	8	0	29	10	0	F	21	8521	0	200
5238	4	0	127	11	0	M	20	6704	0	100
6704	19	0	127	10	0	F	27	5238	0	200
6703	4	0	127	19	0	F	3	0	0	700
6705	19	0	127	1	0	M	53	6706	0	100
6706	1	0	127	8	0	F	23	6705	0	200
8103	19	0	127	10	0	M	13	0	0	300
5416	15	0	128	4	0	M	60	8438	0	100
8438	1	0	128	11	0	F	40	5416	0	200
8597	15	0	128	3	0	M	28	0	0	300
5246	15	0	171	3	0	F	17	0	0	400
5260	8	0	105	9	0	M	19	5258	0	100
5258	9	0	105	4	0	F	16	5260	0	200
5284	4	0	129	8	0	F	46	8459	0	200

8459	12	0	129	16	0	M	54	5284	0	100
5286	4	0	129	12	0	M	22	2900	0	100
2900	14	0	129	17	0	F	19	5286	0	200
5295	16	0	131	8	0	M	28	6496	0	100
6496	20	0	131	1	0	F	32	5295	0	200
7053	7	0	160	8	0	M	38	5313	0	100
5313	5	0	160	20	0	F	46	7053	0	200
5314	7	0	160	5	0	F	18	7059	0	200
5315	6	0	162	17	0	M	17	7435	0	100
7435	4	0	162	20	0	F	19	5315	0	200
5326	19	0	130	8	0	F	50	0	0	200
5327	20	0	130	19	0	M	31	5330	0	100
5330	16	0	130	7	0	F	31	5327	0	200
5329	12	0	130	16	0	F	49	0	0	200
6651	20	0	130	16	0	F	3	0	0	700
5343	16	0	131	2	0	M	35	5342	0	100
5342	17	0	131	19	0	F	22	5343	0	200
8098	3	0	131	16	0	M	13	0	0	300
7769	3	0	131	16	0	F	8	0	0	600
5389	16	0	168	9	0	M	24	5390	0	100
5390	19	0	168	5	0	F	34	5389	0	200
5441	19	0	132	5	0	M	50	5442	0	100
5442	12	0	132	14	0	F	43	5441	0	200
5440	19	0	171	12	0	F	13	0	0	400
7236	19	0	132	12	0	M	7	0	0	500
5452	5	0	133	19	0	M	66	5451	0	100
5451	16	0	133	19	0	F	61	5452	0	200
5453	5	0	133	16	0	M	34	5454	0	100
5454	12	0	133	10	0	F	33	5453	0	200
6866	5	0	133	12	0	M	5	0	0	500
5480	5	0	133	16	0	M	32	6389	0	100
6389	17	0	133	3	0	F	30	5480	0	200
8042	5	0	133	17	0	M	13	0	0	300
6392	5	0	133	17	0	M	1	0	0	700
5787	5	0	133	12	0	M	26	5791	0	100
5791	14	0	133	18	0	F	24	5787	0	200
5492	20	0	134	15	0	F	80	0	0	200
5494	4	0	134	20	0	F	40	5495	0	200
5495	17	0	134	15	0	M	44	5494	0	100
6584	17	0	134	4	0	F	2	0	0	700
7811	17	0	134	4	0	M	9	0	0	500
5496	17	0	134	4	0	M	15	0	0	300
8557	17	0	134	4	0	M	18	8558	0	100

8558	20	0	134	9	0	F	17	8557	0	200
5499	19	0	135	13	0	F	36	5500	0	200
5500	5	0	135	4	0	M	39	5499	0	100
5501	5	0	171	19	0	F	15	0	0	400
5532	19	0	197	2	0	M	17	5534	0	100
5534	8	0	197	16	0	F	22	5532	0	200
5543	6	0	136	1	0	M	35	6393	0	100
6393	17	0	136	15	0	F	35	5543	0	200
7012	6	0	136	17	0	M	9	0	0	500
7232	6	0	136	17	0	F	7	0	0	600
6882	6	0	136	17	0	M	5	0	0	500
6394	6	0	136	17	0	M	1	0	0	700
5557	1	0	171	18	0	F	22	5558	0	400
5558	6	0	136	17	0	M	35	5557	0	300
5560	19	0	132	12	0	M	26	5561	0	100
5561	16	0	132	11	0	F	22	5560	0	200
4776	17	0	201	10	0	M	23	5583	0	100
5583	19	0	201	16	0	F	20	5582	0	200
5626	5	0	208	1	0	M	35	7798	0	100
7798	16	0	208	2	0	F	34	5626	0	200
7965	1	0	137	12	0	M	22	5627	0	100
5627	17	0	137	19	0	F	18	7965	0	200
8943	1	0	137	12	0	M	23	0	0	300
5867	1	0	137	12	0	M	28	5864	0	100
5864	10	0	137	2	0	F	22	5867	0	200
6641	1	0	137	10	0	M	3	0	0	700
5863	10	0	224	12	0	M	38	7884	0	100
7884	1	0	224	10	0	F	38	5863	0	200
5654	5	0	208	19	0	M	27	5655	0	100
5655	19	0	208	2	0	F	29	5654	0	200
5658	16	0	138	6	0	F	38	6097	0	200
6097	18	0	138	3	0	M	34	5658	0	100
7097	18	0	138	16	0	F	8	0	0	600
7096	18	0	138	16	0	M	8	0	0	500
6986	18	0	138	16	0	M	6	0	0	500
6958	18	0	138	1	0	M	37	5674	0	100
5674	20	0	138	15	0	F	42	6958	0	200
7731	18	0	138	20	0	M	8	0	0	500
5676	13	0	139	20	0	M	18	5678	0	100
5678	9	0	139	10	0	F	21	5676	0	200
5677	14	0	220	17	0	F	44	8519	0	200
8519	2	0	220	20	0	M	21	5677	0	100
5687	18	0	140	15	0	M	20	5688	0	100

