2017

A New Evolutionary Algorithm For Mining Noisy, Epistatic, Geospatial Survey Data Associated With Chagas Disease

John P. Hanley
University of Vermont

A NEW EVOLUTIONARY ALGORITHM FOR MINING NOISY, EPISTATIC, GEOSPATIAL SURVEY DATA ASSOCIATED WITH CHAGAS DISEASE

A Dissertation Presented

by

John P. Hanley

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
Specializing in Civil and Environmental Engineering

May, 2017

Defense Date: March 24, 2017
Dissertation Examination Committee:

Donna M. Rizzo, Ph.D., Advisor
Frances E. Carr, Ph.D., Chairperson
Eric M. Hernandez, Ph.D.
Arne Bomblies, Ph.D.
Lori Stevens, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of the Graduate College

ABSTRACT

The scientific community is just beginning to understand some of the profound affects that feature interactions and heterogeneity have on natural systems. Despite the belief that these nonlinear and heterogeneous interactions exist across numerous real-world systems (e.g., from the development of personalized drug therapies to market predictions of consumer behaviors), the tools for analysis have not kept pace. This research was motivated by the desire to mine data from large socioeconomic surveys aimed at identifying the drivers of household infestation by a Triatomine insect that transmits the life-threatening Chagas disease. To decrease the risk of transmission, our colleagues at the laboratory of applied entomology and parasitology have implemented mitigation strategies (known as Ecohealth interventions); however, limited resources necessitate the search for better risk models. Mining these complex Chagas survey data for potential predictive features is challenging due to imbalanced class outcomes, missing data, heterogeneity, and the non-independence of some features.

We develop an evolutionary algorithm (EA) to identify feature interactions in "Big Datasets" with desired categorical outcomes (e.g., disease or infestation). The method is non-parametric and uses the hypergeometric PMF as a fitness function to tackle challenges associated with using p-values in Big Data (e.g., p-values decrease inversely with the size of the dataset). To demonstrate the EA effectiveness, we first test the algorithm on three benchmark datasets. These include two classic Boolean classifier problems: (1) the 'majority-on' problem and (2) the multiplexer problem, as well as (3) a simulated single nucleotide polymorphism (SNP) disease dataset. Next, we apply the EA to real-world Chagas Disease survey data and successfully archived numerous high-order feature interactions associated with infestation that would not have been discovered using traditional statistics. These feature interactions are also explored using network analysis. The spatial autocorrelation of the genetic data (SNPs of Triatoma dimidiata) was captured using geostatistics. Specifically, a modified semivariogram analysis was performed to characterize the SNP data and help elucidate the movement of the vector within two villages. For both villages, the SNP information showed strong spatial autocorrelation albeit with different geostatistical characteristics (sills, ranges, and nuggets). These metrics were leveraged to create risk maps that suggest the more forested village had a sylvatic source of infestation, while the other village had a domestic/peridomestic source. This initial exploration into using Big Data to analyze disease risk shows that novel and modified existing statistical tools can improve the assessment of risk on a fine-scale.

CITATIONS

Material from this dissertation has been published in the following form:

Hanley, John P., Erin Jackson, Leslie A. Morrissey, Donna M. Rizzo, Brian L. Sprague, Indra Neil Sarkar, and Frances E. Carr. (2015). Geospatial and Temporal Analysis of Thyroid Cancer Incidence in a Rural Population. Thyroid, 25 (7), 812–822. doi:10.1089/thy.2015.0039.

AND

Material from this dissertation has been submitted for publication to Evolutionary Computation on April 7, 2017 in the following form:

Hanley, John P., Donna M. Rizzo, Jeffrey S. Buzas, and Margaret J. Eppstein. A Tandem Evolutionary Algorithm for Identifying Optimal Association Rules from Complex Data. Evolutionary Computation.

ACKNOWLEDGEMENTS

I would like to thank the Department of Civil and Environmental Engineering and the National Science Foundation Grant DBC-EID-1216193 for their financial support. I am grateful to my advisor Donna Rizzo for her guidance and tutelage throughout my time at UVM. Also, I would like to thank my collaborators on the myriad of projects I have worked on. First off, the principal investigators of the NSF grant Carlota Monroy, Sergio Melgar, Patricia Dorn, Lori Stevens, Sara Helms Cahan, Leslie Morrissey, and Donna Rizzo. My collaborators at the El Laboratorio de Entomología Aplicada y Parasitología at La Universidad San Carlos de Guatemala, Ciudad de Guatemala, Guatemala: Antonieta Rodas, Raquel Asuncion Lima, Gaby Rodas, Elizabeth Solórzano, Dulce Bustamante, Andrea Solares, and Salvador Castellanos. Also the Ministerio de Salud Pública de Guatemala personnel who were instrumental to our field work. Thanks to my fellow UVM graduate student on the NSF grant Lucia Orantes and her mother Olga Orantes for all her support in Guatemala. I would like to thank my collaborators on for my thyroid research not already mentioned; Erin Jackson, Brian L. Sprague, Indra Neil Sarkar, and Frances E. Carr. Also, my collaborators in developing the evolutionary algorithm Maggie Eppstein and Jeff Buzas. My officemates Lucas Howard and Scott Hamshaw. My Middlebury College advisor Steve Trombulak as well as Jon Isham and Pete Ryan. Finally and most importantly, I would like to thank the support of friends and family especially my mom, Buck, and Laura Obregon.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1: INTRODUCTION

The London cholera outbreaks of the mid 1800s are credited as one of the first times an environmental engineer assessed the risks associated with a disease (Buescher Jr.). The engineer, Mr. Grant, research into the water supply of the houses affected by Cholera on Albion Terrace that helped formulate the epidemiologist John Snow's hypothesis that the source of cholera was water borne and not the result of bad smells as hypothesized by Dr. Milroy (Hempel, 2007). Since that time, engineers have continued to help assess of the risk of diseases ranging from assessing varying climate change scenarios on the abundance of the malaria vector *Anopheles gambiae sensu lato* (Bomblies, 2012), the risks associated with transmission of the parasitic tape worm *Taenia solium* (Enander et al., 2010), or the risk of child-acquired, respiratory illnesses associated with the presence of *Enterococcus* spp on hands (Julian et al., 2013). This dissertation is an environmental engineer's attempt to geospatially assess the risk of Chagas disease and thyroid cancer.

1.1 Summary of Research Contributions

The primary contribution of this research lies in the development and modification of statistical tools to assess the risk of household infestation of *Triatoma dimidiata*, the principal vector of Chagas disease in Guatemala (WHO, 2015). These tools were applied using a unique set of socioeconomic, genetic and entomologic survey data collected from two towns in Jutiapa, Guatemala. The survey data were georeferenced at the household level and include next generation sequencing data on the *T. dimidiata* collected in domestic (i.e., houses) and peridomestic (i.e., areas immediately

surrounding the house such as a yard, chicken coops, wood piles, etc.) ecotopes. The first goal was to determine the risk of household infestation of *T. dimidiata* associated with numerous risk factors present in the socioeconomic and entomologic household surveys. This task was complicated by the presence of missing data, varied datatypes (e.g., nominal, discrete, and ordinal), inherent feature interactions associated with infestation, numerous combinations of risk factors, and implicit heterogeneity. Feature interactions are inherent in this data because, at a minimum, *T. dimidiata* need both shelter and a food source to successfully infest a household. One without the other will inhibit a successful infestation of a household. Heterogeneity is present in the diverse number of combinations of shelters and food sources that may be associated with successful infestation.

The large number of combinations of risk factors in the survey data made exhaustive search of all combinations unrealistic in terms of computational effort. As a result, one of the main contributions of this work is the development of a new algorithm capable of (1) efficiently searching large datasets ("Big Data") for multiple signals (i.e., true associations with an outcome, the opposite of noise), and (2) handling missing data as well as varied datatypes embedded in this large, epistatic, heterogeneous survey data. An evolutionary algorithm (EA) was designed to tackle this problem (Chapter 2). To address the challenges associated with *p*-values and Big Data (Lin et al., 2013; Nuzzo, 2014), the hypergeometric probability mass function (hypergeometric PMF) was introduced as a novel fitness function (i.e., measurement of signal strength). This new EA and fitness were then tested on a set of benchmark problems from the EA community

to assess the EA's ability to perform feature selection and reduce the search space. Next, the EA was tested on the *T. dimidiata* infestation datasets and was able to efficiently identify complex interactions associated with infestation (Chapter 3).

The next generation sequencing of the georeferenced *T. dimidiata* collected in the houses and their peridomestic ecotopes in the two villages in Jutiapa, Guatemala resulted in thousands of single nucleotide polymorphisms (SNPs). Smouse and Peakall (1999) developed a genetics-based algorithm to measure spatial autocorrelation using correlograms. However, given the impact of outliers and the inability to empirically determine confidence intervals for the binned correlations, we modified the Smouse and Peakall (1999) algorithm; and the correlogram was replaced with what is known as a semivariogram (Chapter 4). Rather than measure the similarity between spatially autocorrelated data, semivariograms measure the dissimilarity between paired data points, and are the preferred measure of spatial autocorrelation in the geostatistics community for a variety of reasons. The modified algorithm of Smouse and Peakall (1999) was then used to determine the range of spatial autocorrelation of the genetic structure of *T. dimidiata* in both towns. Because we were unable to find any empirical study that has determined how far *T. dimidiata* moves in the field, the range of spatial autocorrelation identified by our modified semivariogram was then used as a surrogate to vector movement to map the risk of infestation in both towns. These maps provide support to one of the towns having a sylvatic source of infestation, and the other having domestic and peridomestic sources of infestation.

In Chapter 5, we assess the risk of thyroid cancer using 14 years of spatially referenced incidences of thyroid cancer in Vermont. Unlike the Chagas disease datasets, the incidence of thyroid cancer was georeferenced to the larger zip code scale; and the datasets had minimal demographic information. The socioeconomic data available for analyzing risk was the US census data, which aggregates the population data over a zip code. Thus, we were unable to identify novel combinations of risk factors associated with thyroid cancer using this aggregated socioeconomic data. That being said, the geospatial analysis of the thyroid cancer did allow for the visualization of Hot Spots in Vermont.

Throughout the course of this research, novel and adapted statistical tools are developed and applied to disease datasets to assess risk. This work serves to highlight novel ways of analyzing Big Data that are becoming ubiquitous in research. While the methods presented here were successful in analyzing the risk associated with Chagas disease and thyroid cancer, these methods, like many traditional statistics that use *p*-values, should not be shoe-horned into tackling problems they were not specifically designed for. These methods are designed to be part of a larger tool kit when analyzing Big Data associated with complex systems.

1.2 Chagas Disease

Chagas disease is caused by the protozoan parasite, *Trypanosoma cruzi*, and is primarily spread via blood feeding insects in the order Hemiptera, family Reduviidae, and subfamily Triatominae (WHO, 2002). While the primary vector food source is vertebrates, *T. cruzi* only infects mammals (Rassi et al., 2010). Human impacts, such as deforestation for agrarian land use, have caused triatomines to adapt (Coura, 2015); and

one of the main vectors of Chagas disease, *T. dimidiata*, has adapted to human domestic and peridomestic environments (Waleckx et al., 2015a). This vector is endemic from Mexico all the way south to parts of Peru, Ecuador, Colombia, and Venezuela (WHO, 2002). People with Chagas disease often live in remote areas with poor sanitation, low socioeconomic status, and work manual labor jobs (Prata, 2001; Briceño-León et al., 2007). Approximately 70 million people in Latin America are at risk of infection with *T. cruzi* and ~5.7 million people are infected (Chagas, 2015). In Central America, Guatemala has the largest number of vector transmitted cases (~1,275) in 2010 (Chagas, 2015). Furthermore, Guatemala, El Salvador, and Honduras combined account for 85% of the new cases in Central America (Chagas, 2015). Chagas disease has an estimated disability-adjusted life years (DALY) of 546,000 (271,000–1,054,000), and is the second largest proportion of DALY in Latin America, after hookworm disease (Murray et al., 2012). The estimated annual health-care cost per Chagas patient in Latin America is ~$383 (range $207–$636); and the total annual cost to society (i.e., health-care plus productivity losses) per chronic Chagas disease patient in Latin America is ~$4,059 (range $3,569–$4,434) (Lee et al., 2013). Thus, the disease burden of Chagas disease exceeds other infectious diseases such as cholera and rotavirus (Lee et al., 2013).

Humans infected with *T. cruzi* can acquire Chagas disease by the transmission of the *T. cruzi* infected feces into the bite or open wound, or through the mucosa of the eye, nose, or mouth (Prata, 2001; Teixeira et al., 2006; Rassi et al., 2010). Another possible source is via consumption of the insect or infected feces in food items such as juice and vegetables, and possibly from eating wild meat (Rueda et al., 2014). Oral transmission is

believed to be the primary source of infection for wild animals (Coura, 2015); and the odoriferous glands of a marsupial infected with *T. cruzi* can directly transmit the parasite to humans (Coura, 2015).

Chagas disease is broken into three phases. The first is the *acute* phase, which may last 1–4 months after infection with *T. cruzi* (Prata, 2001, Stanaway and Roth, 2015). This phase is characterized by an increase in heart size, heart cell destruction, and depopulation of neurons. (Teixeira et al., 2006). This phase is asymptomatic in 95% of cases (Teixeira et al., 2006; Stanaway and Roth, 2015); however, for the remaining 5%, symptoms may include malaise, fever, jaundice, skin hemorrhages, enlargement of the liver, and muscle and joint pain (Prata, 2001; Teixeira et al., 2006; Rassi et al., 2010; Stanaway and Roth, 2015). Less than 1 in 2,500 infections result in death; the latter is usually attributed to encephalomyelitis or severe cardiac failure (Prata, 2001; Teixeira et al., 2006; Stanaway and Roth, 2015). The *indeterminate* phase is asymptomatic and can last 10–30 years or throughout a lifetime (Prata, 2001; Stanaway and Roth, 2015). Finally, the *chronic* phase, occurs in about one third of those infected and has symptoms that include heart disease (i.e., destruction of target heart cells), megaesophagus, megacolon, nervous system lesions, and sudden death (Prata, 2001; Teixeira et al., 2006; Rassi et al., 2010; Stanaway and Roth, 2015). Heart disease is one of the most common and deadly symptoms; however, there appears to be heterogeneity in Chagas-related heart disease with three distinct groups (Rassi et al., 2006). The 10-year mortality rate across all three groups is ~27%, but ranges from ~10 to ~84% (Rassi et al., 2006). In general, the relative risk ratio of mortality is approximately 1.74 for individuals with

Chagas disease compared to similar individuals without the disease (Cucunubá et al., 2016). In addition to the Chagas-related health effects, there is some evidence that Chagas is a risk factor for high blood pressure (Vicco et al., 2014), cognitive impairment in older adults (Lima-Costa et al., 2008), and ischemic stroke (Lima-Costa et al., 2008).

Currently, there is no preventive medicine for Chagas disease. Nonetheless, there are two anti-trypanosome drugs, nifurtimox and benznidazole, used to treat *T. cruzi* infections (Teixeira et al., 2006; Jannin and Villa, 2007; Rassi et al., 2010; González-Ramos et al., 2016). Both drugs have a common occurrence of adverse reactions that can prevent infected individuals from completing treatment (Hasslocher-Moreno et al., 2012; Sperandio da Silva et al., 2014; Molina et al., 2015; Olivera et al., 2015); studies found that between 13–31% cannot complete drug treatment (Hasslocher-Moreno et al., 2012; Sperandio da Silva et al., 2014; Molina et al., 2015; Olivera et al., 2015).

Benznidazole is the preferred and most effective treatment of *T. cruzi* infections (Prata, 2001; Rassi et al., 2010; González-Ramos, et al. 2016). However, adverse reactions in adults include epigastric pain, skin disorders, nausea, abdominal bloating, sleep disturbance, temporary memory loss, headache, loss of appetite, myalgia, eosinophilia, and central and peripheral nervous system disorders; and the percentage of adults with at least one of these reactions ranges from 49 – 80% (Hasslocher-Moreno et al., 2012; Sperandio da Silva et al., 2014; Molina et al., 2015; Olivera et al., 2015). Extreme cases have resulted in intensive care unit treatment with symptoms that included tonic-clonic seizures and in one case, decreased liver function and multiple general organ failure accompanied by 30% skin detachment in another case (González-Ramos et al.,

2016). Currently, drug treatment is optional for people over 50, because no proven benefits exist for people in this age cohort (Rassi et al., 2010). For other age cohorts, treatment efficacy for people with acute Chagas is between 30–83.5% (Prata, 2001; Teixeira et al., 2006; Jannin and Villa, 2007); and for chronic Chagas, the efficacy is much lower (5–30%) (Teixeira et al., 2006; Jannin and Villa, 2007).

Thus, given the lack of preventative medicine, coupled with the drug reactions and low efficacy of drug treatment, the preferred method of combating Chagas disease is by minimizing human contact with the vector. One of the most common tactics for controlling household infestation of *T. dimidiata* is the use of pyrethroid insecticide (Tabaru et al., 1998; Acevedo et al., 2000; Nakagawa et al., 2003a; 2003b; Dumonteil et al., 2004; Hashimoto et al., 2006; Manne et al., 2012; Yoshioka et al., 2015; Quinde-Calderón et al., 2016). While pyrethroid insecticides have successfully reduced infestation rates of *T. dimidiata*, rarely is the infestation rate reduced to zero (Acevedo et al., 2000; Quinde-Calderón et al., 2016)**.** Nonetheless, the residual effects of pyrethroid spraying appear to last only four months before adult *T. dimidiata* reinfest a house and nine months before nymphs are found in the house (Dumonteil et al., 2004). Thus, while residual pyrethroid spraying has been applied successfully to *Rhodnius prolixus* and *Triatoma infestans*, in most cases the same cannot be said for *T. dimdiata* (Waleckx et al., 2015a). The rebounding of infestation to original levels were observed almost three years after a single round of pyrethroid spraying in Jutiapa, Guatemala (Hashimoto et al., 2006). In addition to spraying houses with pyrethroid insecticides, recent work shows the potential of the fungi *Beauveria bassiana* and *Gliocladium virens* to control *T. dimidiata*

(Vázquez-Martínez et al., 2014). Both fungi were shown to successfully kill *T. dimidiata* in a laboratory setting Vázquez-Martínez et al. (2014); however, short of extirpation of *T. dimidiata*, the vector will always pose the risk of infestation where it is endemic.

The only proven long-term control of *T. dimidiata* infestation is the implementation of home improvements often accompanied by educational outreach on Chagas disease and the vector (Monroy et al., 2009; Ferral et al., 2010; De Urioste-Stone et al., 2015). Home improvements that minimize the risk of household infestation with *T. dimidiata* run the gamut of cleaning and organizing the peridomestic environment immediately surrounding a house (Zeledón and Rojas, 2006; Zeledón et al., 2008; Ferral et al., 2010), plastering walls (Monroy et al., 1998; Monroy et al., 2009; Lucero et al. 2013; Pellecer et al., 2013), replacing dirt floors with cement floors (Lucero, et al., 2013; Pellecer et al., 2013), installing window screens (Ferral et al., 2010; Waleckx et al., 2015b), impregnating curtains with insecticide (Ferral et al., 2010), and domestic rodent control (De Urioste-Stone et al., 2015). These home improvements have led to a reduction in household infestation that often lasts longer than spraying insecticide; however, none have completely eliminated infestation. Some of the aforementioned interventions are considered Ecohealth interventions because they use sustainable methods and locally sourced materials (Monroy et al., 2009; Lucero et al., 2013; Pellecer et al., 2013). Window screens were very effective in nearly eliminating household infestation in pilot villages in the Yucatán penninsula, Mexico; however, in the Yucatán penninsula, *T. dimidiata* has been shown to invade homes seasonally, and thus barriers to entry are more effective (Ferral et al., 2010; Waleckx et al., 2015b). Therefore, risk

analyses capable of discovering unidentified factors or patterns may help successful elimination of infestation.

1.3 Evolutionary Algorithm Background

Evolutionary algorithms (EAs) are biologically inspired algorithms designed to solve complex non-linear problems. EAs have been used to provide a non-invasive means of diagnosing Parkinson's disease (Smith and Timmis, 2008), better diagnose prostate cancer (Llorà et al., 2007), and determine the risk factors associated with bladder cancer (Urbanowicz et al., 2013). The latter two studies used EAs that could be applied to a Chagas disease dataset. Llorà et al. (2007) used an EA called a Pittsburgh-style learning classifier system (LCS) and Urbanowicz et al. (2013) used an EA called a Michigan-style LCS.

The original Michigan-style LCS algorithm, cognitive system (i.e., CS-1), was designed for binary input features and the prediction of associated categorical outcomes in dynamically changing environments (Holland and Reitman, 1978). Thus, they do not perform batch learning, but often learn one observation at a time (Urbanowicz and Moore, 2009), which is computationally inefficient and subject to sampling order bias when the data are available in batch. The most reliable Michigan-style LCS, called XCS, is still in use today (Butz et al., 2003). LCSs evolve a population of classifiers that consist of a condition and an action. The **condition** is a *mask* of input features (F), with each feature having one of the following values $\in\{0, 1, \#\}$, where # symbolizes "don't care" (or wild card) and implies a feature may be either 0 or 1. The classifier **action** is the outcome class (e.g., $\in\{0, 1\}$) associated with the input feature condition. In the LCS

community, the condition is analogous to a **conjunctive clause** (CC). This condition (CC) and action (combined) equal a **classifier**, and the LCS evolves the classifiers using a genetic algorithm (GA). In a broad sense, the classifier fitness is a function of the number of times its condition (input feature mask) and the associated action match both the observed input features and associated outcome data collected from the system under investigation. Michigan-style LCSs use the population of CCs and weighted fitness function first proposed by Wilson (1995) to predict outcomes. The performance of a population of CCs is often evaluated on the prediction accuracy of the most recent input feature vectors.

Smith (1980; 1983; 1984) is credited with the first Pittsburgh-style LCS named LS-1. Like the Michigan-style LCS, LS-1 was designed to predict multiple categorical outcomes given binary input features in a dynamic environment (early LS-1 tests included the ability to play poker). The LS-1 algorithm uses an encoding system similar to the Michigan-style LCS, except that instead of evaluating the fitness of individual classifiers, the fitness is evaluated based on a group of classifiers. Thus, the algorithm maintains a population of Pittsburgh-style classifiers, where each Pittsburgh-style classifier is the disjunction of Michigan-style classifiers, and a GA evolves the population of Pittsburgh-style classifiers.

Thornton-Wells et al. (2004) proclaimed the need for the development of statistical tools that take into account heterogeneity and feature interactions in complex disease datasets. While LCSs have been successfully tested numerous times on the multiplexer problem, a toy Boolean problem that is available in batch and has epistasis

and heterogeneity (Wilson, 1987a; Wilson, 1987b; Booker, 1989; Goldberg, 1989; De Jong and Spears, 1991; Butz et al., 2003; 2004; 2005b; Llorà et al., 2005; Butz and Pelikan, 2006; Ioannides et al., 2011; Iqbal et al., 2012; 2013b; 2013c; 2014; 2015; Urbanowicz and Moore, 2015), this toy problem does not contain noise and as the complexity of the problem increases, the complexity of the solution is so great that it bears little resemblance to a real-world problem. Even though the multiplexer problem is useful in testing an algorithm's ability to find epistatic and heterogeneous features; an algorithm's ability to solve the multiplexer problem does not imply that it can solve complex real-world problems having epistasis and heterogeneity. Therefore, given that LCSs are not designed for solving problems in bulk and given that a number of LCSs have components that are uniquely designed to solve toy Boolean problems a new evolutionary algorithm needed to be developed.

1.4 Spatial Autocorrelation in Disease-Related Studies

We were unable to find any empirical study that measures how far *T. dimidiata* can move during their lifespan. Orantes (personal communication, January 2017) has used next generation Rad-seq to create a database of single nucleotide polymorphisms (SNPs) for *T. dimidiata* collected from two villages. Spatial autocorrelation in the single nucleotide polymorphisms (SNPs) of humans has been observed at various scales. Elhaik et al. (2013) found spatial autocorrelation at the global scale and was relatively successful at leveraging geospatial and SNP data to predict a person's country of origin. On a finer scale, Lao et al. (2013) found spatial autocorrelation in people in the Netherlands, which they attributed to historic settlement patterns.

Spatial autocorrelation in the genetics of *T. dimidiata* have been observed at various scales. Bargues et al. (2008) analyzed 31 haplotypes at 64 locations that spanned a range from Mexico to northern South America. While they did not explicitly characterize spatial autocorrelation, they did show geographic grouping of phenotype trees. More recently, Stevens et al. (2015) investigated spatial autocorrelation using 7 highly polymorphic microsatellite loci from 178 *T. dimidiata* spread across 6 villages in the department of Jutiapa, Guatemala. Using the relatedness function of Lynch and Ritland (1999), Stevens et al. (2015) found some migration of *T. dimidiata* between houses in a village as well as some spatial autocorrelation, despite the signal being weak. These findings are contrary to earlier works that did not find spatial autocorrelation among *T. dimidiata* in nearby villages in Guatemala (Dorn et al., 2003; Calderón et al., 2004). Given that Melgar et al. (2007) found 41 families of *T. dimidiata* in a single house in Guatemala, using hundreds to thousands of markers can allow for fine scale, within-town spatial autocorrelation. As a result, using the thousands of *T. dimidiata* SNPs from the Orantes (personal communication, January 2017) database may provide a unique opportunity to explore spatial autocorrelation at the finer village scale.

Smouse and Peakall (1999) developed a methodology to characterize the range of spatial autocorrelation by using multiple genetic markers to create a correlogram. Their methodology has subsequently been used on a variety of species such as emmer wheat (*T. turgidum* L. ssp. *dicoccoides*) (Volis et al., 2014), beech trees (*Fagus sylvatica* L.) (Piottti et al., 2013), bottlenose dolphins (*Tursiops truncatus*) (Richards et al., 2013), Canada geese (*Branta canadensis*) (Finnegan et al., 2013), and the American black bear

(*Ursus americanus*) (Coster and Kovach, 2012). In a few instances, the correlogram of Smouse and Peakall (1999) has been used to measure the range of spatial autocorrelation of a disease vector. Foley et al. (2004) found the range of spatial autocorrelation for the mosquito vector (*Ochlerotatus notosciptus*) of dog heartworm (*Dirofilaria immitis*) to be ~55 km. While Rašić et al. (2015), found the range of spatial autocorrelation to be 3-6 km for the mosquito *Aedes aegypti*, which is a vector of dengue. Finally, Pérez de Rosas et al. (2013) investigated the range of spatial autocorrelation for *Triatoma infestans*, the principle vector of Chagas disease in South America, and found a range of ~400 m. They also investigated sex-biased dispersal and found that females had a relatively larger range of spatial autocorrelation than males (400 m versus 330 m). Pérez de Rosas et al. (2013) used the range of spatial autocorrelation as a guideline for the radius of insecticide applied around an infested house or peridomestic structure. Therefore, the Smouse and Peakall (1999) methodology for determining the range of spatial autocorrelation can be used as a foundation for determining the range of spatial autocorrelation for *T. dimidiata*.

1.5 Thyroid Cancer

Thyroid cancer incidence is increasing at an annual rate of 3–5%, resulting in the rate tripling over the past 30 years in the United States as well as in other countries (Curado et al., 2007; Kilfoy et al., 2009; Jemal et al., 2011; Morris et al., 2013; Pellegriti et al., 2013). In the United States, the number of cases has risen from 4.3 cases per 100,000 in 1980 to 12.9 cases per 100,000 individuals in 2008. Mortality rates have slightly increased (+0.8% annual percent change [APC]) (Enewold et al., 2009; Cramer et al., 2010; NCI, 2012). A recent study noted a disproportional increase in women

(Edwards et al., 2006). The basis for the increase in thyroid cancer incidence is not known. Some studies suggest enhanced diagnostic scrutiny and better detection of subclinical cancers result in widespread over diagnosis and thus not a true increase in incidence (Davies and Welch, 2006; Ross, 2006; Grodski et al., 2008; Enewold et al., 2009; Hall et al., 2009; Yu et al., 2010; Morris et al., 2013; Reitzel et al., 2014). Other studies note that an increase in both large tumors and microcarcinomas as well as a change in relative frequencies of histological types implicate other contributing factors (Chen et al.; 2005; Kilfoy et al., 2009; Pazaitou-Panayiotou et al., 2013; Ward et al., 2010; Aschebrook-Kilfoy et al., 2013). Of note, recent reports of aggressive, metastatic microcarcinomas of the thyroid that correlate with the risk of second cancers (Kim et al., 2013) suggest that microcarcinomas once considered subclinical might emerge as important new healthcare concerns and reflect an important dimension of the increase in thyroid cancer incidence.

Environmental and demographic factors may be critical determinants in the increase in thyroid cancer incidence (Leux and Guénel, 2010; Morris and Myssiorek, 2010; Li et al., 2013; Pellegriti et al., 2013). A recognized risk factor for thyroid cancer is ionizing radiation exposure through medical procedures, including x-rays, as well as radioactive fallout (Richardson, 2009; Wartofsky, 2010; NCI, 2013). A study of the overall geographic distribution of thyroid cancer in the United States revealed a higher incidence in areas proximate to nuclear power reactors (Mangano, 2009). High levels of nitrate in public drinking water supplies have been linked to increased thyroid cancer incidence (Ward et al., 2010), and environmental endocrine disruptors including

polyhalogenated aromatic hydrocarbons (PHAHs), notably polybrominated diphenyl ethers (PBDEs) and organochlorine insecticides, are postulated factors (Grimalt et al., 1994; Zhang et al., 2008; Zhu et al., 2009; Leux and Guénel, 2010). Leux and Guénel (2010) noted that many environmental chemicals interfere with thyroid function and increase the risk of goiters, nodules, and possibly neoplasia. Additional known risk factors include family history, sex, and age (Pellegriti et al., 2013). Socioeconomic factors (SES) may also indicate that access to healthcare affects incidence (Sprague et al., 2008; Morris et al., 2013). Thus, novel analyses are needed to elucidate both incidence and contributing factors.

With the capability to visualize, analyze, interpret, and map geo-located data, the field of geostatistics, notably the geographic information system (GIS) tool, has emerged as a powerful geospatial technology that is gaining prominence in healthcare applications (Musa et al., 2013). GIS-based cancer mortality maps produced by the National Cancer Institute and Centers for Disease Control and Prevention (CDC) are widely used by public health officials to guide disease surveillance and control activities throughout the United States (Shaw, 2012). Beyond traditional GIS mapping capabilities, more sophisticated spatial statistical analyses have been utilized to identify spatial disease clusters (i.e., nonrandom spatial distributions of disease cases, incidence, or prevalence), map and monitor disease patterns and trends over time and space, and assess the impact of ecological and socioeconomic factors on the spatial distribution of diseases. Although there are still many technical (e.g., knowledgeable users, data quality control) and organizational (e.g., access and sharing) barriers to the wide-scale adoption of geospatial

technologies in the healthcare sector (Boulos et al., 2011), recent advances in the understanding of disease dynamics, healthcare management has demonstrated the power of geospatial technologies to identify new drivers of public health concerns and advance the field of public health research.

CHAPTER 2: A TANDEM EVOLUTIONARY ALGORITHM FOR IDENTIFYING OPTIMAL ASSOCIATION RULES FROM COMPLEX DATA

2.1 Introduction

The causal rules underlying emergent properties of complex systems often exhibit heterogeneity, epistasis, and/or overlap. Empirical observations of such systems may be high-dimensional and typically include missing data, noise, and/or imbalanced classes. All of these complexities complicate our ability to infer meaningful (potentially causal) associations between observed system features and outcomes of interest.

Heterogeneity exists when there are multiple underlying causes for the same outcome class. Evidence for heterogeneity exists in many systems, including bladder cancer (Urbanowicz et al., 2013), autism (Buxbaum et al., 2001), and American political parties (Poole and Rosenthal, 1984). Epistasis occurs when combinations of different feature values exhibit non-additive effects on outcomes. Epistasis is believed to be ubiquitous for many diseases (Moore, 2003), including breast cancer (Ritchie et al., 2001), blood pressure in rats (Rapp et al., 1998), and Behçet's disease (Kirino et al., 2013). Many systems exhibit both heterogeneity and epistasis. For example, different (i.e., heterogneous) combinations of non-linearly interacting (i.e., epistasic) transmission line outages can cause cascading failures that lead to the same patterns of power loss in the electrical grid (Eppstein and Hines, 2012). Similarly, the ecological niche of the American black bear (*Ursus americanus*) is epistatic (in that the species requires both a secluded area for denning and specific combinations of spring, summer, and autumn food

sources (Larivière, 2001)) and heterogeneous (because of the widely different combinations of denning and three-season diets that accommodate the bear population, contributing to a vast geographic range that spans from southern Mexico to northern Canada (Larivière, 2001)). Furthermore, real world datasets often include correlated features that can cause significant overlap in heterogeneous explanatory rules, highly imbalanced classes (i.e., when the outcome classes are not equally represented in the dataset), noise in measured outcomes, and missing data (Chapter 3).

There are many practical applications that require an understanding of such complex relationships, such as in the development of personalized drug therapies (Wilson, 2009), making market predictions of consumer behaviors (Young Kim and Kim, 2004), identifying gene-gene and gene-environment causes for complex disease (Moore, 2003), and developing eco-intervention strategies to minimize the spread of disease in less developed countries (Chapter 3). However, while the size and complexity of available datasets has exploded in recent years, computational tools for analyzing such systems have not kept pace (Wu et al., 2014).

Traditional statistical and data mining methods, such as analysis of variance (Wilson et al., 2017; Yousefi et al., 2016), logistic regression (Jarlenski et al., 2016; Li et al., 2016; Nesheli et al., 2016), and decision trees (Markellos et al., 2016; Nesheli et al., 2016) are well suited for univariate analysis of additive models. Some studies perform feature selection using univariate logistic regression models and then test higher-order interactions between the selected features (Kaplinski et al., 2015; Molina et al., 2015;

Olivera et al., 2015). However, if main effects are small or non-existent, these traditional approaches will fail.

In very high-dimensional problems, researchers have used iterative feature reduction methods to reduce the search space (e.g., Moore and White (2007); McKinney et al. (2007)), often using the data-mining algorithm ReliefF (Robnik-Šikonja and Kononenko, 2003) to assess feature importance. For example, Eppstein et al. (2007) developed a computationally efficient feature reduction approach (logarithmic in the number of features) for identifying parsimonious epistatic interactions that can predict an outcome of interest, dubbed 'Random Chemistry'. This general approach can also be used to find heterogenous, possibly overlapping, sets of different epistatic interactions associated with a given outcome, *via* independent runs (Eppstein and Hines, 2012). However, in noisy and high-dimensional association problems where there are many more features than input samples (e.g., as in genome wide association studies), algorithms such as ReliefF become unreliable (Eppstein and Haake, 2008). Furthermore, even when such methods are successful in identifying individual epistatic interactions, they are not designed to identify maximally explanatory combinations of such interactions in heterogeneous systems.

Learning classifier systems (LCS) are a type of evolutionary algorithm (EA) often employed to analyze classification problems with epistatic, heterogeneous and/or overlapping rules (Urbanowicz and Moore, 2009). The most common type of LCS is the so-called Michigan-style LCS, first introduced by Holland and Reitman (1978). A Michigan-style LCS uses a genetic algorithm to evolve a population of classifiers, with

each classifier comprising a condition/action pair. For example, consider a problem with 5 binary input features and binary outcome classes. The classifier 0##1# → 1 (where # is a wild card symbol) is interpreted as "if feature 1 has value 0 and feature 4 has value 1, then the outcome class is predicted to be 1". The condition 0##1# is thus equivalent to the conjunctive clause ($F_1 \in \{0\} \wedge F_4 \in \{1\}$), where F_i refers to the value of feature *i* and $\wedge$ represents the Boolean operator "AND." (We use set notation so that this is easily generalizable to nominal, ordinal, or continuous features F_i with arity > 2). Prediction is typically evaluated based on a weighted combination of all classifiers in the population, and fitness is based on the number of times a classifier correctly predicts the outcome of an input feature vector (Wilson, 1995). However, because Michigan-style LCS approaches have focused on prediction accuracy, they return large "black box" sets of classifiers, rather than seeking to identify parsimonious "white box" models that are potentially causal. Furthermore, Michigan-style LCS approaches were designed for real-time data assimilation in dynamically changing environments (Holland and Reitman, 1978) and can be inefficient and subject to bias (based on sampling order) when applied to data that are available in batch.

In preliminary work, motivated by the desire to mine complex survey data, we introduced a new evolutionary approach for finding heterogeneous and epistatic associations between input features and multiple outcome classes in large datasets (Hanley et al., 2016). In the current work, we further develop this method, compare the results to published results on test problems from the LCS community, and discuss how

our approach can be applied to seek potentially causal rule sets in real-world survey data, with important practical implications (Chapter 3).

Our approach uses two EAs in tandem, each using an age-layered population structure (Hornby, 2006), and assesses fitness using a hypergeometric probability mass function (Kendall, 1952) that accounts for the size of the dataset, the amount of missing data, and the distribution of outcome categories. The first EA is used to evolve an archive of conjunctive clauses (CCs) that have a high probability of a statistically significant association with a given outcome. The second EA evolves disjunctions of these archived CCs to create an archive of probabilistically significant clauses in disjunctive normal form (DNF). Problem-specific post-processing methods of the DNF archive can then be applied to identify potentially causal parsimonious rule sets for predicting the outcome.

This paper is organized as follows. In Section 2.2, we present our evolutionary approach and in Section 2.3, we describe the test problems used. In Section 2.4, we show how our method efficiently finds the most parsimonious, explanatory models in all problems tested and compare our results to published results from the LCS community. Finally, in Section 2.5, we discuss our findings and propose some directions for future work in algorithm and benchmark problem development.

2.2 Proposed Evolutionary Algorithm

We propose a system of two EAs in tandem that is capable of mining large, heterogeneous datasets of N feature vectors, for possibly epistatic and heterogeneous associations between combinations of L nominal, ordinal, and/or real-valued features that

are possibly predictive of a given target class outcome *k*. This tandem algorithm is run independently for each target class *k* present in the input dataset.

The first EA (dubbed CCEA) evolves an archive of probabilitistically significant conjunctive clauses (CCs) of various orders, where the order is the number of interacting features in an epistatic interaction; 1st-order clauses correspond to main (i.e, univariate) effects. The second EA (dubbed DNFEA) combines archived CCs with disjunctions to evolve an archive of probabilistically significant clauses in disjunctive normal form (DNFs) of various orders, where the order is the number of conjunctions in a heterogeneous rule sets; 1st-order DNFs comprise a single CC. Additional post-processing of the DNF archive seeks the optimal rule set for the target class *k*. Further details of the algorithm are described below and a hard copy of the Matlab code is available in Chapter 8.

2.2.1 Fitness Function

For each target class *k*, we define the fitness of a given *clause* using a hypergeometric probability mass function (PMF) (Kendall 1952), as follows::

$$Fitness(clause, k) = \frac{\binom{N_k}{N_{match,k}}\binom{N_{tot}-N_k}{N_{match}-N_{match,k}}}{\binom{N_{tot}}{N_{match}}} \quad , \qquad (2.1)$$

where clause is a given CC or DNF; N_k = the total number of input feature vectors with the target class *k* that do not have missing values for any features present in the *clause*; $N_{match,k}$ = the number of input feature vectors for which the given *clause* is true and that have the target class *k*; N_{tot} = the total number of input feature vectors that do not have missing values for any features present in the *clause* (regardless of class); and N_{match} = the number of input feature vectors for which the *clause* is true (regardless of class). Note

that these definitions are slightly modified from those in Hanley et al. (2016) to better accommodate missing data.

Eq. (2.1) quantifies the likelihood that the observed association between the *clause* and the target class k is due to chance, taking into account the size of the dataset, the amount of missing data, and the distribution of outcome categories. We thus seek to minimize Eq. (2.1), since lower values are indicative of greater probability of association between a *clause* and a target class. Henceforth, when we refer to the "most fit" clauses, we mean those with the lowest values using Eq. (2.1).

2.2.2 Population Structure

Both the CCEA and the DNFEA are implemented using a customized version of an Age-Layered Population Structure (ALPS) (Hornby, 2006), with 5 linearly-spaced age-layers and an age gap of 5. In this study, we restrict each CCEA layer to a population size of L (where L is the total number of features in the input vectors), whereas we restrict each DNFEA layer to population size of 20. In both the CCEA and the DNFEA, there is an additional 6th layer that it is used as an archive of probabilistically significant clauses.

We run the CCEA and DNFEA separately for each target class k that is present in the input data, thus creating separate archives for each possible outcome class. In Boolean benchmark problems, the optimal generative rule set for each target class is easily identified as the single archived DNF with the lowest fitness per Eq. (2.1). In the more realistic problems, additional problem-specific post-processing of the CC and DNF archives can be applied to identify parsimonious explanatory rule sets, as discussed later. A high-level flowchart of the algorithm is shown in Fig. 2.1.

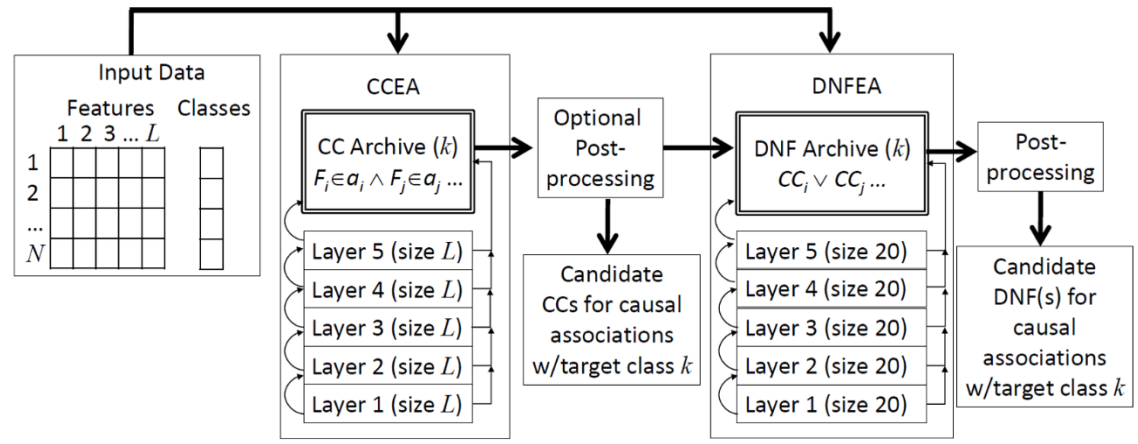


Figure 2.1: Flowchart for the proposed tandem ALPS-based EAs. For each target class *k*, we use the CCEA to evolve an archive of conjunctive clauses (CCs) that have a statistically significant probability of being associated with outcome class *k*; the CCs can be of arbitrary order, thus representing epistatic interactions. The DNFEA then evolves disjunctions of these archived CCs (after optional post-processing of the CC archive) and archives the resulting probabilistically significant disjunctive normal forms (DNFs); the DNFs can be of arbitrary order, thus representing heterogeneity. Further postprocessing of the archived DNFs seeks potentially causal rule set(s), in the form of DNFs that are predictive of outcome class *k*. For benchmark problems, we seek the single known optimal generative rule set.

2.2.2.1 Initialization, Reproduction, and Aging

At the start of the first generation (and every 5 generations thereafter), a novel population of clauses, each with age 1, is introduced into the first age layer. Further details of the initialization of CCs and DNFs are described in Sections 2.2.3 and 2.2.4, respectively. During each generation, all of the individuals in layers 1-5, plus up to $L\times5$ of the youngest individuals from the archived layer 6 (or fewer, if the archive doesn't yet hold this many individuals) are selected to reproduce with variation. The ages of these selected parents are incremented by 1 and they remain in the population. Variation is introduced either through crossover (with probability $P_C = 0.5$) or through mutation. If selected for crossover, a second parent is selected from the same or preceding (if one exists) age layer, using tournament selection with replacement (tournament size of 3);

the age of the second parent is not incremented. Further details of the crossover and mutation operators on CCs and DNFs are described in Sections 2.2.3 and 2.2.4, respectively. The children are given the same age as the oldest parent.

Upon creation of a new clause in any layer (whether via random initialization, mutation, or recombination), if $\frac{N_{match,k}}{N_{match}} < \frac{N_k}{N_{tot}}$, then clause is discarded; this biases the algorithm toward retaining clauses that are useful in finding associations with the target class *k*. Of those retained, clauses with order *i* for which Eq. (2.1) is less than or equal to an order-specific threshold T_i are put directly into the archive bin for order *i* (as further explained in Section 2.2.2.2); otherwise, they are added into the appropriate age layer.

Every fifth generation, individuals in layers 1-5 age out of their layers into the next higher age layer and a new random population is created for layer 1. Those aging out of layer 5 are discarded from the population.

At the end of each generation, all individuals within the same age layer compete with each other during survivor selection, as follows. For any of the layers 1-5 in the CCEA and DNFEA that exceed the maximum layer size of *L* or 20 individuals, respectively, we determine survivors through truncation selection retaining the *L* or 20 most-fit CCs in that layer for the CCEA and DNFEA, respectively.

2.2.2.2 Maintaining the Archive

Each archive is partitioned into bins for different orders of clauses, to ensure diversity in the complexity of the archived clauses. The maximum order of these bins and the lower bounds on the sizes of these bins are dataset-dependent (see Table 2.5). In all cases, the upper bounds on the bin sizes were 10 more than the lower bound. All

clauses in an archive bin for order i have fitness values that are less than or equal to a dynamicallyadjusted order-specific threshold T_i. The highest order bin may accept clauses ≥ the order of the bin. In the CCEA and DNFEA, thresholds T_i for all orders i are initialized to $1/N$. This translates to an initial probability of 1 in N that a CC with fitness $= T_i$ is randomly associated with the target class k.

2.2.3 Representation of Conjunctive Clauses (CC's)

We represent possibly epistatic interactions, which are predictive of a target class k, with CCs in the following form:

$$CC_k \coloneqq F_i \in a_i \wedge F_j \in a_j \ldots \quad , \tag{2.2}$$

where := means "is defined as", F_i represents a feature that may be nominal, ordinal, or continuous, and whose value lies in a_i, and ∧ represents conjunction (i.e., logical AND). Note that a_i is a specified range or set of values that is a proper non-empty subset of a pre-specified universal set or maximum range of each feature. The meaning of such a clause is interpreted as "if CC_k is true for a given input feature vector, then the class outcome is predicted to be k."

Each CC is represented by two parallel data structures. The first is a Boolean vector of length L (where L is the number of features in each input vector) that encodes presence (1) or absence (0) of each possible feature F_i in the clause. Thus, the sum of this Boolean vector represents the order of the CC and each feature i can appear at most once in a CC. Note that 1st-order CCs represent main effects of individual features; and if a feature is absent from a clause, this is equivalent to the LCS notation of having a wild card in that feature's position. We store the corresponding ranges or sets of values a_i in

a parallel data structure (we represent this as a vector of L pointers, each of which points to a binary vector representation of the range or set of values, similar to that used by De Jong and Spears (1991); although this is not space-efficient, it is very time-efficient when checking to see if a given CC matches a given instance in the dataset). These parallel structures comprise the genome of an individual in the CCEA, and the values in the binary vectors representing feature presence/absence and feature ranges/sets are co-evolved.

We enforce that there is at least 1 feature present in each CC, and that the allowable set or range for each included feature is non-empty, to preclude the problem of evolving clauses that cannot match any instances in the dataset (as discussed in Llorà et al. (2005)). We allow CCs to have up to L features present, since we do not wish to make arbitrary *a priori* assumptions on the maximum order of epistasic interactions that may exist. Iqbal et al. (2015) showed that higher-order CCs are useful in finding epistatic lower-order CCs.

2.2.3.1 CC initialization

Novel CCs are randomly created for layer 1 such that they are guaranteed to match at least one input feature vector that is associated with the target class k (a process known as "covering"). To accomplish this, we first generate a uniformly distributed random integer $j \in \{1, \ldots, L\}$ to specify the order of the CC, and then extract the subset of input feature vectors with class k that have at least this many non-missing values. From this subset, we choose one of these at random. While the archive is empty, this input feature vector is selected according to a uniform distribution. However, once the archive

has been populated with clauses, we use a non-uniform distribution to bias the selection toward input feature vectors that are not yet well-covered in the archive. Specifically, we first tally the number of archived clauses that match each input feature vector in the extracted subset. We then sum this tally and add one, and subtract each feature vector's tally from this value. We normalize the resulting vector and select an input feature vector according to this probability distribution.

We then randomly select j of the non-missing features in the selected feature vector to be present in the clause. For each selected feature i, we then randomly initialize the corresponding range or set a_i as follows. If the feature is nominal, the set a_i is initialized to contain only the value for feature i that occurs in the selected input feature vector. If the feature is ordinal or continuous, the range stored in ai is initialized such that both the lower and upper bounds of the range are assigned the value for feature i that occurs in the selected input feature vector, so that the range contains exactly this value.

2.2.3.2 CC mutation

When a CC is selected for mutation, we do the following. Each position in a copy of the binary feature array from the parent is selected with probability $1/L$ (if zero features were initially selected, we select one at random). For each feature i that was selected, if the value at position i in the binary feature array is 0 (feature not present in the clause), then it is set to 1 (feature is added to the clause); and a_i is randomly initialized to a non-empty set or range of allowable values that does not include the entire allowable subset or range of values. However, if the value at position i in the binary feature array is 1 (i.e., F_i was present in the clause), then with probability P_w, the bit is flipped to 0 (i.e., the

feature is removed from the clause). For this work, we selected a high $P_w = 0.5$ so that mutation favors order reduction and thus aids in evolving parsimonious clauses that contain as few features as possible. If the value at position i in the binary feature array remains a 1 (feature F_i is still present), then the corresponding a_i is mutated as follows. If F_i is nominal, we randomly change, add, or delete a categorical value to a_i, ensuring that the set remains non-empty and less than the allowable universal set of values. If F_i is ordinal or continuous, we randomly change the lower or upper bound of a_i, ensuring that the range remains non-empty and less than the maximum allowable range.

2.2.3.3 CC crossover

When a CC is selected for crossover, we perform uniform crossover between copies of the CC and its mate (selected as described in Section 2.2.2). Specifically, we initially create two children, swapping values between random positions in the binary feature arrays of the copies of the two parents, as well as between the same positions in the corresponding arrays of sets/ranges. If the first child contains at least one feature, we discard the second child; otherwise, we discard the first child.

2.2.4 Representation of Clauses in Disjunctive Normal Form (DNF's)

We represent possibly heterogeneous interactions, which are predictive of a target class k, with DNFs in the following form:

$$DNF_k \coloneqq CC_i \vee CC_j \ldots \quad , \tag{2.3}$$

where each CC_i is of the form shown in Eq. (2.2) and $\vee$ represents disjuntion (i.e., logical OR). The meaning of such a clause is interpreted as "if DNF_k is true for a given input feature vector, then the outcome class is predicted to be k." Each DNF is represented by

a binary array of length $N_{CC,k}$, where $N_{CC,k}$ is the number of CCs archived by the CCEA for outcome class k (see Section 2.2.2). The binary values encode presence (1) or absence (0) of a given CC in the DNF, so the sum of this array represents the order of the DNF. Each DNF is constrained to include at least 1 CC but may have up to $N_{CC,k}$ CCs. This binary array comprises the genome of an individual in the DNFEA. For implementation efficiency, prior to running the DFNEA, each *CC* in the archive is associated with a precomputed binary array of length N that encodes whether the CC matches (1) or doesn't match (0) each of the N input feature vectors and its associated outcome class. In general, the implementation of the DNFEA operators is simpler than that of the CCEA operators, since we no longer need to worry about allowable sets/ranges or covering of input feature vectors.