5688	6	0	140	13	0	F	19	5687	0	200
6075	18	0	140	6	0	F	2	0	0	700
5711	20	0	5	4	0	M	43	5710	0	100
5710	3	0	5	17	0	F	40	5711	0	200
5731	7	0	11	8	0	M	35	5732	0	100
5732	2	0	11	16	0	F	29	5731	0	200
5743	20	0	70	7	0	M	29	5744	0	100
5744	11	0	70	8	0	F	27	5743	0	200
5752	17	0	141	7	0	M	37	5751	0	100
5751	4	0	141	13	0	F	33	5752	0	200
6723	17	0	141	4	0	M	3	0	0	700
5755	6	0	142	1	0	M	39	8392	0	100
8392	13	0	142	1	0	F	40	5755	0	200
742	6	0	142	13	0	F	7	0	0	600
5756	6	0	142	13	0	M	20	0	0	300
5757	2	0	171	13	0	F	18	0	0	400
8387	13	0	80	10	0	F	41	5776	0	200
5776	3	0	80	6	0	M	41	8387	0	100
6725	16	0	143	8	0	F	63	0	0	200
6726	5	0	143	16	0	M	19	5777	0	100
5777	20	0	143	3	0	F	23	6726	0	200
5782	3	0	144	4	0	F	36	5785	0	200
5785	12	0	144	3	0	M	38	5782	0	100
8843	12	0	144	3	0	M	20	0	0	300
8680	12	0	144	3	0	M	17	0	0	300
7296	12	0	144	3	0	M	7	0	0	500
6690	12	0	144	3	0	F	3	0	0	700
5810	9	0	107	19	0	M	39	8490	0	100
8490	6	0	107	8	0	F	23	5810	0	200
7524	1	0	145	18	0	F	20	0	0	200
8919	12	0	171	1	0	F	23	0	0	400
6417	12	0	145	1	0	M	31	5817	0	100
5817	9	0	145	20	0	F	31	6417	0	200
8021	12	0	145	9	0	M	12	0	0	500
6787	9	0	171	20	0	F	43	0	0	400
6788	5	0	145	9	0	F	4	0	0	600
6625	1	0	146	9	0	F	28	6232	0	200
6232	12	0	146	5	0	M	22	6625	0	100
6628	7	0	146	1	0	F	3	0	0	700
6233	8	0	119	2	0	M	14	0	0	300
5818	13	0	147	6	0	M	21	0	0	300
5823	13	0	147	20	0	F	54	0	0	200
5825	8	0	147	13	0	M	37	5828	0	100

5828	1	0	147	2	0	F	24	5825	0	200
6743	8	0	147	1	0	M	4	0	0	500
6545	8	0	147	1	0	M	2	0	0	700
5827	2	0	148	17	0	F	56	0	0	200
7286	4	0	148	10	0	M	47	6666	0	100
6666	1	0	148	2	0	F	33	7286	0	200
5833	18	0	149	16	0	M	47	6174	0	100
6174	6	0	149	20	0	F	14	5833	0	200
5834	18	0	149	4	0	M	29	5837	0	100
5837	14	0	149	8	0	F	17	5834	0	200
5835	8	0	171	17	0	F	40	0	0	400
6464	14	0	171	8	0	F	2	0	0	700
5842	1	0	171	18	0	F	14	0	0	400
5854	11	0	150	1	0	M	30	5855	0	100
5855	9	0	150	3	0	F	35	5854	0	200
8043	11	0	150	9	0	M	13	0	0	300
5881	13	0	151	2	0	M	35	5883	0	100
5883	10	0	151	17	0	F	30	5881	0	200
9169	13	0	151	4	0	M	10	0	0	500
7361	13	0	151	10	0	M	7	0	0	500
6829	13	0	151	10	0	F	5	0	0	600
5902	10	0	152	20	0	M	24	5947	0	100
5947	16	0	152	11	0	F	25	5902	0	200
6570	10	0	152	16	0	F	2	0	0	700
5946	16	0	153	15	0	M	54	6081	0	100
6081	14	0	153	6	0	F	43	5946	0	200
7736	16	0	153	14	0	F	8	0	0	600
6692	16	0	153	14	0	M	3	0	0	700
8858	16	0	153	11	0	M	21	0	0	300
6451	13	0	153	19	0	M	17	5948	0	100
5948	16	0	153	11	0	F	23	6451	0	200
5905	12	0	171	4	0	F	29	0	0	400
7551	13	0	230	16	0	M	26	2926	0	100
2926	16	0	230	18	0	F	18	5925	0	200
5930	12	0	154	3	0	M	31	5929	0	100
5929	5	0	154	4	0	F	24	5930	0	200
6801	12	0	154	5	0	F	4	0	0	600
6640	12	0	154	5	0	M	3	0	0	700
6352	12	0	154	5	0	F	1	0	0	700
5945	5	0	135	10	0	M	25	6932	0	100
6932	9	0	135	14	0	F	27	5945	0	200
5954	3	0	200	10	0	M	40	5955	0	100
5955	7	0	200	8	0	F	18	5954	0	200

5960	20	0	155	6	0	M	40	5961	0	100
5961	1	0	155	17	0	F	37	5960	0	200
7455	20	0	155	1	0	F	7	0	0	600
6583	20	0	155	1	0	F	2	0	0	700
5983	13	0	156	20	0	M	30	5984	0	100
5984	1	0	156	18	0	F	46	5983	0	200
7020	13	0	156	1	0	F	7	0	0	600
5987	8	0	147	13	0	M	29	8546	0	100
8546	4	0	147	17	0	F	17	5987	0	200
6682	8	0	147	3	0	F	3	0	0	700
5996	3	0	148	9	0	M	25	5997	0	100
5997	11	0	148	19	0	F	19	5996	0	200
6026	3	0	148	15	0	M	44	6027	0	100
6027	16	0	148	14	0	F	41	6026	0	200
6738	3	0	148	15	0	M	4	0	0	500
6035	1	0	206	3	0	M	29	6036	0	100
6036	16	0	206	14	0	F	21	6035	0	200
6040	10	0	149	2	0	M	17	6043	0	100
6043	8	0	149	9	0	F	19	6040	0	200
6041	8	0	149	13	0	M	40	6042	0	100
6042	9	0	149	15	0	F	62	6041	0	200
6936	8	0	149	9	0	F	6	0	0	600
6048	4	0	150	2	0	M	46	6049	0	100
6049	11	0	150	15	0	F	56	6048	0	200
6050	4	0	150	11	0	M	19	6051	0	100
6051	19	0	150	1	0	F	12	6050	0	200
6100	11	0	151	16	0	F	30	6104	0	200
6104	1	0	151	8	0	M	40	6100	0	100
7560	11	0	151	1	0	F	7	0	0	600
6748	11	0	151	1	0	M	4	0	0	500
6362	11	0	151	1	0	M	1	0	0	700
6440	1	0	152	8	0	F	30	6439	0	200
6439	12	0	152	7	0	M	25	6440	0	100
6671	12	0	152	1	0	M	3	0	0	700
6441	12	0	152	1	0	M	1	0	0	700
6101	7	0	152	1	0	F	43	0	0	400
6337	11	0	151	16	0	M	24	6338	0	100
6338	10	0	151	8	0	F	22	6337	0	200
6629	11	0	151	10	0	F	3	0	0	700
6122	6	0	152	4	0	F	59	0	0	200
6124	18	0	152	6	0	M	34	6126	0	100
6126	11	0	152	19	0	F	19	6124	0	200
8099	18	0	171	4	0	F	13	0	0	400

7915	18	0	171	4	0	F	10	0	0	600
8464	18	0	152	6	0	M	29	8416	0	100
8416	13	0	152	14	0	F	30	8464	0	200
6130	12	0	209	4	0	F	17	2899	0	200
2899	8	0	209	20	0	M	18	6130	0	100
6148	15	0	171	18	0	F	23	0	0	400
6149	1	0	9	5	0	F	23	7575	0	200
7575	15	0	9	12	0	M	20	6149	0	100
6150	18	0	60	20	0	F	22	6153	0	200
6153	15	0	60	10	0	M	29	6150	0	100
6151	5	0	66	1	0	F	22	6152	0	200
6152	19	0	66	8	0	M	29	6151	0	100
6154	18	0	138	3	0	M	15	0	0	300
6155	12	0	153	14	0	M	21	6156	0	100
6156	6	0	153	10	0	F	18	6155	0	200
6157	12	0	153	6	0	F	0	0	0	700
7618	9	0	187	15	0	M	27	0	0	300
6161	12	0	154	2	0	F	47	6162	0	200
6162	10	0	154	5	0	M	47	6161	0	100
7977	12	0	171	10	0	F	11	0	0	400
8770	12	0	171	10	0	F	19	0	0	400
6163	10	0	155	12	0	M	30	6164	0	100
6164	13	0	155	10	0	F	27	6163	0	200
6165	10	0	155	13	0	M	0	0	0	700
6169	20	0	171	1	0	F	42	0	0	400
7077	15	0	171	20	0	F	10	0	0	600
6175	12	0	7	20	0	M	17	0	0	300
6178	18	0	156	15	0	F	56	6179	0	200
6179	13	0	156	2	0	M	58	6178	0	100
8747	13	0	171	18	0	F	19	0	0	400
6180	13	0	156	18	0	M	21	6181	0	100
6181	1	0	156	17	0	F	19	6180	0	200
6202	1	0	157	16	0	M	30	6203	0	100
6203	17	0	157	13	0	F	17	6202	0	200
6205	1	0	157	4	0	M	7	0	0	500
6212	13	0	15	2	0	M	20	0	0	300
8570	12	0	171	15	0	F	16	0	0	400
6215	4	0	158	7	0	M	26	6216	0	100
6216	15	0	158	5	0	F	22	6215	0	200
6534	4	0	158	15	0	F	2	0	0	700
6227	2	0	159	5	0	F	29	6228	0	200
6228	14	0	159	15	0	M	29	6227	0	100
7279	14	0	159	2	0	F	10	0	0	600