2.2.4.1 DNF initialization

Novel DNFs are randomly created as uniformly distributed binary arrays with anywhere from one CC to the maximum DNF order that will be archived for a given problem.

2.2.4.2 DNF mutation

When a DNF is selected for mutation, it will undergo one of five types of mutation with equal probability. Type 1 mutation is simple bit flip where each position in a copy of the binary feature array from the parent is selected with probability $1/N_{CC,k}$ (if zero features were initially selected, we select one at random). We then perform bit-flip mutation at each of these selected positions, subject to the constraint that the DNF must still contain at least one CC.

The other four types of mutation are designed to help expand the diversity of evolved clauses in terms of true positive rate and coverage, or are aimed at reducing the DNF order. Type 2 mutation selects the CC that covers the most input feature vectors with target class *k* that are not covered by the DNF. Type 3 mutation selects the CC that covers the most input feature vectors with target class *k* that are not already covered by the DNF, while avoiding covering input feature vectors that are not associated with target class *k*. Type 4 mutation removes the CC that covers the fewest number of input feature vectors with target class *k* that are not covered by other CCs in the DNF. Finally, Type 5 mutation removes the CC that has the most input feature vectors that are not target class *k* and are not covered by other CCs in the DNF. All five types of mutation ensure that at least one CC will be present in the DNF.

2.2.4.3 DNF crossover

When a DNF is selected for crossover, we perform uniform crossover between copies of the DNF and its mate (selected as described in Section 2.2.2). Specifically, we initially create two children, swapping values between random positions in the binary feature arrays. If the first child contains at least one feature, we discard the second child; otherwise, we discard the first child.

2.3 Test Problem Characteristics

In this manuscript, we test our algorithm on three types of problems previously used to test LCS algorithms. Two of these are classic scalable Boolean benchmark problems (the majority-on and multiplexer problems) and the third is a more realistic

synthetic genome association problem. Each of these problems are challenging and interesting in different ways (Table 2.1).

Table 2.1: Challenging aspects of test problem used in this work; Majority-On (MO), Multiplexer (MP), 4 variants of MP, and the Synthetic Genome problem.

Problem	Heter. Rules	Epistatic Rules	Overlap. Rules	Extran. Features	Imbal. Classes	Noisy Classes	Missing Data
MO	X		X				
MP	X	X					
MP V1	X	X		X			
MP V2	X	X		X	X		
MP V3	X	X		X		X	
MP V4	X	X		X			X
Genome	X	X	X	X		X	

Below, we describe the rule sets used to generate the data for each of these problems, and show how each generative rule set for a particular outcome class can be represented in DNF, where each CC in the disjunction is one of the heterogeneous causes for the outcome class. Our goal is not only to evolve a set of classifiers that can accurately predict the outcome classes from input feature vectors, but to identify each of the true generative CCs as well as the single true generative DNF for each outcome class.

2.3.1 The Majority-On Problem

The majority-on problem, and the related count-ones problem (which is equivalent to majority-on but with extraneous features added) are scalable Boolean benchmark problems still used in the LCS community (Butz et al., 2003; Iqbal et al., 2013a; b; c; 2014), despite known limitations (McDermott et al., 2012).

In the majority-on problem, the number of input features L is always odd and the outcome class is specified by which of the Boolean values (0 or 1) is in the majority in a particular input feature vector. The generative model is the set of all classifiers with order

(L + 1)/2 such that all fixed bits and the action bit have the same value. For example, in the 3-bit majority on problem, the optimal predictive rule set for outcome class 0 is the following disjunction: (00#) ∨ (0#0) ∨ (#00), and the optimal predictive rule set for outcome class 1 is the following disjunction: (11#) ∨ (1#1) ∨ (#11). Since each condition can be considered a conjunctive clause (CC) (see Section 2.1), these optimal rule sets may be considered to be in disjunctive normal form (DNF). Note: These optimal rule sets are heterogeneous (since each is the disjunction of 3 classifiers). The classifiers are overlapping (e.g., (11#) and (1#1) both match the input vector 111 with observed outcome class 1), but are not epistatic (i.e., all features have additive main effects). Despite the presence of overlap, each of the 6 optimal condition/action classifiers are needed since there are input vectors that are only matched by one classifier (e.g., for outcome class 1, the input vector 110 is only matched by classifier 11#).

We note that, for noiseless 2-class benchmarks problems like this, it is not actually necessary to evolve explicit rules for class 0, since one could simply assume the implicit rule of "if class 1 is not predicted, then predict class 0." However, to demonstrate the generality of a given method's ability to evolve explicit sets of classifiers for problems that are potentially noisy and may have an arbitrary number of outcome classes, it is the norm in the LCS community to explicitly evolve classifiers for both outcome classes, and we follow that convention here.

2.3.2 The Multiplexer Problem

The multiplexer problem, designed to predict the output of a electronic multiplexer circuit, is another scalable Boolean benchmark problem. The multiplexer

problem was first introduced to the machine learning community by Barto (1985), and has been a standard benchmark problem for testing LCS approaches for decades (Booker, 1989; De Jong and Spears, 1991; Goldberg, 1989;Wilson, 1987a; b; Butz et al., 2003; 2004; 2005; Butz and Pelikan, 2006; Ioannides et al., 2011; Iqbal et al., 2012; 2013a; b; c; 2014; 2015; Llorà et al., 2005; Urbanowicz and Moore, 2015).

The generative model is the disjunction of 2^{b+1} classifiers, each with order $b + 1$, where b is the total number of address bits used to identify a location in a vector of 2^b data bits that contains the outcome class. An example of the 6-bit multiplexer architecture is presented in Table 2.2. When using the multiplexer as a benchmark classifier problem, the input feature vectors comprise both the address bits and the data bits, so are $b + 2^b$ bits long; the outcome classes associated with particular input feature vectors are thus only discovered as the address bits of the classifiers evolve. The optimal predictive rule set for outcome class 0 in the 6-bit multiplexer can thus be considered as the following DNF: {(000###) ∨ (01#0##) ∨ (10##0#) ∨ (11###0), and the optimal predictive rule set for outcome class 1 is the following DNF: (001###) ∨ (01#1##) ∨ (10##1#) ∨ (11###1)}. This benchmark problem is purely epistatic (the address features do not have main effects and all optimal classifiers are of order > 1) and heterogeneous (different classifiers match different different subsets of the possible input vectors).

Table 2.2: Example of the generative rule set for a 6-bit multiplexer problem. Each feature vector X is 6 bits long, with the first b = 2 bits representing the address bits A, which are interpreted as a 2-digit binary number (equivalent to decimal 0, 1, 2, or 3) that is used as an index into the the next 2^b data bits D. The data bit at this index represents the class outcome, whereas all other data bits are irrelevant to the classifier (wild cards).

Address Bits		Data Bits			
X_1	X_2	X_3	X_4	X_5	X_6
A_1	A_2	D_0	D_1	D_2	D_3
0	0	0	#	#	#
0	1	#	0	#	#
1	0	#	#	0	#
1	1	#	#	#	0
0	0	1	#	#	#
0	1	#	1	#	#
1	0	#	#	1	#
1	1	#	#	#	1

As with the majority-on problem, one could simply preclude the need for explicitly evolving classifiers for class 0 by assuming the implicit rule of "if class 1 is not predicted, then predict class 0." However, to demonstrate generality, it is the norm in the LCS community to explicitly evolve classifiers for both outcome classes, and we follow that convention here.

2.3.3 Synthetic Genome Problem

Urbanowicz and Moore (2010) designed a noisy dataset to represent a synthetic genome association study for a complex disease that incorporates both genetic epistasis and heterogeneity. For the remainder of this manuscript we refer to this as the synthetic genome problem. The dataset contains 1,600 input feature vectors, and is perfectly balanced in that 800 input feature vectors are associated with class 1 (disease) and 800 are associated with class 0 (no disease). Each input feature vector contains 20 ternary features, each representing whether a particular locus in the genome is homozygous for the major (most common) allele, heterozygous, or homozygous for the minor allele.

The dataset was designed with the intent that only four of these features would have a statisically meaningful association with the disease. Specifically, there were four heterogeneous causes for the simulated disease, in two pairs of purely epistatic interactions (i.e., no main effects) between two different pairs of loci (Table 2.3). Since the association between each of these 4 optimal rules and class 1 (disease) was designed to be noisy, we also indicate their true positive rate, coverage, and fitness by Eq. (2.1) (Table 2.3).

Table 2.3: The four generative rules that are designed to have a statistically meaningful association with class 1 (disease) in the synthetic genome problem. In each of the 4 rules, only two loci $\in \{F_1, F_2, F_3, F_4\}$ out of 20 are not wild cards. True positive rate, coverage, and fitness (by Eq. (2.1)) of each of these true generative rules for class 1 (disease) are also shown.

F_1	F_2	F_3	F_4	F_5	…	F_{20}	True Positive	Coverage	Fitness
0	1	#	#	#	…	#	72%	27%	1.1×10^{-17}
1	0	#	#	#	…	#	74%	23%	5.7×10^{-17}
#	#	0	1	#	…	#	66%	28%	4.2×10^{-12}
#	#	1	0	#	…	#	71%	21%	8.7×10^{-13}

Due to noise, the true generative DNF for class 1 (i.e., the disjunction of the 4 true generative rules shown in Table 2.3) has an overall positive prediction rate for class 1 of only 64% (see Table 2.4), coverage of 76%, and fitness by Eq. (2.1) of 3.2×10^{-44}. Note that there are no explicit rules that predict class 0, so if one assumes the default rule that "if class 1 is not predicted, then predict class 0," then the overall positive prediction rate for both classes is 67%.

Table 2.4: Confusion matrix that results when predicting class 1 (disease) from the 1,600 sample noisy synthetic genome dataset, using the optimal generative rule set for class 1. Samples that are not predicted to be class 1 are predicted to be class 0 (no disease).

	Predicted: Class 1	Predicted: Class 0
Actual: Class 1	607	193
Actual: Class 0	336	464

Unlike in the majority-on and multiplexer problems, the synthetic genome problem was not designed to have any classifiers that are explicitly associated with class 0 (no disease). Thus, it is most appropriate to only evolve rules for class 1 and then assume the implicit rule of "if class 1 is not predicted, then predict class 0," and we take that approach here.

2.3.4 Experimental Design

Control parameters for the different types of problems and problem sizes tested are shown in Table 2.5. We note that, while preliminary experimentation showed that these parameters were sufficient for identifying the true generative clauses, it is likely they could be further optimized to improve performance. Each problem was run for 30 random repetitions. For the Boolean benchmark problems we ran the CCEA and DNFEA separately for each of class 0 and class 1, so actually performed a total of 60 runs for each problem size. On the synthetic genome problem we only ran the CCEA and DNFEA for class 1 (since there was no true generative rule for class 0), so performed a total of 30 runs on this problem. For all test problems, the runs were terminated when all of the true generative clauses had been archived and we recorded the total number of fitness evaluations performed per run. We used the same set of parameters for all 4 variants of the 6-bit multiplexer problem with 14 extraneous features added, including (a) a base

case (with balanced classes, no noise in the output classes, and no missing data), (b) imbalanced classes (15% class 0 and 85% class 1), (c) 20% noise added to the class outcomes (i.e., we flipped the outcome bit in 20% of random input data samples), and (d) 20% missing data (i.e., we randomly removed 20% of feature values from the input data samples).

Table 2.5: Control parameters on the CCEA and DNFEA for the test problems. TIT stands for Ternary Digit, and EF stands for Extraneous Features. Genome refers to the Synthetic Genome Problem.

Control Params.	Majority-on					Multiplexer					Genome
Problem Size	3-bit	5-bit	7-bit	9-bit	11-bit	6-bit	11-bit	20-bit	37-bit	6-bit+ 14EF	4-TIT +16EF
Dataset Size	1,000	2,000	3,000	4,000	5,000	500	1,000	2,000	4,000	2,000	1,600
CCEA Parameters											
Bin Size Order 1	3	5	7	9	11	6	11	20	37	20	20
Bin Size Order ≥2	3	10	35	350	3,500	25	50	100	200	100	100
Max Bin Order	3	5	6	6	7	6	6	6	7	6	6
Max Archive	33	85	232	1,809	21,071	181	311	570	1,297	570	570
Max Popsize	48	110	267	1,824	21,126	211	366	670	1,482	670	670
DNFEA Parameters											
Max Bin Order	5	12				6	10	18	34	6	6
Max Archive	150	360				180	300	540	1,020	180	180
Max Popsize	250	460				280	400	640	1,120	280	280

2.4 Results

On all repetitions of all problems tested, the CCEA was successful in archiving all true generative CCs and the DNFEA was successful in archiving the true generative DNF (see Tables 2.6-2.7).

Table 2.6: CCEA and LCS results on the test problems. TIT stands for Ternary Digit, and EF stands for Extraneous Features. Genome refers to the Synthetic Genome Problem. For the CCEA results, we report the median number of evaluations out of 30 repetitions. For the LCS results, # instances refers to the number of instances used before the system achieved 100% accuracy. For the 6-bit multiplexer problem with 14 EF we report the median # evaluations for the base case, but we note that the runs with imbalanced classes, noise in the class data, and missing data had very similar median values.

Control Params.	Majority-on					Multiplexer					Genome
Problem Size	3-bit	5-bit	7-bit	9-bit	11-bit	6-bit	11-bit	20-bit	37-bit	6-bit+ 14EF	4-TIT +16EF
Search Space	52	484	4,372	39,364	354,292	1,456	354,292	7e9	9e17	7e9	1e12
CCEA Results											
# Evals	105	749	4,432	30,418	295,369	2,201	19,891	193,929	2,811,841	16,214	8,316
Iqbal et al. (2013c) results using actions e											
# Classifiers			3,000			500	1,000	2,000	6,000		
# Instances			20,000			3,333	9,698	59,549	1,367,925		
Iqbal et al. (2014) results											
# Classifiers	500	1,000	2,000				1,000	2,000	6,000		
# Instances	16,738	63,862	250,000				31,098	41,921	109,123		
Urbanowicz and Moore [a](2015), [b](2010) results											
# Classifiers						500[a]	1,000[a]	2,000[a]	5,000[a]		1,600[b]
# Instances						12,203[a]	17,966[a]	43,729[a]	75,932[a]		1e6[b]

Table 2.7: DNFEA results on the test problems. TIT stands for ternary digit, and EF stands for Extraneous Features. Genome refers to the Synthetic Genome Problem. We report the median number of DNFs in the search space and the median number of evaluations out of 30 repetitions. For the 6-bit multiplexer problem with 14 EF we report the # of evaluations for the base case; but we note that the runs with imbalanced classes, noise in the class data, or missing data had very similar median values.

DNFEA	Majority-on		Multiplexer					Genome
Problem Size	3-bit	5-bit	6-bit	11-bit	20-bit	37-bit	6-bit+ 14EF	4-TIT+ 16EF
Search Space	739	1e12	6e9	1e18	3e33	1e66	5e13	9,933
# Evals	491	4,789,542	3,724	16,377	67,758	393,857	3,578	2,089

2.4.1 Results on Binary Benchmark Problems

For problems with relatively small search spaces and a large number of true generative CCs (e.g., the majority-on problems), the CCEA was no more efficient than exhaustive search (Table 2.6, Fig. 2.2a). Although total evaluations are not commonly

reported in the LCS community, we note that the fewest reported number of required data instances reported for majority-on (Iqbal et al., 2013c; 2014) (which is a strong lower bound on the total number of evaluations required) are actually orders of magnitude higher than exhaustive search would require (Table 2.6, Fig. 2.2a). However, for the multiplexer problems (which have relatively few true generative CCs) the CCEA proved increasingly efficient relative to exhaustive search as the problem size increased (Table 2.6, Fig. 2.2c). In the multiplexer problem, the fewest number of required instances reported by the LCS community (Iqbal et al., 2013b; 2014; Urbanowicz and Moore, 2015) appears to be scaling slightly better than the number of evaluations required by the CCEA (Table 2.6, Fig. 2.2c). However it is not clear whether the CCEA is inherently less computationally efficient on the multiplexer problem than LCS because (a) different LCS methods gave the best results on different problem sizes (Fig. 2.2c), (b) the number of required instances is only a lower bound on the number of evaluations required by the LCS approaches, and (c) there may be further efficiencies to be gained by additional optimization of the CCEA parameters.

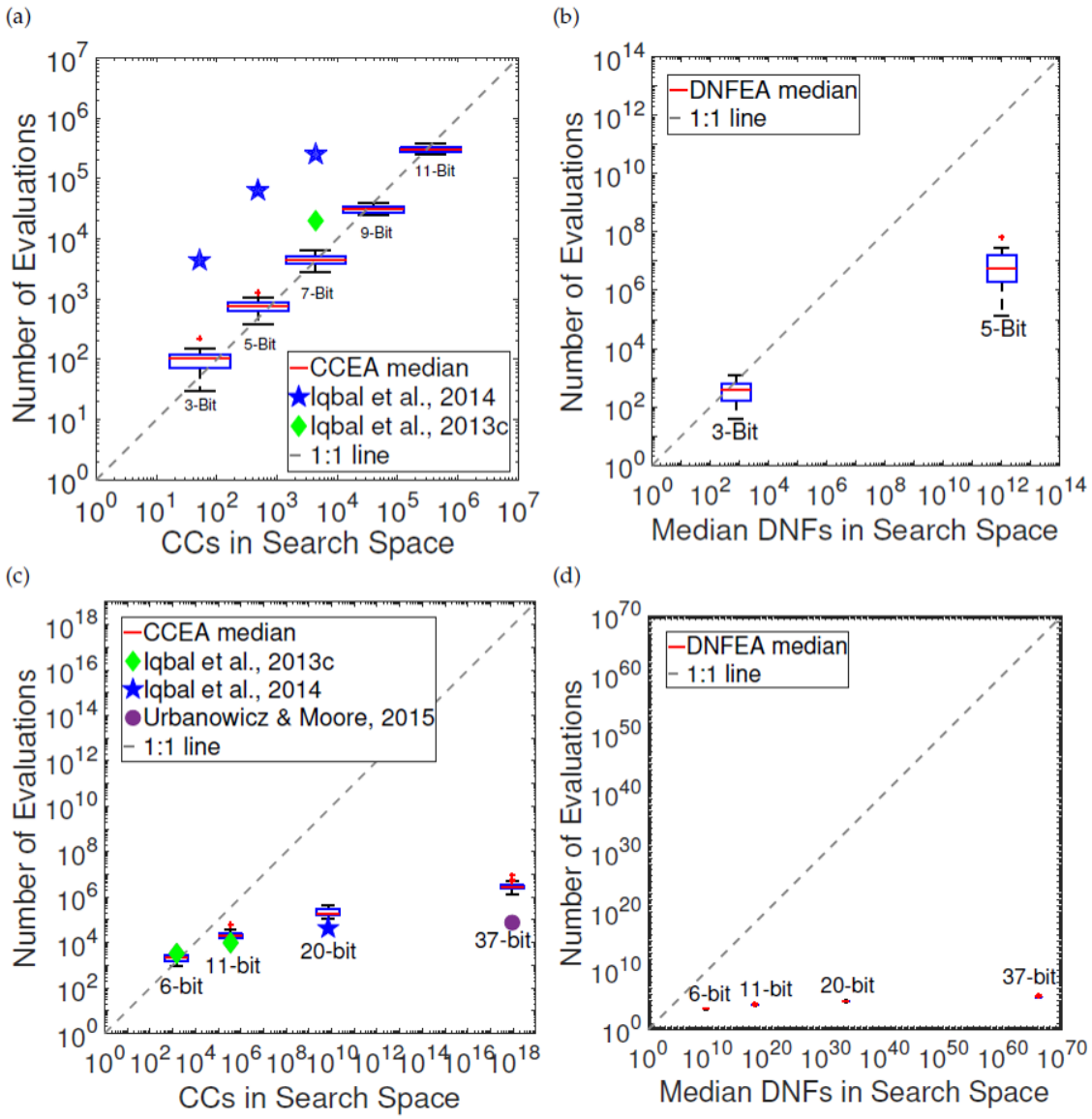


Figure 2.2: Efficiency of the algorithm on the binary benchmark problems. (a) Majority-on: Box plots of the number of CCEA fitness evaluations as a function of the size of the search space, compared to the published results for the number of instances evaluated (a lower bound on the number of fitness evaluations), and exhaustive search (the 1:1 line); (b) Majority-on: Box plots of the number of fitness evaluations of the DNFEA as a function of the median size of the search space over 30 repetitions, (c) Multiplexer: Box plots of the number of CCEA fitness evaluations as a function of the size of the search space, compared to the published results for the number of instances evaluated (a lower bound on the number of fitness evaluations), and exhaustive search (the 1:1 line); (c) Multiplexer: Box plots of the number of fitness evaluations of the DNFEA as a function of the median size of the search space over 30 repetitions.

Regardless of whether the CCEA is more or less efficient than LCS, it is important to note that the number of CCs archived by the CCEA is much smaller than the population of classifiers returned by the LCS methods (Tables 2.5, 2.6), and the CCEA archives (unlike LCS classifiers) always included the true generative CCs (i.e., also had 100% coverage of the complete search space, not just 100% true positive predictions on recent instances sampled).

On both the majority-on and multiplexer problems, the DNFEA was always able to archive the true generative DNF and scaled much better than exhaustive search (Table 2.7, Fig. 2.2b,d).

A closer examination of the archived CC and DNF clauses illustrates the power of using Eq. (2.1) (rather than classification accuracy) as a measure of fitness. For example, in Fig. 2.3a we show results from a typical 6-bit multiplexer run with 14 extraneous features. In this figure, archived CCs are shown with green squares, where darker shading indicates higher-order conjunctions. Similarly, archived DNFs are shown with blue circles, where darker blue indicates higher-order disjunctions. For clarity, the 8 true archived 3rd order generative CCs are shown in orange hexagrams and the single true archived 4th order generative DNF for class 0 is shown with the red pentagram. In this noise-free problem, even though the 2,000 instances in the dataset represent a tiny fraction of the CC and DNF search spaces (Table 2.5), the true DNF is clearly identifiable as the single solution that has 100% true positive rate (a.k.a. accuracy) and 100% coverage and has the highest fitness according to Eq. (2.1). Note: There are many

suboptimal CCs and DNFs with 100% true positive rate, which highlights why true positive rate (accuracy) alone is an insufficient fitness metric.

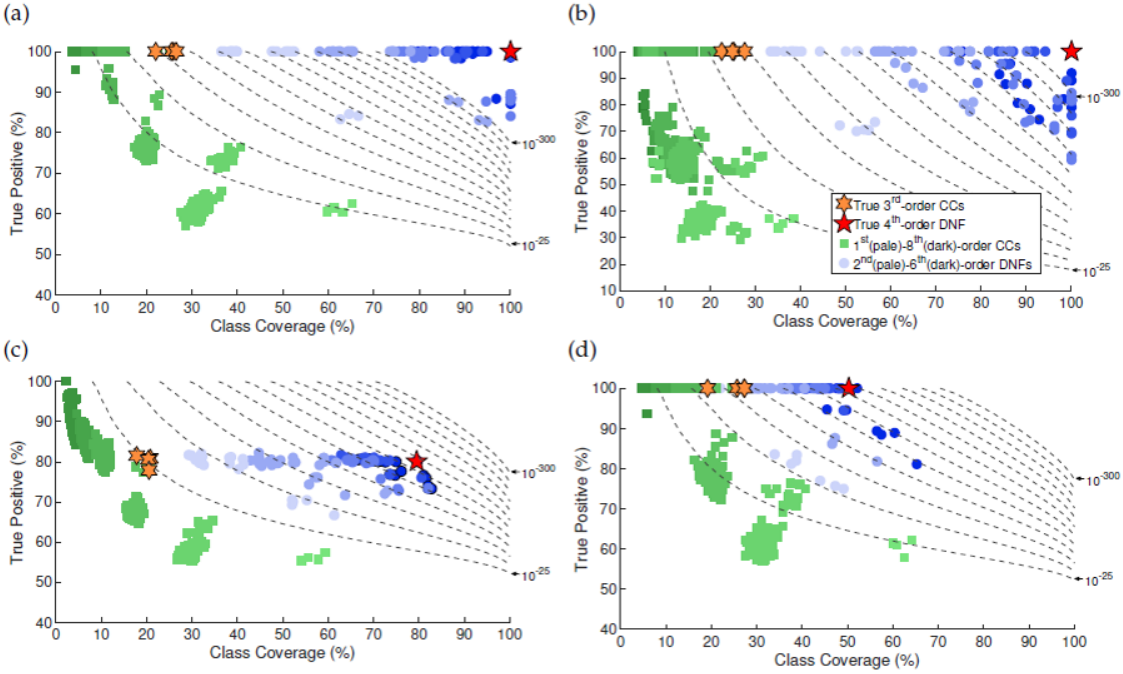


Figure 2.3: Archived results in typical results (arbitrarily selected as the first of 30 repetitions) for target class 0 on the 6-bit multiplexer problem with 14 extraneous features added and 2,000 random instances in the dataset for (a) balanced classes with no noise and no missing data, (b) imbalanced class outcomes (class 0 at 15%, class 1 at 85%) with no noise and no missing data, (c) 20% random errors in class outcome in the dataset, balanced classes and no missing data, and (d) 20% randomly missing feature data, balanced classes and no noise. The legend on panel (b) applies to all panels. We illustrate the true positive prediction rate on the training instances, class coverage of the training instances, and fitness by Eq. (2.1); the true generative CCs are shown in orange hexagrams and all other CCs archived by the CCEA with green squares, and the true generative DNF is shown by the red pentagram and all other DNFs archived by the DNFEA with blue circles. Darker shades of green or blue represent higher order clauses and the contour lines indicate evenly-spaced fitness values.

Even with highly imbalanced classes (15%/85%), the tandem algorithm is able to reliably find the exact generative DNF for the minor class (e.g., Fig. 2.3b). When 20% noise is added to the outcome classes, the true positive rate and coverage are necessarily reduced, but the true generative DNF still consistently stands out as the archived DNF with the highest fitness according to Eq. (2.1) (e.g., Fig. 2.3c). Finally, we observed that

even with 20% missing data in the input dataset, the true generative DNF always had orders of magnitude better fitness than any other 4th-order DNF; and in 77% of trials, this was also the clause with the best fitness. However, in 12%, 10%, and 2% of the trials we found, 1, 2, or 3 higher-order clause(s), respectively, with a slightly better fitness (e.g., see one example in Fig. 2.3d). In these cases, the true generative DNF could still be identified as the most parsimonious (i.e., lowest order) of the most fit DNFs.

2.4.2 Results on Synthetic Genome Problem

The synthetic genome problem includes extraneous features and noise in class outcomes, so it is not possible to achieve 100% true positive prediction or coverage. However, we still observed that the 4 true generative 2nd-order CCs were consistently archived in 30 out of 30 trials, and required 2 orders of magnitude fewer evaluations than reported instances required by XCS (Table 2.6), even though the latter did not report finding the true generative CCs.

The true generative 4th-order DNF was also archived in all 30 trials. While no 4th-order DNF had higher coverage than the true generative DNF, we found numerous DNFs that had higher true positive rate than the true generative DNF and still had relatively high (> 70%) coverage (Fig. 2.4). In general, in real problems one does not know how many, or which, features are part of the true CCs and which are potentially extraneous. However, in post-processing of the CC archive, we observed that the 4 true features occurred in archived CCs (across all 30 repetitions) twice as often as any of the extraneous features. Thus, to reduce the size of the DNF search space, we then presented the DNFEA with those archived CCs that contained only the most prevalent features.

The true generative 4th-order DNF is consistently identifiable as a highly-fit DNF with the greatest coverage (e.g., see Fig. 2.4).

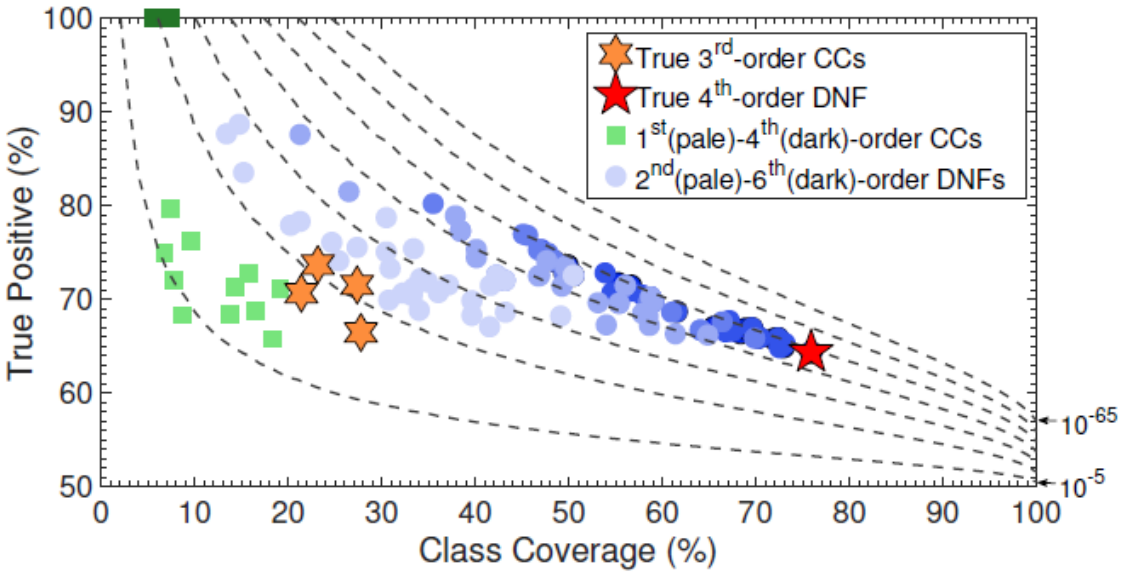


Figure 2.4: Archived results on the Synthetic Genome Problemtrained on 1,600 instances, where the CC archive was reduced by post-processing to include only those features that were most prevalent prior to running the DNFEA. We illustrate the true positive prediction rate on the training instances, class coverage, and fitness by Eq. (2.1) of the CCs archived by the CCEA (green squares) and the DNFs archived by the DNFEA (blue circles), where darker shades represent higher-order clauses and the contour lines indicate evenly-spaced fitness values. For clarity, the true generative CCs are shown in orange hexagrams and the true generative DNF is shown by the red pentagram.

2.5 Discussion

There is a growing availability of Big Data and an increasing recognition that many (probably most) systems of interest are complex. These observations highlight the need for new data analysis tools that are capable of discovering interesting (potentially causal) complex rule sets (that may contain non-linearities, overlap, and heterogeneity) from potentially messy datasets (that itself contain heterogeneous features, missing data,

imbalanced outcome classes, and imperfect relationships between features and outcomes).

2.5.1 Binary Benchmark Problems

Unfortunately, there are relatively few benchmark datasets with tunable heterogeneity and/or epistasis. Two classic benchmark problems that have been widely used in the EA community include the majority-on problem, which includes heterogeneous and overlapping conditions, but not epistasis (Iqbal et al., 2013c; 2014; McDermott et al., 2012) and the multiplexer problem, which includes both heterogeneity and epistasis (Booker, 1989; De Jong and Spears, 1991; Goldberg, 1989; Wilson, 1987a; b; Butz et al., 2003; 2004; 2005; Butz and Pelikan, 2006; Ioannides et al., 2011; Iqbal et al., 2012; 2013a; b; c, 2014; 2015; Llorà et al., 2005; McDermott et al., 2012; Urbanowicz and Moore, 2015). However, both are noise-free, balanced, binary, classification problems with no missing data and; as we show later, both have other characteristics not representative of real-world problems. We note that the genetic programming community has acknowledged that better methods exist for solving these Boolean problems (White et al., 2013) and many of the test problems that have been reported on have so few features that exhaustive search is more efficient than using an EA search strategy. To compare to published results from the LCS community, we have tested our proposed approach on these two classic benchmark problems. However, our work underscores the need for better benchmarks with tunable epistasis and heterogeneity, which are more representative of real-world applications.

The presence of overlapping CCs is the primary reason that the majority-on problem has been used as a benchmark in the LCS community (Iqbal et al., 2013b; c; 2014). One of the most reliable Michigan-style LCSs, referred to as XCS, struggles with this overlap. Kovacs (2002) noted that the XCS algorithm penalizes against overlapping CCs; and Ioannides et al. (2011) showed that even when XCS is initialized with a population containing the overlapping true signals, they are selected out of the CC population. When Iqbal et al. (2013c) used XCS to tackle the 7-bit majority-on problem, the evolved CCs were an order or two below that of the true generative CCs. On the other hand, when a variant of XCS was used that evolves a logical representation of the action set, the CCs found were usually (23 out of 30 times) at least one order greater than the true generative CCs (Iqbal et al., 2013c). Therefore, even when 100% classification accuracy was reported for small majority-on problems (3-, 5-, and 7-bit) (Iqbal et al., 2013b; c; 2014), the true generative CCs were not identified. It is likely that significant overfitting is occurring in these large populations of overly-specific classifiers.

One additional limitation of the majority-on problem is that, as the problem increases from 3-bit to 11-bit, the number of true generative CCs increases from 6 to 924, while the expected coverage of each true CC decreases from 25% to only 1.6% (Table 2.8). In real-world association problems, if one reported identifying a 924-order disjunction of order-6 conjunctions, each with only 1.6% coverage of the dataset, this would be dismissed as extreme overfitting. Despite these limitations, the proposed CCEA was consistently able to archive all of the true generative CCs in up to 11-bit majority-on problems, and the proposed DNFEA was able to archive the single true generative

DNF in up to 5-bit majority-on problems, and this optimal DNF was easily identifiable as the archived clause with the best fitness by Eq. (2.1).

Table 2.8: Characteristics for Majority-On (MO) and Multiplexer (MP) benchmark problems.

Problem	# Possible CCs	Order of True CCs	# of True CCs in Generative Rule Set	Expected Coverage of Each True CC
3-bit MO	52	2	6	25.0%
5-bit MO	484	3	20	12.5%
7-bit MO	4,372	4	70	6.3%
9-bit MO	39,364	5	252	3.1%
11-bit MO	354,292	6	924	1.6%
6-bit MP	1,456	3	8	12.5%
11-bit MP	354,292	4	16	6.3%
20-bit MP	7×10^{9}	5	32	3.2%
37-bit MP	9×10^{17}	6	64	1.6%

The presence of tunable degrees of heterogeneity and epistasis is the primary reason why the multiplexer problem continues to be a standard benchmark problem in both the LCS and genetic programming (GP) communities. White et al. (2013) noted that between 2009-2012, approximately 10% of GP papers submitted to EuroGP and GECCO used the multiplexer problem, despite acknowledgment that these problems are trivial to solve using non-GP techniques (White et al., 2013). In the LCS community, some (Kovacs, 1998; Butz et al., 2003) have noted (1) the existence of many non-optimal CCs that have the same true positive rate and expected coverage as the true generative CCs, and (2) LCS typically returns populations of classifiers (e.g., see Table 2.6) that are much larger than the number of CCs in the true generative DNF (Table 2.8), and sometimes even larger than the maximum possible number of CCs in the search space (Table 2.6). Additionally, as the size of the multiplexer problem increases, the number of true CCs in the true generative DNF increases (albeit not as rapidly as in the majority-on problem)

and the individual coverage rapidly decreases (Table 2.8). Despite these issues, our proposed approach consistently evolved and identified the single true generative DNF in all multiplexer problems tested (up to 37-bit).

Furthermore, even when we introduced extraneous features, imbalanced classes, noise in the class associations, and missing data into a 6-bit multiplexer problem, our proposed method was able to reliably evolve and identify the single true generative 8th-order DNF of 3rd-order CCs (Fig. 2.3). It is encouraging that the CCEA and DNFEA continued to perform so strongly even in the face of significant amounts of class imbalance, noise in class associations, and missing data, since these are often characteristics of real-world datasets. Of particular importance is the ability to handle missing data gracefully, without the need for imputation with potentially misleading fake data.

2.5.2 Synthetic Genome Problem

The synthetic genome problem introduced in Urbanowicz and Moore (2010) was defined to be a more realistic dataset representing a heterogeneous, purely epistatic problem, in which the true generative DNF is a 4th-order disjunction of four 2nd-order CCs. This dataset includes 16 extraneous features and an imperfect association between the true features and balanced binary outcome classes. Our approach consistently archived all 4 true generative CCs and also archived the single true generative DNF, which was readily identifiable as the very fit clause with the highest coverage in the resulting DNF archive.

Although there was no true generative DNF for class 0 in the synthetic genome problem, Urbanowicz and Moore (2010) used XCS to evolve rules for predicting both class 0 and class 1 and reported an average classification accuracy of over 88% using 10-fold cross validation with 1,600 classifiers trained on 1,440 unique training instances (repeatedly sampled for a total of 1,000,000 instances shown to XCS), and up to 72% on the testing data. However, recall that the positive prediction rate on the actual dataset using the true generative DNF for class 1, and assuming class 0 otherwise, is only 67%. These results highlight the danger of overfitting that is inherent in LCS approaches.

2.5.3 Fitness Landscape Analysis

It is common in the LCS community to use "classification accuracy" (more appropriately described as the true positive rate of class predictions on some number – typically 100 or 1,000 – of the most recent instances tested) as the primary metric of success. We contend that, more often than not, using this metric as a proxy for fitness results in overfitting. LCS algorithms have been touted for achieving 100% classification accuracy on the 3-, 5-, and 7-bits majority-on problems (Iqbal et al., 2013b; c; 2014) and the 6-, 11-, 20- and 37-bits multiplexer problems (Iqbal et al., 2013c; 2014; Urbanowicz and Moore, 2015). However, when Iqbal et al. (2013b; c; 2014) solved the 3-bit and 5-bit majority-on problems, the classifier population sizes were larger than the number of possible CCs and for the 7-bit majority-on problem the population size was ~46 – 69% of the number of possible CCs (Table 2.6).

As illustrated in Figs. 2.3-2.4, there are many sub-optimal clauses that have 100% true positive prediction rate (a.k.a. accuracy). It is even more informative to analyze the

entire search space for the 3 types of test problems used here. In Fig. 2.5 we show the true positive prediction rate and coverage of all possible CCs in the search space (up to 6th-order CCs) for an example 11-bit majority-on problem evaluated on 5,000 random instances (Fig. 2.5a), an 11-bit multiplexer problem evaluated on 1,000 random instances (Fig. 2.5b), and the synthetic genome problem evaluated on the 1,600 instances (Fig. 2.5c). In all 3 panels, the true generative CCs are shown with orange hexagrams.

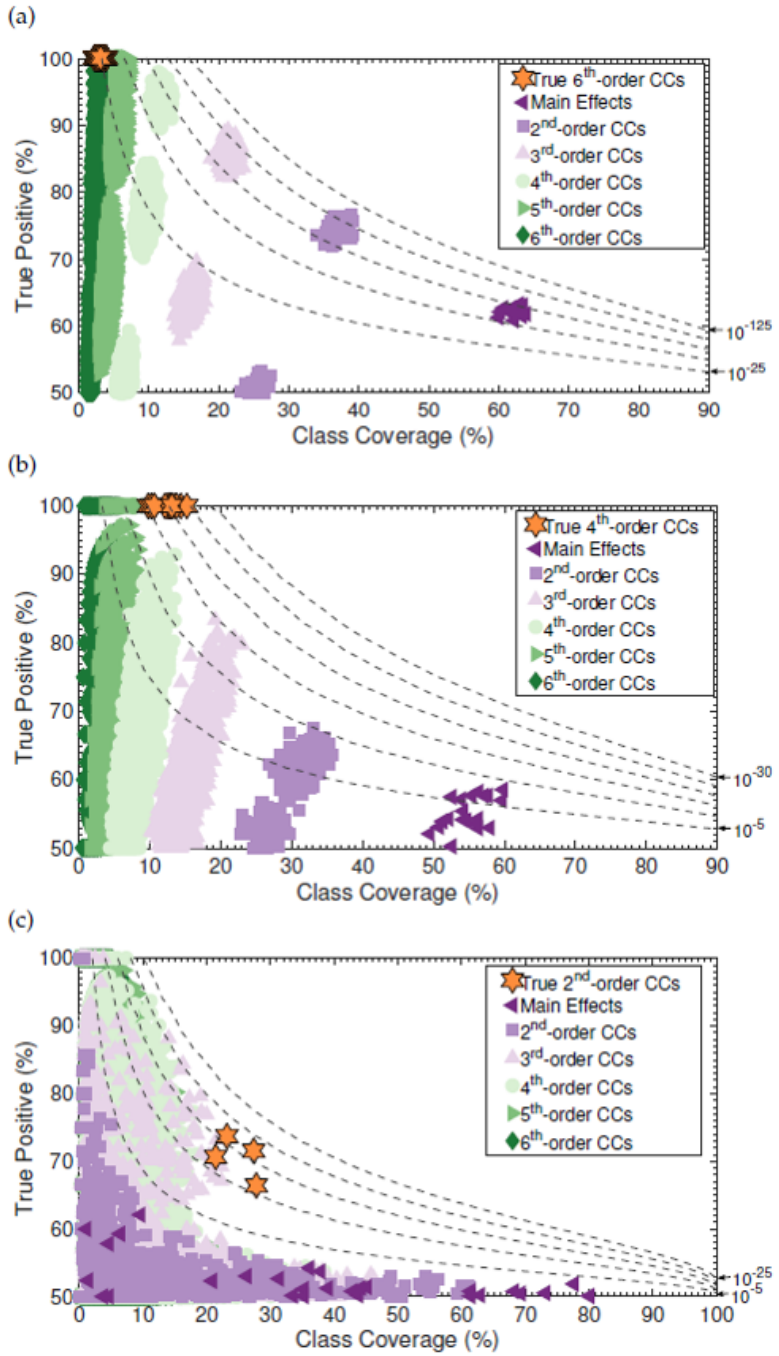


Figure 2.5: Results of using exhaustive search to examine the CC search spaces for (a) a randomly created 11-bit majority-on dataset containing 5,000 input feature vectors, (b) a randomly generated 11-bit multiplexer dataset containing 1,000 input feature vectors, and (c) the simulated SNP disease problem containing 1,600 input feature vectors. We illustrate the true positive prediction rate, class coverage, and fitness by Eq. (2.1) of all possible CCs, where the order of the CCs is indicated by color and the contour lines indicate evenly-spaced fitness values. Note that the lower bounds on the y-axes are 50%.

As seen previously, there are many sub-optimal CCs with 100% true positive prediction rate in the two noise-free Boolean benchmark problems, and many lower order CCs that still have relatively high true positive prediction rate and much higher coverage than the true generative CCs (Figs. 2.5a,b). In the majority-on problem, there are also many sub-optimal CCs that have not only 100% true positive prediction rate, but also have higher coverage than the true generative CCs (Fig. 2.5a). In the synthetic genome problem, which includes noise in the class association, there are many CCs that actually have much higher true positive prediction rate than the 4 true generative CCs (Figs. 2.5c). These observations underscore the danger of using accuracy as a surrogate for fitness.

Also, note how the structure of the fitness landscapes differs between that of the Boolean benchmark problems and the synthetic genome problem. In the former, there are distinct clusters of CCs of different orders, and these are roughly orthogonal to the fitness contours per Eq. (2.1); specifically, in Fig. 2.5a,b, note how the most fit CCs in each order have lower coverage but higher true positive predictive rate, as the order increases (from purple to green). In contrast, in the more realistic synthetic genome problem (Fig. 2.5c), there is significant overlap in the clusters for different orders of CCs, and these are roughly parallel to the fitness contours per Eq. (2.1). These observations illustrate how the majority-on and multiplexer problems exhibit quirky fitness landscape characteristics that are not likely representative of real-world problems.

2.5.4 Hypergeometric PMF as a Fitness Metric

In this work, we propose the use of a hypergeometric probability mass function as a principled statistic for assessing relative fitness for clauses of a given order.

Specifically, Eq. (2.1) quantifies the likelihood that the observed association between a given clause and a given target class is due to chance, taking into account the size of the dataset, the amount of missing data, and the distribution of outcome categories. We use dynamically-adjusted, order-specific, probability thresholds to determine which CC and DNF clauses to archive, much like the original intention of the *p*-values in traditional statistical analysis. That is, any CCs or DNFs below a probability threshold is worth further examination; but while these clauses are *potentially* causal, a low value of Eq. (2.1) alone does not imply causation (Nuzzo, 2014). Unlike relying on accuracy or other *ad hoc* measures as a fitness proxy, using Eq. (2.1) enables our algorithm to archive clauses with different combinations of true positive prediction rate and coverage while gracefully handling imbalanced classes, missing data, and noisy class associations. However, one of the drawbacks is that the rounding error becomes problematic at values below 10^{-300} on a 64-bit computer. Although real-world datasets will likely contain too much noise for this to happen, further research is needed to explore an estimate for very small values of the hypergeometric PMFs.

In Fig. 2.5, note how the true generative CCs (orange hexagrams) have better fitness (per the fitness contours from Eq. (2.1)) than any other CCs of the same order, but how the maximum fitness of a given order of CC varies (sometimes nonlinearly) by CC order. In particular, while the 4 true generative CCs in the synthetic genome problem (Fig. 2.5c) have higher fitness than any other 2nd-order CCs in this landscape, there are other higher-order CCs that have better fitnesses than some of the true CCs. This illustrates the importance of maintaining order-specific thresholds for the

hypergeometric PMFbased fitnesses that determine which clauses are retained in the CCEA and DNFEA archives.

2.5.5 CCEA and DNFEA

LCSs were designed to learn collectively predictive rules from dynamically changing datasets; they were not designed to find optimal parsimonious (potentially causal) rule sets, or for working efficiently on datasets available in batch. Since LCSs learn one instance at a time, LCSs cannot explicitly learn the coverage of classifiers; so it is not clear how well the resulting classifiers cover the dataset. Furthermore, they typically rely on classification accuracy as a major component of the fitness function, which we have shown to be unable to discriminate between optimal and sub-optimal classifiers and can lead to overfitting, especially when there is noise in the dataset. However, even though Urbanowicz and Moore (2010) used cross-validation when applying XCS to the noisy synthetic genome problem, there is still evidence of overfitting since the average training accuracy was 20% higher than the accuracy of the true generative rule set.

To tackle the challenge of analyzing complex real-world datasets that include missing data as well as imbalanced and noisy class associations, we have proposed a new approach using tandem age-layered EAs on batch data. The CCEA creates an archive of CCs that are likely to have a probabilistically significant association with a given outcome class. The DNFEA subsequently creates an archive of probabilistically significant disjunctions of the archived CCs. As in Hornby (2006), we found the age-layering to be very important in maintaining diversity, which facilitated continual

improvement over the course of the evolutionary process. By maintaining separate archive bins for clauses of different orders, the tandem algorithm is able to evolve parsimonious rule sets without making *a priori* assumptions on the maximum order of interactions.

It is important to note that the CCEA and DNFEA algorithms do not necessarily need to be run in tandem, and can each be used independently. For example, in (Chapter 3), the CCEA was used to mine data from large socioeconomic surveys aimed at identifying the drivers of household infestation with an insect that transmits Chagas disease, which if untreated is life-threatening. We discuss this real-world application below, in Section 2.5.6. Similarly, the DNFEA can also be used independently of the CCEA. For example, one could apply the DNFEA to identify heterogeneous rule sets comprised of CCs that were identified by means other than the CCEA, such as through LCS, GP, Random Chemistry, or exhaustive search (if the size of the CC search space is small enough).

While both the CCEA and the DNFEA do some implicit feature reduction by archiving only very fit clauses, in high-dimensional problems one could employ feature reduction methods to first reduce the size of the search space to the more promising features, before applying these methods.

2.5.6 Real-World Application

In the past 5 years, a collaborative effort between the University of Vermont, Loyola University New Orleans, and La Universidad de San Carlos Guatemala have performed detailed socioeconomic and entomological surveys on over 20 towns in

Guatemala, El Salvador, and Honduras to study the risk of Chaagas disease. Mining these complex Chagas survey datasets for useful information has proven to be a major challenge, due to a variety of factors including missing data, imbalanced class outcomes, heterogeneity of drivers of infestation, non-independence of some features, and the expectation of complex high-order nonlinear and overlapping interactions between many of the potential predictive features. Initial attempts to apply the ExSTraCS 1.0 LCS to this data were unsuccessful, which is what motivated the development of the CCEA.

The surveys contain 64 risk factors that experts believe are associated with infestation of households with *Triatoma dimidiata*, a vector of Chagas disease. Fourteen of the risk factors are ordinal/continuous and the remaining 50 are nominal, with 26% missing data and imbalanced class outcomes (32% infested households). In analyzing this real-world dataset, we did not seek a single "optimal" DNF, but rather used the CCEA to find a variety of very fit CCs that could be more closely examined by domain experts to assess (a) whether new insights could be achieved regarding combinations of risk factors associated with *T. dimidiata* infestation, and (b) whether very fit CCs might inform the design of new ecohealth intervention strategies that could prove to be feasible, effective, and cost-effective ways to slow the spread of Chagas disease. The CCEA discovered several interesting heterogeneous and overlapping CCs (ranging from main effects through 7th-order epistatic CCs). Some of the feature interactions evolved by the CCEA had already been previously identified as potential drivers of infestation, which increases our confidence in the CCEA results. However, the CCEA analysis also provided ranges of co-evolved values of interacting features that were most strongly

associated with infestation as well as new feature interactions previously not recognized to be associated with infestation. These new findings will be useful for informing the design of eco-interventions aimed at slowing the spread of Chagas disease. While a full discussion of this application is beyond the scope of this paper, we refer the interested reader to (Chapter 3) for more details.

2.5.7 Summary

In summary, we developed a new approach for discovering parsimonious predictive rule sets that contain potentially heterogeneous, epistatic, and overlapping rules. The method was designed to work on complex batch datasets that may include features of different data types, extraneous features, imbalanced classes, noisy associations between rules and class outcomes, and missing data. Key aspects of our proposed method include (a) the use of a hypergeometric probability mass function as a principled statistic for assessing fitness, which properly accounts for class imbalance and missing data, (b) tandem age-layered evolutionary algorithms for evolving archives of probabilistically significant conjunctive clauses, and disjunctions of these archived conjunctions that are optimally predictive of outcome classes, and (c) separate archive bins for clauses of different orders, with dynamically-adjusted order-specific fitness thresholds. The method was validated on standard binary majority-on and multiplexer benchmark classification problems, including several variants of the multiplexer problem with extraneous features that included class imbalance, noise, extraneous features, or missing data. The method was also applied to a more realistic synthetic genome problem with heterogeneous, purely epistatic, and noisy association rules. In all problems tested,

we were consistently able to evolve the true generative rule sets in the form of a single clause in disjunctive normal form. An in-depth examination of the search space of all possible conjunctive clauses exposed unusual characteristics of the majority-on and multiplexer problems that are not likely representative of real-world problems. This highlights the need for more realistic benchmark classification problems with tunable epistasis, heterogeneity, and overlap in the generative rule sets. Finally, we briefly discussed the application of the method to the complex real-world survey dataset that actually motivated us to develop the CCEA. The results of this analysis provided important practical insights that will inform eco-intervention strategies aimed at slowing the spread of the deadly Chagas disease.