6229	14	0	159	2	0	F	0	0	0	700
6244	7	0	160	9	0	M	43	6245	0	100
6245	1	0	160	18	0	F	41	6244	0	200
6243	7	0	171	1	0	F	22	0	0	400
6663	7	0	160	1	0	F	3	0	0	700
6247	7	0	161	12	0	M	26	6248	0	100
6248	11	0	161	5	0	F	35	6247	0	200
7399	7	0	161	11	0	F	9	0	0	600
6330	6	0	162	1	0	M	62	6239	0	100
6329	16	0	162	10	0	F	51	6330	0	200
6331	6	0	162	16	0	M	24	6334	0	100
6334	11	0	162	12	0	F	19	6331	0	200
6333	11	0	163	5	0	M	48	6332	0	100
6332	12	0	163	14	0	F	50	6333	0	200
7699	11	0	163	12	0	M	9	0	0	500
7694	11	0	171	12	0	F	13	0	0	400
6340	3	0	171	17	0	F	18	0	0	400
6342	2	0	171	3	0	F	16	0	0	400
6901	15	0	164	19	0	F	5	0	0	600
5772	15	0	164	9	0	M	23	5773	0	100
5773	19	0	164	2	0	F	22	5772	0	200
6343	15	0	164	19	0	F	1	0	0	700
6346	11	0	165	9	0	M	27	6347	0	100
6347	2	0	165	18	0	F	18	6346	0	200
6348	11	0	165	2	0	F	1	0	0	700
6732	11	0	165	6	0	F	4	0	0	600
6368	9	0	166	17	0	F	35	6369	0	200
6369	13	0	166	10	0	M	32	6368	0	100
6370	13	0	166	9	0	M	1	0	0	700
6372	18	0	167	20	0	M	36	6371	0	100
6371	17	0	167	3	0	F	36	6372	0	200
7137	18	0	167	17	0	M	10	0	0	500
7136	18	0	167	17	0	M	8	0	0	500
6756	18	0	167	17	0	M	4	0	0	500
6376	3	0	171	4	0	F	17	0	0	400
7489	16	0	168	1	0	M	54	7490	0	100
7490	10	0	168	12	0	F	34	7489	0	200
8576	16	0	168	10	0	M	17	0	0	300
7559	16	0	168	10	0	M	14	0	0	300
7547	16	0	168	10	0	M	10	0	0	500
6381	9	0	171	19	0	F	21	0	0	400
6385	11	0	169	8	0	M	35	6386	0	100
6386	20	0	169	3	0	F	22	6385	0	200

6750	11	0	169	20	0	F	4	0	0	600
6387	11	0	169	20	0	M	1	0	0	700
6409	12	0	171	2	0	F	16	0	0	400
6420	3	0	171	20	0	F	16	0	0	400
6430	7	0	11	11	0	M	23	0	0	300
6431	5	0	12	2	0	M	21	8545	0	100
8545	12	0	12	13	0	F	20	6431	0	200
7762	12	0	12	6	0	M	42	7765	0	100
7765	13	0	12	1	0	F	41	7762	0	200
8694	12	0	12	13	0	M	18	0	0	300
7763	1	0	12	18	0	F	61	0	0	200
6435	5	0	170	6	0	F	1	0	0	700
7347	5	0	170	12	0	M	33	0	0	300
7398	5	0	170	6	0	F	9	0	0	600
6448	19	0	171	6	0	F	19	0	0	400
6619	1	0	18	20	0	M	17	0	0	300
6449	8	0	171	6	0	F	17	0	0	400
7122	16	0	22	1	0	M	19	0	0	300
6458	8	0	178	1	0	M	21	6459	0	100
6459	7	0	178	8	0	F	21	6458	0	200
6457	8	0	178	7	0	F	2	0	0	700
6462	2	0	179	17	0	M	30	6461	0	100
6461	16	0	179	6	0	F	27	6462	0	200
6836	2	0	179	16	0	M	5	0	0	500
6463	2	0	179	16	0	F	2	0	0	700
6478	19	0	21	14	0	M	33	6480	0	100
6480	3	0	21	12	0	F	19	6478	0	200
6491	12	0	171	13	0	F	15	0	0	400
6492	4	0	171	2	0	F	37	0	0	400
6493	10	0	171	14	0	F	19	0	0	400
6494	4	0	171	12	0	F	20	0	0	400
6495	20	0	171	15	0	F	16	0	0	400
7484	15	0	180	19	0	M	40	7237	0	100
7237	18	0	180	20	0	F	42	7484	0	200
7241	4	0	171	18	0	F	19	0	0	400
7239	4	0	171	18	0	F	11	0	0	400
7240	4	0	180	18	0	M	10	0	0	500
7486	15	0	180	20	0	M	7	0	0	500
6500	15	0	180	20	0	F	22	7436	0	200
7436	4	0	180	15	0	M	22	6500	0	100
8748	13	0	181	15	0	M	19	0	0	300
6501	13	0	181	15	0	M	22	6752	0	100
6752	8	0	181	9	0	F	18	6501	0	200