CHAPTER 3: AN EVOLUTIONARY ALGORITHM APPROACH TO IDENTIFYING COMPLEX INTERACTIONS ASSOCIATED WITH THE INFESTATION OF *TRIATOMA DIMIDIATA*, A VECTOR OF CHAGAS DISEASE

3.1 Introduction and Significance

This work was motivated by a desire to mine data from large socioeconomic surveys with an aim toward identifying the drivers of house infestation by an insect that transmits Chagas disease. The disease is transmitted by insects in the subfamily Triatominae (Lent and Wygodzinsk, 1979) and, if left untreated, is life-threatening in about 30% of cases. To decrease risk of transmission, mitigation strategies (known as Ecohealth interventions) have been implemented to remove known hiding locations and lessen the chance of house infestation of the Triatomine vectors (Monroy et al., 2009; Lucero et al., 2013; Pellecer et al., 2013). Because many areas where the disease is endemic have limited resources for these preventative house improvements, it is useful to conduct detailed entomologic and socioeconomic surveys (Bustamante et al., 2014; Bustamante Zamora et al., 2015) to help (1) identify the drivers of infestation and (2) monitor, improve and assess cost-effective mitigation strategies. Mining these complex survey datasets for useful information is challenging due to a variety of factors including imbalanced categorical outcomes, heterogeneity, missing data, and complex, possibly high-order, nonlinear interactions between many of the potential predictive features. As a result, we developed an evolutionary algorithm to find additive feature interactions

(features with main affects) as well as heterogeneous feature interactions for complex real-world datasets.

The scientific community is just beginning to understand some of the profound affects that these nonlinear (i.e., epistatic) feature interactions have on natural systems. Feature interaction is a phenomenon that arises when features combine to produce an effect, which neither alone controls (i.e., feature X does one thing; feature Y does another; and when combined, X and Y do a third thing that has no single controlling element). These feature interactions have been observed in cascading power failures (Eppstein and Hines, 2012), breast cancer (Ritchie et al., 2001), blood pressure in rats (Rapp et al., 1998), and are believed to be ubiquitous in human diseases (Moore, 2003). In addition to feature interactions, heterogeneity is when multiple features independently predict of the same output. Evidence of heterogeneity has been observed in bladder cancer (Urbanowicz et al., 2013), autism (Buxbaum et al., 2001), and American political parties (Poole and Rosenthal, 1984). Studies of systems that consider both heterogeneity and feature interactions are just beginning to appear in the literature. In the context of this work, examples of heterogeneous two–way feature interactions (habitat and food source) that are associated with houses infested with triatomine vectors might be (1) cracks in adobe walls and chicken coops in the house or (2) firewood stacked adjacent to the house and dogs sleeping in the home. Each two-way feature interaction set may be equally important drivers of infestation; our algorithm development was motivated by a desire to preserve main effects as well as higher-order heterogeneous, feature interactions.

Despite the belief that heterogeneity and feature interactions exist across numerous real-world systems (e.g., from the development of personalized drug therapies (Wilson, 2009) to the market prediction of consumer behaviors (Young Kim and Kim, 2004)), the development of tools for analyzing these systems and accommodating these complex feature interactions have not kept pace. We hypothesize that a large source of error in "Big Data" science is a result of feature interactions and heterogeneity. Feature interaction error is not random; it is complicated, but predictable. If we are to develop tools to assist in unraveling these complex datasets, feature interactions and heterogeneity must be considered (Thornton-Wells et al., 2004).

Traditional statistical methods such as analysis of variance (Yousefi et al. 2016; Wilson et al. 2017), logistic regression (Heller et al. 2011; de Campos Franci et al. 2016; Ding et al. 2016; Jarlenski et al. 2016; Larouche et al. 2016; Li et al. 2016; Nesheli et al. 2016; Nicholls et al. 2016), and decision trees (Markellos et al. 2016; Nesheli et al. 2016) are well suited for univariate analysis or additive models. Some studies perform feature selection using univariate logistic regression models, and then test higher-order interactions between the selected features (De Andrade et al. 1995; Enger et al. 2004; Rassi et al. 2006; King et al. 2011; Weeks et al. 2013; Sperandio da Silva 2014; Kaplinski et al. 2015; Molina et al. 2015; Olivera et al. 2015). For systems with significant feature interactions, traditional statistics, designed for additive multivariate relationships, are not well-suited.

Another well-documented issue is that p-values decrease inversely with the size of the dataset, making them an unreliable statistic for Big Data applications (Lin et al.,

2013). Our algorithm development leverages the hypergeometric probability mass function (PMF) as a probabilistic threshold (Hanley et al., *In Review*). The hypergeometric PMF is derived from Pearson's (1899) hypergeometrical series and may be thought of as a pseudo-Bayesian equation. One benefit of the hypergeometric PMF-derived probability is that it accounts for both the size of the dataset and the distribution of the output categories; and using it as a threshold allows the user to readily compare the probabilities (and thus the likelihood) of individual models.

In this work, we present an evolutionary algorithm (EA) that was specifically designed as a non-parametric method for identifying feature interactions in "Big Datasets" that contain missing values, heterogeneity, and additive probabilistic models associated with a desired categorical outcome (e.g., disease or infestation). Our EA searches for combinations of feature sets using the logical AND operator; these feature combinations (e.g., cracks in adobe walls and chicken coops in the house) are referred to as conjunctive clauses throughout this work. To demonstrate the EA effectiveness, we first test the algorithm on the benchmark dataset of Urbanowicz and Moore (2010); the latter was specifically designed to include heterogeneity and feature interactions associated with a complex disease. Next, we use the EA to identify complex multivariate interactions (i.e., risk factors) in real-world datasets associated with house infestation of the Chagas disease vector Triatoma dimidiata. Finally, we show the EA's ability to efficiently search for potential drivers of T. dimidiata infestation and discuss how these models might be implemented by domain experts familiar with stakeholder needs.

3.2 Background

3.2.1 Background on Chagas Disease

Chagas disease is caused by the protozoan parasite, *Trypanosoma cruzi*, and is primarily spread via blood feeding insects in the order Hemiptera, family Reduviidae, and subfamily Triatominae (Lent and Wygodzinsky, 1979). While vector food sources include all vertebrates, *T. cruzi* only infects mammals (Rassi et al., 2010). Human impacts, such as deforestation for agrarian land use, have caused triatomines to adapt (Coura, 2015); and one of the main vectors of Chagas disease, *Triatoma dimidiata*, has adapted to human domestic and peridomestic environments (Waleckx et al., 2015a). This vector is endemic from Mexico through Central America, all the way south to parts of Peru, Ecuador, Colombia (Lent and Wygodzinsky, 1979). People with Chagas disease often live in remote areas with poor sanitation, low socioeconomic status, and work manual labor jobs (Prata, 2001; Briceño-León et al., 2007). Approximately 70 million people in Latin America are at risk of infection with *T. cruzi* and ~5.7 million people are already infected (Chagas, 2015). In Central America, Guatemala, the most populous country, was estimated to have the largest number of new vector transmitted cases (~1,275) in 2010 (Chagas, 2015). However, rates of new infections are also high in El Salvador and Honduras.

The insect vectors deposit parasite laden feces and humans can become infected by transmission of *T. cruzi* into the bite or other open wound, or through the mucosa of the eye, nose, or mouth (Prata, 2001; Teixeira et al., 2006; Rassi et al., 2010). Another possible source is via consumption of the infected feces in food items such as vegetables,

juice, and possibly wild meat (Rueda et al., 2014). Oral transmission is believed to be the primary source of infection for wild animals (Coura, 2015); and the odoriferous glands of a marsupial infected with *T. cruzi* can directly transmit the parasite to humans (Coura, 2015).

Chagas disease is broken into three phases. The first is the *acute* phase, which may last 1–4 months after infection with *T. cruzi* (Prata, 2001, Stanaway and Roth, 2015). This phase is asymptomatic in 95% of cases (Teixeira et al., 2006; Stanaway and Roth, 2015); however, for the remaining 5%, symptoms may include malaise, fever, jaundice, skin hemorrhages, enlargement of the liver, and muscle and joint pain (Prata, 2001; Teixeira et al., 2006; Rassi et al., 2010; Stanaway and Roth, 2015). The *indeterminate* phase is asymptomatic and can last 10–30 years or throughout a lifetime (Prata, 2001; Stanaway and Roth, 2015). Finally, the *chronic* phase of Chagas disease has symptoms that include heart disease, megaesophagus, megacolon, nervous system lesions, and sudden death (Prata, 2001; Teixeira et al., 2006; Rassi et al., 2010; Stanaway and Roth, 2015). Currently, there is no preventive medicine for Chagas disease. Nonetheless, there are two anti-trypanosome drugs, nifurtimox and benznidazole for treating *T. cruzi* infections (Teixeira et al., 2006; Jannin and Villa, 2007; Rassi et al., 2010; González-Ramos et al., 2016). Both drugs have common adverse reactions that have prevented 13-31% ID infected people from completing treatment (Hasslocher-Moreno et al., 2012; Sperandio da Silva et al., 2014; Molina et al., 2015; Olivera et al., 2015); (Hasslocher-Moreno et al., 2012; Sperandio da Silva et al., 2014; Molina et al., 2015; Olivera et al., 2015).

Thus, given the lack of preventative medicine coupled with low efficacy of drug treatment, the preferred method of combating Chagas disease is to minimize human contact with the vector. One of the most common tactics for controlling *T. dimidiata* infestation at the house level is the use of pyrethroid insecticide (Tabaru et al., 1998; Acevedo et al., 2000; Nakagawa et al., 2003a; 2003b; Dumonteil et al., 2004; Hashimoto et al., 2006; Manne et al., 2012; Yoshioka et al., 2015; Quinde-Calderón et al., 2016). However, the residual effects appear to last only four months before adult *T. dimidiata* re-infest a house and nine months before nymphs are found in the house (Dumonteil et al., 2004). The rebound to original infestation levels were observed almost three years after a single spraying in Jutiapa, Guatemala (Hashimoto et al., 2006). Thus, short of extirpation of *T. dimidiata*, the vector will always pose a risk for infestation where it is endemic.

The only proven long-term control of *T. dimidiata* infestation is the implementation of home improvements often accompanied by educational outreach on Chagas disease and the vector (Monroy et al., 2009; Ferral et al., 2010; De Urioste-Stone et al., 2015). Home improvements that minimize the risk of *T. dimidiata* infestation run the gamut of cleaning and organizing the peridomestic ecotope (Zeledón and Rojas, 2006; Zeledón et al., 2008; Ferral et al., 2010), plastering walls (Monroy et al., 1998; Monroy et al., 2009; Lucero et al., 2013; Pellecer et al., 2013), replacing dirt floors with cement (Lucero, et al. 2013; Pellecer et al., 2013), installing window screens (Ferral et al., 2010; Waleckx et al., 2015b), impregnating curtains with insecticide (Ferral et al., 2010), and domestic rodent control (De Urioste-Stone et al., 2015). While these home

improvements have led to reductions in infestation that often last longer than spraying, none have completely eliminated infestation. Some of the aforementioned interventions are considered Ecohealth interventions because they use sustainable methods, locally sourced materials, and often include house level surveys of hypothesized risk factors (Monroy et al., 2009; Lucero et al., 2013; Pellecer et al., 2013). Risk analyses capable of identifying complex multivariate interactions in these ever-evolving, real-world datasets would be invaluable for guiding EcoHealth interventions.

3.2.2 Challenges Associated with Modeling/Analyzing Chagas Disease

A number of studies have used univariate statistical analysis as a feature selection tool; and features below a designated *p*-value (e.g., $p < 0.05$) are often selected for a follow-on multivariate analysis (Rassi et al., 2006; King et al., 2011; Weeks et al., 2013; Sperandio da Silva et al., 2014; Kaplinski et al., 2015; Molina et al., 2015; Olivera et al., 2015). Bustamante Zamora et al. (2015) held a workshop to pre-select features for multivariate modeling of *T. dimidiata*. As an initial starting point, features were selected based on previous studies indicating they increased the odds of infestation. Given the large number of potential features associated with the risk of triatomine infestation, it is natural inclination to first reduce the number of model features because their inclusion makes exhaustive search of all possible models (feature combinations) prohibitively expensive and/or impossible. However, when viewed in light of ecological niche modeling or developing risk maps, this is of particular concern because *a priori* use of univariate statistical analysis will do exactly what it is designed to do - select for main effects, and therefore, prematurely eliminate features interactions (i.e., feature

combinations with no main effects) that could be identified in an exhaustive search multivariate analysis.

House infestation with triatomine vectors is an inherently complex, nonlinear system with the potential for a large number of feature interactions. At a minimum, the vectors require a source of shelter and a readily available food source to survive and infest a house; and when viewed as a complex system, other features may be important (e.g., initial vector entry and/or passive modes of transportation into the house).

Another challenge is that many statistical methods cannot include missing data, resulting in removal of data that may contain drivers of infestation. Lastly, not all statistical models allow for the inclusion of multiple data types (e.g., continuous, ordinal, nominal, and binary); and as a result, the features, especially continuous features associated with survey questions may get binned into a limited number of categories. Such binning benefits from and relies on expert knowledge to reduce the input data (types and number); and while the future success of Big Data analysis requires that the tools be used in tandem with expert knowledge, the posterior tinkering of features runs the risk of biasing and/or reinforcing of preconceived conditions.

3.3 Methods and Study Sites

3.3.1 Study Sites of *Triatoma dimidiata* Infestation

Our study sites are the small rural towns of El Chaperno and El Carrizal located in the dry highlands of in Jutiapa, Guatemala (red and yellow dots of Figure 3.1). Jutiapa, Guatemala (highlighted in red, Panel A) borders El Salvador with the study site locations shown as a yellow star. El Carrizal (Panel B) has spur roads radiating from the main road

making the town less linear in shape. While El Chaperno (Panel C) is more linear in shape since most of the houses are adjacent to the principal road running through the town. Also, El Chaperno is more heavily forested than El Carrizal due to forest conservation efforts.

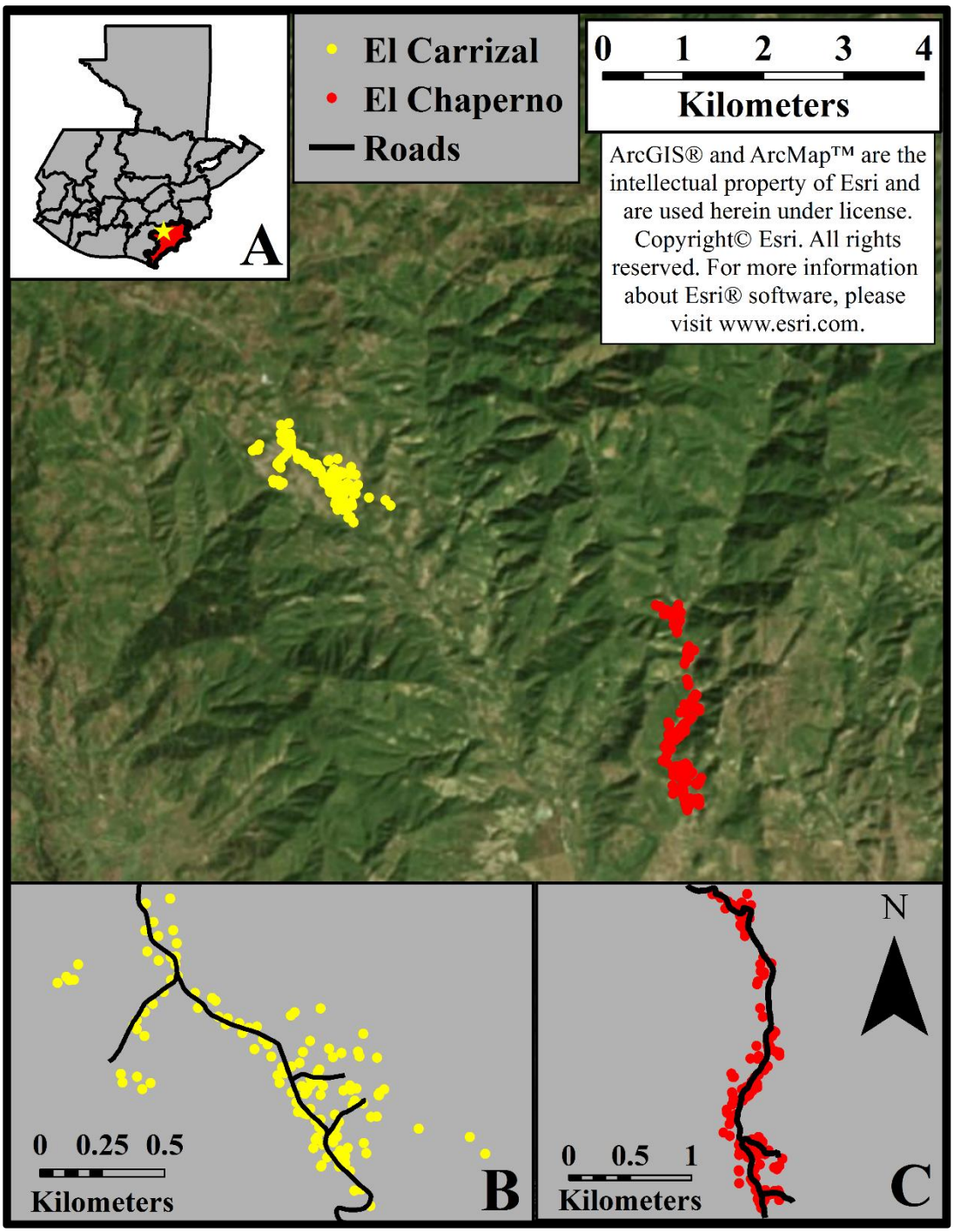


Figure 3.1: Satellite image of the study sites with the houses in El Chaperno and El Carrizal represented as red and yellow dots, respectively. Panel A is a map of the departments of Guatemala with the department of Jutiapa highlighted in red and the location of the study sites represented as a yellow star. Panels B and C show the locations of the houses and roads in El Carrizal and El Chaperno, respectively.

The El Chaperno and El Carrizal house surveys contained 64 features thought to be potential risk factors for infestation with *T. dimidiata* (Table 3.S1). The dataset of each community was analyzed separately, and then combined and re-analyzed to test for larger-scale regional patterns. Given the challenges of finding live *T. dimidiata* (Monroy et al., 1998) and because we are interested in identifying features associated with the risk of house infestation that help further the development of intervention strategies, we define infestation as any sign of *T. dimidiata* presence in the house (i.e., live or dead vectors, eggs, exuviae, or feces) as we believe these signs of *T. dimidiata* are indicative that the house is either currently infested or has been infested in the recent past.

3.3.2 Combinatorial Datasets

While the number of houses in a given dataset (i.e., 129 – 311) may be small, the total number of features and all possible multivariate combination of features make exhaustive search infeasible even on today's computers. For example, let's take the following simplified example. Assume that all features, L, in the dataset have the same number of values, v. If we take a hypothetical dataset with $L = 50$ nominal features, each with $v = 5$ categorical values, and limit each model to one category per feature, then the number of Oth-order models is $v^O \binom{L}{O} = 3.06 \times 10^4$, 2.45×10^6, and 1.44×10^8, for 2nd-, 3rd-, and 4th-order models, respectively. It should be noted that models that do not allow the range of ordinal features values to evolve as part of the model solutions, can bias models against ordinal features. Therefore, when testing models with ranges of ordinal and nominal feature values, the number of two-way interactions (i.e., bivariate models)

is on the order of hundreds of thousands (Table 3.1). For five-way interactions, there are over one trillion possible models for two of the three datasets.

Table 3.1: Possible number of models comprised of 2nd- to 5th-order feature interactions for the El Chaperno, El Carrizal, and the combined datasets.

Dataset	Number of Combinations per Order of Feature Interaction			
	2nd	3rd	4th	5th
El Chaperno	3.91×10^{5}	8.75×10^{7}	1.25×10^{10}	1.25×10^{12}
El Carrizal	3.41×10^{5}	7.07×10^{7}	9.45×10^{9}	8.90×10^{11}
Combined	6.00×10^{5}	1.60×10^{8}	2.70×10^{10}	3.13×10^{12}

3.3.3 Simulated SNP Disease Dataset

To test our ability to identify significant features and interactions with our novel algorithm, we first tested the algorithm on a set of benchmark problems used in the CS community (Hanley et al., *In Review*); and in this work, further test the algorithm on a simulated single nucleotide polymorphism (SNP) dataset. The simulated dataset was designed based on a need for better tools for analyzing complex diseases (Thornton-Wells et al., 2004), where, in general, benchmark datasets are lacking (specifically those that contain gene interactions, heterogeneity, and missing data). Urbanowicz and Moore (2010) present one of the few synthetic datasets designed with both heterogeneity and feature interactions. The dataset was designed to represent a single nucleotide polymorphism (SNP) gene association for a complex disease. It is a balanced dataset (half of the observations are associated with the “disease” and half are not) with 1,600 observations, each with 20 features or SNPS. Each SNP is a ternary representation of homozygous major, heterozygous, or homozygous minor. The dataset was designed such that no individual feature had a significant main effect, and there are four, two-way interactions that comprise the four true signals (i.e., the known mapping between the

input features and the associated outcome disease class) (Table 3.2). While each true signal covers 21-28% of the diseased individuals, the four signals combined cover 76% of the diseased individuals. Thus, multiple true signals cover the same individual, which one might expect in a real-world dataset.

Table 3.2: Accuracy, coverage, and hypergeometric PMF fitness (last 3 columns) associated with the four true signals of the Urbanowicz and Moore (2010) benchmark SNP disease dataset. The dataset is balanced – half of 1,600 input feature vectors are associated with disease; half are not.

True Signals	X_0	X_1	X_2	X_3	Accuracy	Class Coverage	Fitness
1	0	1	–	–	72%	27%	1.1×10^{-17}
2	1	0	–	–	74%	23%	5.7×10^{-17}
3	–	–	0	1	66%	28%	4.2×10^{-12}
4	–	–	1	0	71%	21%	8.7×10^{-13}

3.3.4 Conjunctive Clause Evolutionary Algorithm (CCEA)

We designed a conjunctive clause evolutionary algorithm (CCEA) to efficiently search for multivariate interactions across multiple data types (i.e., binary, nominal, ordinal, continuous) in survey datasets with $k = 1, 2, \ldots K$ outcomes. The details of the algorithm have been presented in Hanley et al. (2016; *In Review*). Briefly, the CCEA is a non-parametric statistical tool that searches across the entire range of multivariate feature interactions. Each feature represents a survey response that varies in data type and range of values. The CCEA evolves feature sets as well as *the range of feature values* using conjunctive clauses in the following form:

$$F_i \in a_i \wedge F_j \in a_j \ldots \wedge F_L \in a_L \quad , \tag{3.1}$$

where each F_i represents a feature, i, that may be nominal, ordinal, or continuous, and whose value lies in a_i, a specified range or set of values. The number of features in a conjunctive clause can vary between one and the total number of features, L, in the

dataset. The only inherent model assumption is that ordinal and continuous features can only evolve monotonic or unimodal ranges. The conjunctive clauses are stored in different populations following an age-layered population structure (ALPS) similar to that created by (Hornby, 2006). The age-layer population structure helps bias or protect newer (more recently evolved) conjunctive clauses compared to older conjunctive clauses. Unlike ALPS, the CCEA version of ALPS has an archived age layer that consists of probabilistically significant conjunctive clauses (Hanley et al., 2016). To help the CCEA detect the possibility of multiple optima and thus find a global optimum, underrepresented observations in the population of conjunctive clauses are preserved by biasing their selection in subsequent generations. Preserving diversity ensures the CCEA explores a larger decision space and safeguards against being trapped in local optima.

To determine whether a conjunctive clause is probabilistically significant, the CCEA estimates the "fitness" of a conjunctive clause using the hypergeometric probability mass function (PMF) (Kendall, 1952). Eq. (3.2) quantifies the likelihood that the observed association between the conjunctive clause and the target class is due to chance; thus, lower values of this fitness function (i.e., lower *p*-values) are indicative of potential association. If the hypergeometric PMF of a conjunctive clause is less than or equal to a user-defined threshold, it is considered probabilistically significant and worthy of being archived. For conjunctive clauses evolved in the CCEA, the hypergeometric PMF is defined as follows:

$$Hypergeometric\ PMF = \frac{\binom{X_{tot}}{x_{match}}\binom{N_{tot}-X_{tot}}{n_{match}-x_{match}}}{\binom{N_{tot}}{n_{match}}} \quad , \qquad (3.2)$$

where,

N_{tot} = the total number of observations in the dataset,

X_{tot} = the total number of observations associated with a desired target class, k,

n_{match} = the total number of sampled observations whose features match a given conjunctive clause, and

x_{match} = the number of sampled observations that match the conjunctive clause and are in target class k.

It should be noted that a novel feature of the EA is the implementation of the hypergeometric PMF and ability to handle missing data. More detailed definitions for N_{tot} and X_{tot} as the number of observations with non-missing values for features present in the conjunctive clause are provided in Hanley et al., (*In Review*). Traditionally, features with lots of missing data are less likely to form probabilistically significant multivariate conjunctive clauses; however, for a dataset where this is not true, significant features with a lot of missing data can be detected.

The CCEA can have a static threshold (i.e., the threshold will not heuristically decrease), or the threshold can deterministically evolve based on the number of archived conjunctive clauses for a given conjunctive clause order. In this work, we use a static threshold: we archive conjunctive clauses that cover at least 10% or more of the houses infested with *T. dimidiata* by setting the hypergeometric fitness threshold to the fitness of a conjunctive clause that has 100% accuracy and 10% coverage of infested houses. Accuracy is defined as $\frac{x_{match}}{n_{match}}$ and is analogous to the true positive rate of the conjunctive clause. Infested house coverage is the number of times a sampled conjunctive clause is associated with a target outcome over the total number of target outcomes in the dataset,

$\frac{x_{match}}{X_{tot}}$. If only a few conjunctive clauses are archived, we risk that the archived signals contain large amounts of noise and are subject to overfitting. As mentioned above, the CCEA used a static threshold to maintain a large population of archived conjunctive clauses. This is consistent with the concept in "Big Data" that more data can be used to find patterns of correlations with a desired output (true signal) (Mayer-Schönberger and Cukier, 2014).

In addition to setting a fitness threshold, there are a number of other input parameters (see Table 3.3) that typically need to be initialized when using an EA. The only parameter that is not typical is the number of archived offspring (Off_A); the latter keeps the number of archived and non-archived offspring balanced; Off_A caps the number of offspring evolved from the archived population every generation. For the portion of the population that is not archived, each conjunctive clause undergoes either crossover or mutation each generation.

Table 3.3: Conjunctive Clause Evolutionary Algorithm (CCEA) parameter settings.

CCEA Parameters	Value
Total # of Features (L)	Dataset dependent
Threshold (T)	Dataset dependent
# Non-Archived Age Layers (AL_{NA})	5
Novel Population (Pop_N)	2 x L
Non-Archive Pop. (Pop_{NA})	Pop_N
# Archive Offspring (Off_A)	AL_{NA} x Pop_N
Generations (Gen)	200
Generations Until Novel Pop (Gen_N)	5
Crossover Function (F_X)	Uniform
Probability of Crossover (Pr_X)	0.50
Probability of Wild Card (Pr_{WC})	0.75
Mutation Function (F_M)	{uniform, $p_m = 1/L$}
Crossover Mate Selection	{tournament, size = 3}

The CCEA was run on four test cases (i.e., the benchmark SNP disease dataset of Urbanowicz and Moore, (2010), the dataset of El Chaperno, El Carrizal, and the two towns combined). Each test case of 200 generations had 5 randomly seeded repetitions to decrease the likelihood of the algorithm becoming trapped in a population of local optima. For each dataset, we calculated the accuracy and infested house coverage for every archived conjunctive clause.

3.3.5 Feature and Feature Pair Importance (*FI* and *FPI*)

The feature importance (*FI*) and feature pair importance (*FPI*) are calculated using only those observed conjunctive clauses that match the target outcome (e.g., diseased individual, infested house). For each observation matching the target output, the *FI* is sum total of features in matching conjunctive clause; and the summed *FIs* are normalized across all features associated with a given target outcome. The latter ensures that the smallest feature sum total is 0 and the largest is 1 (Table 3.S2.A). The *FPI* is

similar to the *FI* except it is the normalized number of times a pair of features are present in conjunctive clauses that match the target observation. For a given observation, any features with missing values, the *FI* and *FPIs* are designated as null values (i.e., a lack of a value).

The *FI* and *FPI* values may be viewed as networks or as heat maps similar to the way Urbanowicz et al. (2012a; 2013) displayed feature interaction metrics. Each node of the network represents a feature, and each edge represents a feature pair with the size of the nodes and thickness of the edges proportional to the *FI* and *FPI*. We sort the *FI* and *FPI* values independently by individual features as well as all paired features, respectively. Because we are most interested in *FI* and *FPI* values that represent at least 10% (user-defined threshold) of the target outcome (e.g., diseased individuals or infested houses), we selected the 90th percentile *FI* and *FPI* values across each individual feature vector and feature-pair vector (Table 3.S2.B). We use the 90th percentile as a conservative user-defined threshold to account for unbalanced heterogeneity; however, this threshold is problem dependent. The Gephi 0.9.1 software (Bastian et al., 2009) is used to visualize the feature network.

3.3.6 Feature Sensitivity

For every archived conjunctive clause, we remove each feature one by one to determine feature sensitivity. Feature sensitivity was designed to be the difference between the $\log_{10}$ of the new fitness value (hypergeometric PMF associated with the new conjunctive clause, i.e., one with a feature removed) and the $\log_{10}$ of the original hypergeometric PMF:

$$Feat.Sens. = log(New\ Hypergeo\ PMF) - log(Original\ Hypergeo\ PMF)\ , \quad (3.3)$$

Because the hypergeometric PMF $\in [0,1]$, taking the $\log_{10}$ will result in values that are ≤ 0. Therefore, positive feature sensitivities of Eq. (3.3) will be associated with features that are important to the conjunctive clause to which they belong; and negative feature sensitivities will be associated with features that add noise. We should note that each unit of feature sensitivity represents an order of magnitude change in the hypergeometric PMF (fitness of the conjunctive clause) and that the scale of the feature sensitivity is relative to the size of the dataset. Thus, direct comparison across datasets is not possible (e.g., feature sensitivity will likely be smaller for El Carrizal compared to El Chaperno; and both towns individually will be smaller than when the data are combined). To help with comparison, we plot the median values of the feature sensitivity for each dataset.

3.4 Conjunctive Clause Evolutionary Algorithm (CCEA) Results

3.4.1 Results of the Simulated SNP Disease Dataset containing all 20 Features

Each single nucleotide polymorphism (SNP) is regarded as a feature, defined X0 – X19. When Hanley et al. (2016) performed exhaustive search on the benchmark dataset of Urbanowicz and Moore (2010), the best fitness was four orders of magnitude less fit than that of a conjunctive clause with 100% accuracy and 10% coverage. As a result, we lowered the criteria of the hypergeometric PMF for a conjunctive clause with 100% accuracy and 1% coverage of the diseased individuals. The accuracy, class coverage, and hypergeometric PMF (contour lines spaced at 10^{-4} intervals) for the conjunctive clauses archived by the CCEA (Figure 3.2) show that most of the conjunctive clauses have low

coverage of the diseased individuals. Solutions (2nd- through higher-order conjunctive clauses) along hypergeometric contours closer to the upper right-hand corner of Figure 3.2 have lower fitness and therefore, are considered more optimal. The CCEA was successful in archiving all four of the true (two-way) signals of Table 3.2 (boxed red circles). These 2nd-order CCs have 20-30% coverage of the diseased individuals and 65-75% accuracy. The algorithm evaluated ~3.74×10^5 of the ~1.10×10^{12} possible conjunctive clauses in the dataset. It should also be noted that the four main effects archived by the CCEA (black circles) have lower fitness; and only one of which (feature X1: accuracy = 54%, diseased individual coverage = 36%) belongs to two of the four true (two-way) signals.

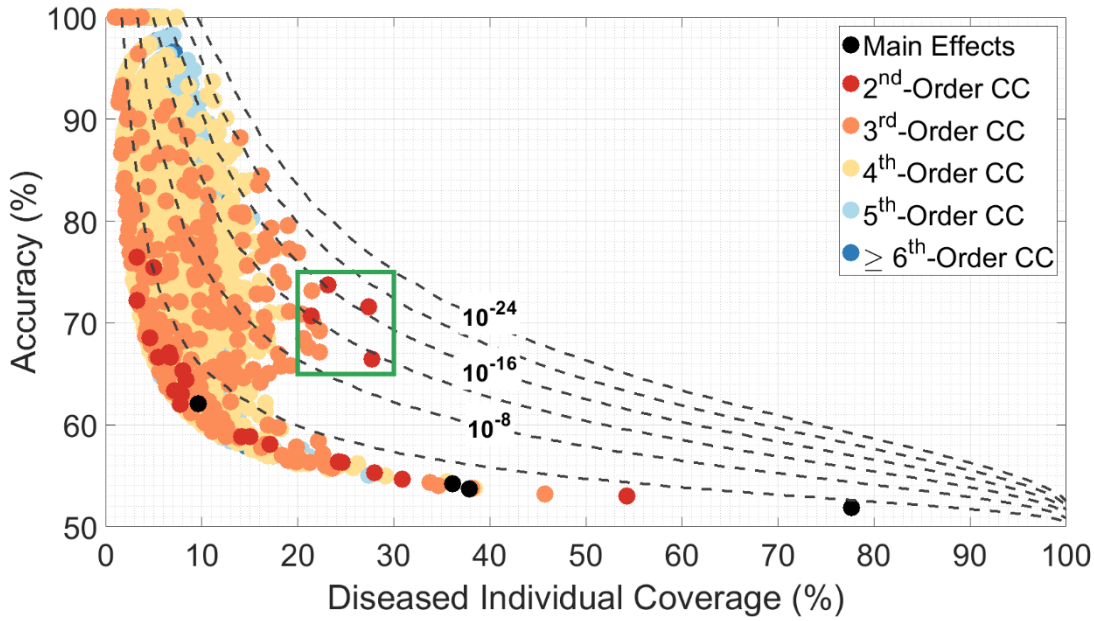


Figure 3.2: Accuracy, class coverage, and hypergeometric PMF (contour lines spaced at 10^{-4} intervals) for the conjunctive clauses identified using the CCEA for the simulated SNP disease dataset. Each color-coded circle represents the order of a conjunctive clause. The green box shows the location of the four true signals (i.e., 2nd-order CCs in red).

Without filtering for feature pairs (2-way interactions), the network representation of Figure 3.3 (panel A) shows the first four features (X0 – X3) and X13 to have the highest *FI* values. Once the filter ($FPI \geq 0.95$) is applied the network (panel B) identifies the four true signals (two-way, feature pairs X0 & X1 and X2 & X3 of Table 3.3) to be the most important.

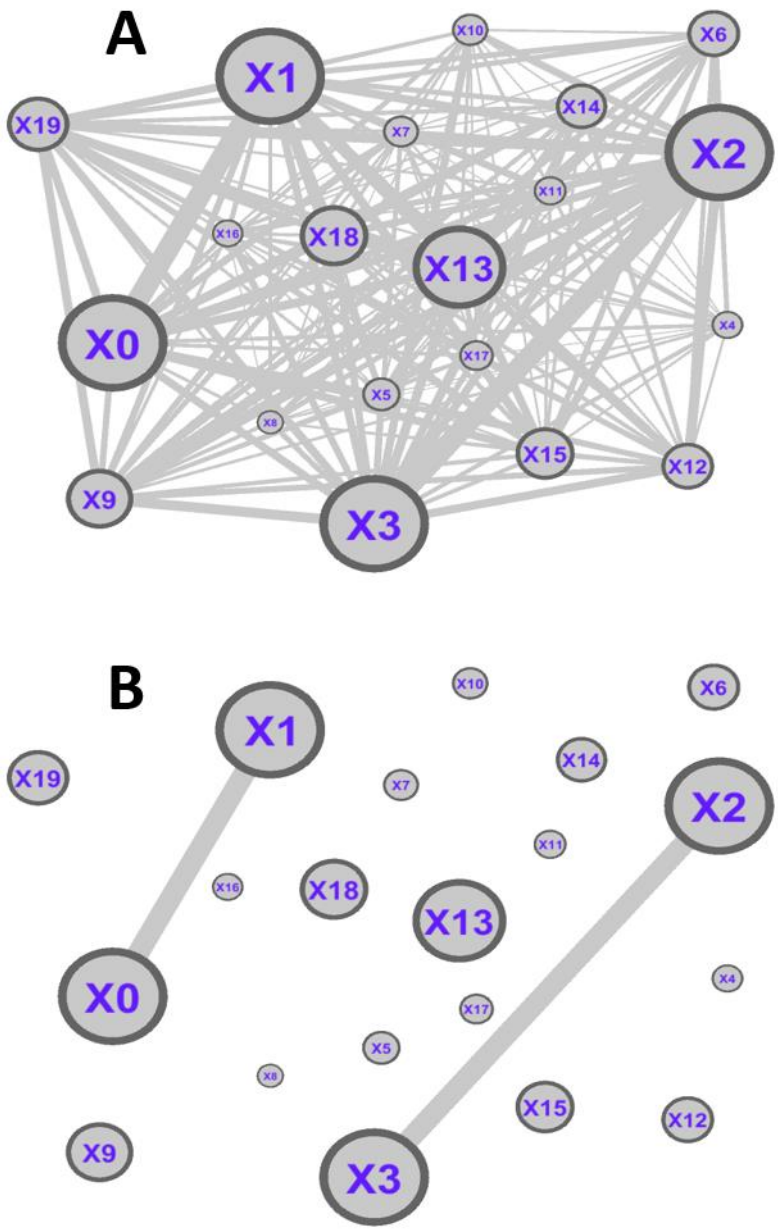


Figure 3.3: Feature importance (*FI*) and feature-pair importance (*FPI*) are represented as a network. The nodes and edges are proportional to the *FI* and *FPI* values, respectively. Panel A) Contains all feature-pair connections and B) is filtered so that only *FPI* ≥ 0.95 are visible.

The median feature sensitivities are represented in the bar chart of Figure 3.4. Only features X0 – X3 have positive median values indicating that removal of these

features from conjunctive clauses would change the probability of the hypergeometric PMF by at least two-orders of magnitude. Taken together, the feature importance (Figure 3.3) and feature sensitivity (Figure 3.4) indicate that the four features (X0 – X3) that comprise the true signals are the most important features in the simulated SNP disease dataset. As a result, the dataset could be reduced to these four features (X0 – X3).

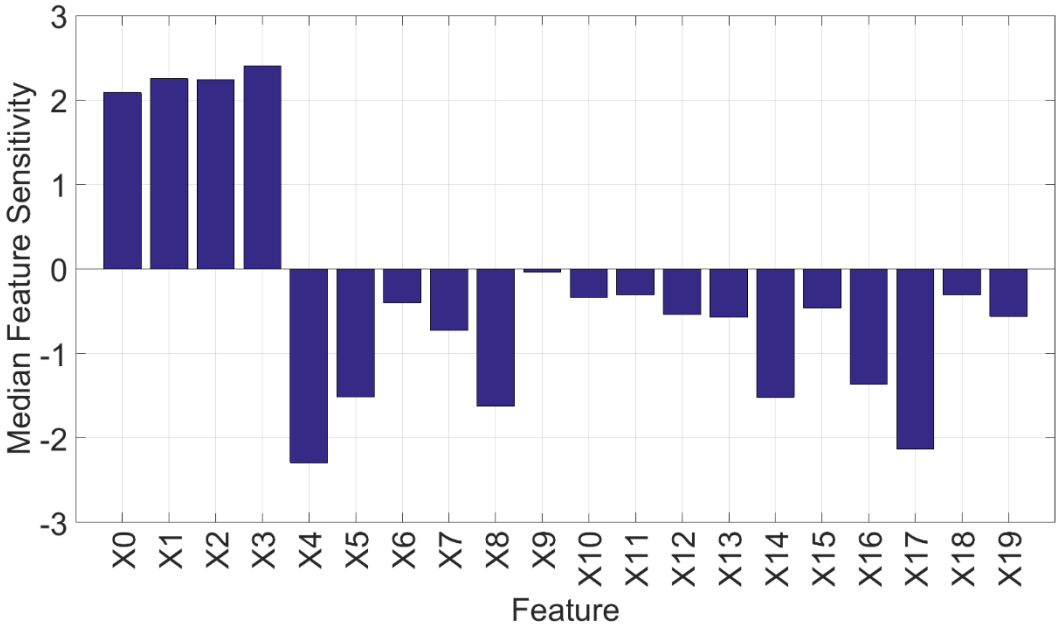


Figure 3.4: Bar graph showing median feature sensitivity for each of the 20 features. Positive bars indicate that the removal of a feature from an archived conjunctive clause decreases the fitness (i.e., the hypergeometric PMF increases and the conjunctive clause becomes more likely due to chance).

Finally, the feature sensitivity for each diseased individual may be viewed as a heat map (Figure 3.5), where the median sensitivity of features is either positive (red), neutral (black), or negative (blue). White indicates that the feature was not present in any conjunctive clause that matched the outcome of interest. The results are re-ordered for visualization purposes and help show the heterogeneity embedded in the dataset (no

individual feature has positive median feature sensitivity across all 800 diseased individuals). Figure 3.5 also shows that the median sensitivity associated with the four features that comprise the true signals is positive for nearly all of the 800 diseased individuals. Similarly, the sensitivity for features X0 & X1 is positive for ~500 of the diseased individuals and the sensitivity of features X2 & X3 is positive for ~500 of the diseased individuals.

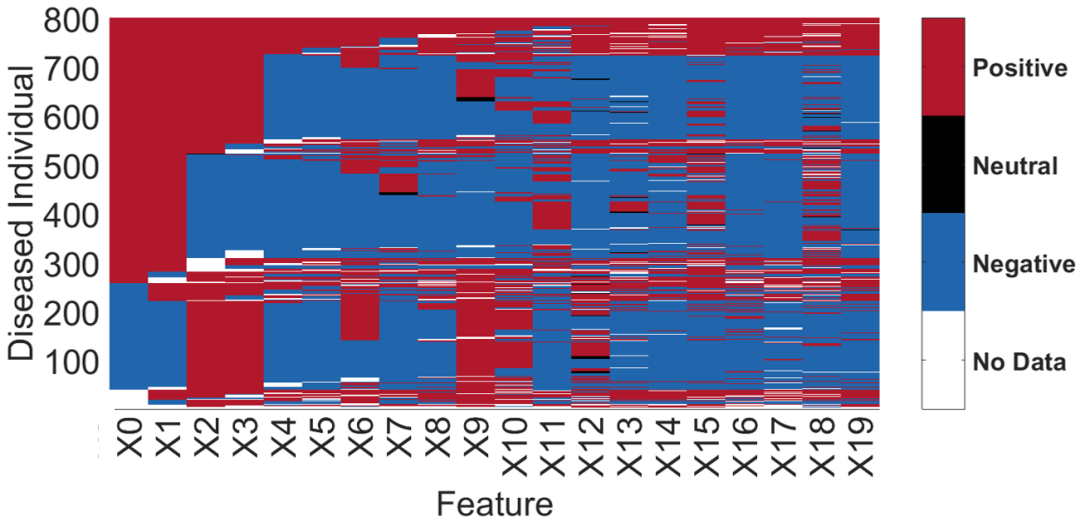


Figure 3.5: Median feature sensitivity associated with an outcome of interest (e.g. diseased individuals) are re-ordered for visualization purposes. The median feature sensitivity across all conjunctive clauses may be positive (red), zero (black), and negative (blue), respectively. White indicates when a feature was not present in any conjunctive clause that matched the outcome of interest.

3.4.1.1 Results of the Simulated SNP Disease Dataset Using Only the Reduced 4-Features

When the simulated SNP disease dataset is reduced to four features, an exhaustive search of the reduced dataset would be trivial. However, because our interest is in exploring how well our feature reduction strategy might work on real-world datasets, we

analyzed all of the CCEA conjunctive clauses that contained features X0 – X3 and had a hypergeometric PMF below the user-defined threshold (n=26). Note: All 26 conjunctive clauses were archived by the CCEA.

Feature importance performed on the reduced (four-feature) dataset was nearly identical to the network of Figure 3.3 (using all 20 features) and as result, the network is not presented here. The median feature sensitivity of the reduced features set is now greater than five indicating there is a five-order magnitude change in the hypergeometric PMF when each of the four features are individually removed (Figure 3.6). The heat map of Figure 3.7 shows the median feature sensitivity and the overlap between the four features that comprise the true signals. The heat map also helps visualize both the heterogeneity and feature interactions embedded in the dataset. Heterogeneous features X0 & X1 are positive for ~400 of the diseased individuals; and features X2 & X3 are positive for ~400 diseased individuals. There are ~200 diseased individuals that are not associated with features that comprise the true signals, showing the noise present in this dataset. These results suggest that without noise, it's even easier to ID important features using the *FI*, *FPI*, and feature sensitivity.

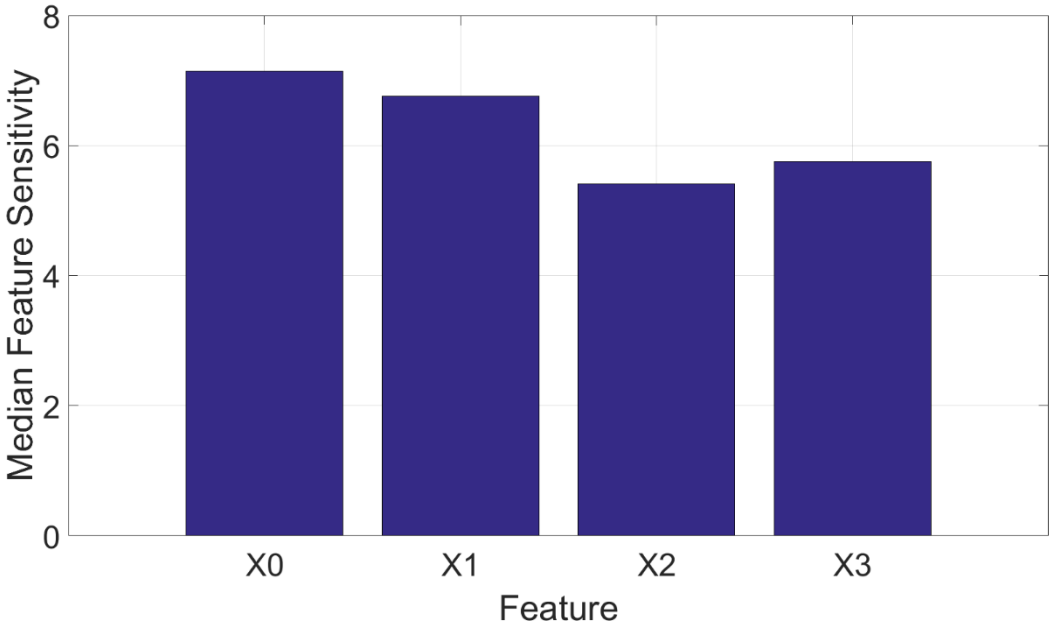


Figure 3.6: Bar graph showing median feature sensitivity for each of the four features that comprise the true signals. Positive bars indicate that removal of the feature from an archived conjunctive clause decreases the fitness (i.e., the hypergeometric PMF increases and the conjunctive clause becomes more likely due to chance).

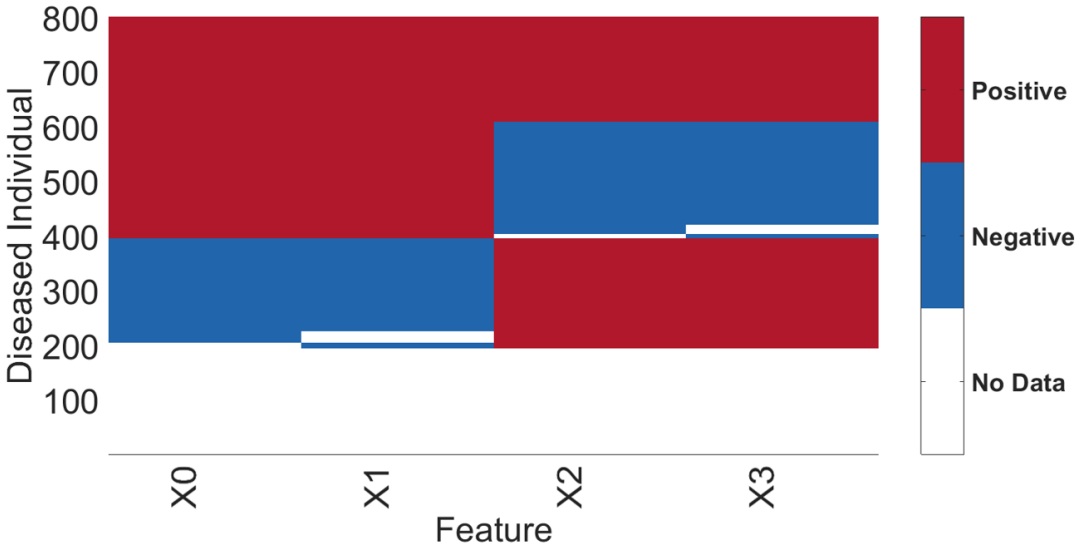


Figure 3.7: Bar graph showing median feature sensitivity for each of the four features that comprise the true signals. Positive bars indicate that removal of the feature from an archived conjunctive clause decreases the fitness (i.e., the hypergeometric PMF increases and the conjunctive clause becomes more likely due to chance).

3.4.2 Results on El Chaperno, El Carrizal, and the Combined Datasets Using all 64 Features

The summary statistics for El Carrizal and El Chaperno (Table 3.4) show that El Carrizal has a higher percentage of infested houses than El Chaperno; and that both datasets have imbalanced outputs with the percentage of infested houses being in the minority (Table 3.4, column 3).

Table 3.4: Summary characteristics for El Chaperno, El Carrizal, and the two towns combined.

Dataset	# Houses	# Infested Houses	# Ordinal, Nominal, Binary Features	% Missing Data	Median % Missing Data per Feature	[Min, Max] % Missing Data per Feature
El Chaperno	182	49 (26.9%)	[12, 8, 44]	28.9	15.7	[0.5, 86.8]
El Carrizal	129	51 (39.5%)	[14, 8, 42]	22.3	3.9	[0.8, 77.5]
Combined	311	100 (32.2%)	[14, 8, 42]	26.1	10.3	[1.2, 78.5]

The conjunctive clauses identified by the CCEA show higher class coverage in the real Chagas survey dataset (Figure 3.8) compared to the simulated SNP disease dataset of Figure 3.2. Given the imbalanced nature of the real datasets, there are conjunctive clauses with accuracy <50%. The CCEA results in a plethora of archived conjunctive clauses that may contain the true drivers of *T. dimidiata* infestation. Again, solutions closer to the upper right-hand corner of the graphs in panels A-C of Figure 3.8 have lower fitness (higher association with infestation). In addition, interesting patterns emerge when comparing the distribution of conjunctive clauses across the three Chagas datasets. The higher percentage of infested houses in El Carrizal (panel B) helps push the distribution of archived conjunctive clauses toward higher accuracy compared to El Chaperno or the combined dataset (panels A and C, respectively).

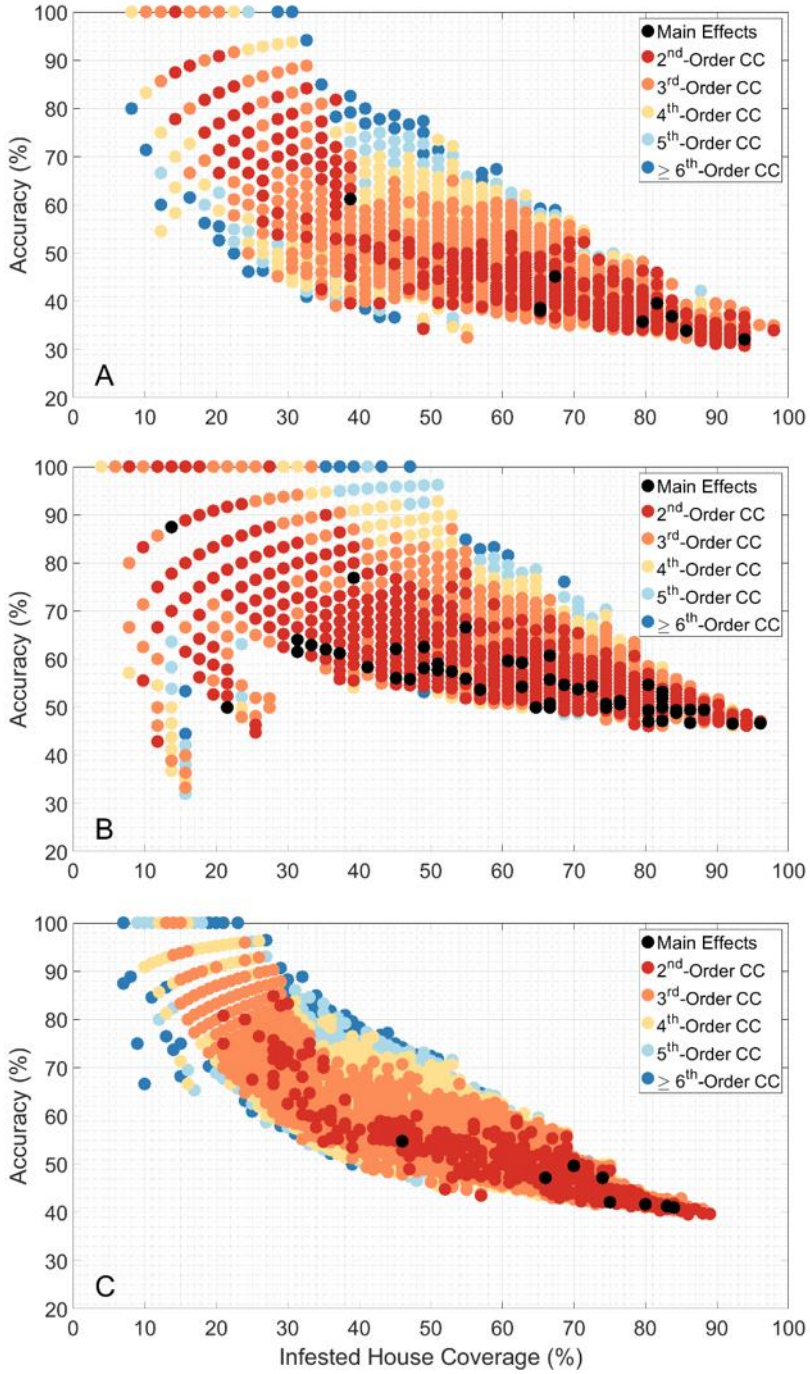


Figure 3.8: The accuracy and infested house coverage of the conjunctive clauses identified using the CCEA for the A) El Chaperno, B) El Carrizal, and C) combined datasets. Each color-coded circle represents the order of a conjunctive clause.

Following the feature reduction strategy used with the simulated SNP disease dataset, we analyzed the *FI* and *FPI* as well as the feature sensitivity for the El Chaperno, El Carrizal, and combined datasets, respectively. Figure 3.9 provides a network representation of the *FI* and *FPI* for each dataset separately; features 9, 10, and 29 appear in all three networks. Features 9 and 10 are the age of the house and the years lived in the house, respectively; while feature 29 provides information on whether there is an accumulation of objects (potential hiding places for bugs) in the house. While features 9 and 10 may appear similar, the survey responses are not always correlated; thus, both features merit future exploration in a reduced dataset. Features 53 and 55 (house wall material and house wall condition) are present in the networks for El Chaperno and El Carrizal (Figure 3.9A & 3.9B), but are not present in the network of the combined dataset (Figure 3.9C). If the user is willing to relax the filter on the importance threshold to *FPI* ≥ 0.85, then feature 54 (condition of the bedroom walls) would have been common to all three networks, stressing the importance of using feature identification tools in concert with domain experts.

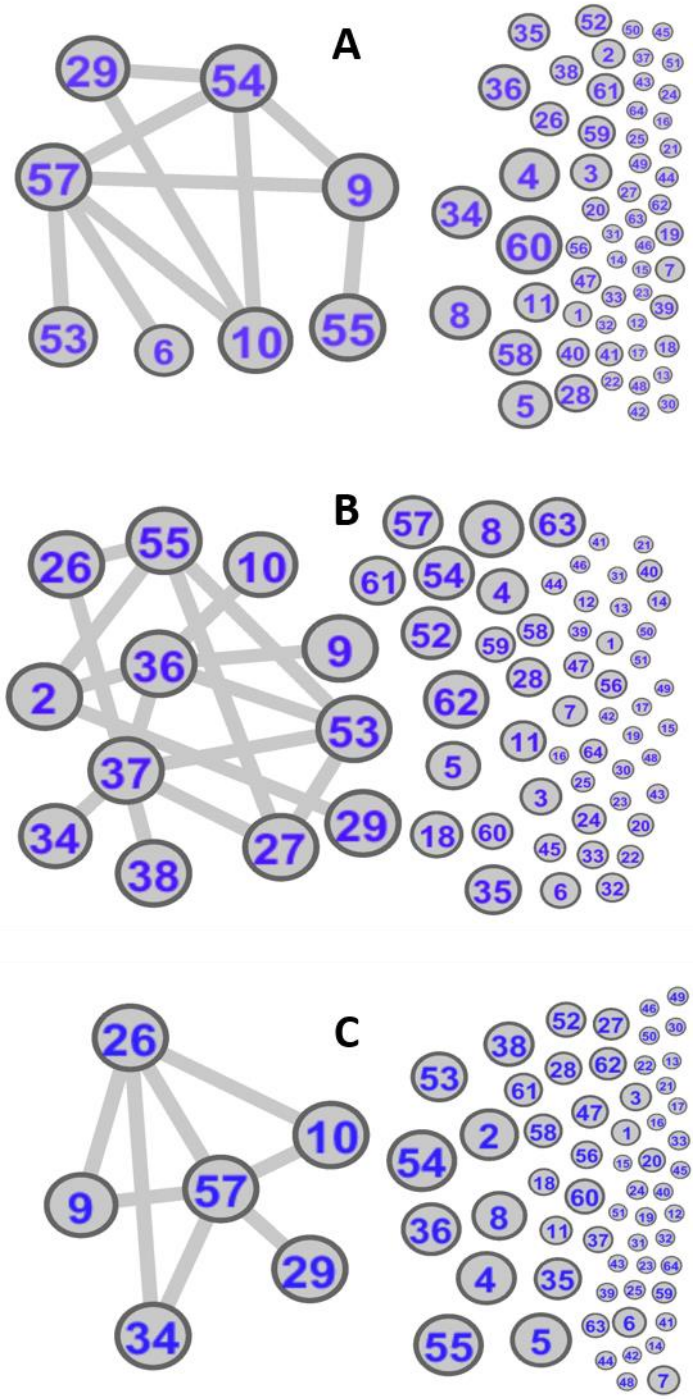


Figure 3.9: The feature and feature pair importance are represented as a network. The nodes and edges are sized based on the *FI* and *FPI*, respectively. The networks are filtered so that only *FPI* ≥ 0.95 are visible for the A) El Chaperno, B) El Carrizal, and C) combined datasets.

The median feature sensitivities for the El Chaperno, El Carrizal, and combined datasets vary considerably (Figure 3.10). However, features 9, 10, and 34 have feature sensitivity values greater than zero for all three datasets. Again, features 9 and 10 are the age of the house and the years lived in the house, respectively; while feature 34 is the hygienic condition of the beds. Interestingly, both the minimum and maximum feature sensitivity (Feature 16: Type of house improvement is other has a feature sensitivity = -5.0; Feature 17: How often are the walls plastered? (feature sensitivity = 2.4)) are found in the combined dataset (Figure 3.10C). The latter is likely a result of the inverse relationship that exists between probabilities (decreasing p-values) and increasing dataset size and highlights the danger of relying on p-value criteria for datasets with large n values. See Lin et al., (2013) for more detail.

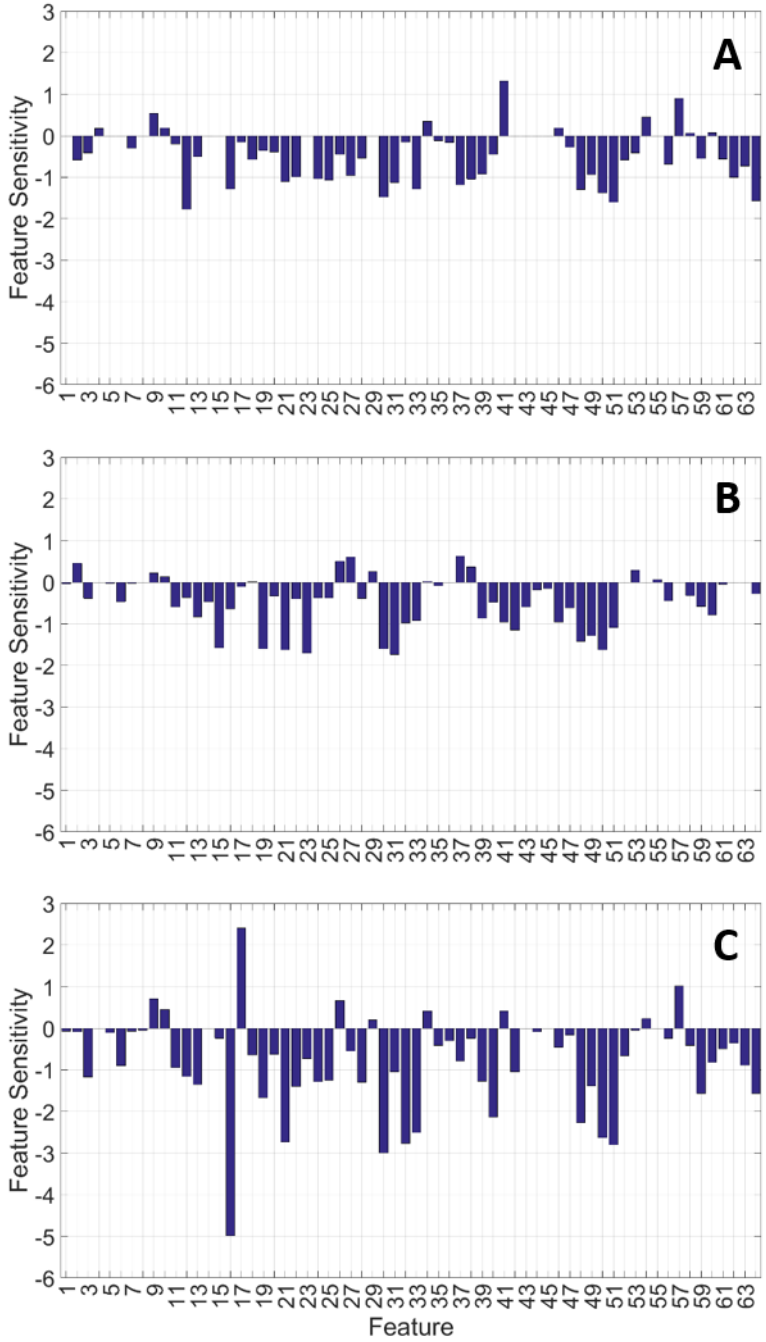


Figure 3.10: Bar graphs showing median feature sensitivities for each of the 64 features for the A) El Chaperno, B) El Carrizal, and C) combined datasets. Positive values indicate that removal of a feature from an archived conjunctive clause decreases the fitness (i.e., the hypergeometric PMF increases and the conjunctive clause becomes more likely due to chance).

The union of all features (i.e., binary, nominal, and ordinal) meeting the selection criteria of the previous SNP disease dataset (i.e., filtered network feature-pair importance and median feature sensitivities > 0) resulted in reducing the 64-feature dataset to the 22 in Table 3.5.

Table 3.5: Features selected using feature-pair importance and feature sensitivity for the El Chaperno, El Carrizal, and combined datasets.

ID	Survey Question	*FPI* ≥ 0.95			Median Feature Sensitivity > 0		
		Chaperno	Carrizal	Combined	Chaperno	Carrizal	Combined
2	Source of income is day laborer		**X**			**X**	
4	Source of income is business				**X**		
6	Source of income is other	**X**					
9	Age of the house	**X**	**X**	**X**	**X**	**X**	**X**
10	Years lived in the house	**X**	**X**	**X**	**X**	**X**	**X**
17	How often are the walls plastered?						**X**
18	Number of dogs					**X**	
26	Presence/signs of animals in the house		**X**	**X**		**X**	**X**
27	Presence/signs of bird nests in the house		**X**			**X**	
29	Accumulation of objects in the house	**X**	**X**	**X**		**X**	**X**
34	Hygienic condition of the beds		**X**	**X**	**X**	**X**	**X**
36	Hygienic condition of the house		**X**				
37	Are grains stored in the main room?		**X**			**X**	
38	Accumulation of firewood		**X**			**X**	
41	Type of accumulated construction material is adobe				**X**		**X**
46	Location of construction materials				**X**		
53	Primary house wall material	**X**	**X**			**X**	
54	Condition of the bedroom walls	**X**			**X**		**X**
55	Condition of walls in rest of house	**X**	**X**			**X**	
57	Primary house floor material	**X**		**X**	**X**		**X**
58	Is the main room dark?				**X**		
60	Does the main room have windows?				**X**		

3.4.2.1 CCEA Results for the Chagas Datasets Using the Reduced 22 Features

The simulated SNP disease results suggests that feature reduction helps shrink the search space and elucidates drivers of a system. As a result, we perform similar analysis on the three house infestation datasets. The accuracies and infested house coverages for the archived CCEA conjunctive clauses are shown (Figure 3.11) for El Chaperno, El Carrizal, and the combined dataset in panels A, B and C, respectively. Interestingly, the range of accuracies and coverages that evolve when using all 64 features (Figure 3.8) is nearly the same for the reduced set of 22 features (Figure 3.11). However, these 22 feature datasets contain less 2nd- and 3rd-order conjunctive clauses with accuracy greater than 70%.

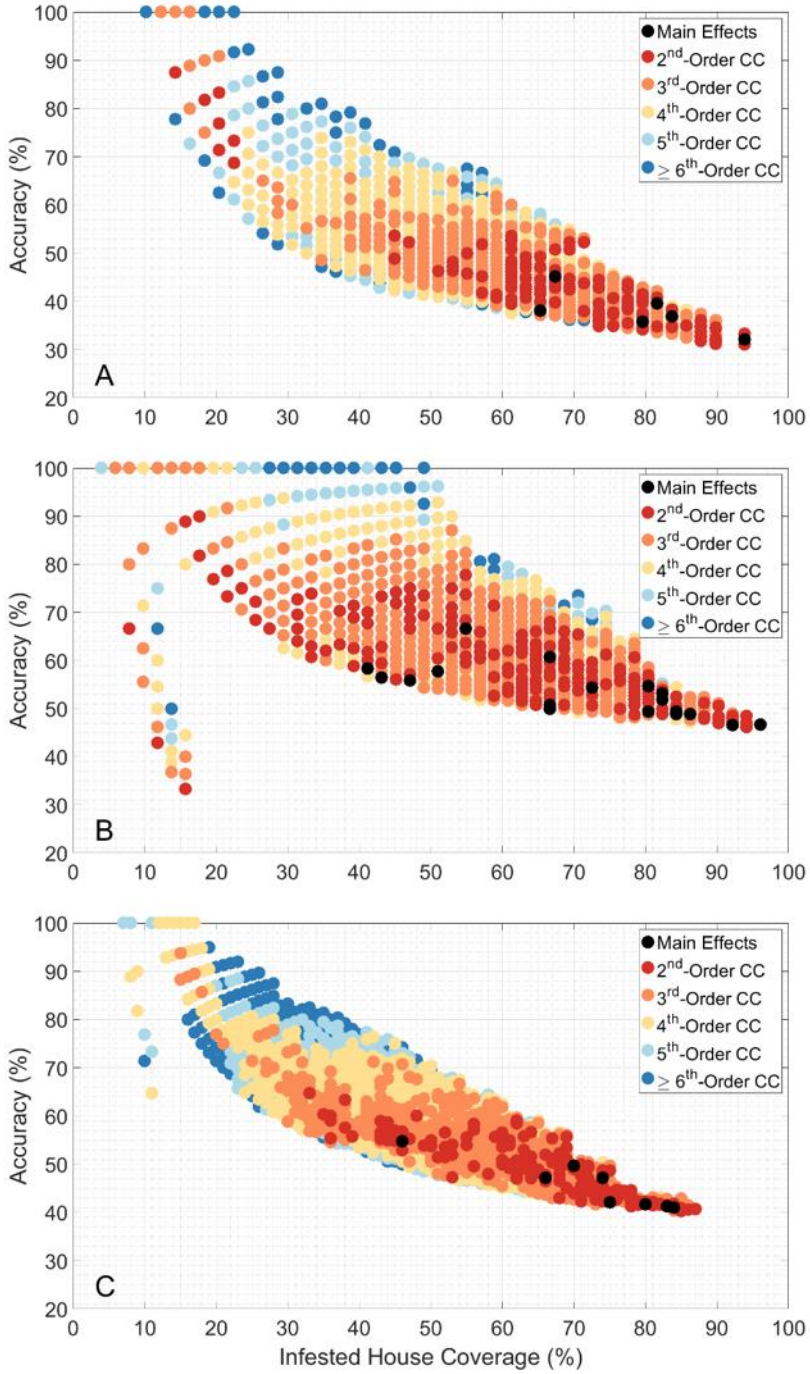


Figure 3.11: The accuracy and infested house coverage of the conjunctive clauses identified using the CCEA on the reduced 22 features for the A) El Chaperno, B) El Carrizal, and C) combined datasets. Each color-coded circle represents the order of a conjunctive clause.

The network representation of features (*FPI* ≥ 0.95) in the El Chaperno and El Carrizal (Figure 3.12A & 3.12B) are a subset of feature pairs identified when all 64 features are used (Figure 3.9A & 3.9B). Figure 3.12C has three interconnected features (9 = age of the house, 53 = primary wall material, and 57 = the primary floor material) with 68% coverage and 56% accuracy implying that this 3rd-order conjunctive clause is important.

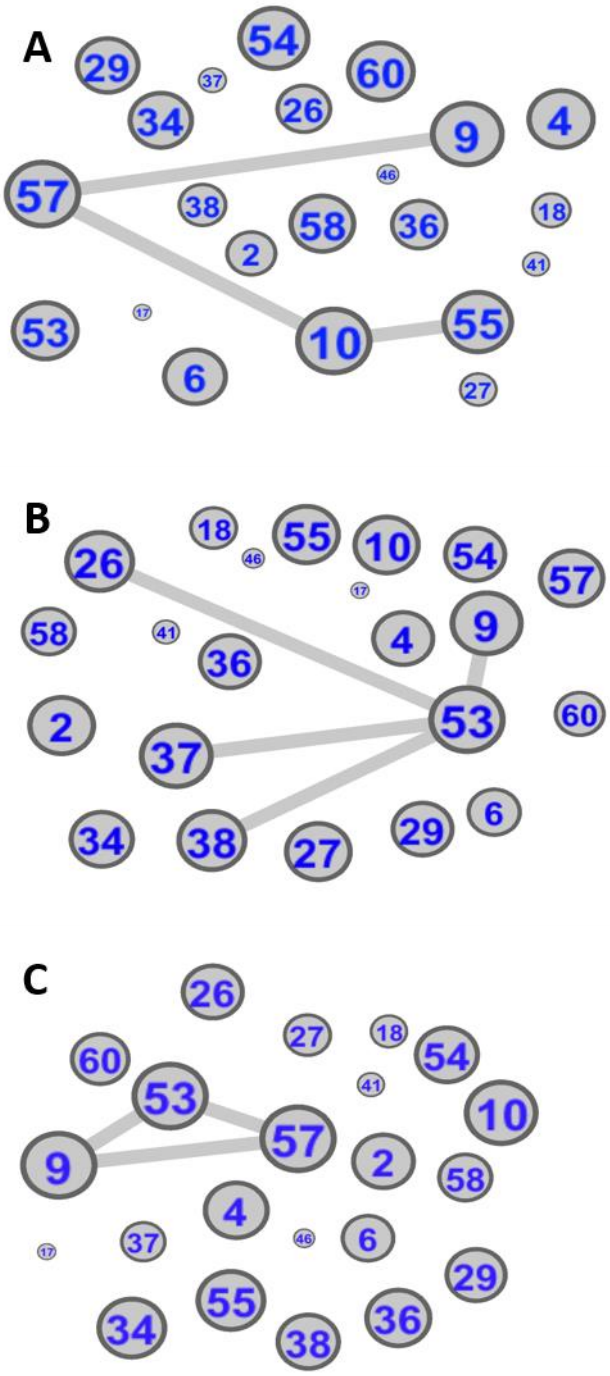


Figure 3.12: The network representation of *FI* and filtered *FPI* ≥ 0.95 for the reduced 22-features (Table 3.5) for the A) El Chaperno, B) El Carrizal, and C) combined datasets. Nodes and edges are proportional to the *FI* and *FPI*, respectively.

The range of maximum and minimum median feature sensitivity is smaller for the reduced 22-features compared to the 64-feature sensitivity (Figure 3.13). For each Chagas dataset, only a minority of the features have positive median feature sensitivity. Features 17, 18, 29, 36, 58, and 60 never have positive median feature sensitivity (Figure 3.13) and are not present in any of the filtered networks of Figure 3.12. Therefore, if further feature reduction was desired, the reduced 22 feature set could be further reduced to 16 features.

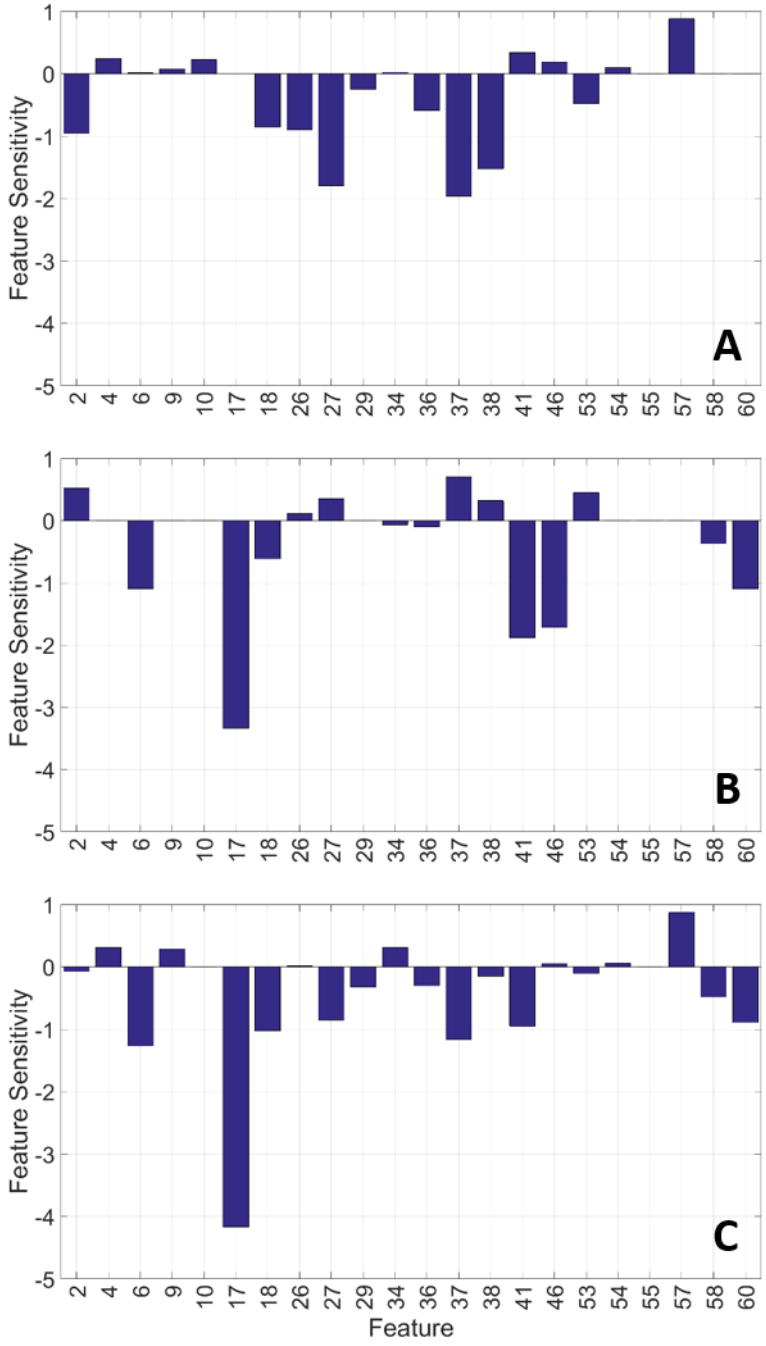


Figure 3.13: Bar graphs showing median feature sensitivities for each of the reduced 22 features for the A) El Chaperno, B) El Carrizal, and C) combined datasets. Positive values indicate that removal of the feature from an archived conjunctive clause decreases the fitness (i.e., the hypergeometric PMF increases and the conjunctive clause becomes more likely due to chance).

3.4.3 Example of Conjunctive Clauses Archived by the CCEA

To better analyze how the feature sensitivities associated with the archived conjunctive clauses might be used in practice, we selected three conjunctive clauses along the Pareto front (Figure 3.14) for the combined (two-town, 64 feature) Chagas.

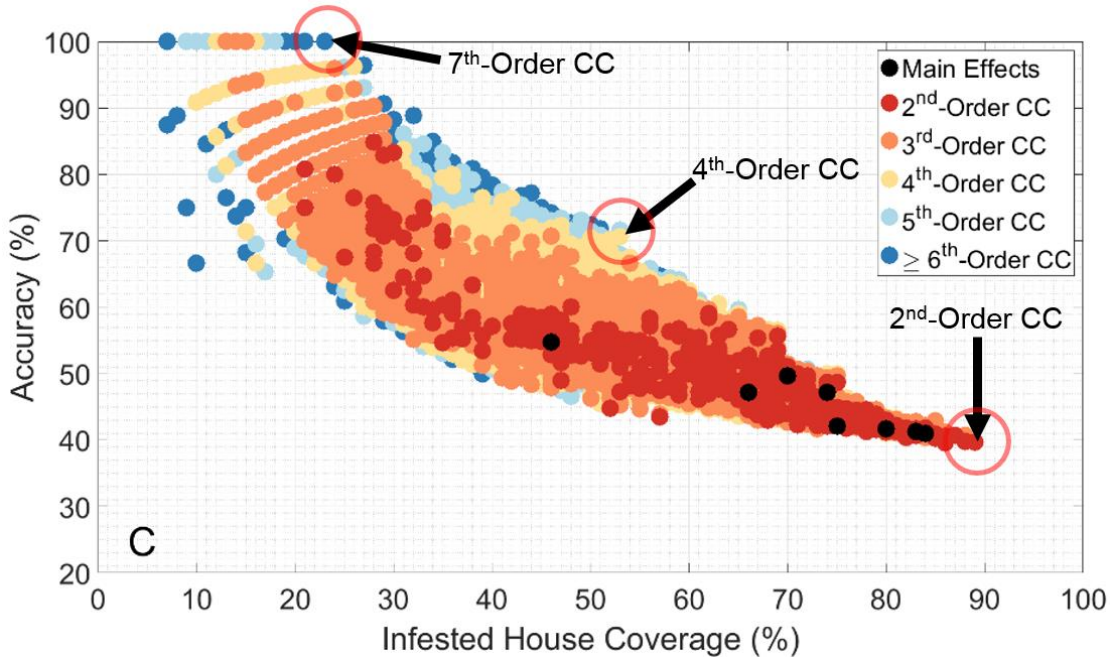
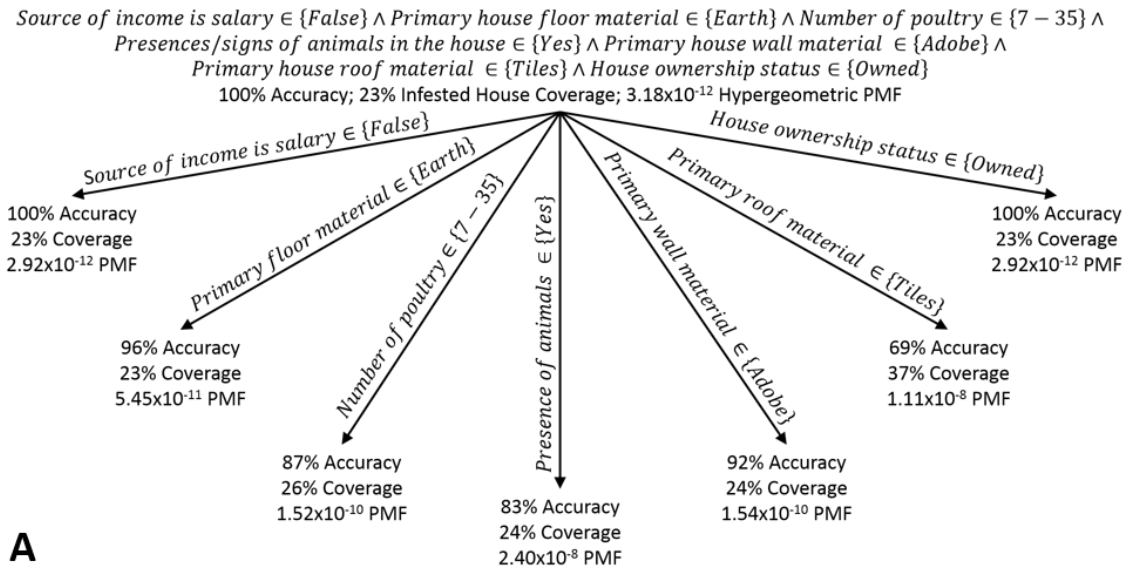


Figure 3.14: The accuracy and infested house coverage of the conjunctive clauses identified using the CCEA and all 64 features for the combined datasets. Three CCs selected along the Pareto front are circled; the circle color of the CC represents the order of a conjunctive clause.

The features associated with the 7th-order conjunctive clause of Figure 3.14 are presented in Figure 3.15A and include binary, nominal, and discrete data types. This conjunctive clause has the highest infested house coverage for clauses with 100% accuracy. We select a 100% accurate conjunctive clause to critically examine the possibility of overfitting. The arrows of Figure 3.15 point to the accuracy, coverage, and hypergeometric PMF of the conjunctive clause when that particular feature is removed.

Because two of the seven features (one identifying that household source of income is not salary and the other indicating that the house is owned) show no change in accuracy or coverage (and negligible change in the hypergeometric PMF) when those features are removed, we reduce the 7th-order conjunctive clause to produce the 5th-order conjunctive clause of panel B.



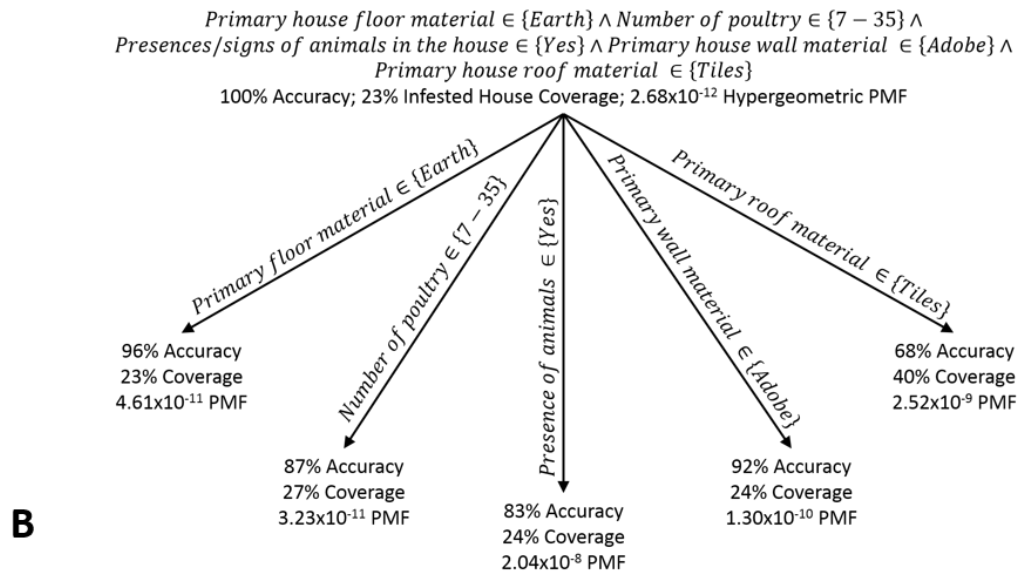


Figure 3.15: The archived conjunctive clause resulting from the 64-feature, combined (two-town) Chagas dataset with 100% accuracy and highest coverage. The arrows point to the resulting accuracy, coverage, and hypergeometric PMF when the feature associated with the line is removed from the conjunctive clause Panel A) shows the 7th-order conjunctive clause. Because there was no change in accuracy or coverage for removal of two of the features, panel B shows the resulting 5th-order conjunctive clause. The arrows in panel B again point to accuracy, coverage and fitness when the associated feature is removed.

The two additional archived CCs selected along the Pareto front of Figure 3.14 include a 4th-order and 2nd-order CC. The 4th-order was selected because it is near the knee of the Pareto front and is depicted in Figure 3.16A. This CC has high accuracy (71%) and high infested house coverage (53%). Removing any one of these four features results in less-fit 3rd-order CCs. Figure 3.15B shows a 2nd-order CC with the highest infested house coverage for all archived CCs. The feature (house is owned or rented) covers 96% (300/311) of the observations in the combined dataset. Note that when the feature associated with the condition of the walls in the remainder of the house is removed, the resulting main effect is many orders of magnitude less-fit.

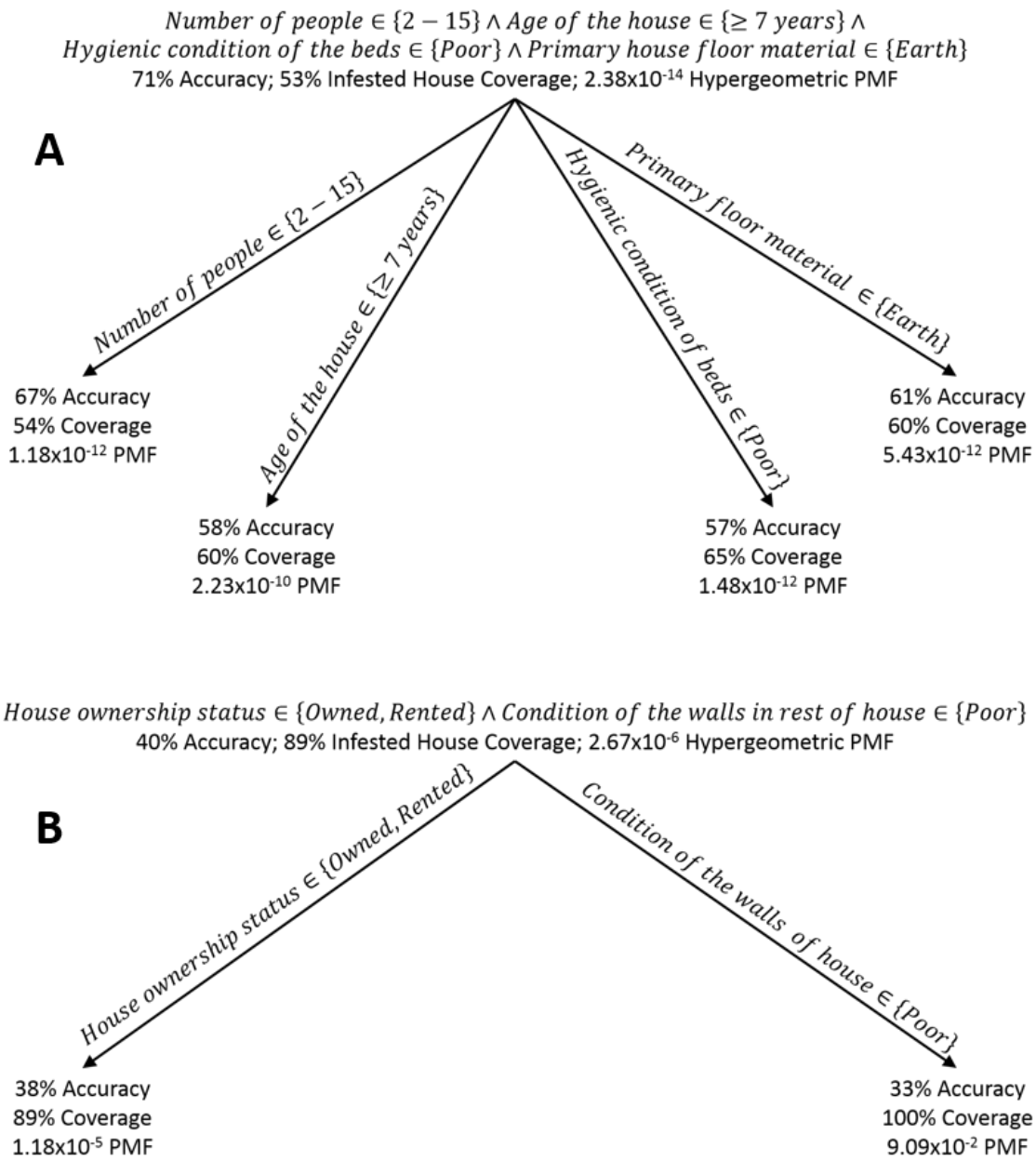


Figure 3.16: Description of two additional archived conjunctive clauses from the Pareto front of the 64-feature, combined (two-town) Chagas dataset. Panel A) describes the 4th-order conjunctive clause near the knee of the Pareto front. Panel B) shows 2nd-order conjunctive clause with the highest coverage for the archived CCs. Arrows point to the resulting accuracy, coverage, and hypergeometric PMF when the feature associated with the line is removed from the conjunctive clause.

3.5 Discussion

We develop a novel evolutionary algorithm (CCEA) to efficiently explore complex feature interactions associated with disease datasets. The algorithm is capable of dealing with survey data that contain missing data, varied data types (i.e., binary, nominal, and ordinal), and additive features. We demonstrate, using a benchmark SNP disease dataset specifically designed for feature interactions and heterogeneity by Urbanowicz and Moore, (2010), that the CCEA is capable of identifying the four (two-way) signals embedded in a benchmark SNP dataset. The algorithm successfully archives the conjunctive clauses that are then mined for the most important features using *FI* and *FPI* network models and feature sensitivity. When feature-reduction methods were applied to the Chagas datasets, the coverage and accuracy for the archived conjunctive clauses associated with infested houses did not increase; and the number of highly accurate 2nd- and 3rd-order conjunctive clauses decreased. This is likely because the risk factors (features) associated with the 2- and 3-way feature interactions were subsequently removed during feature reduction showing the dangers of performing feature reduction when datasets contain feature interactions and heterogeneity.

Feature sensitivity shows that while some of the archived conjunctive clauses contain noise, we can obtain equally-fit, lower-order conjunctive clauses. We identify a 5th-order conjunctive clause that contains three features (earth floors, adobe walls, and tile roofs) previously identified (King et al., 2011; Weeks et al., 2013; Bustamante et al., 2014; Bustamante Zamora et al., 2015) as individually associated with infestation. However, what is unique to this 5th-order conjunctive clause is that the CCEA is capable

of evolving both the feature set and the range of feature values. For example, the range of values that evolved for the number of poultry lies between 7 – 35. Given the computational limitations imposed by the large combinatorial constraints associated with real multivariate interactions comprised of multiple data types, previous studies have no choice but to bin survey data responses (e.g., bin the number of poultry into ranges 0, 1-9 10-19, and ≥19) prior to statistical analysis (Bustamante Zamora et al., 2015). The CCEA is able to relax this constraint and mine the entire set of feature combinations and simultaneously evolve ranges for the number of poultry using statistical signals. The 5th-order conjunctive clause described above paints a picture of a home that many Chagas experts would suspect as being at high risk for *T. dimidiata* infestation (e.g., numerous hiding places for the vector and readily available food sources such as large numbers of poultry and animals inside the home). With that being said, we caution against making any generalizations regarding this (100% accurate) 5th-order conjunctive clause for other nearby towns. The lack of false positives is likely due to the towns being small rural towns with only 311 houses. However, given that this clause covers nearly a quarter of the infested houses across the two towns, this conjunctive clause may represent a driver of *T. dimidiata* infestation. In addition, this conjunctive clause contains many of the features already targeted by existing Ecohealth interventions, such as replacing dirt floors with cement floors, plastering walls, and moving chickens outside of the home and into coops constructed with wire (Monroy et al., 2009; Lucero et al., 2013; Pellecer et al., 2013). These interventions, especially when taken together, are some of the most efficient (i.e., lowest number of false positives) interventions.

There is evidence of heterogeneity in identifying models of infestation with triatomine vectors of Chagas disease. Both Bustamante et al. (2014) and Bustamante Zamora et al. (2015) found no statistical support for a single-best model of infestation. Our results show there are conjunctive clauses that cover nearly every infested house (Figure 3.8); however, these conjunctive clauses tend to have low accuracy and thus a large number of false positives. Alternatively, stakeholders may want a compromise between high infested house coverage and accuracy; and thus, they may be willing to accept some false positives in favor of a simpler more general conjunctive clause such as the 4th-order CC depicted in Figure 3.16A. Alternatively, multiple CCs with high coverage and accuracy may be combined to cover all or nearly all infested houses, while limiting false positives so that stakeholders can efficiently direct limited resources (e.g., it may be more cost effective to perform a preliminary screening of houses at risk of infestation using information such as the age of the house than to acquire information on the household source of income or number of animals). The range of conjunctive clauses allows stakeholders to select the best combination that maximizes their desired coverage, accuracy, model complexity, and the presence of features that have interventions that can be easily and affordably applied.

3.6 Supplementary Tables

Table 3.S1: The first column is the feature number for the 64 features that are input into the CCEA. The second column is the survey question associated with each feature.

Feature	Survey Question
1	Total people in the house
2	Binary Source of Income: Day Laborer (1 = yes, 0 = no)
3	Binary Source of Income: Agriculture (1 = yes, 0 = no)
4	Binary Source of Income: Business (1 = yes, 0 = no)
5	Binary Source of Income: Salary (1 = yes, 0 = no)
6	Binary Source of Income: Other (1 = yes, 0 = no)
7	Highest household level of education
8	Is the house owned, rented, or borrowed?
9	Age of the house
10	Years lived in the house
11	Have you improved the house?
12	Binary Type of House Improvement: Plastered the Walls (1 = yes, 0 = no)
13	Binary Type of House Improvement: Improved the roof (1 = yes, 0 = no)
14	Binary Type of House Improvement: Improved the floor (1 = yes, 0 = no)
15	Binary Type of House Improvement: Addition to the house (1 = yes, 0 = no)
16	Binary Type of House Improvement: Other (1 = yes, 0 = no)
17	How often do you plaster the walls
18	Number of dogs
19	Where do the dogs sleep?
20	Number of poultry
21	Where do poultry birds sleep?
22	Number of cats
23	Where do cats sleep?
24	Number of pigs
25	Number of beasts (i.e., horses, cows, mules)
26	Presence or signs of animals in the house
27	Presence or signs of bird nests in the house
28	Presence or signs of mouse in the house
29	Accumulation of objects in the house
30	Binary Types of Objects Accumulated: Boxes (1 = yes, 0 = no)
31	Binary Types of Objects Accumulated: Sacks (1 = yes, 0 = no)
32	Binary Types of Objects Accumulated: Clothes (1 = yes, 0 = no)
33	Binary Types of Objects Accumulated: Other (1 = yes, 0 = no)
34	Hygienic condition of the beds
35	Are beds separated from the wall
36	Hygienic condition of the house

37	Are grains stored in the main room
38	Accumulation of firewood
39	Where is the firewood
40	Accumulation of construction materials
41	Binary Type of Accumulated Construction Material: Adobe (1 = yes, 0 = no)
42	Binary Type of Accumulated Construction Material: Tiles (1 = yes, 0 = no)
43	Binary Type of Accumulated Construction Material: Wood (1 = yes, 0 = no)
44	Binary Type of Accumulated Construction Material: Cinder Blocks (1 = yes, 0 = no)
45	Binary Type of Accumulated Construction Material: Other (1 = yes, 0 = no)
46	Where are the construction materials
47	Presence of chicken coop
48	Where is the chicken coop located
49	Primary material of the chicken coop walls
50	Primary material of the chicken coop roof
51	Hygienic condition of the chicken coop
52	Presence of another animal corral
53	Primary material of the house walls
54	Condition of the walls in the bedroom
55	Condition of the walls in the rest of the house
56	Primary material of the house roof
57	Primary material of the house floor
58	Is the main room dark?
59	Does the main room have a skylight?
60	Does the main room have windows?
61	Location of the kitchen
62	Does the house have running water?
63	Does the house have electricity?
64	How long ago was the electricity installed?

Table 3.S2: Table A contains the feature importance (*FI*) values for a dataset with eight features and 10 observations with a target outcome (e.g., infested house). Table B contains the same *FI* values that are present in Table A, however, each feature's *FI* is independently sorted. The 90th percentile *FI* values are highlighted in red.

A								
Obs.	FI_1	FI_2	FI_3	FI_4	FI_5	FI_6	FI_7	FI_8
1.	0.44	0.80	0.26	0.25	0.07	0.00	0.04	0.00
2.	0.99	1.00	0.06	0.10	0.00	0.29	0.07	0.04
3.	0.94	1.00	0.00	0.26	0.19	0.27	0.08	0.41
4.	1.00	0.98	0.51	0.64	0.00	0.05	0.21	0.38
5.	0.00	0.26	0.15	0.11	0.19	0.27	0.94	1.00
6.	0.35	0.34	0.23	0.00	0.26	0.17	0.95	1.00
7.	0.24	0.14	0.14	0.27	0.00	0.38	1.00	0.98
8.	0.28	0.58	0.09	0.00	0.16	0.28	0.99	1.00
9.	0.39	0.80	0.26	0.25	0.07	0.00	0.04	1.00
10.	0.99	1.00	0.06	0.10	0.00	0.29	0.07	0.04

B								
Sort	FI_1	FI_2	FI_3	FI_4	FI_5	FI_6	FI_7	FI_8
1.	0.00	0.14	0.00	0.00	0.00	0.00	0.04	0.00
2.	0.24	0.26	0.06	0.00	0.00	0.00	0.04	0.04
3.	0.28	0.34	0.06	0.10	0.00	0.05	0.07	0.04
4.	0.35	0.58	0.09	0.10	0.00	0.17	0.07	0.38
5.	0.39	0.80	0.14	0.11	0.07	0.27	0.08	0.41
6.	0.44	0.80	0.15	0.25	0.07	0.27	0.21	0.98
7.	0.94	0.98	0.23	0.25	0.16	0.28	0.94	1.00
8.	0.99	1.00	0.26	0.26	0.19	0.29	0.95	1.00
9.	0.99	1.00	0.26	0.27	0.19	0.29	0.99	1.00
10.	1.00	1.00	0.51	0.64	0.26	0.38	1.00	1.00

CHAPTER 4: USING NEXT GENERATION SEQUENCING TO DETERMINE THE RANGE OF SPATIAL AUTOCORRELATION OF *TRIATOMA DIMIDIATA*

4.1 Introduction

4.1.1 Chagas Disease Background

Chagas disease is a lethal, neglected, tropical disease that is endemic to every Central American country (Chagas, 2015). Historically *Rhodnius prolixus*, and to a lesser extent *Triatoma dimidiata*, were the principle vectors of Chagas disease in Central America (WHO, 2002; Schofield and Dujardin, 1997). *R. prolixus* was accidentally introduced to Central America (Zeledón, 2004) but, through an intensive insecticide campaign in August 2011, transmission of Chagas disease via *R. prolixus* was successfully eliminated from Central America (Hashimoto and Schofield, 2012). *T. dimidiata* on the other hand is endemic to Central America and is found in domestic, peridomestic, and sylvatic ecotopes from Mexico to northern South America (De León, 1959; Arzube Rodríguez, 1966; Zeledón et al., 1970; Petana, 1971; Zeledón et al., 1973; Whitlaw and Chaniotis, 1978; Tabaru et al., 1999; Zeledón et al., 2001a; Zeledón et al., 2001b; Dumonteil et al., 2002; Monroy et al., 2003a; Monroy et al., 2003b; Sasaki et al., 2003; Ramírez et al., 2005; Zeledón et al., 2005; Hernández et al., 2006; Zeledón and Rojas, 2006; Bustamante et al., 2007; Dorn et al., 2007). Efforts to eliminate *T. dimidiata* through intensive insecticide spraying have proven generally unsuccessful (Tabaru et al., 1998; Nakagawa et al., 2003a; Nakagawa et al., 2003b; Dumonteil et al., 2004; Hashimoto et al., 2006; Manne et al., 2012; Yoshioka et al., 2015). Approximately 70

million people in Latin America are at risk of infection with *T. cruzi* and ~5.7 million people are already infected (Chagas, 2015). In Central America, Guatemala has the largest number of vector transmitted cases (~1,275) in 2010 (Chagas, 2015). Since insecticides have proven to be ineffective at eliminating *T. dimidiata* in Guatemala (Tabaru et al., 1998; Nakagawa et al., 2003a; Nakagawa et al., 2003b; Hashimoto et al., 2006; Manne et al., 2012), recent interventions help reduce contact between people and *T. dimidiata* using Ecohealth interventions (Monroy et al., 2009; Lucero et al., 2013; Pellecer et al., 2013). In Guatemala, these interventions aim to remove domestic shelters and food sources of *T. dimidiata* by plastering walls (Monroy et al., 2009; Lucero et al., 2013; Pellecer et al., 2013), replacing dirt floors with cement floors (Lucero et al., 2013; Pellecer et al., 2013), and distancing the chicken coops from the house while also replacing the more common construction materials (i.e., wood and adobe) with metal fencing (Lucero et al., 2013; Pellecer et al., 2013). Unfortunately, the initial costs of these Ecohealth interventions are relatively high for many communities; and thus, resources should be prioritized toward houses most at risk of *T. dimidiata* infestation. While extensive evidence suggests that *T. dimidiata* move between the domestic, peridomestic, and sylvatic ecotopes (Arzube Rodríguez, 1966; Zeledón et al., 1973; Monroy et al., 2003b; Sasaki et al., 2003; Ramírez et al., 2005; Zeledón et al., 2005), the distance traveled is currently unknown, thus complicating estimates of disease transmission. A number of studies have modeled the movement of *T. dimidiata* from the sylvatic ecotope to domestic ecotope in the Yucatan, Mexico (Barbu et al., 2010; Ramirez-Sierra et al., 2010; Barbu et al., 2011), but the vector is associated with seasonal infestation in the

Yucatan, which is not the case in Guatemala. In Guatemala, Lucero et al. (2013) used geostatistics and hot spot analysis to identify areas within a village most at risk for infestation. Their analysis identified a range of spatial autocorrelation for bugs per house found using man-hour collection methodology. However, Monroy et al. (1998) have demonstrated that this collection method has a high variance and most likely results in finding only 0.7 – 10.8% of the true population (Monroy et al., 1998; Valença-Barbosa et al., 2014). In addition, the method is biased toward houses with high numbers of bugs (Abad-Franch et al., 2014). All of the above challenges help motivate the need for alternative models for estimating the range of spatial autocorrelation associated with *T. dimidiata*.