7942	2	0	182	20	0	M	27	7110	0	100
7110	6	0	182	8	0	F	20	7942	0	200
6502	2	0	182	6	0	M	2	0	0	700
6514	12	0	183	5	0	M	71	0	0	300
9168	12	0	183	20	0	M	46	0	0	300
8378	12	0	183	20	0	M	43	8379	0	100
8379	7	0	183	17	0	F	39	8378	0	200
6531	10	0	171	2	0	F	20	0	0	400
6538	9	0	184	10	0	F	31	6539	0	200
6540	8	0	184	3	0	M	2	0	0	700
6539	8	0	184	9	0	M	18	6538	0	100
6542	17	0	185	20	0	M	2	0	0	700
6543	17	0	185	4	0	M	23	6544	0	100
6544	20	0	185	19	0	F	23	6543	0	200
6551	5	0	186	8	0	M	33	6550	0	100
6550	3	0	186	16	0	F	32	6551	0	200
7695	5	0	171	3	0	F	15	0	0	400
7703	5	0	186	3	0	M	10	0	0	500
6552	5	0	186	3	0	F	2	0	0	700
6553	9	0	187	17	0	M	36	6554	0	100
6554	16	0	187	5	0	F	46	6553	0	200
6555	9	0	187	16	0	F	2	0	0	700
6618	9	0	171	16	0	F	15	0	0	400
7006	9	0	187	16	0	M	19	6561	0	100
6561	17	0	187	12	0	F	27	7006	0	200
6557	16	0	231	15	0	F	18	8506	0	200
8506	14	0	231	6	0	M	22	6557	0	100
6564	16	0	188	5	0	F	23	6565	0	200
6565	6	0	188	9	0	M	22	6564	0	100
6563	6	0	188	16	0	F	2	0	0	700
6591	6	0	188	16	0	M	26	6592	0	100
6592	15	0	188	3	0	F	19	6591	0	200
6590	6	0	188	15	0	M	3	0	0	700
6616	20	0	189	5	0	M	30	6613	0	100
6613	4	0	189	6	0	F	31	6616	0	200
6617	20	0	189	4	0	M	3	0	0	700
6635	16	0	171	19	0	F	31	0	0	400
6637	7	0	190	15	0	M	32	6638	0	100
6638	4	0	190	18	0	F	31	6637	0	200
7421	7	0	190	4	0	F	9	0	0	600
6770	7	0	190	4	0	F	4	0	0	600
6639	7	0	190	4	0	M	3	0	0	700
6642	20	0	185	2	0	M	22	7673	0	100

7673	10	0	185	13	0	F	28	6642	0	200
6649	9	0	23	11	0	M	37	8478	0	100
8478	4	0	23	7	0	F	27	6649	0	200
8522	20	0	191	17	0	M	24	6660	0	100
6660	1	0	191	11	0	F	21	8522	0	200
8146	20	0	191	10	0	M	43	0	0	300
8147	20	0	171	17	0	F	14	0	0	400
6668	15	0	191	17	0	F	3	0	0	700
6680	17	0	27	12	0	M	43	6679	0	100
6679	5	0	27	11	0	F	59	6680	0	200
6701	1	0	192	11	0	M	18	6702	0	100
6702	14	0	192	18	0	F	21	6701	0	200
6700	1	0	192	14	0	M	3	0	0	700
6714	17	0	51	16	0	F	50	0	0	200
6715	12	0	51	17	0	M	33	0	0	300
8493	7	0	171	9	0	F	34	0	0	400
6736	11	0	193	4	0	M	37	6737	0	100
6737	1	0	193	3	0	F	46	6736	0	200
7207	11	0	193	1	0	M	17	0	0	300
6735	11	0	193	1	0	F	4	0	0	600
6746	15	0	194	13	0	M	40	6745	0	100
6745	2	0	194	18	0	F	37	6746	0	200
7528	15	0	194	2	0	F	10	0	0	600
6747	15	0	194	2	0	F	4	0	0	600
6755	2	0	195	14	0	M	45	0	0	300
6754	2	0	195	18	0	M	24	7352	0	100
7352	19	0	195	7	0	F	26	6754	0	200
6767	6	0	196	20	0	M	22	6768	0	100
6768	1	0	196	3	0	F	18	6767	0	200
6766	6	0	196	1	0	M	4	0	0	500
6773	19	0	197	4	0	M	4	0	0	500
9129	19	0	197	13	0	M	30	0	0	300
8709	19	0	171	13	0	F	18	0	0	400
8119	19	0	171	13	0	F	14	0	0	400
8520	19	0	197	13	0	M	22	8562	0	100
8562	1	0	197	20	0	F	14	8520	0	200
8528	9	0	69	6	0	M	22	6797	0	100
6797	17	0	69	9	0	F	24	8528	0	200
6804	2	0	198	17	0	M	33	6805	0	100
6805	11	0	198	8	0	F	40	6804	0	200
7162	2	0	198	11	0	M	11	0	0	500
6823	9	0	171	2	0	F	36	0	0	400
7881	15	0	199	9	0	M	10	0	0	500

6828	15	0	199	9	0	F	5	0	0	600
6827	13	0	199	8	0	M	47	6826	0	100
6826	9	0	199	2	0	F	37	6827	0	200
6933	13	0	199	9	0	F	6	0	0	600
6851	9	0	199	8	0	F	45	0	0	200
9074	12	0	199	9	0	M	27	0	0	300
6857	3	0	200	8	0	M	43	8547	0	100
8547	13	0	200	5	0	F	22	6857	0	200
6859	3	0	200	13	0	M	5	0	0	500
6862	4	0	201	14	0	M	25	6863	0	100
6863	17	0	201	13	0	F	25	6862	0	200
7022	4	0	201	17	0	F	9	0	0	600
6864	4	0	201	17	0	F	5	0	0	600
6880	2	0	202	10	0	M	49	6879	0	100
6879	1	0	202	18	0	F	49	6880	0	200
7632	2	0	202	1	0	M	12	7629	0	100
7629	2	0	202	1	0	F	10	7632	0	200
6881	2	0	202	1	0	M	5	0	0	500
6885	3	0	203	11	0	M	25	6886	0	100
6886	18	0	203	17	0	F	22	6885	0	200
6884	3	0	203	18	0	F	5	0	0	600
7620	3	0	82	2	0	M	34	6897	0	100
6897	4	0	82	2	0	F	35	7620	0	200
6923	11	0	204	3	0	M	57	6922	0	100
6922	4	0	204	18	0	F	29	6923	0	200
7516	11	0	171	4	0	F	12	0	0	400
6924	11	0	204	4	0	M	11	0	0	500
7521	11	0	204	4	0	M	11	0	0	500
6943	11	0	205	6	0	M	48	6942	0	100
6942	10	0	205	11	0	F	41	6943	0	200
6944	11	0	205	10	0	F	6	0	0	600
8040	11	0	205	10	0	M	13	0	0	300
8609	11	0	205	10	0	M	19	0	0	300
6968	1	0	206	11	0	M	24	6969	0	100
6969	14	0	206	7	0	F	27	6968	0	200
6967	1	0	206	14	0	M	6	0	0	500
7827	19	0	232	10	0	M	28	6985	0	100
6985	5	0	232	9	0	F	35	7827	0	200
6995	8	0	19	4	0	M	19	0	0	300
6997	8	0	19	4	0	M	28	0	0	300
6998	11	0	233	3	0	M	19	0	0	300
7017	2	0	233	9	0	F	53	0	0	200
7024	17	0	207	11	0	M	36	7023	0	100

7023	1	0	207	4	0	F	23	7024	0	200
7025	17	0	207	1	0	F	10	0	0	600
7036	12	0	61	20	0	M	29	7037	0	100
7037	15	0	61	18	0	F	46	7036	0	200
7041	17	0	171	1	0	F	34	0	0	400
7052	5	0	208	3	0	M	26	7051	0	100
7051	7	0	208	10	0	F	44	7052	0	200
7098	5	0	208	7	0	F	9	0	0	600
7054	7	0	171	20	0	F	44	0	0	400
7157	15	0	171	7	0	F	11	0	0	400
7156	15	0	171	7	0	F	9	0	0	600
7057	12	0	235	5	0	M	41	7056	0	100
7056	2	0	235	16	0	F	43	7057	0	200
7058	11	0	171	20	0	F	16	0	0	400
7062	19	0	53	4	0	M	22	7531	0	100
7531	7	0	53	18	0	F	20	7062	0	200
7067	2	0	234	10	0	M	40	0	0	300
7047	2	0	234	18	0	M	44	7046	0	100
7046	6	0	234	14	0	F	36	7047	0	200
3435	2	0	234	6	0	M	21	0	0	300
7111	13	0	209	3	0	F	22	8279	0	200
8279	8	0	209	7	0	M	49	7111	0	100
8799	8	0	171	5	0	F	20	0	0	400
7131	15	0	236	12	0	F	28	7602	0	200
7602	10	0	236	3	0	M	20	7131	0	100
7134	10	0	236	16	0	M	35	0	0	300
7143	2	0	171	20	0	F	39	0	0	400
7145	8	0	171	2	0	F	6	0	0	600
7158	12	0	235	9	0	M	10	0	0	500
7166	1	0	210	4	0	M	37	7165	0	100
7165	8	0	210	12	0	F	33	7166	0	200
7167	1	0	210	8	0	F	7	0	0	600
7179	18	0	237	5	0	M	39	7172	0	100
7172	17	0	237	9	0	F	19	7179	0	200
7201	1	0	211	10	0	M	27	5553	0	100
5553	5	0	211	14	0	F	15	7201	0	200
7191	1	0	171	5	0	F	11	0	0	400
7683	11	0	212	19	0	M	58	7192	0	100
7192	14	0	212	15	0	F	34	7683	0	200
7687	11	0	212	2	0	M	21	7496	0	100
7496	9	0	212	15	0	F	23	7687	0	200
8565	19	0	212	12	0	M	19	7685	0	100
7685	11	0	212	2	0	F	20	8565	0	200

7195	8	0	213	14	0	M	27	7194	0	100
7194	2	0	213	9	0	F	33	7195	0	200
7301	8	0	213	2	0	M	11	0	0	500
7197	1	0	214	17	0	M	26	7196	0	100
7196	18	0	214	11	0	F	30	7197	0	200
7306	1	0	214	18	0	M	9	0	0	500
7204	6	0	6	16	0	M	40	7775	0	100
7775	11	0	6	4	0	F	42	7204	0	200
8653	17	0	171	11	0	F	17	0	0	400
7209	14	0	231	15	0	M	20	8534	0	100
8534	2	0	231	12	0	F	16	7209	0	200
7223	18	0	189	13	0	F	46	0	0	200
8940	20	0	189	18	0	M	27	0	0	300
7245	12	0	183	8	0	M	27	7244	0	100
7244	19	0	183	3	0	F	23	7245	0	200
7251	5	0	170	3	0	M	41	7250	0	100
7250	4	0	170	8	0	F	32	7251	0	200
7258	3	0	203	18	0	F	47	0	0	200
7262	4	0	215	7	0	M	25	8502	0	100
8502	6	0	215	3	0	F	23	7262	0	200
8306	6	0	215	15	0	M	61	0	0	300
8420	6	0	215	3	0	M	43	8421	0	100
8421	2	0	215	19	0	F	35	8420	0	200
7263	12	0	238	19	0	M	24	0	0	300
7267	4	0	4	9	0	M	28	8532	0	100
8532	13	0	4	17	0	F	15	7267	0	200
7280	4	0	216	11	0	M	37	7281	0	100
7281	20	0	216	15	0	F	37	7280	0	200
7282	4	0	171	20	0	F	11	0	0	400
8592	4	0	216	20	0	F	13	0	0	400
7333	4	0	217	9	0	M	47	7332	0	100
7332	13	0	217	5	0	F	41	7333	0	200
7415	4	0	217	13	0	M	9	0	0	500
7335	4	0	217	14	0	M	47	7334	0	100
7334	15	0	217	10	0	F	43	7335	0	200
8568	11	0	218	1	0	M	18	8567	0	100
8567	4	0	218	15	0	F	15	8568	0	200
8166	11	0	218	7	0	M	42	8165	0	100
8165	1	0	218	4	0	F	45	8166	0	200
7341	14	0	239	6	0	M	34	0	0	300
7345	16	0	240	14	0	M	34	0	0	300
7349	10	0	84	3	0	M	33	0	0	300
7311	7	0	171	10	0	F	20	0	0	400