4.1.2 Background on a Genetic Geostatistical Method for Spatial Autocorrelation

Smouse and Peakall (1999) developed a methodology to characterize the range of spatial autocorrelation that uses multiple genetic markers to create a correlogram. Their methodology has subsequently been used on a variety of species such as emmer wheat (*T. turgidum* L. ssp. *dicoccoides*) (Volis et al., 2014), beech trees (*Fagus sylvatica* L.) (Piottti et al., 2013), bottlenose dolphins (*Tursiops truncatus*) (Richards et al., 2013), Canada geese (*Branta canadensis*) (Finnegan et al., 2013), and the American black bear (*Ursus americanus*) (Coster and Kovach, 2012). Foley et al. (2004) used the correlogram to find the range of spatial autocorrelation for the mosquito vector (*Ochlerotatus notosciptus*) of dog heartworm (*Dirofilaria immitis*) to be ~55 km. While Rašić et al. (2015), found a range of 3-6 km for the mosquito *Aedes aegypti*, which is a vector of dengue. Finally, Pérez de Rosas et al. (2013) investigated the range of spatial

autocorrelation for *Triatoma infestans*, the principle vector of Chagas disease in South America, and found a range of ~400 m. They also investigated sex-biased dispersal finding that females had a relatively larger range of spatial autocorrelation than males (400 m versus 330 m), and used the range to guide the radius of insecticide applied around an infested house or peridomestic structure.

4.1.3 Background on Spatial Autocorrelation in Human SNP Data

Spatial autocorrelation in the single nucleotide polymorphisms (SNPs) of humans has been observed at various scales. Elhaik et al. (2013) found spatial autocorrelation at the global scale and was relatively successful at leveraging georeferenced SNP data to predict a person's country of origin. On a finer scale, Lao et al. (2013) used Smouse and Peakall's (1999) correlogram to determine the range of spatial autocorrelation in people in the Netherlands, which they attributed to historic settlement patterns, using to georeferenced SNP data.

4.1.4 Genetics of *Triatoma dimidiata*

Spatial autocorrelation in the genetics of *T. dimidiata* have been observed at various scales. Bargues et al. (2008) analyzed 31 haplotypes at 64 locations that spanned a range from Mexico to northern South America. While they did not explicitly characterize spatial autocorrelation, they did show geographic grouping of phenotype trees. More recently, Stevens et al. (2015) investigated spatial autocorrelation using 7 highly polymorphic microsatellite loci from 178 *T. dimidiata* spread across 6 villages in the department of Jutiapa, Guatemala. Using the relatedness function of Lynch and Ritland (1999), Stevens et al. (2015) found some migration of *T. dimidiata* between

houses in a village as well as some spatial autocorrelation, despite the signal being weak. These findings are contrary to earlier works that did not find spatial autocorrelation among *T. dimidiata* in nearby villages in Guatemala (Dorn et al., 2003; Calderón et al., 2004). Given that Melgar et al. (2007) found 41 families of *T. dimidiata* in a single house in Guatemala, using tens of genetic markers is unlikely to provide sufficient genetic information to sufficiently capture (characterize) within-town, spatial autocorrelation. As a result, using the thousands of *T. dimidiata* SNPs from the Orantes (personal communication, January 2017) database may provide a unique opportunity to explore spatial autocorrelation at the finer village scale.

4.1.5 Summary of Work

In this work, we use the SNP database of Orantes (personal communication, January 2017) and the genetic distance of Smouse and Peakall (1999) and relatedness of Lynch and Ritland (1999) to explore spatial autocorrelation at the finer village scale. For two towns in Jutiapa, Guatemala, Orantes (personal communication, January 2017) extracted the DNA of 216 T. dimidiata, and through Rad-seq was able to create a database of single nucleotide polymorphisms (SNPs). We use these T. dimidiata SNPs to (1) create semivariograms, (2) characterize the range of spatial autocorrelation of the vector in both villages, and (3) use these metrics as a surrogate for vector movement to produce maps of homesteads most at risk of infestation.

4.2 Study Sites and Methods

4.2.1 Study Sites and Genetic Data

Our study sites are the small rural towns of El Chaperno and El Carrizal located in the dry highlands of in Jutiapa, Guatemala (red and yellow dots of Figure 3.1). Jutiapa, Guatemala (highlighted in red, Panel A) borders El Salvador with the study site locations shown as a yellow star. El Carrizal (Panel B) has spur roads radiating from the main road making the town less linear in shape. While El Chaperno (Panel C) is more linear in shape since most of the houses are adjacent to the principal road running through the town. Also, El Chaperno is more heavily forested than El Carrizal due to forest conservation efforts.

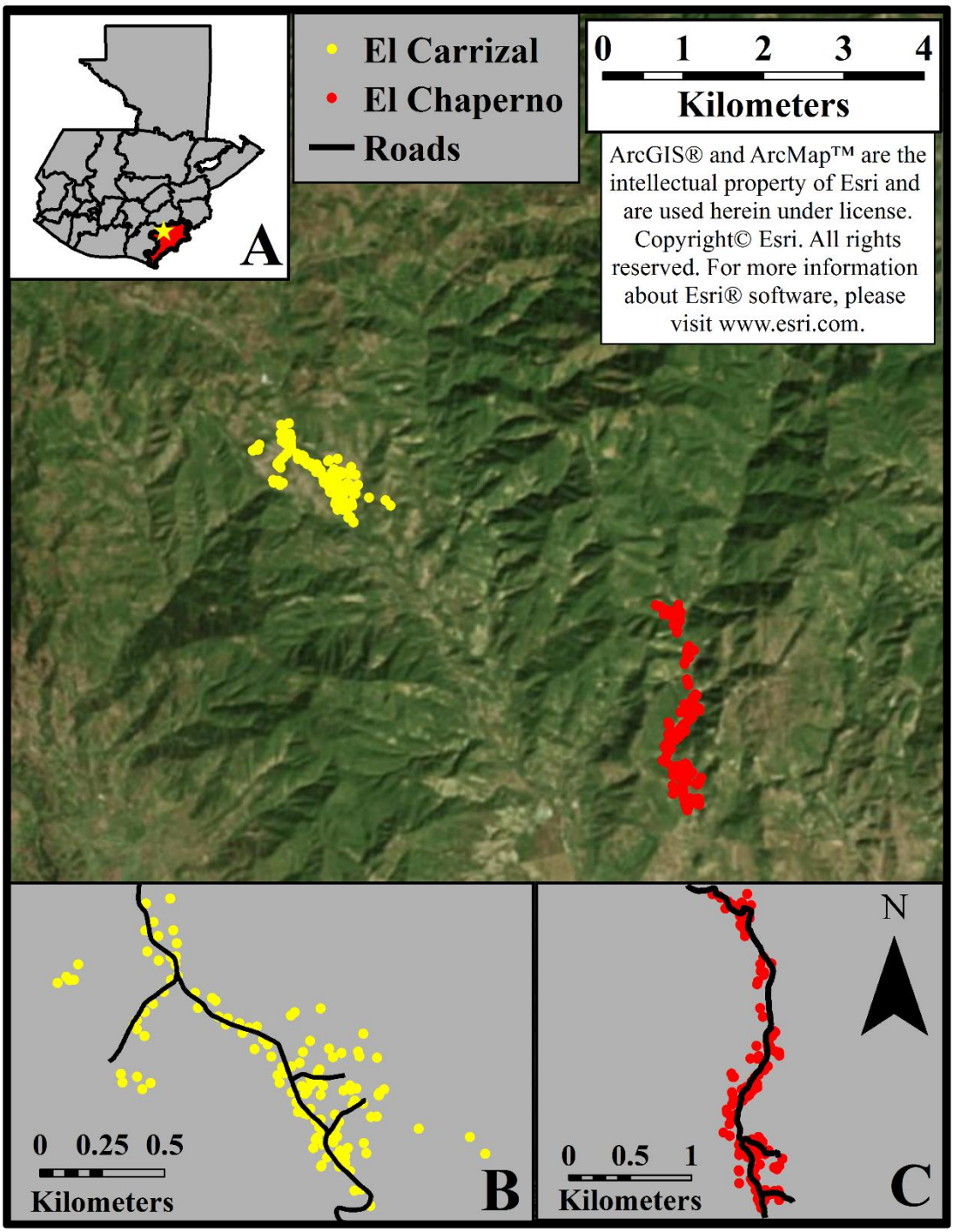


Figure 4.1: Satellite image of the study sites with the houses in El Chaperno and El Carrizal represented as red and yellow dots, respectively. Panel A is a map of the departments of Guatemala with the department of Jutiapa highlighted in red and the location of the study sites represented as a yellow star. Panels B and C are show the locations of the houses and roads in El Carrizal and El Chaperno, respectively.

Teams comprised of personnel from the Escuela de Biología at La Universidad de San Carlos Guatemala, and the Guatemalan Ministry of Health Office of Vector-Borne Diseases conducted entomological surveys for the domestic and peridomestic ecotopes of 182 and 129 homesteads in El Chaperno and El Carrizal, respectively (Table 4.1). Informed consent was obtained from all adult participants and from parents or legal guardians of minors. This project received ethical clearance from the Ministry of Health in Guatemala, La Universidad de San Carlos bioethics committee, and the Panamerican Health Organization.

Given the challenges with finding live *T. dimidiata* (Monroy et al., 1998), we believe that signs of *T. dimidiata* indicate the vector has likely infested a homestead. As a result, we categorized homesteads as infested if either their domestic or peridomestic ecotopes contain any sign of *T. dimidiata* (i.e., live, dead, eggs, exuviae, or feces). In El Chaperno and El Carrizal, Ministry of Health officials collected 276 and 222 live *T. dimidiata* from 35 and 31 homesteads, respectively. Due to the cost associated with next generation sequencing, only a proportion of the collected bugs were sequenced; all bugs from houses with three or fewer bugs were sequenced; and for houses with larger numbers (4-81 bugs), a select proportion were sequenced. For detail, see Table 4.1 & Figures 4.S1 and 4.S2; and for details of the experimental design, see Orantes (personal communication, January 2017).

Table 4.1: The characteristics of the El Chaperno and El Carrizal datasets collected during the periods of October 1-3, 2012 in El Chaperno and February 4-5, 2013 in El Carrizal.

	El Chaperno	**El Carrizal**
Number of Homesteads Surveyed	182	129
Number of Homesteads Infested	56 (31%)	52 (40%)
Number of Homesteads with live *T. dimidiata*	35 (19%)	31 (24%)
Number of live *T. dimidiata* collected	276	222
Number of Homesteads with Sequenced *T. dimidiata*	34 (19%)	30 (23%)
Number of *T. dimidiata* Sequenced	95 (34%)	121 (55%)

Orantes (personal communication, January 2017) used next generation sequencing on a subset of the *T. dimdiata* collected in El Chaperno and El Carrizal (Table 4.1) to determine the subsequent SNP loci for each population of bugs. Figures 4.2 and 4.3 present the sex, instar level, and homestead location of specimens sequenced in El Chaperno and El Carrizal, respectively; the size of the pie charts are proportional to the number of bugs sequenced from the same homestead.

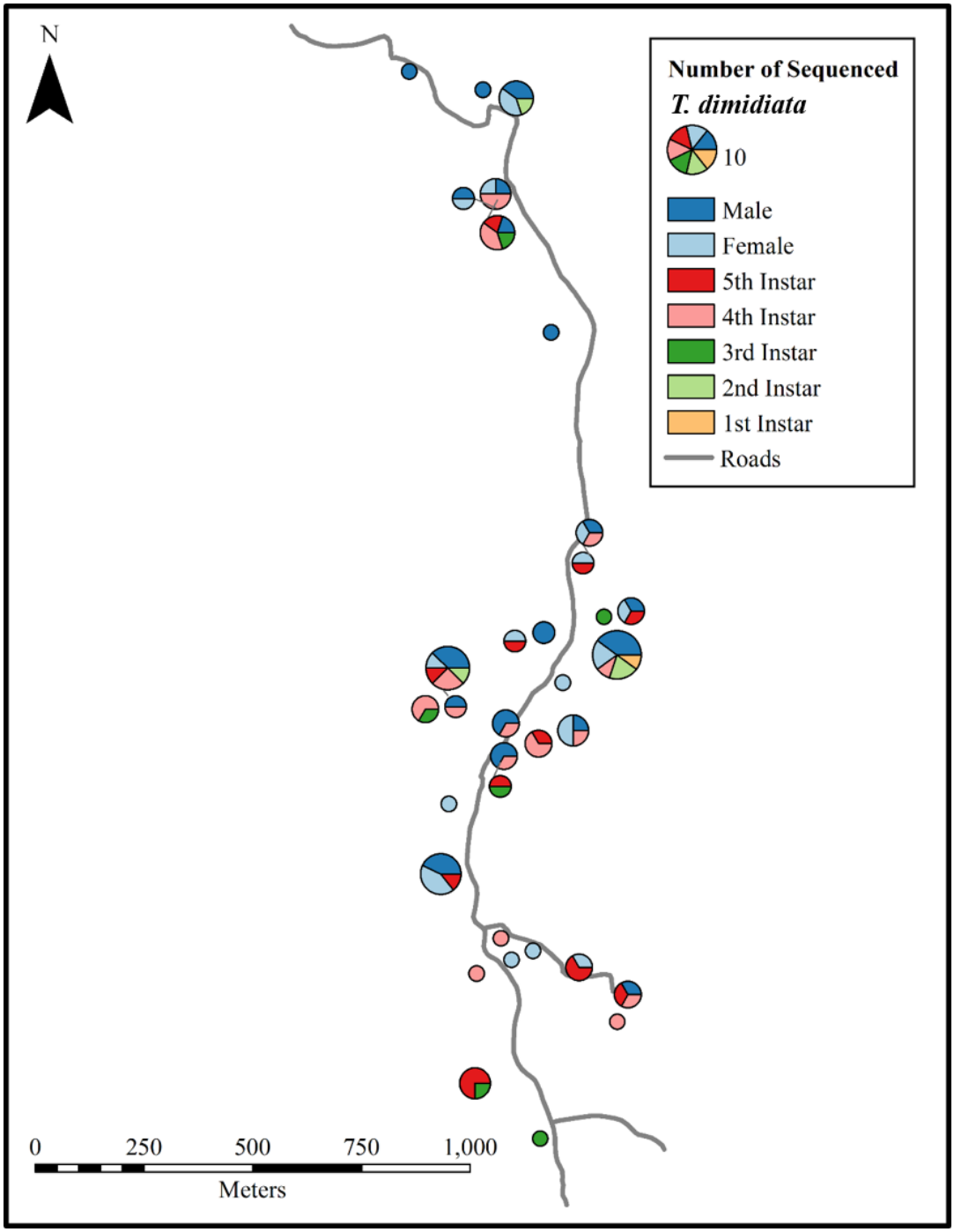


Figure 4.2: The pie charts are proportional to the number of *T. dimidiata* sequenced for a given homestead ranging from 1 to 10 sequenced per house. Colors represent the sex, instar level, and homestead location of collected insects in El Chaperno.

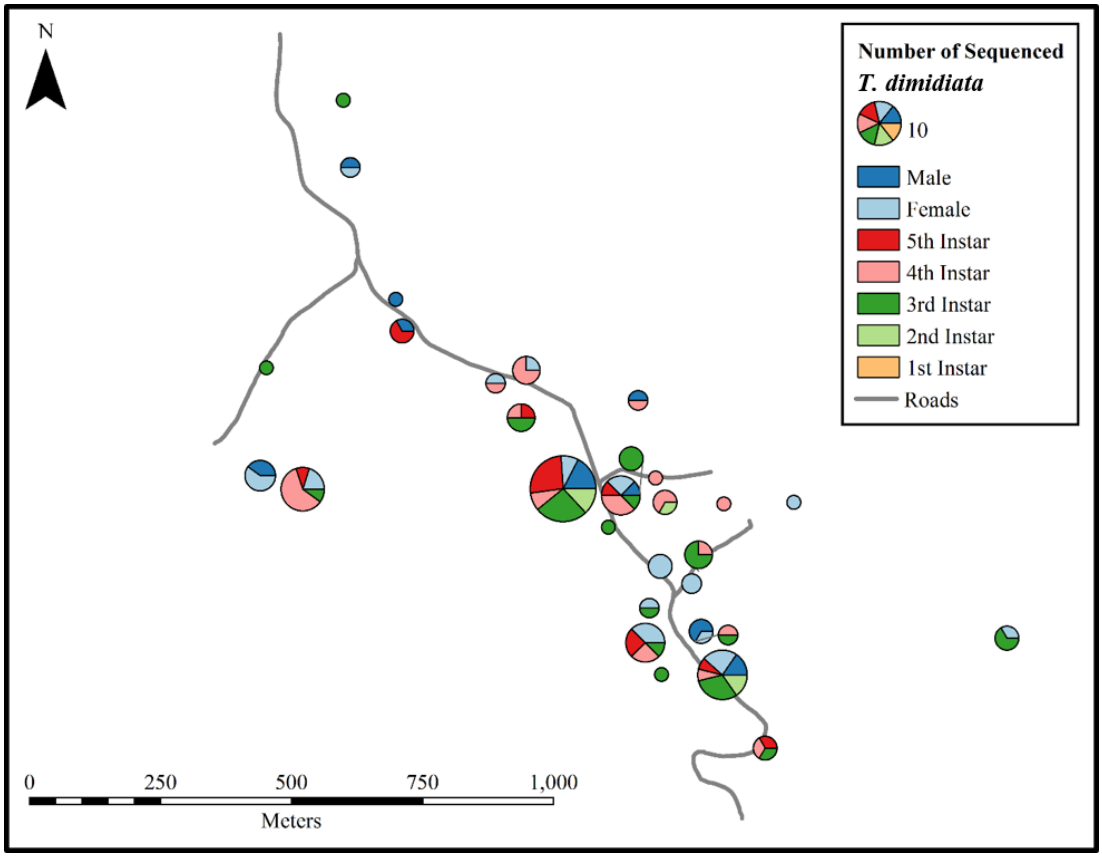


Figure 4.3: The pie charts are proportional to the number of *T. dimidiata* sequenced for a given homestead ranging from 1 to 23 sequenced per house. Colors represent the sex, instar level, and homestead location of collected insects in El Carrizal.

For El Chaperno and El Carrizal, the number of SNPs is 1,870 and 2,265, respectively; and the percent missing data (i.e., specimens and associated SNPs with International Union of Pure and Applied Chemistry (IUPAC) nucleotide code N values) is 37% and 41%, respectively. To test the impact of missing data, we analyzed each dataset with and this filtering. We refer to these two datasets throughout the manuscript as the *original* and *filtered* datasets. Original refers to the dataset containing all sequenced specimens and all loci with a SNP. We then perform multiple levels of filtering on the original dataset. One level refers to retaining only pairs of specimens that

have some threshold of loci in common (referred to Level 1). Level 2 refers to refers to removing all specimens with >50% missing values, followed by removal of any loci with missing values. For Level 2 filtering, the El Chaperno was reduced to 73 specimens and 287 loci across 34 homesteads; El Carrizal was reduced to 97 specimens and 250 loci across 30 homesteads.

4.2.2 Geostatistical methodology

While the correlograms of Smouse and Peakall (1999) were certainly ahead of their time in the field of genetics, semivariograms (the inverse of correlograms) were introduced in the mining industry as early as the 1930s and have been used extensively to leverage spatial autocorrelation in subsurface site investigations. Semivariograms express the range of autocorrelation (spatial or temporal) as a dissimilarity between measurement points rather than normalized similarity (i.e., correlograms). Semivariograms are generally the preferred method for measuring spatial autocorrelation because they (1) allow for empirical derivation of 95% confidence intervals, (2) plotting of raw semivariance (prior to binning), which allows for easier visualization of natural geographic breakpoints for binning, and (3) preserve the relative error variance between measured variables.

The equation of the semivariance (Isaaks and Srivastava, 1989) is given as:

$$\gamma(h) = \frac{1}{2N(h)} \sum_{\alpha=1}^{N(h)} \left[z(u_\alpha) - z\left(u_{\alpha+|h|}\right) \right]^2 \ , \qquad (4.1)$$

where, $z(u_\alpha)$ is the measurement at a point in space or time, $z\left(u_{\alpha+|h|}\right)$ is the measurement of a point at a distance, h, from location u_α, and N is the total number of paired points separated by distance, h.

The correlogram of Smouse and Peakall (1999) calculates a genetic distance d^2 that is similar to the semivariance in equation (4.1). For example, the genetic distance using SNPs for the diploid *T. dimidiata* may be calculated as:

$$d^2(h) = \sum_{k=1}^{K} \frac{1}{2} \begin{bmatrix} \left(A(u_\alpha) - A\left(u_{\alpha+|h|}\right)\right)^2 + \left(C(u_\alpha) - C\left(u_{\alpha+|h|}\right)\right)^2 \\ + \left(G(u_\alpha) - G\left(u_{\alpha+|h|}\right)\right)^2 + \left(T(u_\alpha) - T\left(u_{\alpha+|h|}\right)\right)^2 \end{bmatrix}, \qquad (4.2)$$

where, *A*, *C*, *G*, and *T* represent the total number of purine adenine, pyrimidine cytosine, purine guanine, and pyrimidine thymine, respectively, K is the total number of loci, $A(u_\alpha)$ is the total number of purine adenine (A) (i.e., 0, 1, or 2) at the *k*th locus for the specimen at location u_α, and $A\left(u_{\alpha+|h|}\right)$ is the total number of purine adenine associated with a specimen that is distance *h* from u_α. Both equations 4.1 and 4.2 represent half the squared difference between two points, with the exception that equation 4.2 is designed for nominal genetic data, and equation 4.1 is designed for continuous data. We, therefore, refer to the Smouse and Peakall (1999) genetic distance as a genetic semivariance in the remainder of this manuscript.

A common geostatistical practice is to bin semivariances based on geographic distance in order to more easily visualize the pattern of spatial autocorrelation. Since there were no natural break points and to maintain consistency between the two towns the semivariances were binned using the same fixed distance bins for both towns. The first bin consists of all semivariances in the same house. The median nearest neighbors for El Chaperno and El Carrizal are 35 and 62 meters, respectively, so an approximate midpoint of 50 meters was selected for the second bin. The third bin was all pairs of bugs between 50-100 meters. The fourth through twelfth bins increased at 100 meter intervals

and the final bin contained all bugs >1,000 meters apart. Each binned semivariance is centered on the median of the distances for that bin.

Semivariance is often represented by a model that is best fit to the average semivariance (i.e., all paired data within select bin (or range of distances, *h*) are averaged); and a 95% confidence interval that may be placed around the binned averages. The latter assumes the population follows a normal distribution. Lucero et al. (2013) assumed a normal distribution when creating a semivariogram model for the number of *T. dimidiata* collected at homesteads. However, given that most of the homesteads had zero values (i.e., no insects were found), the majority of the raw semivariance values were zero, which violates the assumption of a normal distribution. Therefore, we propose representing the binned (mean) semivariance as Tukey box plots (Tukey, 1977) and use the box plot median as a surrogate for the mean. When the binned semivariances do not follow a normal distribution, the median, midspread (i.e., interquartile range), and outliers of the box plots, help visualize the distribution of binned semivariances. When the binned semivariances are normally distributed, then the mean and median should be equivalent. Once the semivariances are binned and plotted as box plots, a domain expert can select the parameters (i.e., the nugget, range, and sill) needed to characterize/model spatial (or temporal) autocorrelation of the data. The nugget represents measurement error or the general variability within the measured parameter that is not spatially dependent. Range (also referred to as the range of decorrelation) defines the distance beyond which the variable (in our case, genetic distance and relatedness) is no longer correlated. The sill represents the median variance in vector genetic distance or

relatedness for homesteads separated by distances greater than the range of decorrelation (Isaaks and Srivastava, 1989). Summary statistics (e.g., mean, median) are used to describe the nugget, range, and sill of spatial autocorrelation in order to fit a model a model to the semivariance. The model is then used to map a variable such as genetic distance. The model will produce two spatial maps using kriging; one map will plot the estimated variable value and the other will plot the error variance associated with the estimate.

In addition to using the genetic distance of Smouse and Peakall (1999), we used the relatedness of Lynch and Ritland (1999) as an alternative (genetic semivariance) equation, because it is a frequency-based, pairwise equation. Since higher values of relatedness are associated with more similar individuals, we simply invert the vertical axis such that paired individuals who are more related plot below the x-axis and those less-related plot above the x-axis. We used GenAlEx 6.503 (Peakall and Smouse, 2006; 2012) to calculate both genetic distance and relatedness for all pairs of insects collected in El Chaperno and El Carrizal. However, GenAlEx 6.503 has two options for handling missing data; one treats missing data as a base pair; and the other interpolates the missing data. Smouse and Peakall (2006) do not recommend interpolating missing data for individual statistics; whereas, filtering data often results in the loss of large amounts of data. As a result, we tested the impact of filtering in two ways. First, we encoded the genetic distance formula of Smouse and Peakall (1999) using Matlab® 2016a (MathWorks, Natick, MA) and used the Level 1 filtering. We normalize each pair-wise

genetic distance by the maximum genetic distance for the given pair, given the number of loci that have data.

To test the impact of data reduction (pairs of individuals that do not have loci in common) when filtering, we created semivariograms using three thresholds for Level 1 filtering: (1) all data having at least one locus in common, (2) all data having 287 or 250 loci in common for El Chaperno and El Carrizal, respectively (i.e., the same number of loci as Level 2 filtering), and (3) all data having at least 1,000 loci in common. We also compare the Smouse and Peakall (1999) genetic distance semivariograms to the Lynch and Ritland (1999) relatedness semivariograms for the Level 2 filtering of the El Chaperno and El Carrizal datasets.

The semivariograms were each fit with a spherical model (Marsily, 1993) using the nugget, range, and sill to help characterize the change in semivariance with distance. Finally, a buffer sized by the range of spatial autocorrelation was placed around each infested homestead for both villages; and the resulting overlap is plotted as increased risk of infestation for homesteads that lie within multiple buffers. This risk assumes that infestation comes from within the system (only infested homesteads are sources of risk) and that spatial genetic structure is indicative of vector movement.

4.3 Results

4.3.1 Results Level 1 Filtering

For both towns, the genetic distance of equation (4.2) was modified to use only the base pairs that were common between individuals. Figures 4.4 and 4.5 show the binned semivariances for El Chaperno and El Carrizal, respectively. The nugget for each

panel was set to the median of the semivariance bin corresponding to a distance of zero. Panels A, B, and C represent increasing thresholds of minimum common loci, respectively; and a spherical model (red dashed line) was fit to the box plot medians. For El Chaperno (Figure 4.4), the best fit spherical model has the same median range (28 meters; minimum and maximum bin distance are 9 and 48 meters, respectively) and sill (0.08) for all three levels of filtering (minimum common loci thresholds of 1, 287, and 1,000, respectively). There is little difference between the three semivariograms, with the exception that panel A, which uses all 4,465 pairs of *T. dimidiata* in the dataset, has larger inter-whisker semivariance distances for each bin and more outliers than panels B or C. That being said, the median range of spatial autocorrelation is 28 meters ([9, 48] meters) regardless of the threshold, which is most likely due to the fact that the majority of our data have over 1,000 loci in common.

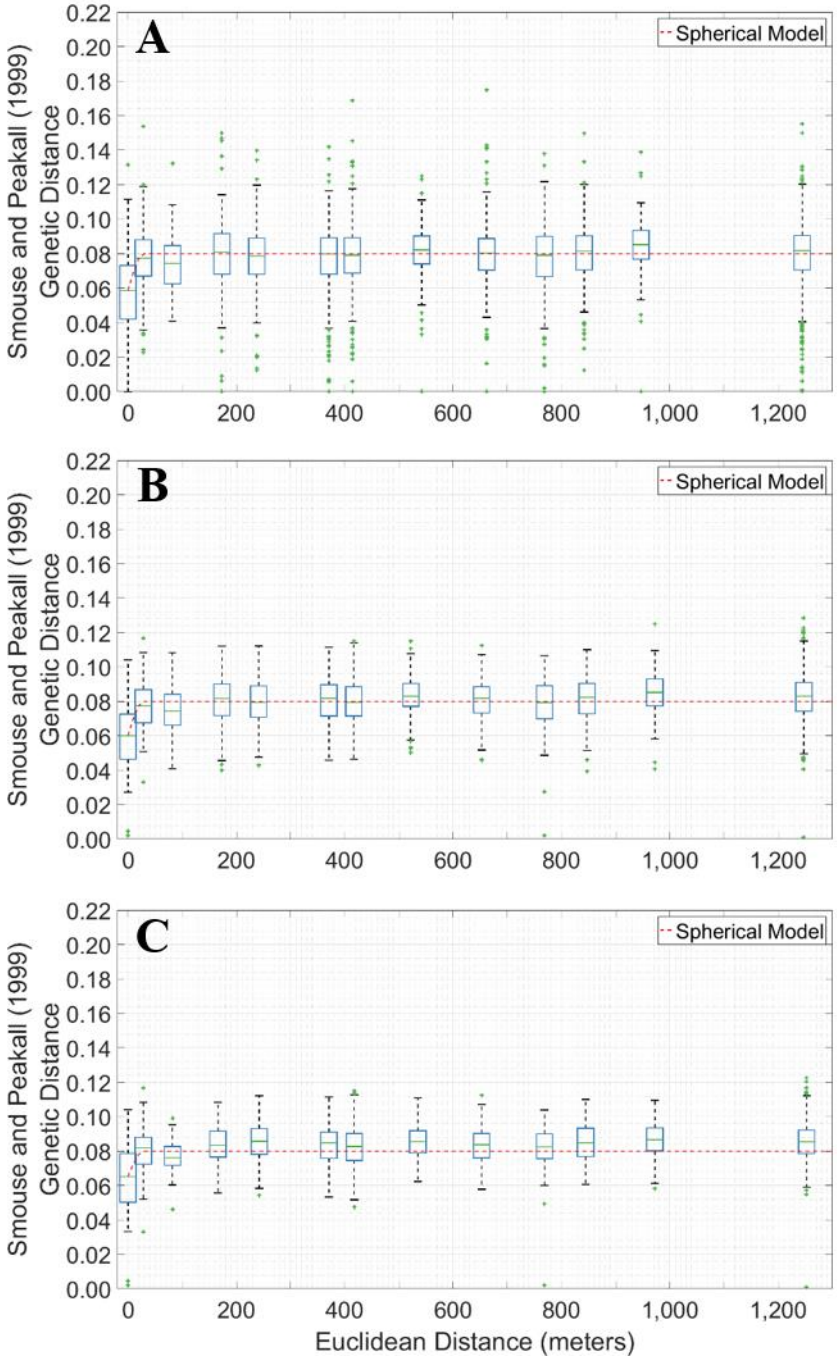


Figure 4.4: Box plot semivariograms of genetic distance for El Chaperno generated using Level 1 filtering. Semivariogram using A) all paired individuals with at least 1 loci in common, B) all pairs with at least 287 loci in common, and C) all pairs with 1,000 common loci. Best fit spherical models (red dashed line) have the same range (28m) and sills (0.08) across all three panels.

A spherical model was also fit to the genetic distance data of El Carrizal (Figure 4.5). Again, all three datasets (panels A, B, and C using minimum common loci thresholds of 1, 250, and 1,000, respectively) have the same median range (88 meters; minimum and maximum bin distance are 59 and 97 meters, respectively) and sill (0.07). Again, there is little difference between the three semivariograms, and panel A (semivariogram generated with the largest amount of paired (7,260) *T. dimidiata*) has a larger inter-whisker distances and more outliers than panels B and C. With that being said, the median range of spatial autocorrelation is constant 88 meters ([59, 97] meters) regardless of the amount of data used to generate the semivariogram. Interestingly, the midspread of binned genetic distances appears to be similar for both El Chaperno (Figure 4.4) and El Carrizal (Figure 4.5).

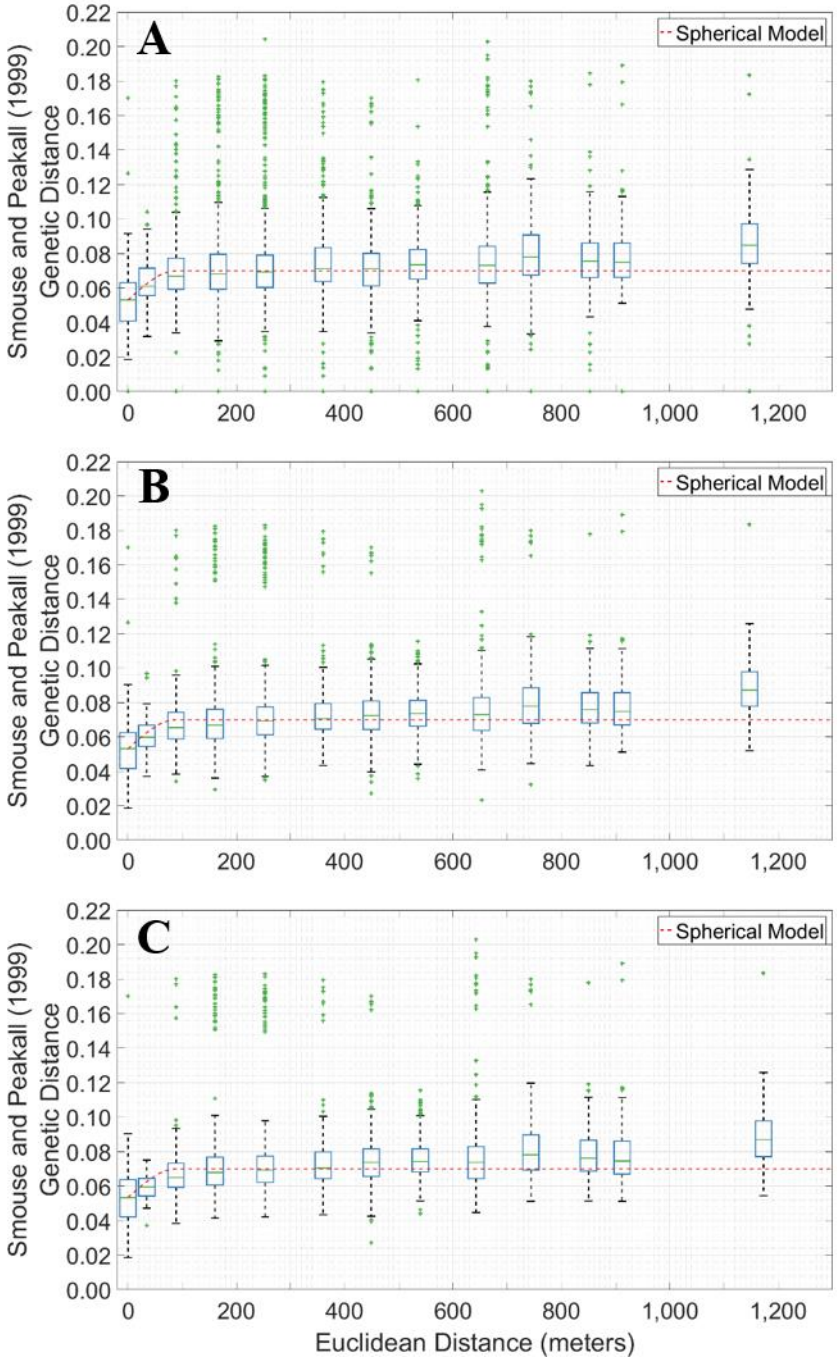


Figure 4.5: Box plot semivariograms of genetic distance for El Carrizal generated using Level 1 filtering. Semivariogram using A) all paired individuals with at least 1 loci in common, B) all pairs with at least 250 loci in common, and C) all pairs with 1,000 common loci. Best fit spherical models (red dashed line) have the same range (88m) and sills (0.07) across all three panels.

We also generated semivariograms for El Chaperno and El Carrizal using only the nymphs to characterize the spatial autocorrelation of *T. dimidiata* of *T. dimidiata* that are not capable of flight. However, these results are not shown because the semivariograms and range of spatial autocorrelation were nearly identical to the semivariograms plotted in Figure 4.3 and 4.4, respectively.

To test for sex-biased dispersal, we created separate semivariograms for male and female *T. dimidiata*. Again, there was little difference between sex in terms of the range of spatial autocorrelation; however, the sex-based semivariograms were noisier, most likely due to the smaller sample sizes.

4.3.2 Results Level 2 Filtering

The Level 2 filtering enabled us to use GenAlEx 6.503 to calculate the relatedness of Lynch and Ritland (1999) and compare the results to the genetic distance of Smouse and Peakall's (1999). The semivariograms for El Chaperno (Figure 4.6) characterize relatedness (panel A) and genetic distance (panel B) for the Level 2 dataset comprised of 73 specimens and 287 SNPs. The best-fit spherical model (red dashed line) shows a median range of spatial autocorrelation of 28 meters ([9, 48] meters) for both relatedness and genetic distance. This is the same range identified in the spherical models of Figure 4.4. The midspread for within-homestead semivariance (first box plot of Figure 4.6A) is much larger than any other bins of relatedness. This midspread pattern was not observed in Figure 4.6B for the within-homestead genetic distance.

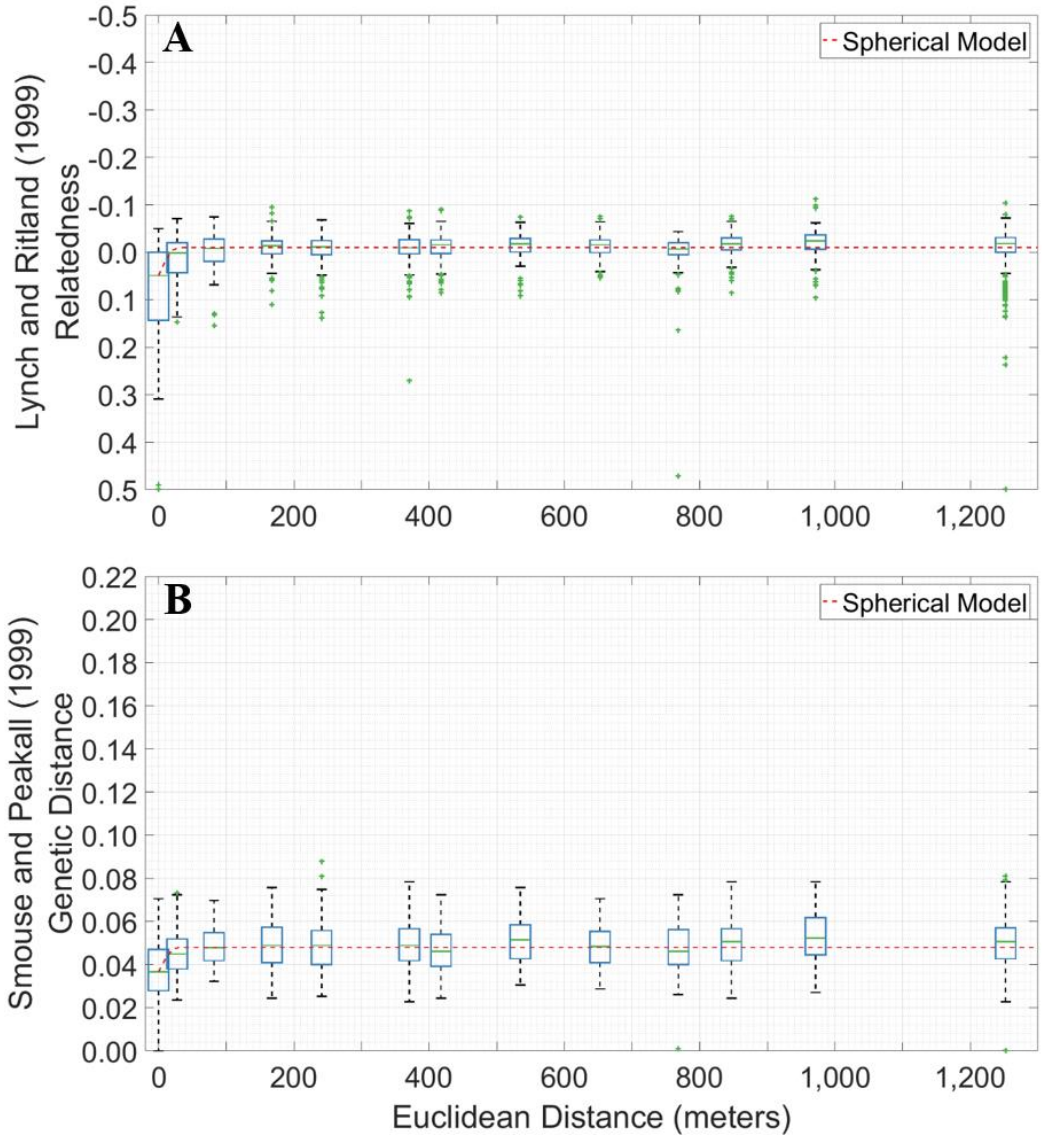


Figure 4.6: Box plot semivariograms for El Chaperno characterize the A) relatedness of Lynch and Ritland (1999) and B) genetic distance of Smouse and Peakall (1999) using a SNP dataset with 73 specimens and 287 loci. The best-fit spherical models (red dashed line) have the same range (28m); and the sills are -0.01 and 0.05, respectively. The vertical-axis of panel A was flipped so that more similar bugs (high relatedness) have positive values and those with low relatedness have negative values.

Figure 4.7 displays the semivariograms for genetic distance (panel A) and relatedness (B) generated using 97 specimens and 250 SNPs of El Carrizal. The best-fit

spherical model (dashed line) shows a median range of spatial autocorrelation of 160 meters ([104, 200] meters), which is nearly twice the median range (88m) of using the Level 1 data (Figure 4.5). Similar to the town of El Chaperno (Figure 4.6A), the midspread of semivariance for within-homestead relatedness is much larger than other bins (Figure 4.7A).

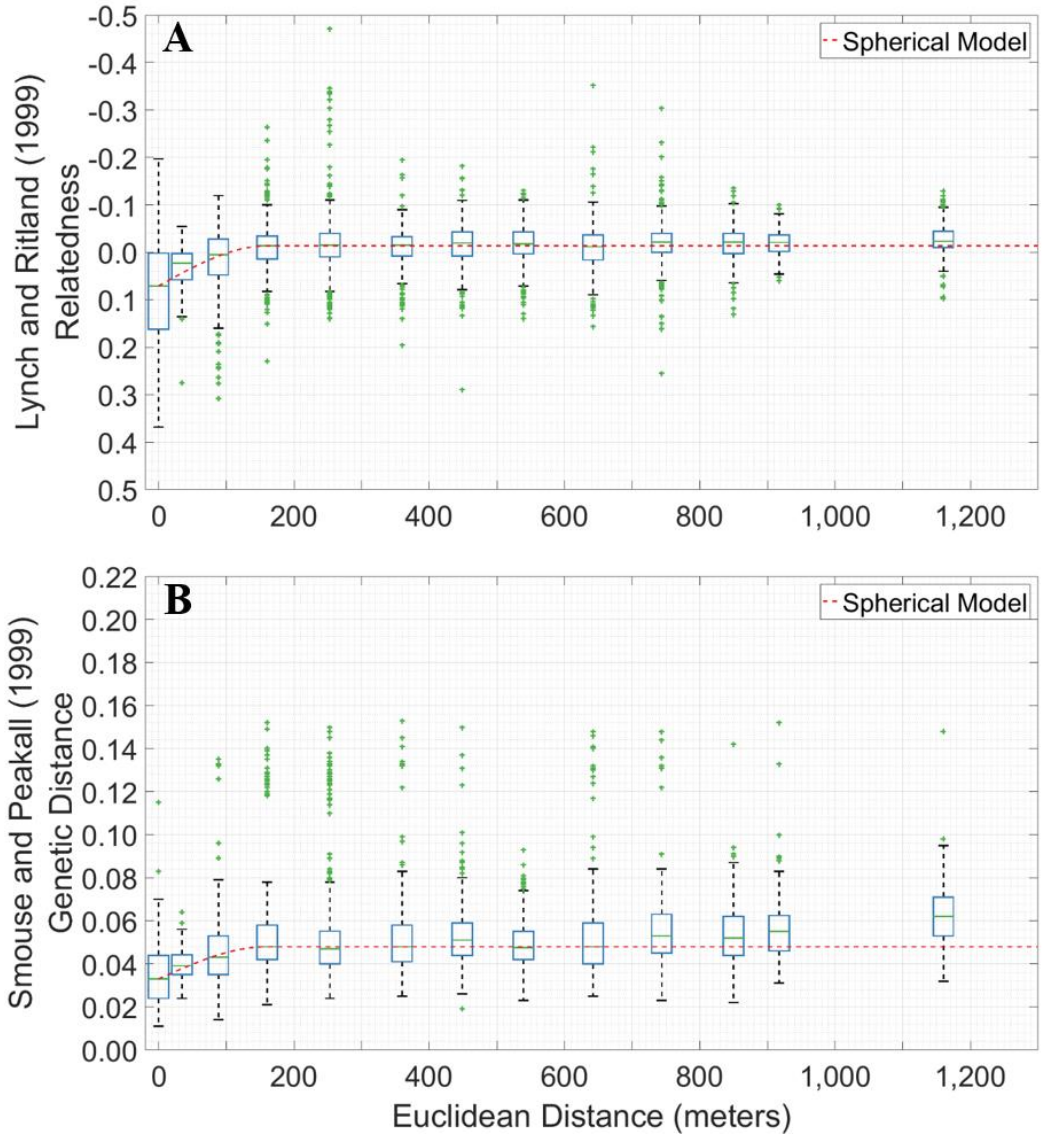


Figure 4.7: Box plot semivariograms for El Carrizal characterize the A) relatedness of Lynch and Ritland (1999) and B) genetic distance of Smouse and Peakall (1999) using SNP data with 97 specimens and 250 loci. The best fit spherical models (red dashed line) have the same range (160m); and the sills are -0.01 and 0.05, respectively. The vertical-axis of panel A was flipped such that more similar bugs (high relatedness) have positive values and those with low relatedness have negative values.

4.3.3 Results – Risk Maps for El Chaperno and El Carrizal

If one assumes that every infested homestead is a potential source of *T. dimidiata*, then a risk map can be created to highlight the parts of town that have higher risk of infestation due to proximity to an infested neighbor. For El Chaperno, the *T. dimidiata* median range of spatial autocorrelation was consistently 28 meters regardless of the (1) level of filtering, and (2) metric used for genetic semivariance (relatedness or genetic distance). The range of spatial autocorrelation was plotted as a red circle around every infested homestead in El Chaperno, with overlap represented in deeper shades of red (Figure 4.8); the deepest shade of red (i.e., maximum overlap) corresponds to 5 overlapping ranges. Infested homesteads are plotted with an x, and non-infested homesteads with a +. Given the relatively small median range of spatial autocorrelation (28 meters) in El Chaperno, we see that 68% (86/126) of non-infested homesteads fall outside the range of risk of infestation; and only 6% (8/126) of the non-infested homesteads lie within in the range of multiple infested homesteads.

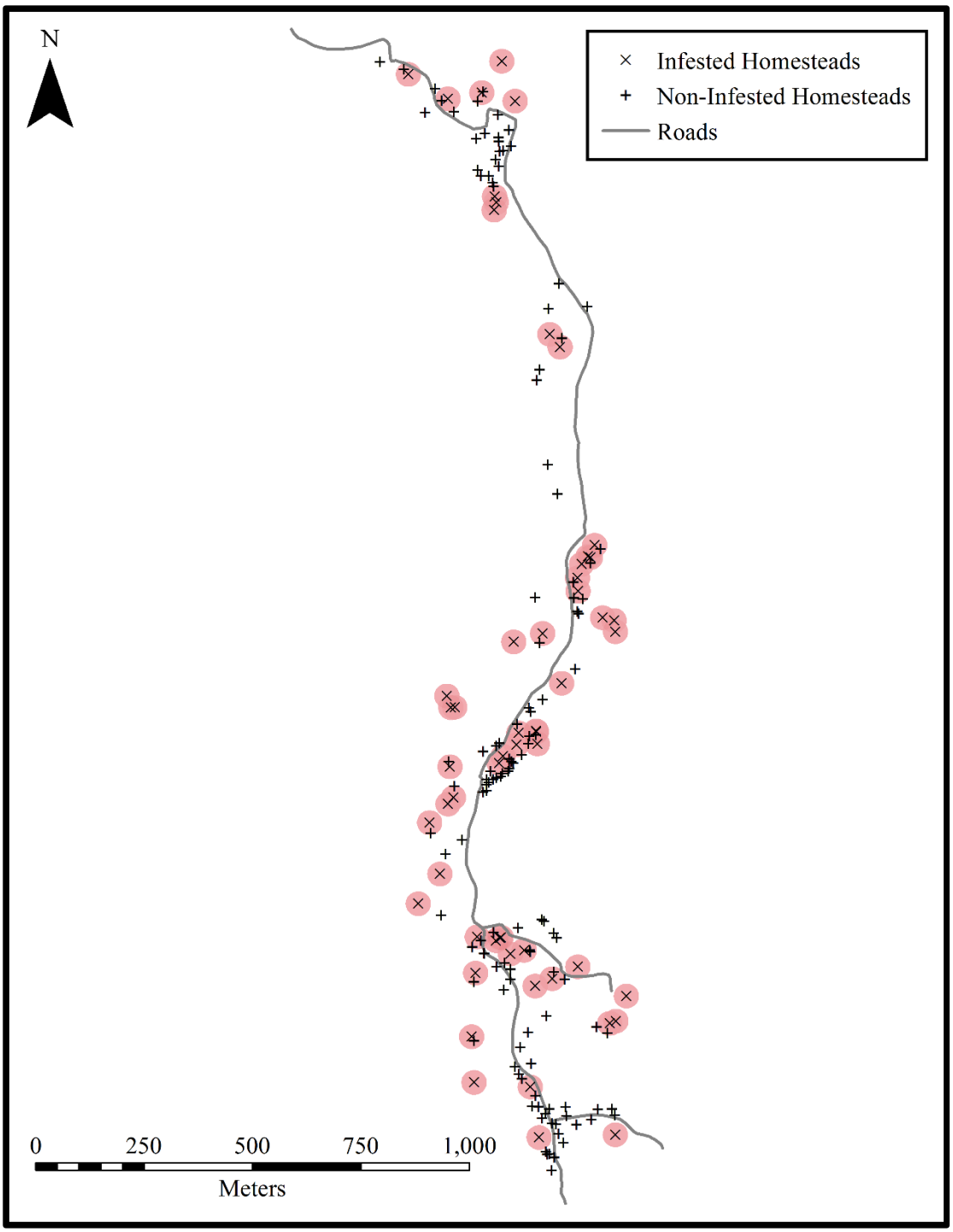


Figure 4.8: Risk map for infestation of *T. dimidiata* in El Chaperno. Red circles show the range of spatial autocorrelation for each homestead. Overlap is represented in darker shades of red using the range of 28 meters. The deepest shade of red (i.e., maximum overlap) corresponds to 5 overlapping ranges. Infested homesteads are plotted as an x and non-infested homesteads as a +. Every infested homestead is treated as a possible source of *T. dimidiata*.