7355	12	0	233	19	0	F	27	7356	0	200
7356	11	0	233	10	0	M	31	7355	0	100
7360	4	0	4	20	0	M	11	0	0	500
7363	3	0	171	14	0	F	37	0	0	400
7376	8	0	219	1	0	M	29	7375	0	100
7375	11	0	219	14	0	F	29	7376	0	200
7377	8	0	219	11	0	F	7	0	0	600
8095	1	0	157	17	0	M	33	7380	0	100
7380	19	0	157	17	0	F	20	8095	0	200
7385	11	0	77	5	0	F	52	0	0	200
7392	14	0	240	13	0	F	8	0	0	600
7394	17	0	223	2	0	M	27	7395	0	100
7395	7	0	223	8	0	F	27	7394	0	200
7429	10	0	96	12	0	M	19	8526	0	100
8526	19	0	96	13	0	F	19	7429	0	200
7431	6	0	24	5	0	M	23	0	0	300
7432	20	0	96	15	0	M	29	7442	0	100
7442	18	0	96	4	0	F	20	7432	0	200
7433	4	0	24	5	0	M	47	0	0	300
7441	14	0	171	12	0	F	19	0	0	400
7452	13	0	1	7	0	M	56	7451	0	100
7451	3	0	1	6	0	F	55	7452	0	200
7454	11	0	165	13	0	M	19	0	0	300
7458	18	0	241	13	0	M	23	0	0	300
7460	2	0	220	16	0	M	37	7459	0	100
7459	11	0	220	9	0	F	33	7460	0	200
7512	2	0	220	11	0	M	14	0	0	300
7462	18	0	237	20	0	M	25	7461	0	100
7461	15	0	237	1	0	F	23	7462	0	200
7546	7	0	171	14	0	F	29	0	0	400
7473	3	0	171	7	0	F	9	0	0	600
5657	10	0	242	11	0	M	26	7474	0	100
7474	11	0	242	20	0	F	23	5657	0	200
8533	13	0	3	17	0	M	21	7532	0	100
7532	6	0	3	18	0	F	19	8533	0	200
8418	13	0	221	1	0	F	43	0	0	200
7002	3	0	221	13	0	M	18	7535	0	100
7535	15	0	221	20	0	F	20	7002	0	200
7539	1	0	171	8	0	F	23	0	0	400
7621	12	0	146	2	0	M	34	7541	0	100
7541	4	0	146	2	0	F	26	7621	0	200
7264	8	0	112	13	0	M	25	7545	0	100
7545	10	0	112	2	0	F	24	7264	0	200

7555	13	0	139	5	0	M	29	7552	0	100
7552	6	0	139	5	0	F	29	7555	0	200
7561	15	0	163	2	0	F	34	7562	0	200
7562	11	0	163	16	0	M	30	7561	0	100
7565	2	0	222	8	0	F	54	7564	0	200
7564	5	0	222	19	0	M	55	7565	0	100
7605	5	0	222	2	0	F	10	0	0	600
7585	5	0	222	2	0	F	9	0	0	600
7567	6	0	225	2	0	M	52	7661	0	100
7661	1	0	225	18	0	F	25	7567	0	200
7569	17	0	223	6	0	M	57	7568	0	100
7568	6	0	223	12	0	F	52	7569	0	200
7594	17	0	223	6	0	M	13	0	0	300
7595	17	0	223	6	0	F	9	0	0	600
7571	10	0	224	13	0	M	57	7570	0	100
7570	7	0	224	20	0	F	57	7571	0	200
7633	10	0	224	7	0	M	13	0	0	300
7631	10	0	171	7	0	F	11	0	0	400
7628	10	0	224	7	0	F	9	0	0	600
7574	16	0	35	2	0	M	19	8544	0	100
8544	14	0	35	20	0	F	17	7574	0	200
7577	6	0	225	2	0	M	27	7578	0	100
7578	10	0	225	19	0	F	34	7577	0	200
7614	1	0	171	19	0	F	35	0	0	400
7622	10	0	171	8	0	F	22	0	0	400
7623	7	0	171	1	0	F	29	0	0	400
7666	19	0	226	18	0	M	27	7624	0	100
7624	12	0	226	4	0	F	29	7666	0	200
7625	1	0	171	8	0	F	33	0	0	400
7657	17	0	171	1	0	M	11	0	0	500
7656	13	0	171	1	0	F	9	0	0	600
7638	16	0	171	10	0	F	8	0	0	600
7643	7	0	190	14	0	M	10	0	0	500
7667	13	0	166	12	0	M	35	7664	0	100
7664	15	0	166	19	0	F	35	7667	0	200
7689	9	0	100	20	0	M	35	7678	0	100
7678	12	0	100	3	0	F	32	7689	0	200
8589	2	0	222	1	0	M	35	7679	0	100
7679	3	0	222	19	0	F	30	8589	0	200
7681	11	0	226	13	0	F	29	7688	0	200
7688	19	0	226	8	0	M	29	7681	0	100
7690	19	0	226	6	0	M	55	0	0	300
7705	19	0	226	11	0	M	15	0	0	300