For El Carrizal, the Level 1 filtered dataset had a median range of spatial autocorrelation of 88 meters while the Level 2 filtered dataset had a median range of 160 meters. We use these two ranges of autocorrelation to create the risk maps of Figure 4.9 for El Carrizal. Figure 4.9A uses a range of 88 meters and shows that 23% (18/77) of the non-infested homesteads lie outside the range of an infested homestead. The maximum number of overlapping ranges is equal to 10; Figure 4.9B has a maximum overlap of 18, and uses a range of 160 meters. Only 8% (6/77) of the non-infested homesteads lie outside the range of an infested homestead; and all but one of these homesteads is located in the northwest part of the village. Note: If one uses the less conservative estimate of spatial autocorrelation (i.e., 88 meters), then 70% (54/77) of the non-infested homesteads fall within the range of multiple infested homesteads; this increases to 92% (71/77) when the range increases to 160 meters. Both risk maps show an area located between the two eastern spur roads of El Carrizal to be high risk (darker overlapping areas of red in Figures 4.9A and 4.9B).

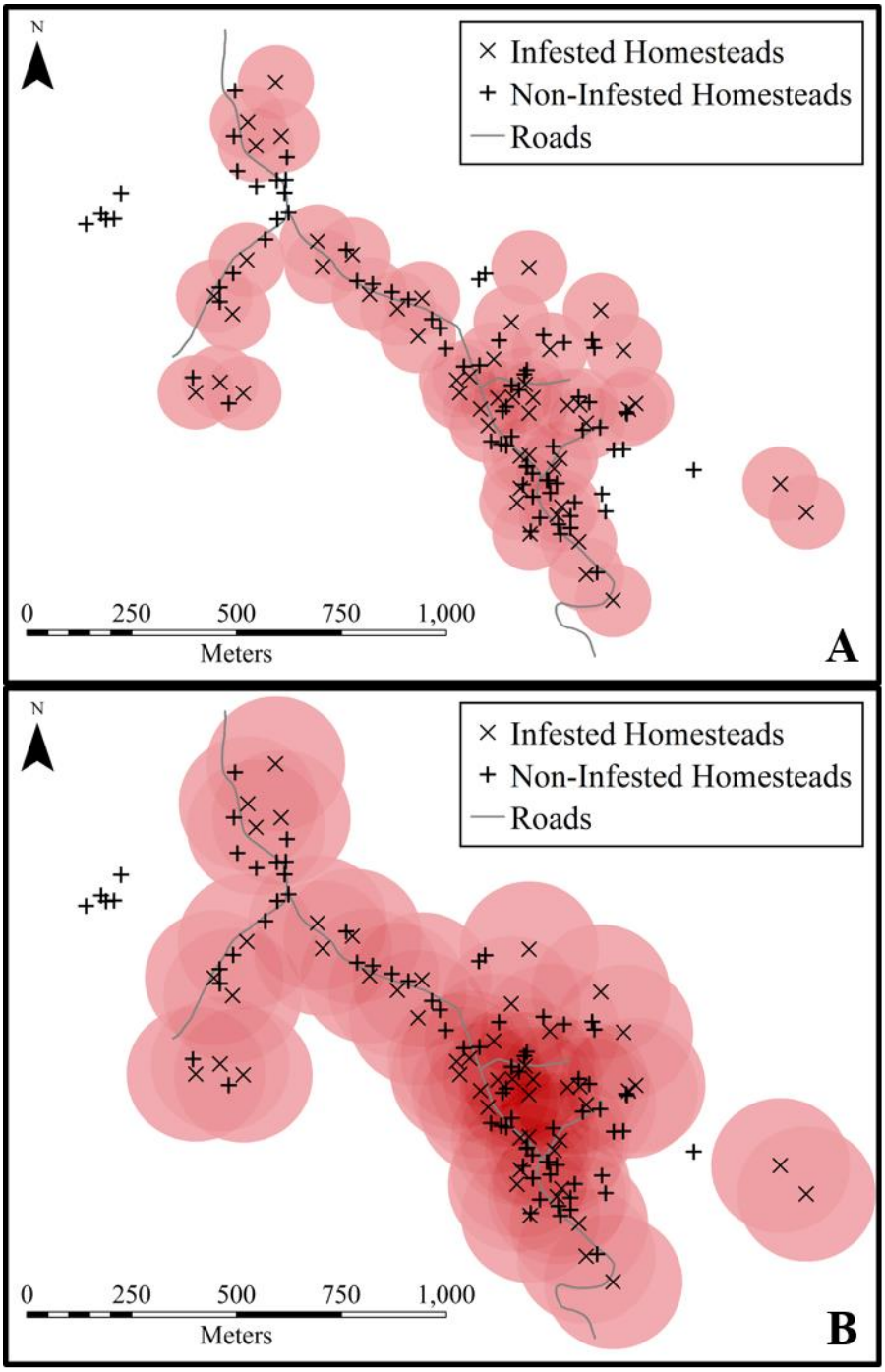


Figure 4.9: Risk maps for infestation of *T. dimidiata* in El Carrizal. Red circles show the range of spatial autocorrelation for each homestead. Overlap is represented in darker shades of red using the range of A) 88 meters and B) 160 meters. The deepest shade of red (i.e., maximum overlap) corresponds to A) 10 and B) 18 overlapping ranges. Infested homesteads are plotted as an x and non-infested homesteads as a +. Every infested homestead is treated as a possible source of *T. dimidiata*.

4.4 Discussion

In this work, we first used the genetic distance of Smouse and Peakall (1999) to determine the range of spatial autocorrelation as a surrogate for the movement of the Chagas vector *T. dimidiata* in two towns in Guatemala. For each town, the original datasets had ~100 collected specimens and around 2,000 SNPs. Given the time and money that goes into collecting the *T. dimidiata*, extracting their DNA, and then sequencing each bug's genome, we thought it prudent to explore how filtering the data might impact the semivariograms. We modified genetic distance of Smouse and Peakall (1999) to use only the loci that are common to both specimens and then standardized the genetic distance between each pair. Next, we performed a semivariogram analysis on this modified genetic distance using several thresholds for the number of loci common to both specimens. For both El Chaperno and El Carrizal, the semivariograms identify the same median range of spatial autocorrelation, regardless of whether all paired specimens were included in the analysis or whether only specimens with at least 1,000 loci in common were used. This resiliency to filtering may result from the majority of our specimen pairs having at least 1,000 loci in common. Also, the resiliency is due in part to our decision to use box plots to represent binned genetic semivariance.

Next we performed Level 2 filtering for both the datasets for both towns reducing the dataset size. We reanalyzed these Level 2 filtered datasets using semivariograms analysis and the normalized Smouse and Peakall (1999) genetic distance and Lynch and Ritland (1999) relatedness, a frequency based method. The ranges of spatial autocorrelation were independent of both measures of genetic similarity.

For El Chaperno, the median range of spatial autocorrelation remained at 28 meters for the reduced dataset regardless of which measure of similarity is selected for the genetic semivariance analysis. El Carrizal, on the other hand, saw an increase in the median range of autocorrelation from 88 meters in the original dataset and Level 1 filtering to 160 meters when using the Level 2 filtering. This difference in range is concerning since it leaves us uncertain as to which dataset is better for measuring the range of spatial autocorrelation. The Level 2 filtering of the El Carrizal dataset has 80% (97/121) of the pre-filtered specimens and only 11% (250/2,265) of the pre-filtered SNPs. We would like to err toward not filtering the data, or at most using low thresholds for Level 1 filtering, due to the big data concept that more data are better and perfect data are not necessary (Mayer-Schönberger and Cukier, 2014). Also, when we performed Level 1 filtering, we did not observe a change in the range of spatial autocorrelation.

Finally, we used the ranges of spatial autocorrelation derived from the semivariogram analyses to create risk maps of *T. dimidiata* infestation. Given that we cannot find any empirical study demonstrating the distances that *T. dimidiata* move in a natural environment, we used the range of genetic structure as a surrogate for vector movement at the village scale. In El Chaperno, the median range of spatial autocorrelation was only 28 meters; and thus, most of the non-infested homesteads fell outside the range of an infested homestead and would be less likely to be infested by an infested homestead. However, the El Chaperno semivariograms show that *T. dimidiata* within the same homestead are more similar (i.e., closely related) than *T. dimidiata* outside the homestead. This relationship could be explained by nymphs having the same

parent(s). Therefore, it is possible that the El Chaperno population is panmictic. Thus, the distance of a non-infested homestead from an infested homestead will not affect the risk of infestation. For El Carrizal, the majority of non-infested homesteads lie within the range of multiple infested homesteads, regardless of whether the median range of spatial autocorrelation is 88 meters or 160 meters. Therefore, the *T. dimidiata* population of El Carrizal are not panmictic and the non-infested homesteads within the range of an infested homestead have a higher risk of infestation from nearby homesteads. That being said, the reported median ranges of spatial autocorrelation are summary statistics and thus the true range of spatial autocorrelation can lie anywhere between the reported minimum and maximum bin range. This difference in spatial autocorrelation between the two towns may be attributed to their land use/cover. El Chaperno is less open and has more forest due to conservation efforts than El Carrizal. Therefore, *T. dimidiata* may need to travel further in El Carrizal to find suitable shelter. Also, if *T. dimidiata* is sylvatic in Jutiapa, Guatemala as postulated by Hernández et al. (2006), then the nearby forests in El Chaperno may serve as a source of *T. dimidiata* infestation; whereas, for El Carrizal, particularly in the part of town most at risk (Figure 4.9), the primary sources of infestation may be other homesteads.

When using the relatedness of Lynch and Ritland (1999) as a metric of similarity, both El Chaperno (Figure 4.5A) and El Carrizal (Figure 4.6A) exhibited a within-homestead midspread that was larger than the midspreads corresponding to greater spatial ranges. This may be indicative of having a large number of *T. dimidiata* families with individual households similar to that documented by Melgar et al. (2007) in

neighboring Santa Rosa, Guatemala. It is interesting that despite the large within-homestead diversity for El Chaperno and El Carrizal, both towns still exhibited within village spatial autocorrelation using SNPs.

While using semivariogram analysis on genetic markers has not been offered as an alternative to the correlograms of Smouse and Peakall (1999), semivariograms are not a foreign concept in genetics. Elhaik et al. (2013) plot what is essentially a semivariogram and determine a range of spatial autocorrelation for humans using loess distribution fitting and SNPs; and Bradburd et al. (2013) do something similar for the ancestors of corn *Teosinte zea mays mexicana* and *Teosinte zea mays parviglumis*. In addition, the methodology of Smouse and Peakall has been used to study the range of spatial autocorrelation of mosquito disease vectors (Foley et al., 2004; Rašić et al., 2015) and the principal Chagas disease vector *Triatoma infestans* (Pérez de Rosas et al., 2013). However, one advantage of fitting the semivariogram data with a monotonic model is that it enables the use of estimation methods that can minimize the error variance in an unbiased way (Isaaks and Srivastava, 1989). The nugget, range, sill, and spherical model can enable one to estimate/interpolate parameter fields (e.g., map genetic distance/relatedness) using multiple data and the estimates of error variance may be used to improve risk maps. For example, the semivariogram model of genetic distance could be paired with another variable such homestead attractiveness (e.g., house risk level, sources of light) to perform co-kriging that can be leverage two variables to create potentially better risk maps.

To our knowledge this is the first study that uses the range of spatial autocorrelation to plot the risk of homestead infestation. Our risk maps enable stakeholders to assess the parts of town that are in most need of intervention and provide clues as to whether the source of infestation is sylvatic or domestic/peridomestic.

4.5 Conclusion

In this study, we demonstrate that Level 2 filtering of data may not be necessary when using SNPs to identify the range of spatial autocorrelation. In addition, we observed spatial autocorrelation among *T. dimidiata* at relatively small distances within towns implying that they are likely moving between neighboring homesteads, at least in El Carrizal. Finally, we were able map the risk to non-infested homesteads using the range of spatial autocorrelation derived from semivariogram analysis. Since there is little empirical evidence for how far *T. dimidiata* might travel in the field, we feel that semivariogram analysis using metrics of genetic similarity may provide a sufficient model for vector movement or disease transmission.

4.6 Supplementary Figures

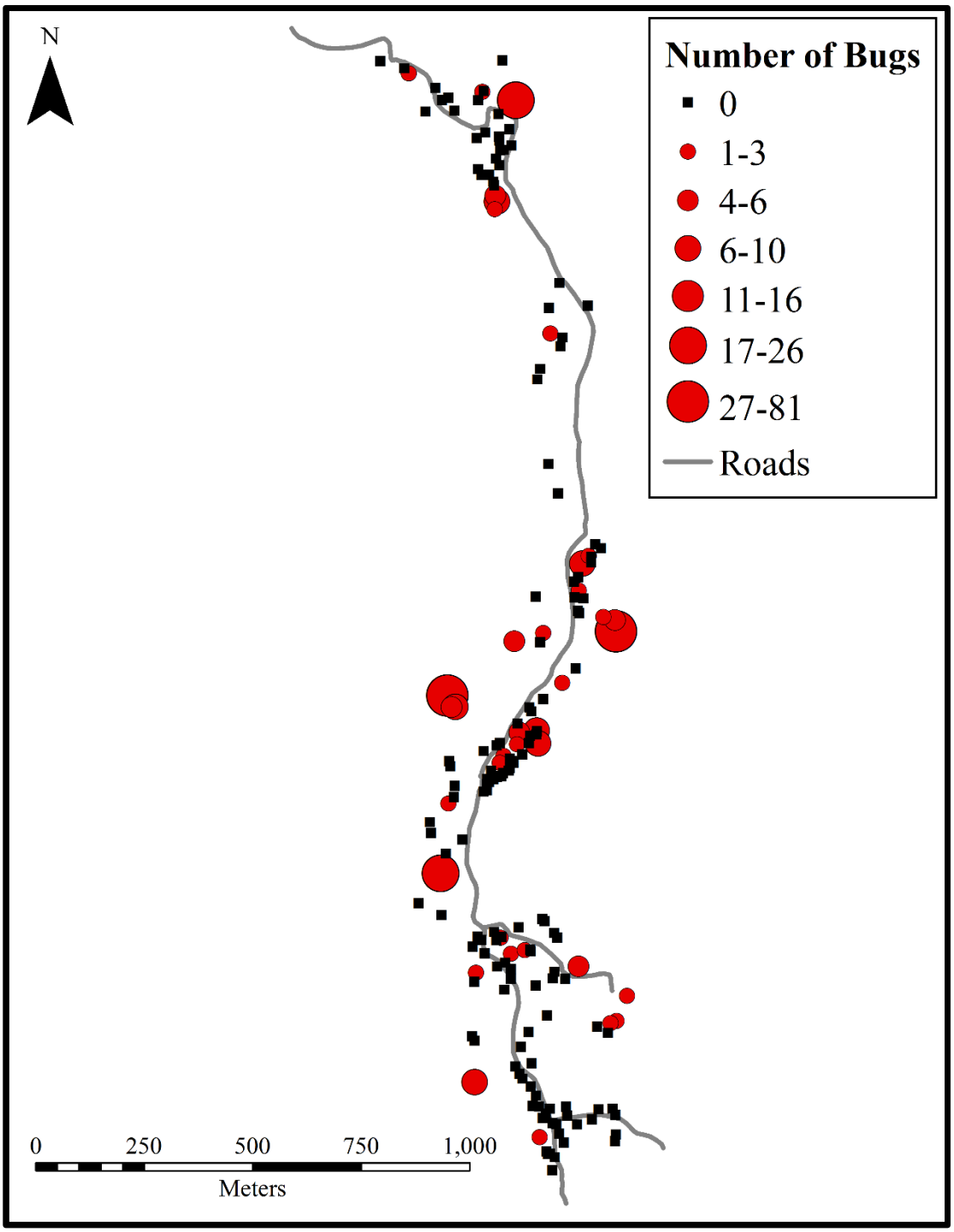


Figure 4.S1: The black squares represent homesteads where no *T. dimidiata* were found during the entomologic search in El Chaperno. The red circles are proportional to the number of *T. dimidiata* collected for a given homestead ranging from 1 to 81.

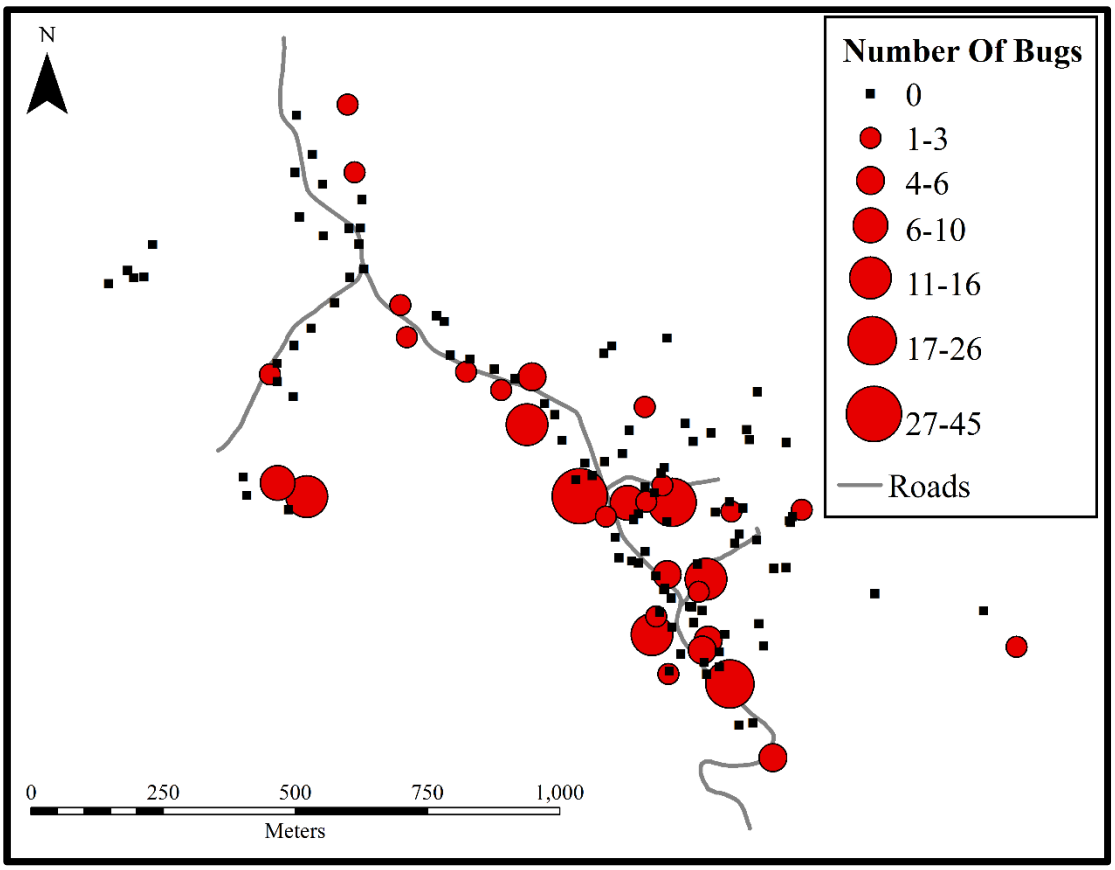


Figure 4.S2: The black squares represent homesteads where no *T. dimidiata* were found during the entomologic search in El Chaperno. The red circles are proportional to the number of *T. dimidiata* collected for a given homestead ranging from 1 to 45.

CHAPTER 5: GEOSPATIAL AND TEMPORAL ANALYSIS OF THYROID CANCER INCIDENCE IN A RURAL POPULATION

5.1 Introduction

Thyroid cancer incidence is increasing at an annual rate of 3–5%, resulting in the rate tripling over the past 30 years in the United States as well as in other countries (Curado et al., 2007; Kilfoy et al., 2009; Jemal et al., 2011; Morris et al., 2013; Pellegriti et al., 2013). In the United States, the number of cases has risen from 4.3 cases per 100,000 in 1980 to 12.9 cases per 100,000 individuals in 2008. Mortality rates have slightly increased (+0.8% annual percent change [APC]) (Enewold et al., 2009; Cramer et al., 2010; NCI, 2012). A recent study noted a disproportional increase in women (Edwards et al., 2006). The basis for the increase in thyroid cancer incidence is not known. Some studies suggest enhanced diagnostic scrutiny and better detection of subclinical cancers result in widespread over diagnosis and thus not a true increase in incidence (Davies and Welch, 2006; Ross, 2006; Grodski et al., 2008; Enewold et al., 2009; Hall et al., 2009; Yu et al., 2010; Morris et al., 2013; Reitzel et al., 2014). Other studies note that an increase in both large tumors and microcarcinomas as well as a change in relative frequencies of histological types implicate other contributing factors (Chen et al.; 2005; Kilfoy et al., 2009; Pazaitou-Panayiotou et al., 2013; Ward et al., 2010; Aschebrook-Kilfoy et al., 2013). Of note, recent reports of aggressive, metastatic microcarcinomas of the thyroid that correlate with the risk of second cancers (Kim et al., 2013) suggest that microcarcinomas once considered subclinical might emerge as

important new healthcare concerns and reflect an important dimension of the increase in thyroid cancer incidence.

Environmental and demographic factors may be critical determinants in the increase in thyroid cancer incidence (Leux and Guénel, 2010; Morris and Myssiorek, 2010; Li et al., 2013; Pellegriti et al., 2013). A recognized risk factor for thyroid cancer is ionizing radiation exposure through medical procedures, including x-rays, as well as radioactive fallout (Richardson, 2009; Wartofsky, 2010; NCI, 2013). A study of the overall geographic distribution of thyroid cancer in the United States revealed a higher incidence in areas proximate to nuclear power reactors (Mangano, 2009). High levels of nitrate in public drinking water supplies have been linked to increased thyroid cancer incidence (Ward et al., 2010), and environmental endocrine disruptors including polyhalogenated aromatic hydrocarbons (PHAHs), notably polybrominated diphenyl ethers (PBDEs) and organochlorine insecticides, are postulated factors (Grimalt et al., 1994; Zhang et al., 2008; Zhu et al., 2009; Leux and Guénel, 2010). Leux and Guénel (2010) noted that many environmental chemicals interfere with thyroid function and increase the risk of goiters, nodules, and possibly neoplasia. Additional known risk factors include family history, sex, and age (Pellegriti et al., 2013). Socioeconomic factors (SES) may also indicate that access to healthcare affects incidence (Sprague et al., 2008; Morris et al., 2013). Thus, novel analyses are needed to elucidate both incidence and contributing factors.

With the capability to visualize, analyze, interpret, and map geo-located data, the field of geostatistics, notably the geographic information system (GIS) tool, has emerged

as a powerful geospatial technology that is gaining prominence in healthcare applications (Musa et al., 2013). GIS-based cancer mortality maps produced by the National Cancer Institute and Centers for Disease Control and Prevention (CDC) are widely used by public health officials to guide disease surveillance and control activities throughout the United States (Shaw, 2012). Beyond traditional GIS mapping capabilities, more sophisticated spatial statistical analyses have been utilized to identify spatial disease clusters (i.e., nonrandom spatial distributions of disease cases, incidence, or prevalence), map and monitor disease patterns and trends over time and space, and assess the impact of ecological and SES on the spatial distribution of diseases. Although there are still many technical (e.g., knowledgeable users, data quality control) and organizational (e.g., access and sharing) barriers to the wide-scale adoption of geospatial technologies in the healthcare sector (Boulos et al., 2011), recent advances in the understanding of disease dynamics, healthcare management has demonstrated the power of geospatial technologies to identify new drivers of public health concerns and advance the field of public health research. The present objective was to examine the characteristics of thyroid cancer incidence and determine the geospatial distribution in the state of Vermont, United States.

This study postulated that geospatial analyses would reveal important risk factors of thyroid cancer incidence in a rural population that would provide the framework for investigation of potential drivers of disease patterns. It was determined that the characteristics of thyroid cancer incidence, including significant nonrandom clusters, are most likely due to environmental and lifestyle factors. Spatial statistical analyses

revealed that the overall distribution of thyroid cancer incidence and higher APC in these rural regions provide the framework for evaluating demographic and environmental drivers that may contribute to thyroid cancer incidence.

5.2 Methods

5.2.1 Data Sources

Data on thyroid cancer (1994–2007) were obtained from the Vermont Department of Health, and U.S. data on thyroid cancer were obtained from the National Cancer Institute at the United States National Institutes of Health Surveillance, Epidemiology, and End Results (SEER) Program. State mandated data collection began in 1994 and included year of initial diagnosis, age at diagnosis, sex, primary site of disease at diagnosis, histology code, histological grade, behavior code, size of tumor, postal code at diagnosis, year last contacted, vital status, and death place code. Data exchange agreements between neighboring states minimize underreporting in border counties. Data pertaining to residents of neighboring states were not included in this study. Thyroid cancers were grouped based on histology codes, including papillary (8050, 8052, 8130, 8260, 8340–8344, 8450, 8452), follicular (8290, 8330–8332, 8335), medullary (8345, 8346), anaplastic (8021), and other/indeterminate/not specified (8012, 8032, 8046, 8070, 8140, 8190, 8335, 8337, 8347, 8350) (Fritz et al., 2000).

Population data, used to calculate incidence, were obtained from the Vermont Department of Health's intercensal population estimates (VPE, 2013). The Vermont population in 1994, 2000, and 2007 was 585,544, 608,827, and 623,481, respectively. Incidence and mortality rates were age adjusted to the U.S. 2000 Standard Population (as

per SEER practice (SPAA, 2013)) and normalized per 100,000 person-years (Breslow and Day, 1987). For the geospatial analyses, zip code boundaries were downloaded from the U.S. Census Bureau, and all map layers projected to the Vermont State Plane Coordinate System North American Datum 1983. Information regarding SES was obtained from the 2000 U.S. Census variables, which included percent of the population by age, length of household occupancy, median household income, and post–high school education. The percent of the population with health insurance was obtained from Vermont Household Health Insurance Survey, Department of Financial Regulation, State of Vermont (VDB, 2010).

The study was approved by the Institutional Review Board of the University of Vermont Committee on Human Research and the Vermont Cancer Center.

5.2.2 Statistical Analyses

Age-adjusted incidence (also known as age-standardized rate) was calculated as described by Boyle and Parkin (1991). This method adjusts each age group's contribution to the overall population incidence so that incidence is based on the same age structure. Proportional age-adjusted incidence was also calculated that quantified the contribution of various age strata (e.g., 30–39 year olds) to the age-adjusted incidence. The proportional age-adjusted incidence for each age group of interest was calculated by summing the product of the crude incidence and the respective frequency of the standard population for each single year of age within the age group of interest (e.g., for age group 30–39, sum product for ages 30, 31,.,39). The standard errors of the overall age-adjusted

incidence and proportional age-adjusted incidence were calculated using the Poisson approximation method (Boyle and Parkin, 1991).

The estimated APC is a summary statistic used to measure trends over time by taking the average rate of change in incidence over several years (Breslow and Day, 1987). The values were calculated by fitting a regression line to the natural logarithm of the incidence using the calendar year as the independent variable (Ries et al., 2000). The estimated APC is equal to $100\times(e^{slope}-1)$. The statistical significance ($p < 0.05$) of the linear slope was GEOSPATIAL ANALYSIS OF THYROID CANCER INCIDENCE 813 compared to zero, and confidence intervals (CI) were calculated from the standard error of the slope. The time period was split into 1994–2000 and 2001–2007 in order to compare trends from the first half of the study period to the second half of the study period, and the estimated APC was calculated for incidence for time periods 1994–2007, 1994–2000, and 2001–2007 for males, females, and both sexes combined, respectively.

The age-adjusted incidence for each county was compared to the overall age-adjusted incidence of Vermont by creating a standardized rate ratio (SRR) (Boyle and Parkin, 1991). To determine whether national incidence was significantly different from the incidence in Vermont, the confidence interval of each SRR was approximated as described by Smith (1987). There was a significant difference between incidences if the confidence interval did not include SRR 1.0, indicating equal incidence. All statistical analysis, including estimation of the APC and age-adjusted incidence, were performed using Excel 2013 (Microsoft Corp., Redmond, WA), JMP® Pro v10.0.0 (SAS Institute, Cary, NC), ArcGIS® v10.2 (esri®, Redlands, CA), and MATLAB® 2014a (MathWorks,

Natick, MA). All incidence data were age adjusted to the U.S. 2000 Standard Population baseline.

5.2.3 Trend Analyses

Significant ($p < 0.05$) annual trends in the age-adjusted incidence for Vermont females, males, and the total population of Vermont were performed using the Ljung-Box Q analysis in JMP® Pro. The same analysis was used to test for significant annual trends for sex-specific proportional age-adjusted incidence for three age groups (<30 years old, 30–59 years old, and >59 years old). In addition, the study tested for significant proportional annual trends in thyroid cancer tumors ≤1.0 cm, 1.1–2.0 cm, and >2.0 cm in size.

5.2.4 Socioeconomic Analyses

Socioeconomic data from the 2000 U.S. census was analyzed at both the zip code and county scale. As a result, the study used both logistic and linear regression analysis to test for significance between the annual age-adjusted incidence of thyroid cancer and with socioeconomic variables related to income, education, length of residency, and access to healthcare at both the zip code and county scales.

5.2.5 Geospatial Analyses

ArcGIS® v10.2 software was used to perform geospatial analyses and map visualization. The number of thyroid cancer cases in each zip code was mapped to show their spatial locations in Vermont. The cases were normalized per 100,000 to the population for each zip code based on the Vermont Department of Health's intercensal population estimates. Due to the nature of zip code data and inconsistencies between the

2010 census zip code boundaries and zip code census data, some zip codes were combined. For two zip codes with recorded thyroid cancer cases and no zip code associated with those zip codes, the cases were added to the zip code that shared the greatest area of the zip code. Calculated normalized incidence was mapped to illustrate the effect of population on incidence distribution. The cases and incidence distributions for each image were classified based on Jenks Natural Breaks. This method of classification partitions data into the specified number of classes based on natural groups or clusters of data values.

Spatial statistics use inferential statistics to test a null hypothesis that the features are randomly distributed in space. In this case, the feature tested is the average annual age-adjusted incidence of thyroid cancer for each zip code. A *p*-value and *z*-score are computed to determine the statistical significance of observed spatial patterns. A *p*-value calculates the probability that the observed patterns were due to random chance; statistically significant clustering is evident at a *p*-value of < 0.05. The *z*-score is the standard deviation of the result, which is calculated using the logistic regression model. Very high (>1.96) and very low (<-1.96) *z*-scores correspond to low *p*-values (0.05) and indicate the spatial distribution of age-adjusted incidence is not random.

The Getis-Ord Gi* statistic was calculated for each age-adjusted incidence in a weighted set of zip codes using the Hot Spot Analyses tool. Although a particular zip code may have high incidence, Hot Spot Analysis identifies those zip codes with statistically higher incidence of cancer cases, that is, those zip codes that have significantly higher values than can be expected by chance. The Gi* local statistic

identifies individual members (zip codes) of local clusters by looking at each target zip code compared to neighboring zip codes within a specified ‘‘Zone of Indifference.’’ This distance metric calculation enables each age-adjusted incidence within the critical distance to be equally weighted and the age-adjusted incidence of each zip code outside the specified distance with diminishing weights as distance increases. A significant Hot Spot ($p < 0.05$) is identified if the sum of a zip code’s value and the values of all its neighboring zip codes is proportionally higher than expected when compared to the sum of all zip codes in the state. Likewise, a zip code is a significant Cold Spot ($p < 0.05$) if the sum of its value and the values of its neighboring zip codes is proportionally lower than expected.

The Hot Spot Analysis tool requires the input of a specified distance, which determines the scale of the analysis. This value was calculated using the ‘‘Calculate Distance Band from Neighbor Count’’ geoprocessing tool to determine the distance between every zip code and, in this work, its eight nearest neighbors, and returns the minimum, maximum, and average distance. The minimum value is the distance (in meters) one would travel away from a zip code to ensure that at least one zip code has eight neighbors, the maximum value is the distance one would travel away from a zip code to ensure that each zip code has at least eight neighbors, and the average value is the average distance between each zip code and its eight nearest neighbors. Maximum and average distances were chosen to test for clustering at multiple scales across the state (Supplementary Figure 5.S1; Supplementary Data are available online at www.liebertpub.com/thy).

5.3 Results

5.3.1 Incidence Trends

The age-adjusted thyroid cancer incidence in Vermont rose significantly 2.4-fold from 5.3 in 1994 to 12.6 in 2007 with a significant estimated APC of 8.3% [CI 5.7–11.0] compared to the national estimated APC of 5.7% [CI 5.2–6.3] (Table 5.1 and Supplementary Figure 5.S2). Although the overall average annual aged-adjusted incidence for females in Vermont was similar to that in the United States (11.8 and 12.3, respectively), the estimated APC was higher at 9.9 for Vermont and 5.9 for the United States. For males, both the average annual age-adjusted incidence and the estimated APC were similar to national trends, with both significantly increased over time (Table 5.1 and Supplementary Figure 5.S1). The thyroid cancer age-adjusted incidence in Vermont (8.0 per 100,000) was comparable to the national incidence (8.4 per 100,000). Also, the overall mortality rate was 0.5 per 100,000 for males and females, which is similar to the national rate (NCI, 2012).

Table 5.1: Age-adjusted incidence of thyroid cancer per 100,000 people for the United States (U.S.) and Vermont (VT), 1994-2007. Annual percent changes were significant at $p < 0.001$ ($df = 12$) or $p < 0.05$ ($df = 12$) as indicated.

	Age-adjusted incidence 1994-2007	Annual percent change	Confidence interval	t-Test
VT	8.0	8.3	[5.7-11.0]	$p < 0.001$
U.S.	8.4	5.7	[5.2-6.1]	$p < 0.001$
VT females	11.8	9.9	[5.9-14.0]	$p < 0.001$
U.S. females	12.3	5.9	[5.4-6.3]	$p < 0.001$
VT males	4.1	4.9	[0.2-9.9]	$p < 0.05$
U.S. males	4.4	5.1	[4.4-5.7]	$p < 0.001$

5.3.2 Trends by Sex and Age

Using the Ljung-Box Q analysis, increasing trends for annual age-adjusted thyroid cancer incidence in Vermont were significant between 1994 and 2000 and 2002 and 2007 for the total population, and between 1994 and 1999 and 2002 and 2007 for females, reflecting changes within the overall increase (Figure 5.1). While the overall ratio of age-adjusted incidence for females to males is 3.1 to 1, the rate of change differed during the time frame. The estimated APC among females was a little more than double that of males: 9.9 versus 4.9, respectively. The estimated APC for both females and males was higher for more recent years (2001–2007) at 13.2% for females [CI 7.3-19.1] and 11% for males [CI 0.7-21.2]. The proportional age-adjusted incidence was higher among females than males for all ages except those younger than 10 years of age (Figure 5.2). From 1994–2000, the peak age of diagnosis was between 30 and 49 years for females and between 40 and 49 years for males. However, from 2001 to 2007, the peak age of diagnosis was between 40 and 49 years for females and between 30 and 69 years for males. Overall, 29.8% of the cases were diagnosed below the age of 40 years, and 57.7% of the cases below the age of 50 years. The overall increase in incidence for females was in the 30–59 year age group for females, while no overall change in incidence by age was noted for males (Figure 5.3). There is no significant difference in the statewide distribution of the population by age or sex.

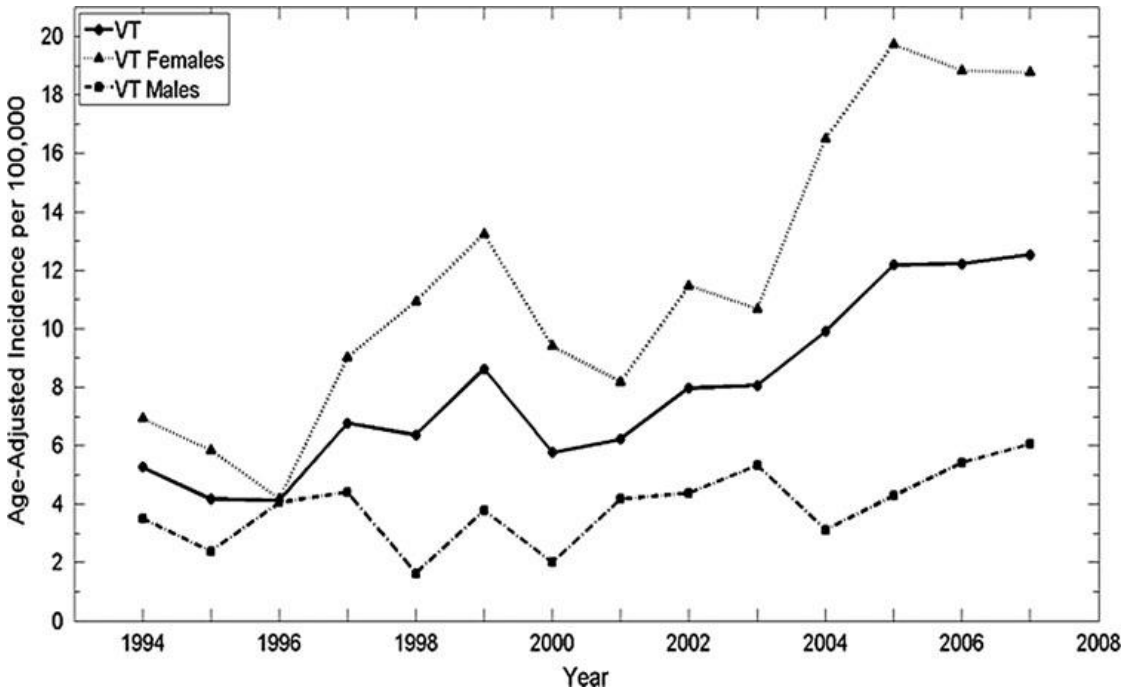


Figure 5.1: Annual age-adjusted thyroid cancer incidence significantly increased in Vermont, 1994-2007. Significant annual trends are noted for Vermont (1994-2000, 2002-2007) and Vermont females (1994-1999, 2002-2007). Significance is $p < 0.05$, $n = 14$, using Ljung-Box Q analysis in JMP® Pro v10.0.0.

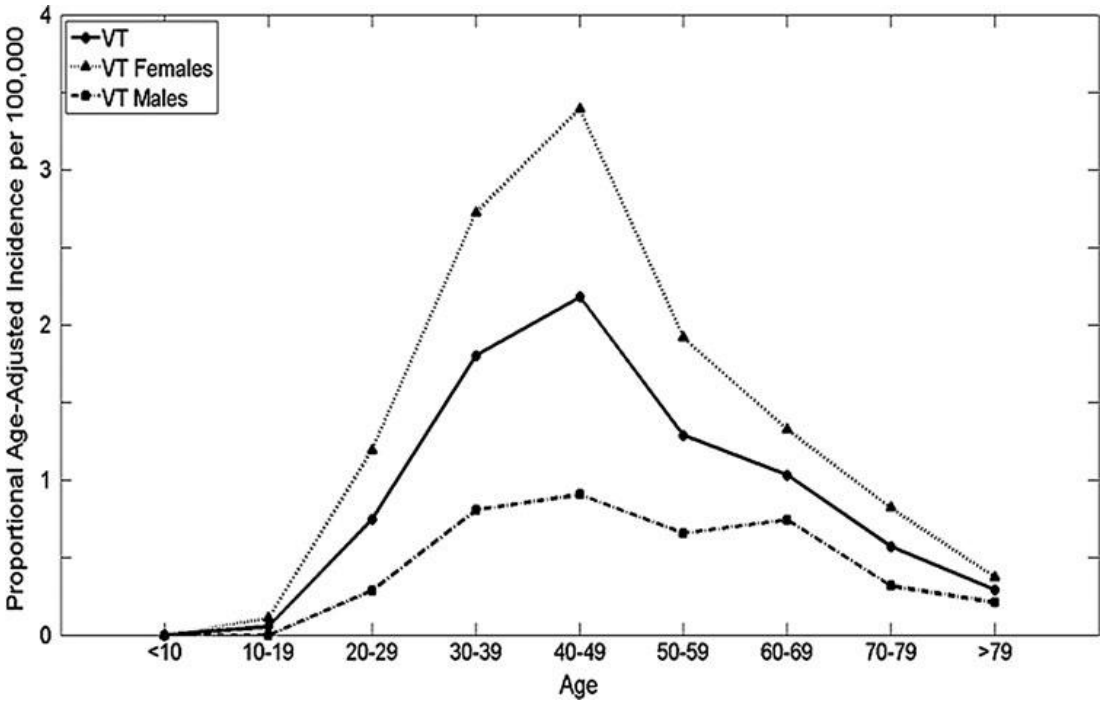


Figure 5.2: Average annual proportional age-adjusted incidence (1994-2007) for Vermont overall, Vermont females, and Vermont males. For Vermont females, the age groups with the three highest annual average age-adjusted incidence are ages 30-39 years, 40-49 years, and 50-59 years.

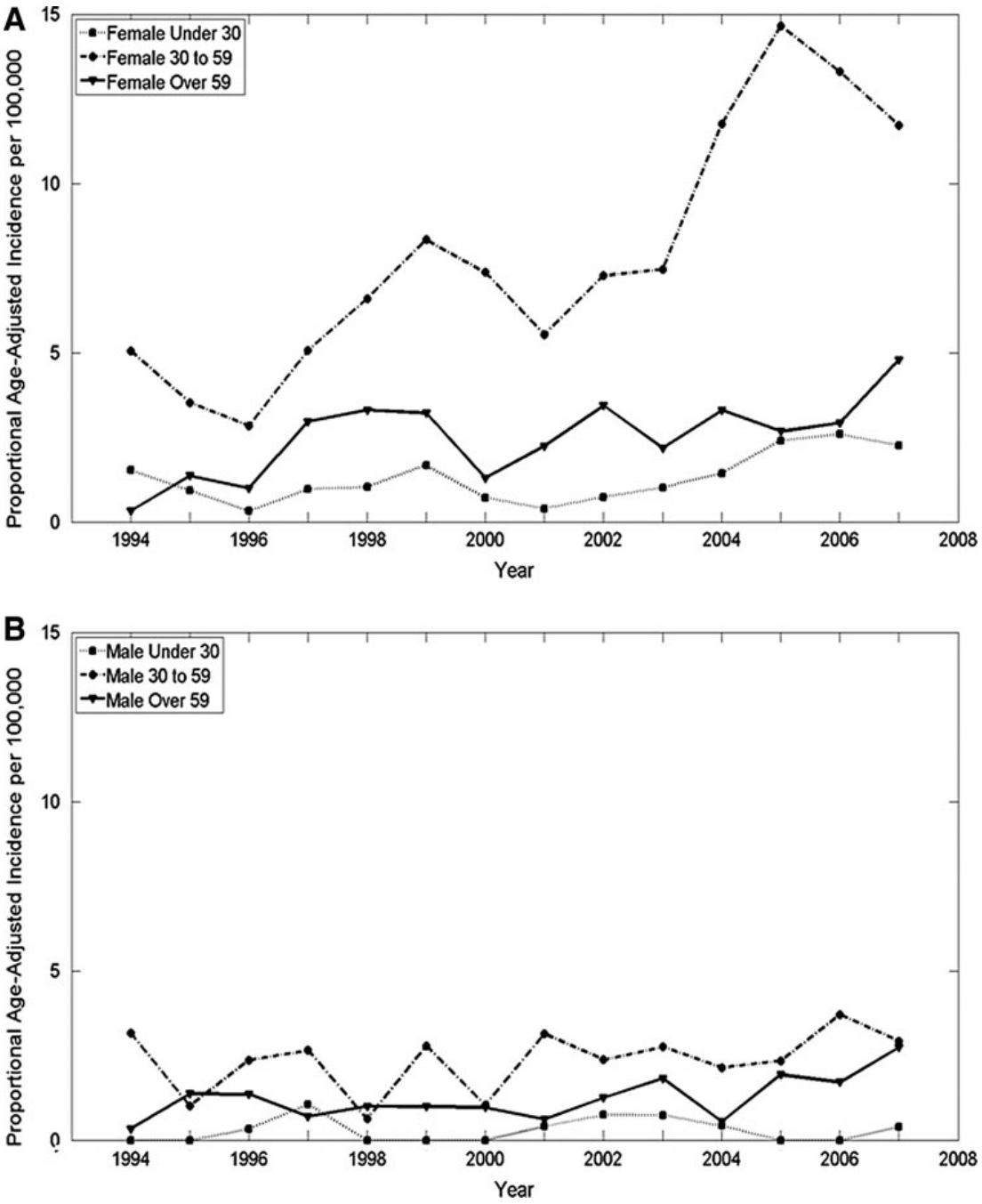


Figure 5.3: Proportional age-adjusted incidence of thyroid cancer differed by age and sex in Vermont, 1994-2007. Significant trends were identified for females (A) younger than 30 years of age (1994-1996), females aged 30-59 years old (1994-2007), females older than 59 years old (2006-2007), and males (B) younger than 30 years of age (1997-2007) by Ljung-Box Q analysis in JMP® Pro v10.0.0 ($p < 0.05$, $n = 14$).

5.3.3 Incidence by Tumor Size and Type

In Vermont, during 1994–2007, 86% of thyroid cancer cases were papillary, 9% follicular, 2% medullary, and <2% anaplastic comparable to national data. Of particular note, the findings reveal that sex is a factor in the distribution of cases by histological type (Figure 5.4). In females, papillary thyroid cancer (PTC) incidence was 89%, follicular (FTC) 8%, medullary (MTC) 2%, and anaplastic (ATC) 0.6%, while in males, PTC was 77%, FTC 15%, MTC 1%, and ATC 3%, respectively. The increase in females encompasses primarily PTCs with a small increase in follicular cancer types, but in males the increase is primarily in differentiated follicular cancers (Table 5.2). National data (Aschebrook-Kilfoy et al., 2013) indicate that PTC and FTC increased for both males and females, whereas data from the present study indicate an increase in PTC for females and FTC and ATC for males.

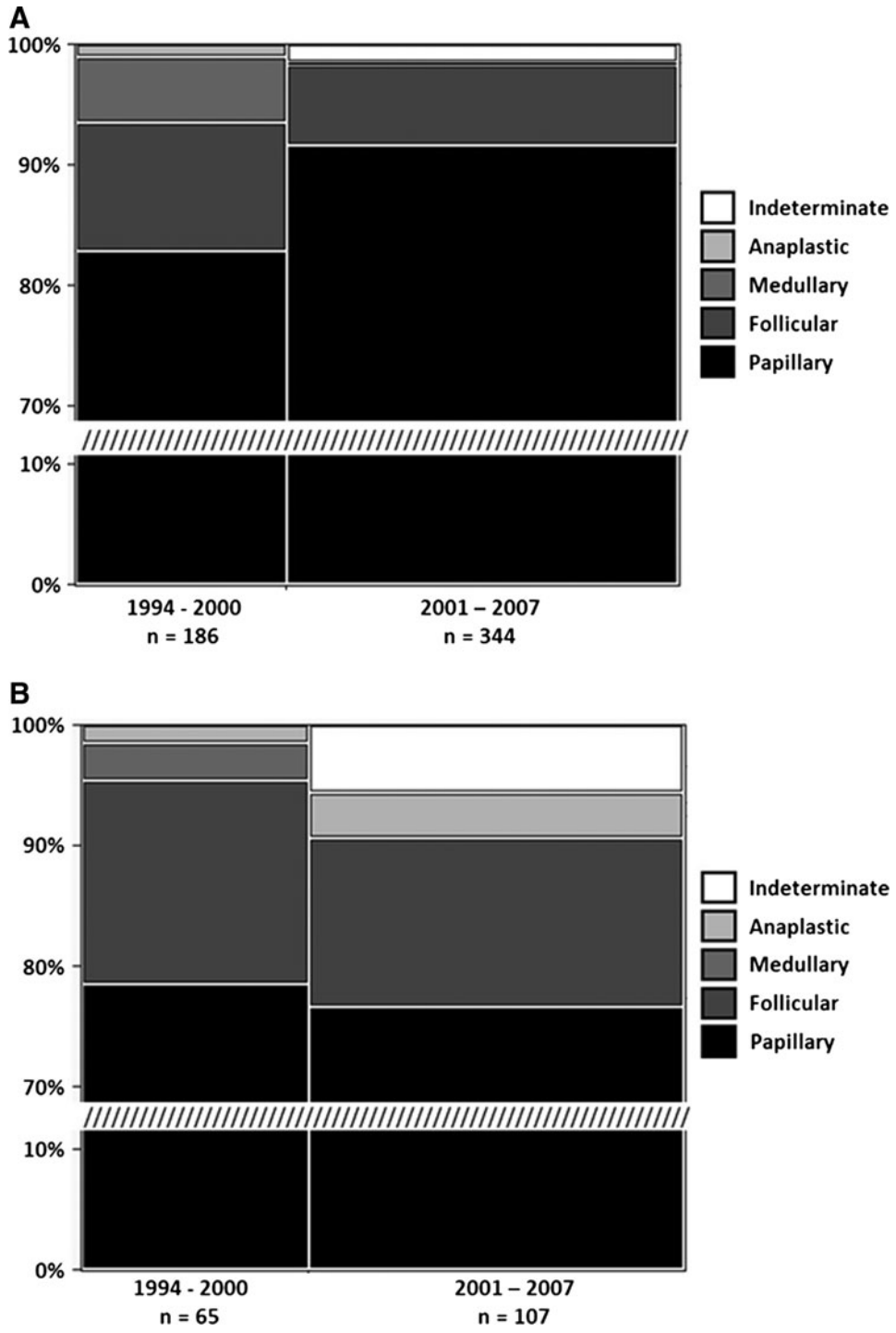


Figure 5.4: The percent of thyroid cancer types between females and males in VT differ significantly. Females (A) have proportionally more cases of papillary cancer and fewer cases of follicular and anaplastic cancer than males (B). (Pearson chi square test; $p < 0.001$, $n = 702$, $df = 4$).

Table 5.2: Thyroid cancer histological type varies by age and sex.

	Age group					
	<30 years, *n*	**<30 years, %**	**30-59 years, *n***	**30-59 years, %**	**>59 years, *n***	**>59 years, %**
Both sexes:						
Papillary	60	92.3	415	88.7	127	75.1
Follicular	2	3.1	41	8.8	26	15.4
Medullary	3	4.6	6	1.3	3	1.8
Anaplastic	0	0	1	0.2	7	4.1
Indeterminate	0	0	5	1.1	6	3.6
Total	65	100	468	100	169	100
Males:						
Papillary	10	90.9	87	83.7	36	63.2
Follicular	1	9.1	14	13.5	11	19.3
Medullary	0	0	1	1	1	1.8
Anaplastic	0	0	1	1	4	7
Indeterminate	0	0	1	1	5	8.8
Total	11	100	104	100	57	100
Females:						
Papillary	50	92.6	328	90.1	91	81.3
Follicular	1	1.9	27	7.4	15	13.4
Medullary	3	5.6	5	1.4	2	1.8
Anaplastic	0	0	0	0	3	2.7
Indeterminate	0	0	4	1.1	1	0.9
Total	54	100	364	100	112	100

Although some studies have indicated that the increase in thyroid cancer could be attributed to an increase in detection of small tumors and microcarcinomas, using the Ljung-Box Q analysis, the present data for Vermont indicate no significant difference in tumor size over time (Figure 5.5). For both females and males, the distribution of tumors by size did not vary over time; ≤1.0 cm, 1.1–2.0 cm, and >2.0 cm represented 38%, 22%, and 40%, respectively. While the distribution of tumors ≤1.0 cm, 1.1–2.0 cm, and >2.0 cm varies from year to year, the increase in thyroid cancer incidence is not due to a

significant increase in small tumors but to an overall increase in cases diagnosed with tumors.

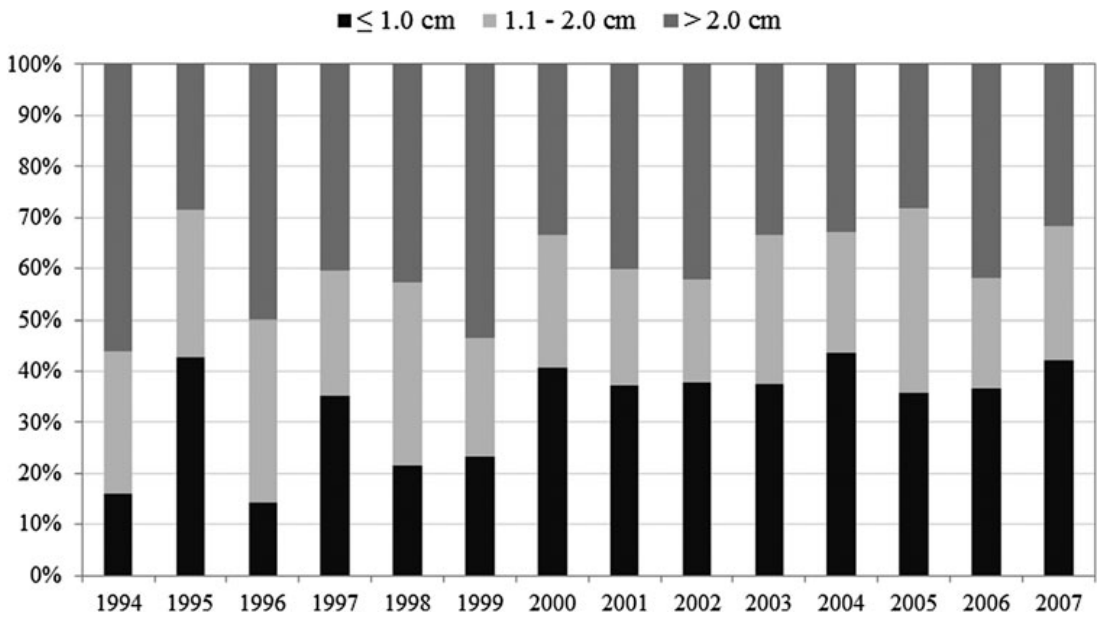


Figure 5.5: Thyroid cancer incidence classified by tumor size in Vermont, 1994-2007. The minimum number of tumors measured in any given year was 14 (1995); the maximum was 79 (2006). Using Ljung-Box Q analysis, the only significant trend occurred for tumors 1.1-2.0 cm in size in 2001-2004. When the 1.1-2.0 cm category was combined with either of the other two categories, there were no significant trends.

5.3.4 Geospatial Distribution of Thyroid Cancer Incidence

Between 1994 and 2007, thyroid cancer age-adjusted incidence varied widely throughout Vermont, ranging from no incidence to >30 per 100,000. The wide variability in incidence is striking as noted across adjacent zip codes (Figure 5.6). This was further supported by no spatial autocorrelation being detected between the annual age-adjusted thyroid cancer incidence at the zip code scale, indicating the high spatial heterogeneity of incidence across the state. Even with the high spatial variability of incidence, nine zip code Hot Spots were identified, highlighting specific focus areas that could provide

insight into future research regarding SES and environmental drivers of thyroid cancer. No other significant relationships between thyroid cancer incidence and other U.S. census variables were found.

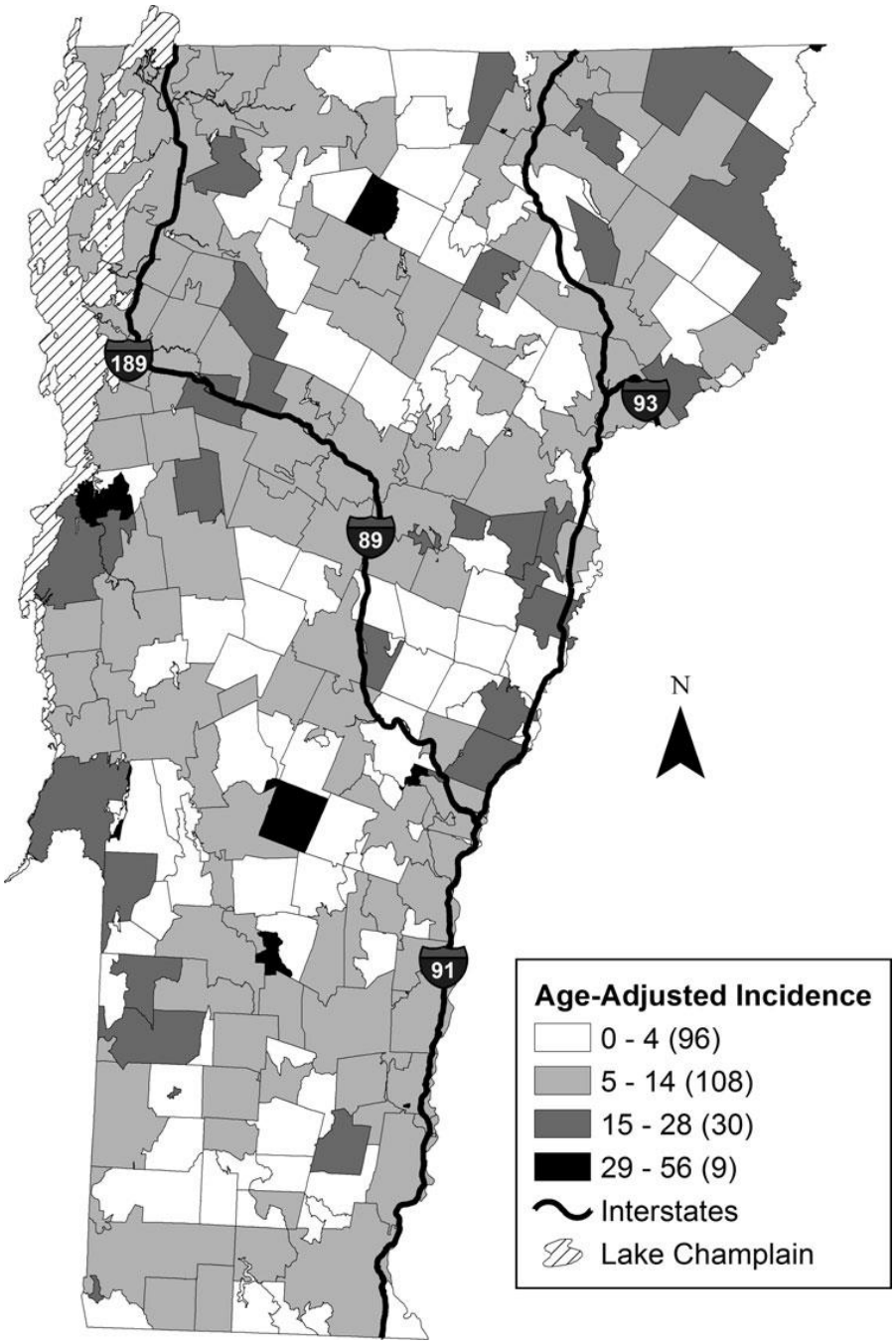


Figure 5.6: Geospatial distribution of thyroid cancer incidence. Average annual age-adjusted incidence for Vermont (1994-2007) mapped to the U.S. 2010 Census zip code tabulation areas (zip codes). Jenks Natural Breaks was used to create the four classification categories of cancer incidence.

At the county scale, Vermont health data showed a significant (df = 13, F = 12.82, p = 0.004, R^2_{adj} = 0.48) negative linear relationship between thyroid cancer incidence and the number of medical practices per 100,000 people. In addition, no significant linear relationship was found between thyroid cancer incidence and the percent insured or the number of primary care physicians per 100,000 people at the county scale. Several nonrandom clusters of high thyroid cancer incidence were revealed by Getis-Ord Gi* analyses. These clusters are located in 8 of 14 counties, and include northern and central regions of the state. The geographic distribution of the clusters occurred predominantly in the regions of highest elevation along the north–south axis of the state, which encompasses the Green Mountain Range.

When SES and demographic factors and measures of health care access were analyzed, thyroid cancer incidence was not correlated with mean family income, education at more than high school level, mean travel time to work, and long-term residents (in residence prior to 1979). At the county scale, the high thyroid cancer incidence was negatively correlated with access to healthcare, as measured by location and concentration of primary care physicians compared to the population (HISA-VT 2008). No Hot Spots were identified in the highest income counties whether by per capita or median household income. According to Vermont Health Insurance Survey, >92% of the population has health insurance coverage (reference BISHCA) since 1990 when the surveys were initiated.

5.4 Discussion

Between 1994 and 2007, the incidence of thyroid cancer more than doubled in Vermont. The present findings suggest that during 1994–2007, the thyroid cancer incidence in Vermont (8.0%) was comparable to the national trend (8.4%). However, the estimated APC for women was higher in Vermont (9.9%) compared with the national APC (5.9%) as also reflected in the total estimated APC in Vermont and the United States (8.3% and 5.7%, respectively). Strikingly, the estimated APC for females in Vermont was double that for males (9.9% and 4.9%, respectively). When categorized by age groups, the thyroid cancer incidence more than doubled for females aged between 30 and 59 years over the study period, while all other categories increased but less dramatically. The total incidence increased for males, but there was no significant difference among age groups. Various studies have indicated a relation between reproductive factors and hormone use that may partially explain the increasing thyroid cancer incidence in younger women (Negri et al., 1999). Although the overall health insurance rate in Vermont (>92%) is near complete, it is unclear in this study whether female access to healthcare is greater than for males, which might contribute to the sex difference in estimated APC.

Overall, PTC accounts for more than 85% and FTC 10% of the tumors detected, as anticipated. However, the distribution varies by age (Table 5.2); PTC represents >92% of the tumors in those younger than 30 years of age, but only 75% in patients older than 59 years of age. The incidence of FTC and ATC increases for those older than 59 years of ages for both men and women. For men, PTC is most common in those younger than

59 years of age (>90%), but in those older than 59 years of age, PTC drops to <63%, and FTC and ATC increase to 19% and 7%, respectively. For females, the change in distribution of thyroid cancer type is less pronounced such that in those aged 59 years and older, PTC accounts for >81% of cases, while FTC and ATC increase to 13.4% and 2.7%, respectively. Aschebrook–Kilfoy et al. (2013) recently reported an increase in age-adjusted FTC in women and men, with an increase in aggressive tumors as well as small tumors particularly in women. Unfortunately, the grade of tumor and metastatic lesions were not reported in the Vermont registry in >80% of cases, so a comparison of aggressive tumors is not possible.

While previous studies have reported a significant increase in small (≤1.0 cm) tumors (Davies and Welch, 2006; Davies et al., 2010; Morris and Myssiorek, 2010; Morris et al., 2013), the present findings did not reveal a significant selective increase in these tumors. An increase in small tumors and a decrease in larger tumors (>2.0 cm) would be predicted if increased diagnostic scrutiny accounted for the increase in thyroid cancer incidence. An incidence of ≤1.0 cm tumor size that does not significantly increase over time would argue against an increased detection due to improved diagnosis.

The present findings for the entire state do not show concordance with higher SES and increase in thyroid cancer incidence as has been previously shown (Sprague et al., 2008; Li et al., 2013; Morris et al., 2013). Unexpectedly, the higher thyroid cancer incidence by county was not located in the counties with the highest per capita income, family income, and education as would have been predicted from previous studies. No correlation was observed between zip codes with high incidence of thyroid cancers and

SES or access to enhanced medical diagnostics. Data on tumor characteristics by zip code would be necessary to determine a potential correlation between SES and tumor size and stage at diagnosis. Nevertheless, the distribution of higher incidence of thyroid cancer incidence is not consonant with higher diagnostic scrutiny that would be expected with higher SES and access to healthcare. Aside from healthcare access, variation in healthcare provider culture and practices could contribute to the geospatial and temporal patterns that were observed in thyroid cancer incidence, but this could not be addressed in this study. Future studies could examine variation across healthcare provider networks.

This study was unable to determine causal relations between healthcare access, diagnostic approaches, environmental factors, and thyroid cancer incidence based on the geospatial analyses, but regions were identified where an assessment of possible environmental and demographic drivers may be focused. Although the geostatistical analysis did not identify a spatial autocorrelation at the zip code scale, the possibility of autocorrelation cannot be ruled out. As with any geo-referenced data set, there is always the possibility that the scale or range of autocorrelation will be missed if the spacing between observations is too large (Goovaerts, 1998). As a result, it is suggested that a database geo-referenced at the household scale is needed to identify spatial correlations better between environmental factors and risk of thyroid cancer. Future studies are necessary to evaluate the role of diagnostic evaluation, environmental factors directly in thyroid cancer incidence trends.

This study may also be limited by the usual concerns of population-based studies, including nonreview of histopathological diagnoses, incomplete data collection, and

variations in tumor classifications related to analyses of registry data. The 1994–2007 data collection time frame is subsequent to the World Health Organization recommended change in thyroid tumor classification that occurred in 1988 (Hedinger et al., 1988). Further, the population of Vermont is generally racially homogenous (>95% white Caucasian), and thus caution must be taken in generalizing the results to other populations with greater representation of racial groups. The finding that variation in access to healthcare does not fully explain temporal and spatial trends in thyroid cancer incidence in Vermont warrants further investigation in other study populations, particularly those with increased racial diversity. Healthcare insurance coverage is high (>92%) in Vermont and should be taken into consideration when generalizing to other states or population groups.

In summary, in rural Vermont with nearly complete healthcare coverage and a relatively stable population, the incidence of thyroid cancer is increasing among both women and men. The increase is most profound for women between the ages of 30 and 59 years. The increase in thyroid cancer is reflected in both small and large tumors; there is no significant difference in tumor size detected over the time period studied. Furthermore, geospatial analysis revealed a distribution of thyroid cancer incidence across the state that did not correlate with proximity to tertiary healthcare centers or SES. Similarly, the data did not support the often-reported hypothesis of increased incidence over time due to improved diagnostic scrutiny. These findings strongly suggest that other SES and environmental factors may likely contribute to the increase in thyroid cancer

incidence. Investigation into naturally occurring and man-made environmental factors as well as lifestyle impact on thyroid cancer development is clearly warranted.

5.5 Supplementary Figures

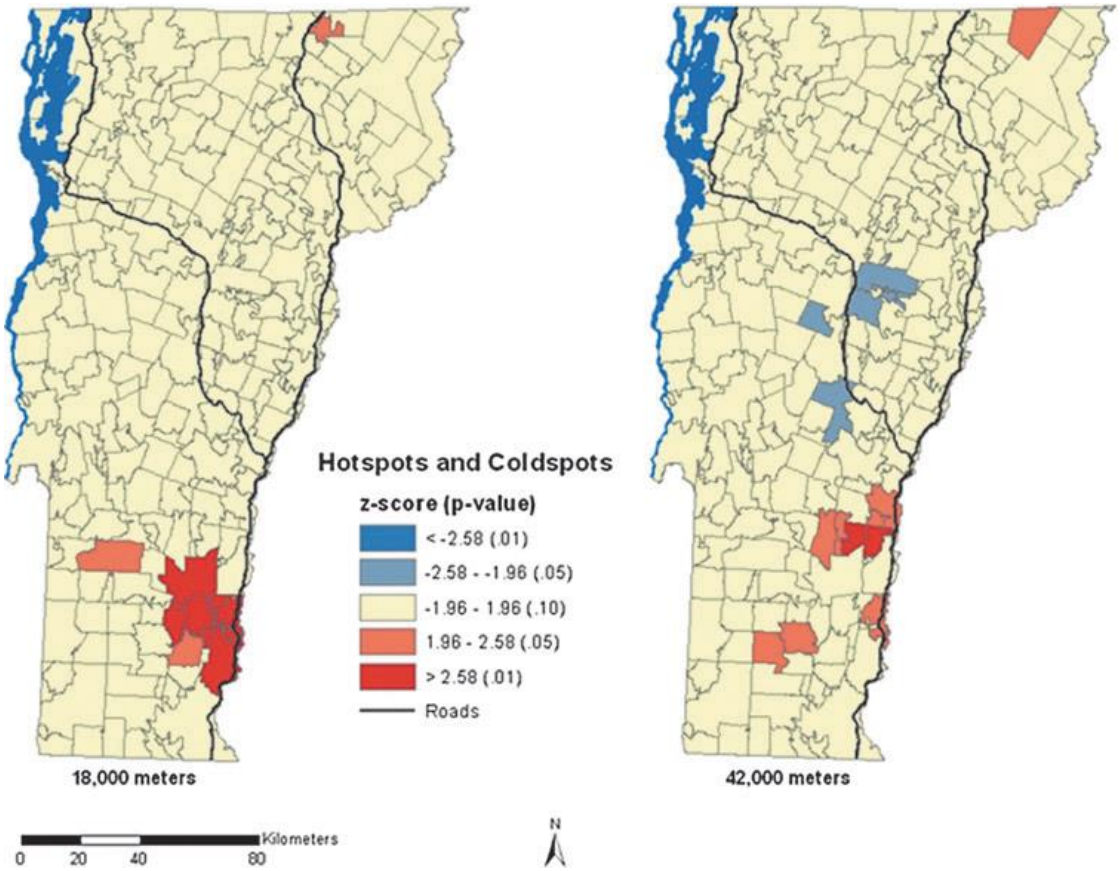


Figure 5.S1: Clusters of thyroid cancer incidence in Vermont, United States, 1994–2007. For the Getis-Ord Gi* statistic, two zones of indifference of 18,000m and 42,000m were used. Clusters were significant ($p < 0.05$) if there was a higher (red) or lower (blue) proportion of thyroid cancer incidence (normalized per 100,000) than expected within the specified distance.

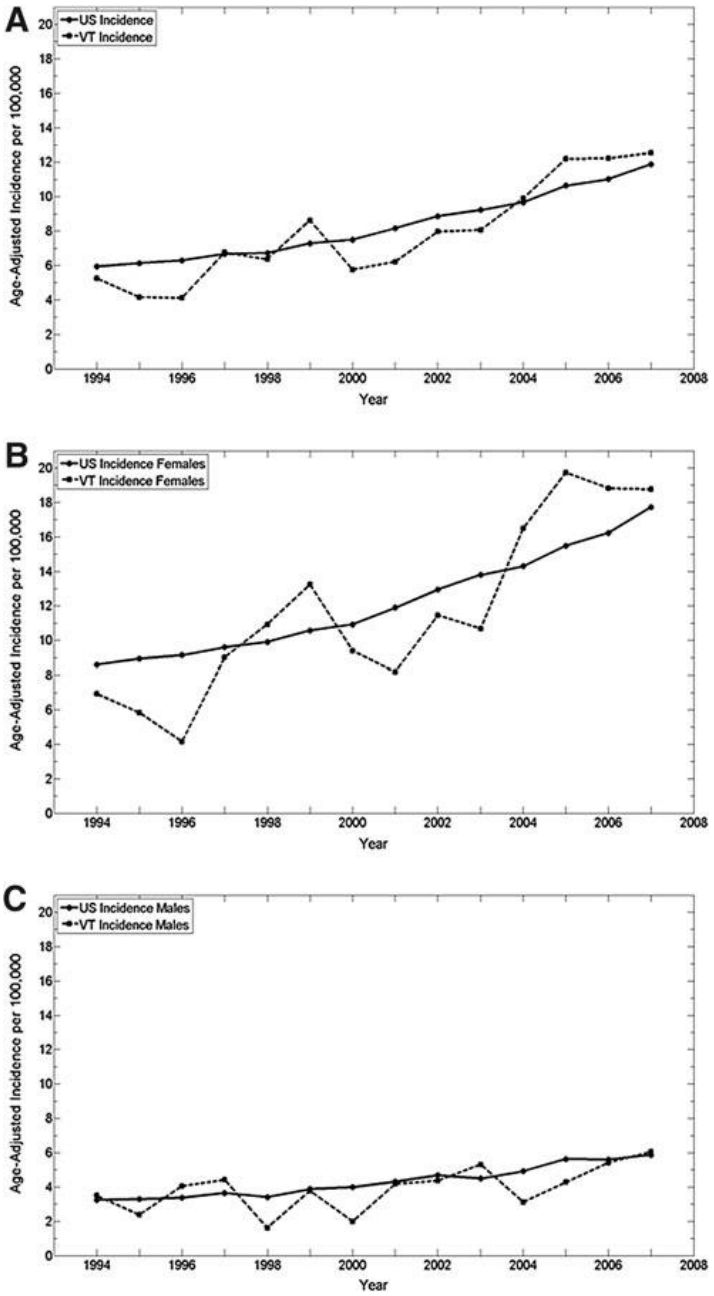


Figure 5.S2: Age-adjusted incidence of thyroid cancer per 100,000 people for the United States and Vermont, 1994–2007. (A) The average annual age-adjusted incidence was 8.0 (VT) and 8.4 (U.S.). The annual percent change (EAPC) at 8.3 [CI 5.7–11.0] for Vermont and 5.7 [CI 5.2–6.3] for the United States were significant ($p < 0.001$). (B) The average annual age-adjusted incidence for females was 11.8 (VT) and 12.3 (U.S.). The EAPC was 9.9 [CI 5.9–4.0] for Vermont and 5.9 [CI 5.4–6.3] for the United States were significant ($p < 0.001$). (C) The average annual age-adjusted for males was 4.1 (VT) and 4.4 (U.S.). The EAPC was 4.9 [CI 0.2–9.9] for Vermont and 5.1 [CI 4.4–5.7] for the United States were significant at $p < 0.05$ (VT) and $p < 0.001$ (U.S.).

CHAPTER 6: CONCLUSION

In this dissertation, I initially develop the conjunctive clause evolutionary algorithm (CCEA) and the disjunction of the conjunctive clauses evolutionary algorithm (DNFEA) a set of evolutionary algorithms (EAs), with the intent that they be used in tandem to mine real-world datasets that include epistatic and heterogeneous associations. The EAs were developed to efficiently explore real-world datasets that contain missing data, varied data types (i.e., nominal, discrete, and ordinal), inherent feature interactions, numerous combinations of risk factors, and implicit heterogeneity. To demonstrate effectiveness of this set of EAs, I first tested my algorithms on three benchmark problems. These three classifier test cases (the majority-on, the multiplexer and a simulated SNP dataset developed by Urbanowicz and Moore (2010)) exhibit some combination of feature interaction (epistasis), heterogeneity, and noise. The CCEA in tandem with the DNFEA was able to successfully solve all three benchmark problems by repeatedly evolving the optimal solution set for each problem. Next, using inspiration from Big Data analysis in Mayer-Schönberger and Cukier (2014), the solution sets archived by the CCEA were mined to perform feature selection. These feature selection techniques were successfully applied to the simulated SNP dataset and consistently selected the features that comprise the true signals. Finally, the CCEA was applied to the *T. dimidiata* infestation datasets for two towns in Jutiapa, Guatemala. The CCEA was able to efficiently search these large, noisy datasets with multiple datatypes to find strong probabilistic signals for complex multivariate interactions associated with infestation. These probabilistically significant interactions could then be utilized by domain experts

and town managers to improve mitigation strategies and make efficient use of limited resources to reduce the risk of *T. dimidiata* infestation. In addition, I developed a proof-of-concept for mapping risk that leverages the SNPs from next generation sequencing of *T. dimidiata*, and the genetic distance equation of Smouse and Peakall (1999) to determine the range of spatial autocorrelation of the vector's genetic structure. This range of spatial autocorrelation was then used as a surrogate to the movement of *T. dimidiata* and plotted to visualize the risk of infestation. The risk maps for the two Guatemalan villages seemed to suggest that the source of infestation for one town was sylvatic, and the other was domestic/peridomestic. These novel and modified statistical tools were successful in characterizing the risk of infestation across the two villages.

Finally, the risk of thyroid cancer over a 14-year period in Vermont was assessed on the zip code scale, which is larger than the individual household scale used for the Chagas disease datasets. This aggregated US census data did not show spatial autocorrelation and limited my ability to determine individual risk factors associated with thyroid cancer on an individual scale. That being said, traditional risk factors such as age and gender were associated with higher rates of thyroid cancer. Also, geospatial analyses of incidence of thyroid cancer at the zip code scale did reveal Hot Spots associated with thyroid cancer. However, whether these Hot Spots are signals or noise could not be determined given the available aggregated data.

This research was an initial venture into assessing disease risk in the age of Big Data. The methods developed in this dissertation were designed specifically for the dataset available (data-driven). With that being said, these methodologies are not so

specific that they cannot be applied to other datasets and applications. For instance, the CCEA is a general, non-parametric classification tool that can be applied to a wide range of problems without violating assumptions inherent to the CCEA. This is not the case with many traditional statistical methods such as analysis of variance and linear regression that are limited to specific data types, assume normality, non-correlation between data and are challenged by missing data. The CCEA is not limited to assessing multivariate interactions associated with disease; it can be applied to variety of fields and topics including but not limited to analysis of political party association, marketing, and ecological niche modeling. As for the modifications made to the genetic distance correlograms of the Smouse and Peakall (1999), switching to a semivariogram allows for the use of the most appropriate pairwise distance or relatedness metric to measure genetic spatial autocorrelation. In addition, using box plots for the semivariograms relaxes the assumption of normality, helps visualize the distribution of the semivariance data, and is not limited to genetic data.

The hope is that the methodology developed in this dissertation will continually be improved by myself and others. I view this as one of many initial explorations in analyzing Big Data and will hopefully inspire new algorithms as this field continues to grow.

CHAPTER 7: LITERATURE CITED

Abad-Franch, Fernando, Carolina Valença-Barbosa, Otília Sarquis, and Marli M. Lima. 2014. "All That Glisters Is Not Gold: Sampling-Process Uncertainty in Disease-Vector Surveys with False-Negative and False-Positive Detections." Edited by Alon Warburg. *PLoS Neglected Tropical Diseases* 8 (9): e3187. doi:10.1371/journal.pntd.0003187.

Acevedo, F, E Godoy, and CJ Schofield. 2000. "Comparison of Intervention Strategies for Control of Triatoma Dimidiata in Nicaragua." *Memórias Do Instituto Oswaldo Cruz* 95 (6). doi:10.1590/S0074-02762000000600022.

Arzube Rodríguez, Manuel. 1966. "Investigación de La Fuente Alimenticia Del T. Dimidiata, Latr. 1811 (Hemiptera: Reduvidae), Mediante La Reacción de Precipitina." *Revista Ecuatoriana de Higiene Y Medicina Tropical* 23: 137–52.

Aschebrook-Kilfoy, Briseis, Raymon H. Grogan, Mary H. Ward, Edwin Kaplan, and Susan S. Devesa. 2013. "Follicular Thyroid Cancer Incidence Patterns in the United States, 1980-2009." *Thyroid: Official Journal of the American Thyroid Association* 23 (8): 1015–21. doi:10.1089/thy.2012.0356.

Barbu, Corentin, Eric Dumonteil, and Sébastien Gourbière. 2010. "Characterization of the Dispersal of Non-Domiciliated Triatoma Dimidiata through the Selection of Spatially Explicit Models." Edited by Ricardo E. Gürtler. *PLoS Neglected Tropical Diseases* 4 (8): e777. doi:10.1371/journal.pntd.0000777.

———. 2011. "Evaluation of Spatially Targeted Strategies to Control Non-Domiciliated Triatoma Dimidiata Vector of Chagas Disease." Edited by Ricardo E. Gürtler. *PLoS Neglected Tropical Diseases* 5 (5): e1045. doi:10.1371/journal.pntd.0001045.

Bargues, María Dolores, Debora R. Klisiowicz, Fernando Gonzalez-Candelas, Janine M. Ramsey, Carlota Monroy, Carlos Ponce, Paz María Salazar-Schettino, et al. 2008. "Phylogeography and Genetic Variation of Triatoma Dimidiata, the Main Chagas Disease Vector in Central America, and Its Position within the Genus Triatoma." Edited by Ricardo E. Gurtler. *PLoS Neglected Tropical Diseases* 2 (5): e233. doi:10.1371/journal.pntd.0000233.

Barto, A. G. 1985. "Learning by Statistical Cooperation of Self-Interested Neuron-like Computing Elements." *Human Neurobiology* 4 (4): 229–56.

Bastian, Mathieu, Sebastien Heymann, and Mathieu Jacomy. 2009. "Gephi: An Open Source Software for Exploring and Manipulating Networks." In *International AAAI Conference on Weblogs and Social Media*. http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154.

Bomblies, Arne. 2012. "Modeling the Role of Rainfall Patterns in Seasonal Malaria Transmission." *Climatic Change* 112 (3–4): 673–85. doi:10.1007/s10584-011-0230-6.

Booker, Lashon B. 1989. "Triggered Rule Discovery in Classifier Systems." In *Proceedings of the Third International Conference on Genetic Algorithms*, 3:265–74.

Boulos, Dina N. Kamel, Ramy R. Ghali, Ezzeldin M. Ibrahim, Maged N. Kamel Boulos, and Philip AbdelMalik. 2011. "An Eight-Year Snapshot of Geospatial Cancer Research (2002-2009): Clinico-Epidemiological and Methodological Findings and Trends." *Medical Oncology (Northwood, London, England)* 28 (4): 1145–62. doi:10.1007/s12032-010-9607-z.

Boyle, P., and D. M. Parkin. 1991. "Cancer Registration: Principles and Methods. Statistical Methods for Registries." *IARC Scientific Publications*, no. 95: 126–58.

Bradburd, Gideon S., Peter L. Ralph, and Graham M. Coop. 2013. "DISENTANGLING THE EFFECTS OF GEOGRAPHIC AND ECOLOGICAL ISOLATION ON GENETIC DIFFERENTIATION: ISOLATION BY GEOGRAPHIC AND ECOLOGICAL DISTANCE." *Evolution* 67 (11): 3258–73. doi:10.1111/evo.12193.

Breslow, Norman E., and Nicholas E. Day. 1987. *Statistical Methods in Cancer Research. Bd. 2: The Design and Analysis of Cohort Studies*. IARC Scientific Publications 82. Lyon: International Agency for Research on Cancer.

Briceño-León, Roberto, and Jorge Méndez Galván. 2007. "The Social Determinants of Chagas Disease and the Transformations of Latin America." *Memorias Do Instituto Oswaldo Cruz* 102 Suppl 1 (October): 109–12.

Buescher Jr., Charles A. 2017. "History of Environmental Engineering." *Washing University in St. Louis: School of Engineering & Applied Science*. Accessed February 23. https://eece.wustl.edu/eeceatwashu/about/Pages/environmental-engineering-history.aspx.

Bustamante, Dulce M., Sandra M. De Urioste-Stone, José G. Juárez, and Pamela M. Pennington. 2014. "Ecological, Social and Biological Risk Factors for Continued Trypanosoma Cruzi Transmission by Triatoma Dimidiata in Guatemala." Edited by Claudio R. Lazzari. *PLoS ONE* 9 (8): e104599. doi:10.1371/journal.pone.0104599.

Bustamante, Dulce Maria, Maria Carlota Monroy, Antonieta Guadalupe Rodas, Jaime Abraham Juarez, and John B. Malone. 2007. "Environmental Determinants of the Distribution of Chagas Disease Vectors in South-Eastern Guatemala." *Geospatial Health* 1 (2): 199–211. doi:10.4081/gh.2007.268.

Bustamante Zamora, D. M., M. M. Hernandez, N. Torres, C. Zuniga, W. Sosa, V. de Abrego, and M. C. Monroy Escobar. 2015. "Information to Act: Household Characteristics Are Predictors of Domestic Infestation with the Chagas Vector Triatoma Dimidiata in Central America." *American Journal of Tropical Medicine and Hygiene* 93 (1): 97–107. doi:10.4269/ajtmh.14-0596.

Butz, Martin V., David E. Goldberg, and Kurian Tharakunnel. 2003. "Analysis and Improvement of Fitness Exploitation in XCS: Bounding Models, Tournament Selection, and Bilateral Accuracy." *Evolutionary Computation* 11 (3): 239–77. doi:10.1162/106365603322365298.

Butz, Martin V., and Martin Pelikan. 2006. "Studying XCS/BOA Learning in Boolean Functions: Structure Encoding and Random Boolean Functions." In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 1449. ACM Press. doi:10.1145/1143997.1144236.

Butz, Martin V., Kumara Sastry, and David E. Goldberg. 2005. "Strong, Stable, and Reliable Fitness Pressure in XCS due to Tournament Selection." *Genetic Programming and Evolvable Machines* 6 (1): 53–77. doi:10.1007/s10710-005-7619-9.

Butz, M.V., T. Kovacs, P.L. Lanzi, and S.W. Wilson. 2004. "Toward a Theory of Generalization and Learning in XCS." *IEEE Transactions on Evolutionary Computation* 8 (1): 28–46. doi:10.1109/TEVC.2003.818194.

Buxbaum, Joseph D., Jeremy M. Silverman, Christopher J. Smith, Mario Kilifarski, Jennifer Reichert, Eric Hollander, Brian A. Lawlor, Michael Fitzgerald, David A. Greenberg, and Kenneth L. Davis. 2001. "Evidence for a Susceptibility Gene for Autism on Chromosome 2 and for Genetic Heterogeneity." *The American Journal of Human Genetics* 68 (6): 1514–20. doi:10.1086/320588.

Calderón, Claudia I., Patricia L. Dorn, Sergio Melgar, Juan José Chávez, Antonieta Rodas, Regina Rosales, and Carlota M. Monroy. 2004. "A Preliminary Assessment of Genetic Differentiation of *Triatoma Dimidiata* (Hemiptera: Reduviidae) in Guatemala by Random Amplification of Polymorphic DNA-Polymerase Chain Reaction." *Journal of Medical Entomology* 41 (5): 882–87. doi:10.1603/0022-2585-41.5.882.

Campos Franci, Luciana de, Jacob Nabe-Nielsen, Jens-Christian Svenning, and Fernando Roberto Martins. 2016. "Short-Term Spatial Variation in the Demography of a Common Neotropical Liana Is Shaped by Tree Community Structure and Light Availability." *Plant Ecology* 217 (10): 1273–90. doi:10.1007/s11258-016-0655-0.

"Chagas Disease in Latin America: An Epidemiological Update Based on 2010 Estimates." 2015. *Releve Epidemiologique Hebdomadaire* 90 (6): 33–43.

Chen, Amy Y., Ahmedin Jemal, and Elizabeth M. Ward. 2009. "Increasing Incidence of Differentiated Thyroid Cancer in the United States, 1988-2005." *Cancer* 115 (16): 3801–7. doi:10.1002/cncr.24416.

Coster, Stephanie S., and Adrienne I. Kovach. 2012. "Anthropogenic Influences on the Spatial Genetic Structure of Black Bears." *Conservation Genetics* 13 (5): 1247–57. doi:10.1007/s10592-012-0368-4.

Coura, José Rodrigues. 2015. "The Main Sceneries of Chagas Disease Transmission. The Vectors, Blood and Oral Transmissions - A Comprehensive Review." *Memórias Do Instituto Oswaldo Cruz* 110 (3): 277–82. doi:10.1590/0074-0276140362.

Cramer, John D., Pingfu Fu, Karem C. Harth, Seunghee Margevicius, and Scott M. Wilhelm. 2010. "Analysis of the Rising Incidence of Thyroid Cancer Using the Surveillance, Epidemiology and End Results National Cancer Data Registry." *Surgery* 148 (6): 1147-1152; discussion 1152-1153. doi:10.1016/j.surg.2010.10.016.

Cucunubá, Zulma M., Omolade Okuwoga, María-Gloria Basáñez, and Pierre Nouvellet. 2016. "Increased Mortality Attributed to Chagas Disease: A Systematic Review and Meta-Analysis." *Parasites & Vectors* 9 (1). doi:10.1186/s13071-016-1315-x.

Curado, M. P., and International Agency for Research on Cancer, eds. 2008. *Cancer Incidence in Five Continents. Vol. 9: [...]*. IARC Scientific Publications 160. Lyon: International Agency for Research on Cancer.

Davies, Louise, Michelle Ouellette, Mark Hunter, and H. Gilbert Welch. 2010. "The Increasing Incidence of Small Thyroid Cancers: Where Are the Cases Coming From?" *The Laryngoscope* 120 (12): 2446–51. doi:10.1002/lary.21076.

Davies, Louise, and H. Gilbert Welch. 2006. "Increasing Incidence of Thyroid Cancer in the United States, 1973-2002." *JAMA* 295 (18): 2164–67. doi:10.1001/jama.295.18.2164.

De Andrade, A. L., F. Zicker, R. M. De Oliveira, I. G. Da Silva, S. A. Silva, S. S. De Andrade, and C. M. Martelli. 1995. "Evaluation of Risk Factors for House Infestation by Triatoma Infestans in Brazil." *The American Journal of Tropical Medicine and Hygiene* 53 (5): 443–47.

De Jong, Kenneth A., and William M. Spears. 1991. "Learning Concept Classification Rules Using Genetic Algorithms." In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 12:651–56. Sydney, Australia: Morgan Kaufmann.

De León, J. Romeo. 1959. "Estado Actual de La Enfermedad de Chagas En Guatemala." *Revista Goiana de Medicina* 5: 445–55.

De Urioste-Stone, S. M., P. M. Pennington, E. Pellecer, T. M. Aguilar, G. Samayoa, H. D. Perdomo, H. Enriquez, and J. G. Juarez. 2015. "Development of a Community-Based Intervention for the Control of Chagas Disease Based on Peridomestic Animal Management: An Eco-Bio-Social Perspective." *Transactions of the Royal Society of Tropical Medicine and Hygiene* 109 (2): 159–67. doi:10.1093/trstmh/tru202.

Dorn, Patricia L., Sergio Melgar, Vanessa Rouzier, Astrid Gutierrez, Crescent Combe, Regina Rosales, Antonieta Rodas, Sarah Kott, Debra Salvia, and Carlota M. Monroy. 2003. "The Chagas Vector, *Triatoma Dimidiata* (Hemiptera:Reduviidae), Is Panmictic within and Among Adjacent Villages in Guatemala : Table 1." *Journal of Medical Entomology* 40 (4): 436–40. doi:10.1603/0022-2585-40.4.436.

Dorn, Patricia L., Carlota Monroy, and Andrew Curtis. 2007. "Triatoma Dimidiata (Latreille, 1811): A Review of Its Diversity across Its Geographic Range and the Relationship among Populations." *Infection, Genetics and Evolution* 7 (2): 343–52. doi:10.1016/j.meegid.2006.10.001.

Dumonteil, Eric, Sebastien Gourbière, Mario Barrera-Pérez, Eugenia Rodriguez-Félix, Hugo Ruiz-Piña, Othón Baños-Lopez, María Jesús Ramirez-Sierra, Frédéric Menu, and Jorge E. Rabinovich. 2002. "Geographic Distribution of Triatoma Dimidiata and Transmission Dynamics of Trypanosoma Cruzi in the Yucatan Peninsula of Mexico." *The American Journal of Tropical Medicine and Hygiene* 67 (2): 176–83.

Dumonteil, Eric, Hugo Ruiz-Piña, Eugenia Rodriguez-Félix, Mario Barrera-Pérez, María Jesús Ramirez-Sierra, Jorge E Rabinovich, and Frédéric Menu. 2004. "Re-Infestation of Houses by Triatoma Dimidiata after Intra-Domicile Insecticide

Application in the Yucatán Peninsula, Mexico." *Memórias Do Instituto Oswaldo Cruz* 99 (3): 253–56. doi:10.1590/S0074-02762004000300002.

Edwards, Brenda K., Elizabeth Ward, Betsy A. Kohler, Christie Eheman, Ann G. Zauber, Robert N. Anderson, Ahmedin Jemal, et al. 2010. "Annual Report to the Nation on the Status of Cancer, 1975-2006, Featuring Colorectal Cancer Trends and Impact of Interventions (Risk Factors, Screening, and Treatment) to Reduce Future Rates." *Cancer* 116 (3): 544–73. doi:10.1002/cncr.24760.

Eiben, Agoston E., and James E. Smith. 2010. *Introduction to evolutionary computing*. 1. ed., 2. printing, Softcover version of original hardcover ed. 2003. Natural computing series. Berlin: Springer.

Elhaik, Eran, Tatiana Tatarinova, Dmitri Chebotarev, Ignazio S. Piras, Carla Maria Calò, Antonella De Montis, Manuela Atzori, et al. 2014. "Geographic Population Structure Analysis of Worldwide Human Populations Infers Their Biogeographical Origins." *Nature Communications* 5 (April). doi:10.1038/ncomms4513.

Enander, Richard T., Antonio Ramírez Amaya, Richard A. Enander, and David M. Gute. 2010. "Neurocysticercosis: Risk and Primary Prevention Strategies Update." *International Journal of Environmental Health Research* 20 (5): 329–65. doi:10.1080/09603123.2010.482152.

Enewold, Lindsey, Kangmin Zhu, Elaine Ron, Aizen J. Marrogi, Alexander Stojadinovic, George E. Peoples, and Susan S. Devesa. 2009. "Rising Thyroid Cancer Incidence in the United States by Demographic and Tumor Characteristics, 1980-2005." *Cancer Epidemiology, Biomarkers & Prevention: A Publication of the American Association for Cancer Research, Cosponsored by the American Society of Preventive Oncology* 18 (3): 784–91. doi:10.1158/1055-9965.EPI-08-0960.

Enger, Kyle S., Rosalinda Ordoñez, Mark L. Wilson, and Janine M. Ramsey. 2004. "Evaluation of Risk Factors for Rural Infestation by Triatoma Pallidipennis (Hemiptera: Triatominae), a Mexican Vector of Chagas Disease." *Journal of Medical Entomology* 41 (4): 760–67.

Eppstein, M. J., and P Haake,. 2008. "Very Large Scale Relieff for Genome-Wide Association Analysis." In *Computational Intelligence in Bioinformatics and Computational Biology, 2008*, 112–19. IEEE.

Eppstein, Margaret J., and Paul D. H. Hines. 2012. "A 'Random Chemistry' Algorithm for Identifying Collections of Multiple Contingencies That Initiate Cascading Failure." *IEEE Transactions on Power Systems* 27 (3): 1698–1705. doi:10.1109/TPWRS.2012.2183624.

Eppstein, Margaret J., Joshua L. Payne, Bill C. White, and Jason H. Moore. 2007. "Genomic Mining for Complex Disease Traits with 'random Chemistry.'" *Genetic Programming and Evolvable Machines* 8 (4): 395–411. doi:10.1007/s10710-007-9039-5.

Finnegan, Laura, Sarrah Castillo, Jack Hughes, Ken F. Abraham, Rodney W. Brook, and Christopher J. Kyle. 2013. "Fine-Scale Analysis Reveals Cryptic Patterns of

Genetic Structure in Canada Geese." *The Condor* 115 (4): 738–49. doi:10.1525/cond.2013.120117.

Foley, D. H., R. C. Russell, and J. H. Bryan. 2004. "Population Structure of the Peridomestic Mosquito Ochlerotatus Notoscriptus in Australia." *Medical and Veterinary Entomology* 18 (2): 180–90. doi:10.1111/j.0269-283X.2004.00497.x.

Fritz, April G., ed. 2013. *International Classification of Diseases for Oncology: ICD-O*. Third edition, First revision. Geneva: World Health Organization.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Mass: Addison-Wesley Pub. Co.

González-Ramos, J., L. Noguera-Morel, H.Y. Tong, E. Ramírez, E. Ruiz-Bravo, T. Bellón, R. Cabañas, L. Cachafeiro, and P. Herranz-Pinto. 2016. "Two Cases of Overlap Severe Cutaneous Adverse Reactions to Benznidazole Treatment for Asymptomatic Chagas Disease in a Nonendemic Country." *British Journal of Dermatology* 175 (3): 604–7. doi:10.1111/bjd.14451.

Goovaerts, P. 1998. "Geostatistical Tools for Characterizing the Spatial Variability of Microbiological and Physico-Chemical Soil Properties." *Biology and Fertility of Soils* 27 (4): 315–34. doi:10.1007/s003740050439.

Grimalt, J. O., J. Sunyer, V. Moreno, O. C. Amaral, M. Sala, A. Rosell, J. M. Anto, and J. Albaiges. 1994. "Risk Excess of Soft-Tissue Sarcoma and Thyroid Cancer in a Community Exposed to Airborne Organochlorinated Compound Mixtures with a High Hexachlorobenzene Content." *International Journal of Cancer* 56 (2): 200–203.

Grodski, Simon, Tani Brown, Stan Sidhu, Anthony Gill, Bruce Robinson, Diana Learoyd, Mark Sywak, Tom Reeve, and Leigh Delbridge. 2008. "Increasing Incidence of Thyroid Cancer Is due to Increased Pathologic Detection." *Surgery* 144 (6): 1038–1043; discussion 1043. doi:10.1016/j.surg.2008.08.023.

Hall, Stephen F., Hugh Walker, Robert Siemens, and Amy Schneeberg. 2009. "Increasing Detection and Increasing Incidence in Thyroid Cancer." *World Journal of Surgery* 33 (12): 2567–71. doi:10.1007/s00268-009-0226-9.

Hanley, John P., Margaret J. Eppstein, Jeffrey S. Buzas, and Donna M. Rizzo. 2016. "Evolving Probabilistically Significant Epistatic Classification Rules for Heterogeneous Big Datasets." In *Proceedings of the 18th Annual Conference on Genetic and Evolutionary Computation*, 445–52. ACM Press. doi:10.1145/2908812.2908931.

Hashimoto, Ken, Celia Cordon-Rosales, Ranfery Trampe, and Masato Kawabata. 2006. "Impact of Single and Multiple Residual Sprayings of Pyrethroid Insecticides against Triatoma Dimidiata (Reduviiade; Triatominae), the Principal Vector of Chagas Disease in Jutiapa, Guatemala." *The American Journal of Tropical Medicine and Hygiene* 75 (2): 226–30.

Hashimoto, Ken, and Christopher J Schofield. 2012. "Elimination of Rhodnius Prolixus in Central America." *Parasites & Vectors* 5 (1): 45. doi:10.1186/1756-3305-5-45.

Hasslocher-Moreno, A. M., P. E. A. A. do Brasil, A. S. de Sousa, S. S. Xavier, M. C. Chambela, and G. M. Sperandio da Silva. 2012. "Safety of Benznidazole Use in

the Treatment of Chronic Chagas' Disease." *Journal of Antimicrobial Chemotherapy* 67 (5): 1261–66. doi:10.1093/jac/dks027.

Hedinger, Christoph Ernst, Edward Dillwyn Williams, and Weltgesundheitsorganisation, eds. 1993. *Histological Typing of Thyroid Tumours*. 2. ed., 1. reprint. International Histological Classification of Tumours 11. Berlin: Springer.

Hempel, Sandra. 2007. *The Strange Case of the Broad Street Pump: John Snow and the Mystery of Cholera*. Berkeley: University of California Press.

Hernández, Marianela Menes, Carlota Monroy, Dulce María Bustamante, Bárbara Moguel, Antonieta Rodas, Elizabeth Solórzano, and Mauricio García. 2006. "Estudio de Las Preferencias de Hábitat No Domiciliar Del Principal Vector de La Enfermedad de Chagas En Guatemala, Triatoma Dimidiata, Y Sus Implicaciones Para El Control Vectorial." Dirección General de Investigación. Laboratorio de Entomología Aplicada y Parasitología – LENAP – Escuela de Biología: Universidad de San Carlos de Guatemala.

Holland, John H, and Judith S Reitman. 1978. "Cognitive Systems Based on Adaptive Algorithms." In *An Overview of Pattern-Directed Inference Systems*, 313–29. Santa Monica, CA: Rand Corporation.

Hornby, Gregory S. 2006. "ALPS: The Age-Layered Population Structure for Reducing the Problem of Premature Convergence." In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 815. ACM Press. doi:10.1145/1143997.1144142.

Ioannides, Charalambos, Geoff Barrett, and Kerstin Eder. 2011. "XCS Cannot Learn All Boolean Functions." In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 1283. ACM Press. doi:10.1145/2001576.2001749.

Iqbal, Muhammad, Will N. Browne, and Mengjie Zhang. 2014. "Reusing Building Blocks of Extracted Knowledge to Solve Complex, Large-Scale Boolean Problems." *IEEE Transactions on Evolutionary Computation* 18 (4): 465–80. doi:10.1109/TEVC.2013.2281537.

Iqbal, Muhammad, Will N. Browne, and Mengjie Zhang. 2012. "Extracting and Using Building Blocks of Knowledge in Learning Classifier Systems." In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, 863. ACM Press. doi:10.1145/2330163.2330283.

———. 2013a. "Extending Learning Classifier System with Cyclic Graphs for Scalability on Complex, Large-Scale Boolean Problems." In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, 1045. ACM Press. doi:10.1145/2463372.2463500.

———. 2013b. "Evolving Optimum Populations with XCS Classifier Systems: XCS with Code Fragmented Action." *Soft Computing* 17 (3): 503–18. doi:10.1007/s00500-012-0922-5.

———. 2013c. "Learning Complex, Overlapping and Niche Imbalance Boolean Problems Using XCS-Based Classifier Systems." *Evolutionary Intelligence* 6 (2): 73–91. doi:10.1007/s12065-013-0091-1.

———. 2015. “Improving Genetic Search in XCS-Based Classifier Systems through Understanding the Evolvability of Classifier Rules.” *Soft Computing* 19 (7): 1863–80. doi:10.1007/s00500-014-1369-7.

Isaaks, Edward H., and R. Mohan Srivastava. 1989. *Applied Geostatistics*. New York: Oxford University Press.

Jarlenski, Marian, Seo Hyon Baik, and Yuting Zhang. 2016. “Trends in Use of Medications for Smoking Cessation in Medicare, 2007–2012.” *American Journal of Preventive Medicine* 51 (3): 301–8. doi:10.1016/j.amepre.2016.02.018.

Jemal, Ahmedin, Freddie Bray, Melissa M. Center, Jacques Ferlay, Elizabeth Ward, and David Forman. 2011. “Global Cancer Statistics.” *CA: A Cancer Journal for Clinicians* 61 (2): 69–90. doi:10.3322/caac.20107.

Julian, Timothy R., Amy J. Pickering, James O. Leckie, and Alexandria B. Boehm. 2013. “Enterococcus Spp on Fomites and Hands Indicate Increased Risk of Respiratory Illness in Child Care Centers.” *American Journal of Infection Control* 41 (8): 728–33. doi:10.1016/j.ajic.2012.10.013.

Kaplinski, Michelle, Malasa Jois, Gerson Galdos-Cardenas, Victoria R. Rendell, Vishal Shah, Rose Q. Do, Rachel Marcus, et al. 2015. “Sustained Domestic Vector Exposure Is Associated With Increased Chagas Cardiomyopathy Risk but Decreased Parasitemia and Congenital Transmission Risk Among Young Women in Bolivia.” *Clinical Infectious Diseases* 61 (6): 918–26. doi:10.1093/cid/civ446.

Kendall, Maurice G. 1952. *The Advanced Theory of Statistics*. 5th ed. Vol. 1. New York, New York: Hafner Publishing Company.

Kilfoy, Briseis A., Tongzhang Zheng, Theodore R. Holford, Xuesong Han, Mary H. Ward, Andreas Sjodin, Yaqun Zhang, et al. 2009. “International Patterns and Trends in Thyroid Cancer Incidence, 1973-2002.” *Cancer Causes & Control: CCC* 20 (5): 525–31. doi:10.1007/s10552-008-9260-4.