7712	20	0	171	13	0	F	8	0	0	600
7718	19	0	171	15	0	F	40	0	0	400
7738	8	0	219	2	0	F	63	0	0	200
7747	2	0	243	7	0	M	24	8446	0	100
8446	16	0	243	13	0	F	37	7747	0	200
7771	7	0	227	3	0	M	39	7770	0	100
7770	4	0	227	5	0	F	38	7771	0	200
8054	7	0	171	4	0	F	13	0	0	400
7780	18	0	214	6	0	M	9	0	0	500
7784	11	0	228	4	0	M	57	7783	0	100
7783	4	0	228	2	0	F	53	7784	0	200
8233	11	0	228	4	0	M	34	0	0	300
8096	11	0	228	4	0	M	25	8529	0	100
8529	2	0	228	9	0	F	19	8096	0	200
8455	12	0	238	10	0	M	33	7789	0	100
7789	15	0	238	8	0	F	31	8455	0	200
7792	4	0	171	12	0	F	35	0	0	400
7895	20	0	216	11	0	M	34	8429	0	100
8429	2	0	216	5	0	F	32	7895	0	200
8531	17	0	229	3	0	M	21	7920	0	100
7920	12	0	229	4	0	F	34	8531	0	200
7921	17	0	171	12	0	F	11	0	0	400
9926	13	0	230	8	0	M	41	7925	0	100
7925	6	0	230	7	0	F	42	9926	0	200
8671	13	0	230	6	0	M	21	0	0	300
8346	6	0	142	14	0	M	39	7940	0	100
7940	18	0	142	8	0	F	29	8346	0	200
8084	1	0	192	11	0	M	27	7947	0	100
7947	12	0	192	8	0	F	38	8084	0	200
7949	14	0	171	6	0	F	36	0	0	400
8816	13	0	171	14	0	F	20	0	0	400
7951	13	0	114	14	0	M	11	0	0	500
8003	7	0	171	15	0	F	24	0	0	400
8026	3	0	171	12	0	F	12	0	0	400
8087	18	0	171	7	0	F	43	0	0	400
8089	6	0	196	18	0	M	13	0	0	300
8101	3	0	203	17	0	M	50	8100	0	100
8100	5	0	203	1	0	F	49	8101	0	200
8114	20	0	246	13	0	M	14	0	0	300
8894	20	0	246	13	0	M	22	0	0	300
9107	20	0	171	13	0	F	30	0	0	400
8121	3	0	221	11	0	M	47	8120	0	100
8120	4	0	221	7	0	F	45	8121	0	200

8213	18	0	241	9	0	M	34	8432	0	100
8432	8	0	241	5	0	F	40	8132	0	200
8226	6	0	247	17	0	M	34	8225	0	100
8225	20	0	247	11	0	F	39	8226	0	200
8269	16	0	171	11	0	F	44	0	0	400
8303	1	0	171	7	0	F	36	0	0	400
8319	17	0	227	7	0	M	49	8390	0	100
8390	7	0	227	5	0	F	44	8319	0	200
8325	17	0	229	4	0	M	60	8326	0	100
8326	16	0	229	14	0	F	46	8325	0	200
8336	20	0	246	13	0	M	44	8337	0	100
8337	11	0	246	8	0	F	46	8336	0	200
8342	13	0	76	3	0	M	53	8343	0	100
8343	10	0	76	9	0	F	59	8342	0	200
8351	6	0	247	4	0	M	42	8434	0	100
8434	2	0	76	11	0	F	35	8351	0	200
8389	9	0	247	10	0	F	50	0	0	200
8394	10	0	242	18	0	F	51	0	0	200
8435	9	0	171	8	0	F	43	0	0	400
8535	3	0	245	5	0	M	28	0	0	300
8599	16	0	243	5	0	M	16	0	0	300
8861	17	0	244	18	0	M	34	0	0	300
8961	16	0	35	17	0	M	23	0	0	300
9015	9	0	171	16	0	F	29	0	0	400
9999	0	0	173	0	0	M	45	0	0	800

VITA

Carolyn Orbann was born in San Diego, California to Carl and Marilyn Orbann. After graduating from Chantilly High School in Chantilly, Virginia, she attended The George Washington University in Washington, D.C. She earned a Bachelor of Arts in Anthropology with a minor in Biology in May 1999. During her time at GWU, she held a long-term internship at the National Museum of Natural History in the Office of Repatriation.

She started graduate work in Anthropology at California State University, Chico, graduating with a Master's of Arts in 2006. While a student at Chico, she volunteered on the Physical Anthropology and Human Identification Laboratory's Forensic Recovery Team, assisted in planning and arranging two Forensic Anthropology Mini Conferences and worked with the Archaeological Research Program as a crew chief and osteological and zooarchaeological consultant.

In the fall of 2006, she started work on a doctoral degree in Anthropology at the University of Missouri, Columbia. During her doctoral work, she taught as an adjunct instructor in Anthropology at UM – Columbia and Westminster College. Currently, she holds a teaching position in the Department of Health Sciences at the University of Missouri – Columbia.