Kim, Christopher, Xiaofeng Bi, Dongsheng Pan, Yingtai Chen, Tobias Carling, Shuangge Ma, Robert Udelsman, and Yawei Zhang. 2013. “The Risk of Second Cancers after Diagnosis of Primary Thyroid Cancer Is Elevated in Thyroid Microcarcinomas.” *Thyroid: Official Journal of the American Thyroid Association* 23 (5): 575–82. doi:10.1089/thy.2011.0406.

King, Raymond J., Celia Cordon-Rosales, Jonathan Cox, Clive R. Davies, and Uriel D. Kitron. 2011. “Triatoma Dimidiata Infestation in Chagas Disease Endemic Regions of Guatemala: Comparison of Random and Targeted Cross-Sectional Surveys.” Edited by Ricardo E. Gürtler. *PLoS Neglected Tropical Diseases* 5 (4): e1035. doi:10.1371/journal.pntd.0001035.

Kirino, Yohei, George Bertsias, Yoshiaki Ishigatsubo, Nobuhisa Mizuki, Ilknur Tugal-Tutkun, Emire Seyahi, Yilmaz Ozyazgan, et al. 2013. “Genome-Wide Association Analysis Identifies New Susceptibility Loci for Behçet’s Disease and Epistasis between HLA-B*51 and ERAP1.” *Nature Genetics* 45 (2): 202–7. doi:10.1038/ng.2520.

Kovacs, T. 2002. “What Should a Classifier System Learn and How Should We Measure It?” *Soft Computing - A Fusion of Foundations, Methodologies and Applications* 6 (3–4): 171–82. doi:10.1007/s005000100114.

Kovacs, Tim. 1998. “XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions.” In *Soft Computing in Engineering Design and Manufacturing*, edited by P. K. Chawdhry, R. Roy, and R. K. Pant, 59–68. London: Springer London. http://link.springer.com/10.1007/978-1-4471-0427-8_7.

Lao, Oscar, Eveline Altena, Christian Becker, Silke Brauer, Thirsa Kraaijenbrink, Mannis van Oven, Peter Nürnberg, Peter de Knijff, and Manfred Kayser. 2013. “Clinal Distribution of Human Genomic Diversity across the Netherlands despite Archaeological Evidence for Genetic Discontinuities in Dutch Population History.” *Investigative Genetics* 4 (1): 9. doi:10.1186/2041-2223-4-9.

Larivière, Serge. 2001. “Ursus Americanus.” *Mammalian Species* 647 (January): 1–11. doi:10.1644/1545-1410(2001)647<0001:UA>2.0.CO;2.

Larouche, Richard, Michelle Stone, Ron N. Buliung, and Guy Faulkner. 2016. “‘I’d Rather Bike to School!’: Profiling Children Who Would Prefer to Cycle to School.” *Journal of Transport & Health* 3 (3): 377–85. doi:10.1016/j.jth.2016.06.010.

Lee, Bruce Y, Kristina M Bacon, Maria Elena Bottazzi, and Peter J Hotez. 2013. “Global Economic Burden of Chagas Disease: A Computational Simulation Model.” *The Lancet Infectious Diseases* 13 (4): 342–48. doi:10.1016/S1473-3099(13)70002-1.

Lent, Herman, and Pedro Wygodzinsky. 1979. “Revision of the Triatominae (Hemiptera, Reduviidae), and Their Significance as Vectors of Chagas’ Disease.” *Bulletin of the American Museum of Natural History* 163 (3): 123–520.

Leux, C., and P. Guénel. 2010. “Risk Factors of Thyroid Tumors: Role of Environmental and Occupational Exposures to Chemical Pollutants.” *Revue D’epidemiologie Et De Sante Publique* 58 (5): 359–67. doi:10.1016/j.respe.2010.05.005.

Li, Kaigang, Bruce Simons-Morton, Benjamin Gee, and Ralph Hingson. 2016. “Marijuana-, Alcohol-, and Drug-Impaired Driving among Emerging Adults: Changes from High School to One-Year Post-High School.” *Journal of Safety Research* 58 (September): 15–20. doi:10.1016/j.jsr.2016.05.003.

Li, Nan, Xianglin L. Du, Lorraine R. Reitzel, Li Xu, and Erich M. Sturgis. 2013. “Impact of Enhanced Detection on the Increase in Thyroid Cancer Incidence in the United States: Review of Incidence Trends by Socioeconomic Status within the Surveillance, Epidemiology, and End Results Registry, 1980-2008.” *Thyroid: Official Journal of the American Thyroid Association* 23 (1): 103–10. doi:10.1089/thy.2012.0392.

Llorà, X., K. Sastry, and D.E. Goldberg. 2005. “The Compact Classifier System: Scalability Analysis and First Results.” In *The 2005 IEEE Congress on Evolutionary Computation*, 1:596–603. IEEE. doi:10.1109/CEC.2005.1554737.

Llorà, Xavier, Rohith Reddy, Brian Matesic, and Rohit Bhargava. 2007. “Towards Better than Human Capability in Diagnosing Prostate Cancer Using Infrared Spectroscopic Imaging.” In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, 2098. ACM Press. doi:10.1145/1276958.1277366.

Lucero, D. E., L. A. Morrissey, D. M. Rizzo, A. Rodas, R. Garnica, L. Stevens, D. M. Bustamante, and M. C. Monroy. 2013. "Ecohealth Interventions Limit Triatomine Reinfestation Following Insecticide Spraying in La Brea, Guatemala." *American Journal of Tropical Medicine and Hygiene* 88 (4): 630–37. doi:10.4269/ajtmh.12-0448.

Lynch, M., and K. Ritland. 1999. "Estimation of Pairwise Relatedness with Molecular Markers." *Genetics* 152 (4): 1753–66.

Mangano, Joseph J. 2009. "Geographic Variation in U.S. Thyroid Cancer Incidence and a Cluster near Nuclear Reactors in New Jersey, New York, and Pennsylvania." *International Journal of Health Services: Planning, Administration, Evaluation* 39 (4): 643–61. doi:10.2190/HS.39.4.c.

Manne, Jennifer, Jun Nakagawa, Yoichi Yamagata, Alexander Goehler, John S. Brownstein, and Marcia C. Castro. 2012. "Triatomine Infestation in Guatemala: Spatial Assessment after Two Rounds of Vector Control." *The American Journal of Tropical Medicine and Hygiene* 86 (3): 446–54. doi:10.4269/ajtmh.2012.11-0052.

Markellos, Raphael N., Dimitris Psychoyios, and Friedrich Schneider. 2016. "Sovereign Debt Markets in Light of the Shadow Economy." *European Journal of Operational Research* 252 (1): 220–31. doi:10.1016/j.ejor.2015.12.039.

Marsily, Ghislain de. 1993. *Quantitative Hydrogeology: Groundwater Hydrology for Engineers*. 5. [print.]. San Diego: Acad. Press.

Mayer-Schönberger, Viktor, and Kenneth Cukier. 2014. *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. First Mariner Books edition. Boston: Mariner Books, Houghton Mifflin Harcourt.

McDermott, James, Kenneth De Jong, Una-May O'Reilly, David R. White, Sean Luke, Luca Manzoni, Mauro Castelli, et al. 2012. "Genetic Programming Needs Better Benchmarks." In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, 791. ACM Press. doi:10.1145/2330163.2330273.

McKinney, B.A., D.M. Reif, B.C. White, J.E. Crowe, and J.H. Moore. 2007. "Evaporative Cooling Feature Selection for Genotypic Data Involving Interactions." *Bioinformatics* 23 (16): 2113–20. doi:10.1093/bioinformatics/btm317.

Melgar, Sergio, Juan José Chávez, Patricia Landaverde, Franklin Herrera, Antonieta Rodas, Eunice Enríquez, Patricia Dorn, and Carlota Monroy. 2007. "The Number of Families of Triatoma Dimidiata in a Guatemalan House." *Memórias Do Instituto Oswaldo Cruz* 102 (2): 221–23. doi:10.1590/S0074-02762007005000001.

Molina, I., F. Salvador, A. Sánchez-Montalvá, B. Treviño, N. Serre, A. Sao Avilés, and B. Almirante. 2015. "Toxic Profile of Benznidazole in Patients with Chronic Chagas Disease: Risk Factors and Comparison of the Product from Two Different Manufacturers." *Antimicrobial Agents and Chemotherapy* 59 (10): 6125–31. doi:10.1128/AAC.04660-14.

Monroy, Carlota, Dulce Maria Bustamante, Sandy Pineda, Antonieta Rodas, Xochitl Castro, Virgilio Ayala, Javier Quiñónes, and Bárbara Moguel. 2009. "House

Improvements and Community Participation in the Control of Triatoma Dimidiata Re-Infestation in Jutiapa, Guatemala." *Cadernos De Saude Publica* 25 Suppl 1: S168-178.

Monroy, Carlota, Antonieta Rodas, Mildred Mejía, Regina Rosales, and Yuichiro Tabaru. 2003. "Epidemiology of Chagas Disease in Guatemala: Infection Rate of Triatoma Dimidiata, Triatoma Nitida and Rhodnius Prolixus (Hemiptera, Reduviidae) with Trypanosoma Cruzi and Trypanosoma Rangeli (Kinetoplastida, Trypanosomatidae)." *Memórias Do Instituto Oswaldo Cruz* 98 (3): 305–10. doi:10.1590/S0074-02762003000300003.

Monroy, Carlota, Antonieta Rodas, Mildred Mejia, and Yuichiro Tabaru. 1998. "Wall Plastering and Paints as Methods to Control Vectors of Chagas Disease in Guatemala." *Medical Entomology and Zoology* 49 (3): 187–93. doi:10.7601/mez.49.187.

Monroy, Maria Carlota, Dulce Maria Bustamante, Antonieta Guadalupe Rodas, Maria Eunice Enriquez, and Regina Guadalupe Rosales. 2003. "Habitats, Dispersion and Invasion of Sylvatic <I>Triatoma Dimidiata</I> (Hemiptera: Reduviidae: Triatominae) in Petén, Guatemala." *Journal of Medical Entomology* 40 (6): 800–806. doi:10.1603/0022-2585-40.6.800.

Moore, J. H., and B. C. White. 2007. "Tuning Relieff for Genome-Wide Genetic Analysis." In *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, 166–75. Springer.

Moore, Jason H. 2003. "The Ubiquitous Nature of Epistasis in Determining Susceptibility to Common Human Diseases." *Human Heredity* 56 (1–3): 73–82. doi:10.1159/000073735.

Morris, Luc G. T., and David Myssiorek. 2010. "Improved Detection Does Not Fully Explain the Rising Incidence of Well-Differentiated Thyroid Cancer: A Population-Based Analysis." *American Journal of Surgery* 200 (4): 454–61. doi:10.1016/j.amjsurg.2009.11.008.

Morris, Luc G. T., Andrew G. Sikora, Tor D. Tosteson, and Louise Davies. 2013. "The Increasing Incidence of Thyroid Cancer: The Influence of Access to Care." *Thyroid: Official Journal of the American Thyroid Association* 23 (7): 885–91. doi:10.1089/thy.2013.0045.

Murray, Christopher J L, Theo Vos, Rafael Lozano, Mohsen Naghavi, Abraham D Flaxman, Catherine Michaud, Majid Ezzati, et al. 2012. "Disability-Adjusted Life Years (DALYs) for 291 Diseases and Injuries in 21 Regions, 1990–2010: A Systematic Analysis for the Global Burden of Disease Study 2010." *The Lancet* 380 (9859): 2197–2223. doi:10.1016/S0140-6736(12)61689-4.

Musa, George J., Po-Huang Chiang, Tyler Sylk, Rachel Bavley, William Keating, Bereketab Lakew, Hui-Chen Tsou, and Christina W. Hoven. 2013. "Use of GIS Mapping as a Public Health Tool-From Cholera to Cancer." *Health Services Insights* 6: 111–16. doi:10.4137/HSI.S10471.

Nakagawa, J, C Cordón-Rosales, J Juárez, C Itzep, and T Nonami. 2003. "Impact of Residual Spraying on Rhodnius Prolixus and Triatoma Dimidiata in the

Department of Zacapa in Guatemala." *Memórias Do Instituto Oswaldo Cruz* 98 (2): 277–82. doi:10.1590/S0074-02762003000200019.

Nakagawa, J., K. Hashimoto, C. Cordón-Rosales, J. Abraham Juárez, R. Trampe, and L. Marroquín Marroquín. 2003. "The Impact of Vector Control on Triatoma Dimidiata in the Guatemalan Department of Jutiapa." *Annals of Tropical Medicine and Parasitology* 97 (3): 288–97. doi:10.1179/000349803235001895.

"National Cancer Institute 2012 Surveillance, Epidemiology and End Results (SEER) Stat Fact Sheets: Thyroid Cancer." 2013. Accessed September 30. Available at: http://seer.cancer.gov/statfacts/html/thyro.html.

"National Cancer Institute. Thyroid Cancer." 2013. September 30. www .cancer.gov/cancertopics/types/thyroid.

Negri, E., L. Dal Maso, E. Ron, C. La Vecchia, S. D. Mark, S. Preston-Martin, A. McTiernan, et al. 1999. "A Pooled Analysis of Case-Control Studies of Thyroid Cancer. II. Menstrual and Reproductive Factors." *Cancer Causes & Control: CCC* 10 (2): 143–55.

Nesheli, Mahmood Mahmoodi, Avishai (Avi) Ceder, and Simon Estines. 2016. "Public Transport User's Perception and Decision Assessment Using Tactic-Based Guidelines." *Transport Policy* 49 (July): 125–36. doi:10.1016/j.tranpol.2016.04.007.

Nuzzo, Regina. 2014. "Scientific Method: Statistical Errors." *Nature* 506 (7487): 150–52. doi:10.1038/506150a.

Olivera, M. J., Z. M. Cucunuba, C. A. Alvarez, and R. S. Nicholls. 2015. "Safety Profile of Nifurtimox and Treatment Interruption for Chronic Chagas Disease in Colombian Adults." *American Journal of Tropical Medicine and Hygiene* 93 (6): 1224–30. doi:10.4269/ajtmh.15-0256.

Orantes, Lucia. 2017. "The Dissertation Will Be Defended On April, 19, 2017."

Pazaitou-Panayiotou, K., P. Kappa Iliadou, A. Chrisoulidou, P. Mitsakis, E. Doumala, A. Fotareli, M. Boudina, L. Mathiopoulou, F. Patakiouta, and K. Tziomalos. 2013. "The Increase in Thyroid Cancer Incidence Is Not Only due to Papillary Microcarcinomas: A 40-Year Study in 1 778 Patients." *Experimental and Clinical Endocrinology & Diabetes: Official Journal, German Society of Endocrinology [and] German Diabetes Association* 121 (7): 397–401. doi:10.1055/s-0033-1345125.

Peakall, R., and P. E. Smouse. 2012. "GenAlEx 6.5: Genetic Analysis in Excel. Population Genetic Software for Teaching and Research--an Update." *Bioinformatics* 28 (19): 2537–39. doi:10.1093/bioinformatics/bts460.

Peakall, Rod, and Peter E. Smouse. 2006. "Genalex 6: Genetic Analysis in Excel. Population Genetic Software for Teaching and Research." *Molecular Ecology Notes* 6 (1): 288–95. doi:10.1111/j.1471-8286.2005.01155.x.

Pearson, Karl. 1899. "On Certain Properties of the Hypogeometrical Series, and on the Fitting of Such Series to Observation Polygons in the Theory of Chance." *Philosophical Magazine*, 5, 47 (285): 236–46.

Pellecer, M. J., P. L. Dorn, D. M. Bustamante, A. Rodas, and M. C. Monroy. 2013. "Vector Blood Meals Are an Early Indicator of the Effectiveness of the Ecohealth

Approach in Halting Chagas Transmission in Guatemala." *American Journal of Tropical Medicine and Hygiene* 88 (4): 638–44. doi:10.4269/ajtmh.12-0458.

Pellegriti, Gabriella, Francesco Frasca, Concetto Regalbuto, Sebastiano Squatrito, and Riccardo Vigneri. 2013. "Worldwide Increasing Incidence of Thyroid Cancer: Update on Epidemiology and Risk Factors." *Journal of Cancer Epidemiology* 2013: 1–10. doi:10.1155/2013/965212.

Pérez de Rosas, Alicia R., Elsa L. Segura, Octavio Fusco, Adolfo L. Bareiro Guiñazú, and Beatriz A. García. 2013. "Fine-Scale Genetic Structure in Populations of the Chagas' Disease Vector Triatoma Infestans (Hemiptera, Reduvidae)." *Genetica* 141 (1–3): 107–17. doi:10.1007/s10709-013-9710-0.

Petana, W B. 1971. "American Trypanosomiasis in British Honduras – X: Natrual Habitats and Ecology of Triatoma Dimidiata ((Hemiptera, Reduviidae) in the El Cayo and Toledo Districts, and the Prevalence of Infection with Trypanosoma (Schizotrypanum) Cruzi in the Wild-Caught Bugs." *Annals of Tropical Medicine and Parasitology* 65 (2): 168–78.

Piotti, Andrea, Stefano Leonardi, Myriam Heuertz, Joukje Buiteveld, Thomas Geburek, Sophie Gerber, Koen Kramer, Cristina Vettori, and Giovanni Giuseppe Vendramin. 2013. "Within-Population Genetic Structure in Beech (Fagus Sylvatica L.) Stands Characterized by Different Disturbance Histories: Does Forest Management Simplify Population Substructure?" Edited by Pär K. Ingvarsson. *PLoS ONE* 8 (9): e73391. doi:10.1371/journal.pone.0073391.

Poole, Keith T, and Howard Rosenthal. 1984. "The Polarization of American Politics." *The Journal of Politics* 46 (4): 1061–79.

Quinde-Calderón, Leonardo, Paulina Rios-Quituizaca, Luis Solorzano, and Eric Dumonteil. 2016. "Ten Years (2004-2014) of Chagas Disease Surveillance and Vector Control in Ecuador: Successes and Challenges." *Tropical Medicine & International Health* 21 (1): 84–92. doi:10.1111/tmi.12620.

Ramírez, Carolina J., Carlos A. Jaramillo, María del Pilar Delgado, Néstor A. Pinto, Germán Aguilera, and Felipe Guhl. 2005. "Genetic Structure of Sylvatic, Peridomestic and Domestic Populations of Triatoma Dimidiata (Hemiptera: Reduviidae) from an Endemic Zone of Boyaca, Colombia." *Acta Tropica* 93 (1): 23–29. doi:10.1016/j.actatropica.2004.09.001.

Ramirez-Sierra, Maria Jesus, Melba Herrera-Aguilar, Sébastien Gourbière, and Eric Dumonteil. 2010. "Patterns of House Infestation Dynamics by Non-Domiciliated *Triatoma Dimidiata* Reveal a Spatial Gradient of Infestation in Rural Villages and Potential Insect Manipulation by *Trypanosoma Cruzi*: **Patterns of House Infestation Dynamics by Non-Domiciliated *Triatoma Dimidiata*** ." *Tropical Medicine & International Health* 15 (1): 77–86. doi:10.1111/j.1365-3156.2009.02422.x.

Rapp, J P, M R Garrett, and A Y Deng. 1998. "Construction of a Double Congenic Strain to Prove an Epistatic Interaction on Blood Pressure between Rat Chromosomes 2 and 10." *Journal of Clinical Investigation* 101 (8): 1591–95. doi:10.1172/JCI2251.

Rašić, Gordana, Nancy Endersby-Harshman, Warsito Tantowijoyo, Anjali Goundar, Vanessa White, Qiong Yang, Igor Filipović, Petrina Johnson, Ary A. Hoffmann, and Eggi Arguni. 2015. "Aedes Aegypti Has Spatially Structured and Seasonally Stable Populations in Yogyakarta, Indonesia." *Parasites & Vectors* 8 (1). doi:10.1186/s13071-015-1230-6.

Rassi, Anis, Anis Rassi, William C. Little, Sérgio S. Xavier, Sérgio G. Rassi, Alexandre G. Rassi, Gustavo G. Rassi, Alejandro Hasslocher-Moreno, Andrea S. Sousa, and Maurício I. Scanavacca. 2006. "Development and Validation of a Risk Score for Predicting Death in Chagas' Heart Disease." *The New England Journal of Medicine* 355 (8): 799–808. doi:10.1056/NEJMoa053241.

Reitzel, Lorraine R., Nga Nguyen, Nan Li, Li Xu, Seann D. Regan, and Erich M. Sturgis. 2014. "Trends in Thyroid Cancer Incidence in Texas from 1995 to 2008 by Socioeconomic Status and Race/Ethnicity." *Thyroid: Official Journal of the American Thyroid Association* 24 (3): 556–67. doi:10.1089/thy.2013.0284.

Richards, V. P., T. W. Greig, P. A. Fair, S. D. McCulloch, C. Politz, A. Natoli, C. A. Driscoll, et al. 2013. "Patterns of Population Structure for Inshore Bottlenose Dolphins along the Eastern United States." *Journal of Heredity* 104 (6): 765–78. doi:10.1093/jhered/est070.

Richardson, David B. 2009. "Exposure to Ionizing Radiation in Adulthood and Thyroid Cancer Incidence." *Epidemiology (Cambridge, Mass.)* 20 (2): 181–87. doi:10.1097/EDE.0b013e318196ac1c.

Ries, L. A., P. A. Wingo, D. S. Miller, H. L. Howe, H. K. Weir, H. M. Rosenberg, S. W. Vernon, K. Cronin, and B. K. Edwards. 2000. "The Annual Report to the Nation on the Status of Cancer, 1973-1997, with a Special Section on Colorectal Cancer." *Cancer* 88 (10): 2398–2424.

Ritchie, Marylyn D., Lance W. Hahn, Nady Roodi, L. Renee Bailey, William D. Dupont, Fritz F. Parl, and Jason H. Moore. 2001. "Multifactor-Dimensionality Reduction Reveals High-Order Interactions among Estrogen-Metabolism Genes in Sporadic Breast Cancer." *The American Journal of Human Genetics* 69 (1): 138–47. doi:10.1086/321276.

Robnik-Šikonja, M., and I Kononenko. 2003. "Theoretical and Empirical Analysis of Relieff and Rrelieff." 53 (1–2): 23–69.

Ross, Douglas S. 2006. "Editorial: Predicting Thyroid Malignancy." *The Journal of Clinical Endocrinology and Metabolism* 91 (11): 4253–55. doi:10.1210/jc.2006-1772.

Rueda, Karina, Jorge Eduardo Trujillo, Julio César Carranza, and Gustavo Adolfo Vallejo. 2014. "Transmisión Oral de Trypanosoma Cruzi: Un Nuevo Escenario Epidemiológico de La Enfermedad de Chagas En Colombia Y Otros Países Suramericanos." *Biomédica* 34 (4). doi:10.7705/biomedica.v34i4.2204.

Sasaki, Hitoshi, Regina Rosales, and Yuichiro Tabaru. 2003. "Host Feeding Profiles of Rhodnius Prolixus and Triatoma Dimidiata in Guatemala (Hemiptera: Reduviidae: Triatominae)." *Medical Entomology and Zoology* 54 (3): 283–89. doi:10.7601/mez.54.283.

Schofield, C.J, and J-P Dujardin. 1997. “Chagas Disease Vector Control in Central America.” *Parasitology Today* 13 (4): 141–44. doi:10.1016/S0169-4758(97)89811-0.

Shaw, Nicola T. 2012. “Geographical Information Systems and Health: Current State and Future Directions.” *Healthcare Informatics Research* 18 (2): 88–96. doi:10.4258/hir.2012.18.2.88.

Smith, Stephen F. 1980. “A Learning System Based on Genetic Adaptive Algorithms.” University of Pittsburgh.

———. 1983. “Flexible Learning of Problem Solving Heuristics through Adaptive Search.” In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, 8:422–25. Los Altos, California: William Kaufman Inc.

———. 1984. “Adaptive Learning Systems.” In *Expert Systems: Principles and Case Studies*, 169–89. London, UK: Chapman & Hall, Ltd.

Smith, Stephen L., and Jon Timmis. 2008. “An Immune Network Inspired Evolutionary Algorithm for the Diagnosis of Parkinson’s Disease.” *Biosystems* 94 (1–2): 34–46. doi:10.1016/j.biosystems.2008.05.024.

Smouse, Peter E, and Rod Peakall. 1999. “Spatial Autocorrelation Analysis of Individual Multiallele and Multilocus Genetic Structure.” *Heredity* 82 (5): 561–73. doi:10.1038/sj.hdy.6885180.

Sprague, Brian L., Shaneda Warren Andersen, and Amy Trentham-Dietz. 2008. “Thyroid Cancer Incidence and Socioeconomic Indicators of Health Care Access.” *Cancer Causes & Control: CCC* 19 (6): 585–93. doi:10.1007/s10552-008-9122-0.

Stanaway, Jeffrey D., and Gregory Roth. 2015. “The Burden of Chagas Disease.” *Global Heart* 10 (3): 139–44. doi:10.1016/j.gheart.2015.06.001.

“Standard Populations (Millions) for Age-Adjustment.” 2017. September 30. http://seer.cancer.gov/stdpopulations/.

Stevens, L., M. C. Monroy, A. G. Rodas, R. M. Hicks, D. E. Lucero, L. A. Lyons, and P. L. Dorn. 2015. “Migration and Gene Flow Among Domestic Populations of the Chagas Insect Vector Triatoma Dimidiata (Hemiptera: Reduviidae) Detected by Microsatellite Loci.” *Journal of Medical Entomology* 52 (3): 419–28. doi:10.1093/jme/tjv002.

Tabaru, Yuichiro, Carlota Monroy, Antonieta Rodas, Mildred Mejia, and Regina Rosales. 1998. “Chemical Control of Triatoma Dimidiata and Rhodnius Prolixus (Reduviidae : Triatominae), the Principal Vectors of Chagas’ Disease in Guatemala.” *Medical Entomology and Zoology* 49 (2): 87–92. doi:10.7601/mez.49.87.

———. 1999. “The Geographical Distribution of Vectors of Chagas’ Disease and Populations at Risk of Infection in Guatemala.” *Medical Entomology and Zoology* 50 (1): 9–17. doi:10.7601/mez.50.9_1.

Thornton-Wells, Tricia A., Jason H. Moore, and Jonathan L. Haines. 2004. “Genetics, Statistics and Human Disease: Analytical Retooling for Complexity.” *Trends in Genetics* 20 (12): 640–47. doi:10.1016/j.tig.2004.09.007.

Tukey, John Wilder. 1977. *Exploratory Data Analysis*. Addison-Wesley Series in Behavioral Science. Reading, Mass: Addison-Wesley Pub. Co.

Urbanowicz, Ryan, Ambrose Granizo-Mackenzie, and Jason Moore. 2012. "Instance-Linked Attribute Tracking and Feedback for Michigan-Style Supervised Learning Classifier Systems." In , 927. ACM Press. doi:10.1145/2330163.2330291.

Urbanowicz, Ryan J., and Jason H. Moore. 2009. "Learning Classifier Systems: A Complete Introduction, Review, and Roadmap." *Journal of Artificial Evolution and Applications* 2009: 1–25. doi:10.1155/2009/736398.

———. 2010. "The Application of Michigan-Style Learning Classifier Systems to Address Genetic Heterogeneity and Epistasis in Association Studies." In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, 195. ACM Press. doi:10.1145/1830483.1830518.

———. 2015. "ExSTraCS 2.0: Description and Evaluation of a Scalable Learning Classifier System." *Evolutionary Intelligence* 8 (2–3): 89–116. doi:10.1007/s12065-015-0128-8.

Urbanowicz, Ryan John, Angeline S Andrew, Margaret Rita Karagas, and Jason H Moore. 2013. "Role of Genetic Heterogeneity and Epistasis in Bladder Cancer Susceptibility and Outcome: A Learning Classifier System Approach." *Journal of the American Medical Informatics Association* 20 (4): 603–12. doi:10.1136/amiajnl-2012-001574.

Valença-Barbosa, Carolina, Marli M. Lima, Otília Sarquis, Claudia M. Bezerra, and Fernando Abad-Franch. 2014. "Modeling Disease Vector Occurrence When Detection Is Imperfect II: Drivers of Site-Occupancy by Synanthropic Triatoma Brasiliensis in the Brazilian Northeast." Edited by Martin Donnelly. *PLoS Neglected Tropical Diseases* 8 (5): e2861. doi:10.1371/journal.pntd.0002861.

Vázquez-Martínez, María Guadalupe, Blanca Elva Cirerol-Cruz, José Luis Torres-Estrada, and Mario Henry Rodríguez López. 2014. "Potential for Entomopathogenic Fungi to Control Triatoma Dimidiata (Hemiptera: Reduviidae), a Vector of Chagas Disease in Mexico." *Revista Da Sociedade Brasileira de Medicina Tropical* 47 (6): 716–22. doi:10.1590/0037-8682-0193-2014.

"Vermont Department of Banking ISaHCA." 2010. Vermont Household Health Insurance Survey (2009). Presentation to the State Legislature. Market Decisions.

"Vermont Population Estimates." 2017. September 30. http:// healthvermont.gov/research/pop_estimate.aspx.

Volis, Sergei, Minshu Song, Yong-Hong Zhang, and Irina Shulgina. 2013. "Fine-Scale Spatial Genetic Structure in Emmer Wheat and the Role of Population Range Position." *Evolutionary Biology*, August. doi:10.1007/s11692-013-9256-1.

Waleckx, Etienne, Sébastien Gourbière, and Eric Dumonteil. 2015. "Intrusive versus Domiciliated Triatomines and the Challenge of Adapting Vector Control Practices against Chagas Disease." *Memórias Do Instituto Oswaldo Cruz* 110 (3): 324–38. doi:10.1590/0074-02760140409.

Ward, Elizabeth M., Ahmedin Jemal, and Amy Chen. 2010. "Increasing Incidence of Thyroid Cancer: Is Diagnostic Scrutiny the Sole Explanation?" *Future Oncology (London, England)* 6 (2): 185–88. doi:10.2217/fon.09.161.

Ward, Mary H., Briseis A. Kilfoy, Peter J. Weyer, Kristin E. Anderson, Aaron R. Folsom, and James R. Cerhan. 2010. "Nitrate Intake and the Risk of Thyroid

Cancer and Thyroid Disease." *Epidemiology (Cambridge, Mass.)* 21 (3): 389–95. doi:10.1097/EDE.0b013e3181d6201d.

Wartofsky, Leonard. 2010. "Increasing World Incidence of Thyroid Cancer: Increased Detection or Higher Radiation Exposure?" *Hormones (Athens, Greece)* 9 (2): 103–8.

Weeks, E. N. I., C. Cordón-Rosales, C. Davies, S. Gezan, M. Yeo, and M. M. Cameron. 2013. "Risk Factors for Domestic Infestation by the Chagas Disease Vector, Triatoma Dimidiata in Chiquimula, Guatemala." *Bulletin of Entomological Research* 103 (06): 634–43. doi:10.1017/S000748531300014X.

White, David R., James McDermott, Mauro Castelli, Luca Manzoni, Brian W. Goldman, Gabriel Kronberger, Wojciech Jaśkowski, Una-May O'Reilly, and Sean Luke. 2013. "Better GP Benchmarks: Community Survey Results and Proposals." *Genetic Programming and Evolvable Machines* 14 (1): 3–29. doi:10.1007/s10710-012-9177-2.

Whitlaw, J. T., and B. N. Chaniotis. 1978. "Palm Trees and Chagas' Disease in Panama." *The American Journal of Tropical Medicine and Hygiene* 27 (5): 873–81.

Wilson, I. D. 2009. "Drugs, Bugs, and Personalized Medicine: Pharmacometabonomics Enters the Ring." *Proceedings of the National Academy of Sciences* 106 (34): 14187–88. doi:10.1073/pnas.0907721106.

Wilson, Norbert L.W., Bradley J. Rickard, Rachel Saputo, and Shuay-Tsyr Ho. 2017. "Food Waste: The Role of Date Labels, Package Size, and Product Category." *Food Quality and Preference* 55 (January): 35–44. doi:10.1016/j.foodqual.2016.08.004.

Wilson, Stewart W. 1987a. "Classifier Systems and the Animat Problem." *Machine Learning* 2 (3): 199–228.

———. 1987b. "Quasi-Darwinian Learning in a Classifier System." In *Proceedings of the Fourth International Machine Learning Workshop*, 4:59–65.

———. 1995. "Classifier Fitness Based on Accuracy." *Evolutionary Computation* 3 (2): 149–75. doi:10.1162/evco.1995.3.2.149.

World Health Organization, ed. 2002. *Control of Chagas Disease: Second Report of the WHO Expert Committee*. Control of Chagas Disease, 2.2002. Geneva: WHO.

Yoshioka, K., J. Nakamura, B. Perez, D. Tercero, L. Perez, and Y. Tabaru. 2015. "Effectiveness of Large-Scale Chagas Disease Vector Control Program in Nicaragua by Residual Insecticide Spraying Against Triatoma Dimidiata." *American Journal of Tropical Medicine and Hygiene* 93 (6): 1231–39. doi:10.4269/ajtmh.15-0403.

Young Kim, Eun, and Youn-Kyung Kim. 2004. "Predicting Online Purchase Intentions for Clothing Products." *European Journal of Marketing* 38 (7): 883–97. doi:10.1108/03090560410539302.

Yousefi, Saleh, Hamidreza Moradi, Jan Boll, and Sarah Schönbrodt-Stitt. 2016. "Effects of Road Construction on Soil Degradation and Nutrient Transport in Caspian Hyrcanian Mixed Forests." *Geoderma* 284 (December): 103–12. doi:10.1016/j.geoderma.2016.09.002.

Yu, Guo-Pei, James Chun-Lun Li, Daniel Branovan, Steven McCormick, and Stimson P. Schantz. 2010. "Thyroid Cancer Incidence and Survival in the National Cancer Institute Surveillance, Epidemiology, and End Results Race/Ethnicity Groups." *Thyroid: Official Journal of the American Thyroid Association* 20 (5): 465–73. doi:10.1089/thy.2008.0281.

Zeledón, R., V. M. Guardia, A. Zúñiga, and J. C. Swartzwelder. 1970. "Biology and Ethology of Triatoma Dimidiata (Latreille, 1811). I. Life Cycle, Amount of Blood Ingested, Resistance of Starvation, and Size of Adults." *Journal of Medical Entomology* 7 (3): 313–19.

Zeledon, R., G. Solano, A. Zuniga, and J. C. Swartzwelder. 1973. "Biology and Ethology of Triatoma Dimidiata (Latreille, 1811). III. Habitat and Blood Sources." *Journal of Medical Entomology* 10 (4): 363–70. doi:10.1093/jmedent/10.4.363.

Zeledón, Rodrigo. 2004. "Some Historical Facts and Recent Issues Related to the Presence of Rhodnius Prolixus (STAL, 1859) (HEMIPTERA: REDUVIIDAE) in Central America." *Entomología Y Vectores* 11 (2): 233–46.

Zeledón, Rodrigo, Nidia Calvo, Víctor M Montenegro, Elias Seixas Lorosa, and Carolina Arévalo. 2005. "A Survey on Triatoma Dimidiata in an Urban Area of the Province of Heredia, Costa Rica." *Memórias Do Instituto Oswaldo Cruz* 100 (6): 607–12. doi:10.1590/S0074-02762005000600002.

Zeledón, Rodrigo, Víctor M Montenegro, and Oswaldo Zeledón. 2001. "Evidence of Colonization of Man-Made Ecotopes by Triatoma Dimidiata (Latreille, 1811) in Costa Rica." *Memórias Do Instituto Oswaldo Cruz* 96 (5). doi:10.1590/S0074-02762001000500012.

Zeledón, Rodrigo, and Julio C Rojas. 2006. "Environmental Management for the Control of Triatoma Dimidiata (Latreille, 1811), (Hemiptera: Reduviidae) in Costa Rica: A Pilot Project." *Memórias Do Instituto Oswaldo Cruz* 101 (4): 379–86. doi:10.1590/S0074-02762006000400006.

Zeledón, Rodrigo, Julio C Rojas, Andrea Urbina, Marlen Cordero, Sue H Gamboa, Elias S Lorosa, and Sergio Alfaro. 2008. "Ecological Control of Triatoma Dimidiata (Latreille, 1811): Five Years after a Costa Rican Pilot Project." *Memórias Do Instituto Oswaldo Cruz* 103 (6): 619–21. doi:10.1590/S0074-02762008000600020.

Zeledón, Rodrigo, Jesús A Ugalde, and Luis A Paniagua. 2001. "Entomological and Ecological Aspects of Six Sylvatic Species of Triatomines (Hemiptera, Reduviidae) from the Collection of the National Biodiversity Institute of Costa Rica, Central America." *Memórias Do Instituto Oswaldo Cruz* 96 (6): 757–64. doi:10.1590/S0074-02762001000600002.

Zhang, Yawei, Grace L. Guo, Xuesong Han, Cairong Zhu, Briseis A. Kilfoy, Yong Zhu, Peter Boyle, and Tongzhang Zheng. 2008. "Do Polybrominated Diphenyl Ethers (PBDEs) Increase the Risk of Thyroid Cancer?" *Bioscience Hypotheses* 1 (4): 195–99. doi:10.1016/j.bihy.2008.06.003.

Zhu, Cairong, Tongzhang Zheng, Briseis A. Kilfoy, Xuesong Han, Shuangge Ma, Yue Ba, Yana Bai, Rong Wang, Yong Zhu, and Yawei Zhang. 2009. "A Birth Cohort Analysis of the Incidence of Papillary Thyroid Cancer in the United States, 1973-

2004." *Thyroid: Official Journal of the American Thyroid Association* 19 (10): 1061–66. doi:10.1089/thy.2008.0342.

CHAPTER 8: APPENDIX

8.1 Matlab® Code

8.1.1 Convert Data to Ones and Zeros (Data2Binary)

```
function [DataBin, DataType, FeatVals, FeatInd, NaNMask, DataSum]=...
        Data2Binary(Data,Output,ContData,CatData,UniqCatData)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
% October 14, 2016
% Last updated: October 14, 2016
%
% Data2Binary converts input data to binary and supplies some summary
% information for each of the features.
%
% Inputs:
% Data = A matrix of ones and zeros where each row represents an
%       observation and each column represents a feature.
% Output = The output class for each observation.
% ContData = The feature index for all continuous or discrete input
%        features.
% CatData = The feature index for all nominal features that have more than
%       two categories and/or the user desires an or statement between.
```

```
% UniqCatData = The feature index of any feature that the user desires to
%              not have a range or an or statement between categories
%              and/or there are only 2 unique values for the feature.
%
% Outputs:
% DataBin = The data in logical binary matrix form where each row
%           represents an observation and each column represents a value
%           for a feature.
% DataType = A categorical reference to the type of data for each feature.
%            1 = ContData, 2 = CatData, 3 = UniqCatData
% FeatVals = The unique values for each of the features.
% FeatInd = The column index in the DataBin for each feature where each
%           column indexed for a given feature is represented by the
%           FeatVals.
% NaNMask = A NaN mask for the original dataset.  This represents which
%           features for a given observation have a NaN.
% DataSum = Is a structure array with an overall summary of the data for
%           each feature. Each row represents a unique output class and
%           each column represents a feature.
%           DataSum.NaNs = counts the number of NaNs for each output class
%                          for each feature.
%           DataSum.Tabs = is a tabulation for the number of times values
```

```
%                         are present for each output class.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine the number of features
NumFeat=size(Data,2);
% Create a vector for dataype where 1 = continuous data, 2 = categorical
% data, 3 = unique categorical data.
DataType=NaN(1,NumFeat);
% Now set the DataTypes
DataType(ContData)=1;
DataType(CatData)=2;
DataType(UniqCatData)=3;
% Now for each feature convert the features to binary data
% Determine the output classes
OutClasses=unique(Output);
% Determine the number of output classes
NumOut=length(OutClasses);
% For efficiency
DataSum.NaNs=NaN(NumOut,NumFeat); % # NaNs per feature class
DataSum.Tabs=cell(NumOut,NumFeat); % Tabulate of feature vals
FeatVals=cell(1,NumFeat);
NumUniVals=NaN(1,NumFeat);
```

```
FeatInd=cell(1,NumFeat);
% Determine the NaNMask
NaNMask=isnan(Data);
% Create a counter
count=1;
% First go through for each feature and determine the number of categories
% Note if the number of unique values is 2 then convert the datatype to 3
for i=1:NumFeat
    % First determine the unique values for the feature
    FeatVals(i)={unique(Data(~NaNMask(:,i),i))};
    % Determine the number of unique values
    NumUniVals(i)=length(FeatVals{i});
    % Determine the current feature index
    FeatInd(i)={count:sum(NumUniVals(1:i))};
    % update the counter
    count=sum(NumUniVals(1:i))+1;
    if NumUniVals==2
        % Ensure that the current DataType is set to 3
        DataType(i)=3;
    elseif NumUniVals<2
        % Then end the algorithm and display the warning that the ith
        % feature has less then 2 unique values and is not useful
```

```
        disp(['Warning Feature # ' num2str(i) ...
                        ' Has < 2 Unique Values. Please Remove.'])
        break
    end
    % For each feature determine the number of NaNs for each class
    for j=1:NumOut
        % Create a TempMask for the current class
        TempMask=Output==OutClasses(j);
        % Count the number of NaNs for the current class
        DataSum.NaNs(j,i)=sum(NaNMask(TempMask,i));
        % Tabulate the values for each class
        DataSum.Tabs(j,i)={tabulate(Data(TempMask,i))};
        clear TempMask
    end
end
clear count
clear NumOut
clear OutClasses
clear j
clear i
% Now convert the Data to a binary logical matrix
DataBin=false(size(Data,1),sum(NumUniVals));
```

```
% Now run a for loop to set the values of each input
for i=1:NumFeat
    % Extract the ith features Values
    CurVals=FeatVals{i};
    % Extract the Current Index
    CurInd=FeatInd{i};
    % Now for each value place a true in the new binary input data
    for j=1:NumUniVals(i)
        % Create a Mask for the data with the jth value for the ith feature
        TempMask=Data(:,i)==CurVals(j);
        % Set the current values to true
        DataBin(TempMask,CurInd(j))=true();
        clear TempMask
    end
    clear CurVals
    clear CurInd
end
clear NumFeat
clear i
clear j
clear NumUniVals
```

8.1.2 Conjunctive Clause Evolutionary Algorithm (CCEA)

```
function [ArchCCs, ArchCCFeats, ArchCCFit, ArchCCFitComp, ArchCCOrder,...
        ArchCCMatchLocs, ArchCCAge, NonArchCCs, NonArchCCFeats,...
        NonArchCCFit, NonArchCCFitComp, NonArchCCOrder,...
        NonArchCCMatchLocs, NonArchCCAge, Param, CCstats]=...
         CCEA(Param,DataBin,NaNMask,TargetClass)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
% October 19, 2016
% Last Updated: October 19, 2016
%
% CCEA is the evolutionary algorithm for the conjunctive clauses.
%
% Inputs:
% Param = General parameters for the evolutionary algorithm. The general
%         parameters are set up as a structure and the following
%         paramaters are necessary to run the algorithm.
%         .NumNewPop = The number of offspring created every time a new
%                 randomly created population of offspring is
%                 created.
%         .TotGens = The total number of generations to run the
```

```
%                 algorithm.
%         .DataType = The DataType of each feature. This is a vector
%                 where each value represents the feature data type
%                 (1 = a continuous or discrete feature, 2 = a
%                 categorical feature where more than one category
%                 can be present in the conjunctive clause, 3 = a
%                 feature where only one value or category can be
%                 present (e.g., binary feature)).
%         .Thresh = A matrix with the initial threshold settings. The
%                 matrix has 4 columns with the first column containing
%                 all of the orders of the conjunctive clause that the
%                 user is interested in. For instance if the user wants
%                 to explore conjunctive clauses of orders 1 - 6, then
%                 each row represents the order with the exception of
%                 the last row where the order is the order listed and
%                 any order greater than 6. This way the algorithm does
%                 not assume an order. The second column is the initial
%                 probability threshold [0, 1]. The third column is the
%                 minimum number of conjunctive clauses the user wants
%                 to save for each order.  The fourth and final column
%                 is the maximum number of conjunctive clauses the
%                 user wants to archive for the given order. If the
```

```
%               maximum is exceeded then the threshold for the given
%               order is replaced.
%         .MaxNumFeat = The maximum number of features allowed during
%                 crossover. No offspring that is the product of
%                 crossover will have more features than this
%                 number.
%         .FeatInd = The index for each feature. Each cell represents a
%                 feature and the numbers in the cell represent the
%                 columns in the binary data where the feature is
%                 represented.
%         .ALna = The number of non-archived age layers
%         .GENn = The number of generations until a novel population is
%               introduced.
%         .NonArchLMax = The maximum population for each non-archived
%                 layer.
%         .ArchOff = The maximum number of archived offsrping that will
%                 undergo mutation or crossover.
%         .Px = The probability of crossover.
%         .Pwc = The probability that a feature will be turned into a
%               wild card during mutation.
%         .Pm = the probability that an individual feature will undergo
%               mutation.
```

```
%          .TournSize = The size of the tournament with relacement of
%                       selecting the mate for crossover.
%          .BestFit = Set to true if the user wants to record the best
%                     fitness of each order each generation. Otherwise set
%                     to false.
% DataBin = The data as a binary logical matrix.
% NaNMask = A logical mask of the location of the NaN values in the
%                     dataset.
% TargetClass = A logical vector of the observations that have the output
%               class.
% CCstats = Structure array statistics on the conjunctive clauses.
%
% Outputs:
% ArchCCs = The archived conjunctive clauses.
% ArchCCFeats = The features present in the archived conjunctive clauses.
% ArchCCFit = The fitness of the archived conjunctive clauses.
% ArchCCFitComp = The raw inputs used to calculate the archived fitness.
% ArchCCOrder = The order of the archived conjunctive clause.
% ArchCCMatchLocs = A logical matrix that shows which observations the
%                   archived conjunctive clauses match.
% ArchCCAge = The age of the archived conjunctive clause.
% NonArchCCs = The non-archived conjunctive clauses.
```

```
% NonArchCCFeats = The features present in the non-archived conjunctive
%                  clauses.
% NonArchCCFit = The fitness of the non-archived conjunctive clauses.
% NonArchCCFitComp = The raw inputs used to calculate the non-archived
%           fitness.
% NonArchCCOrder = The order of the non-archived conjunctive clause.
% NonArchCCMatchLocs = A logical matrix that shows which observations the
%            non-archived conjunctive clauses match.
% NonArchCCAge = The age of the non-archived conjunctive clause.
% Param = A structure array with many of the parameters for the algorithm.
% CCstats = Structure array statistics on the conjunctive clauses.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For efficiency
CCstats.EvalsAll=zeros(Param.TotGens,Param.MaxNumFeat);
CCstats.EvalsArchive=zeros(Param.TotGens,Param.MaxNumFeat);
CCstats.EvalsNonArchive=zeros(Param.TotGens,Param.MaxNumFeat);
if Param.BestFit
   % Then the best fitness for each order for each generation will be
   % recorded
   CCstats.BestFit=NaN(Param.TotGens,Param.MaxNumFeat);
end
```

```
% Determine the number of observation and features in the dataset
[Param.NumObs, Param.NumFeat]=size(NaNMask);
% Determine the number of binary columns in the data
Param.NumBinCols=size(DataBin,2);
% Determine the number of Target Class observations
Param.TotTargetClass=sum(TargetClass);
% Determine the number of features that are not NaN for the target class
Param.TargetNotNaNMask=~NaNMask(TargetClass,:);
Param.TargetNumNotNaN=sum(Param.TargetNotNaNMask,2);
% Extract the target observations
Param.TargetObs=DataBin(TargetClass,:);
% Set the current generation parameter
Param.CurGen=1;
% Create a roulette wheel probability distribution for selecting each
% observation to serve as the basis for the conjunctive clauses
Param.PrObsSel=ones(Param.TotTargetClass,1)*1/Param.TotTargetClass;
% Create an initial population of conjunctive clauses
[NewCCs, NewCCFeats, NewCCFit, NewCCFitComp, NewCCOrder, ...
        NewCCMatchLocs, ArchiveMask, CCstats]=...
            CCPopInit(Param,DataBin,NaNMask,TargetClass,CCstats);
% Seperate the CCs into Archive and Non-archive
ArchCCs=NewCCs(ArchiveMask,:);
```

```
ArchCCFeats=NewCCFeats(ArchiveMask,:);
ArchCCFit=NewCCFit(ArchiveMask);
ArchCCFitComp.TotObs=NewCCFitComp.TotObs(ArchiveMask);
ArchCCFitComp.TotObsT=NewCCFitComp.TotObsT(ArchiveMask);
ArchCCFitComp.TotCCObs=NewCCFitComp.TotCCObs(ArchiveMask);
ArchCCFitComp.TotCCObsT=NewCCFitComp.TotCCObsT(ArchiveMask);
ArchCCOrder=NewCCOrder(ArchiveMask);
ArchCCMatchLocs=NewCCMatchLocs(:,ArchiveMask);
% Set the age of the CCs to 1
ArchCCAge=ones(size(ArchCCFit));
% Now seperate the NonarchiveCCs
NonArchCCs=NewCCs(~ArchiveMask,:);
NonArchCCFeats=NewCCFeats(~ArchiveMask,:);
NonArchCCFit=NewCCFit(~ArchiveMask);
NonArchCCFitComp.TotObs=NewCCFitComp.TotObs(~ArchiveMask);
NonArchCCFitComp.TotObsT=NewCCFitComp.TotObsT(~ArchiveMask);
NonArchCCFitComp.TotCCObs=NewCCFitComp.TotCCObs(~ArchiveMask);
NonArchCCFitComp.TotCCObsT=NewCCFitComp.TotCCObsT(~ArchiveMask);
NonArchCCOrder=NewCCOrder(~ArchiveMask);
NonArchCCMatchLocs=NewCCMatchLocs(:,~ArchiveMask);
% Set the age of the CCs to 1
NonArchCCAge=ones(size(NonArchCCFit));
```

```
clear NewCCs
clear NewCCFeats
clear NewCCFit
clear NewCCFitComp
clear NewCCOrder
clear NewCCMatchLocs
clear ArchiveMask
% Run the population reduction algorithm
[ArchCCs, ArchCCFeats, ArchCCFit, ArchCCFitComp, ArchCCOrder,...
        ArchCCMatchLocs, ArchCCAge, NonArchCCs, NonArchCCFeats,...
        NonArchCCFit, NonArchCCFitComp, NonArchCCOrder,...
        NonArchCCMatchLocs, NonArchCCAge, Param, CCstats]=...
          CCreducepop(ArchCCs,ArchCCFeats,ArchCCFit,ArchCCFitComp,...
            ArchCCOrder,ArchCCMatchLocs,ArchCCAge,NonArchCCs,...
            NonArchCCFeats,NonArchCCFit,NonArchCCFitComp,...
            NonArchCCOrder,NonArchCCMatchLocs,NonArchCCAge,Param,...
            CCstats);
% Now work on the for loop for the ALPS like evolution
for gen=2:Param.TotGens
   % Set the current current gen parameter
   Param.CurGen=gen;
   % Increase the age of the non-archived population
```

```
NonArchCCAge=NonArchCCAge+1;
% Now determine if a new population should be added
if mod(gen,Param.GENn)~=0
   % Then just perform crossover or mutation on population
   [ArchCCs, ArchCCFeats, ArchCCFit, ArchCCFitComp, ArchCCOrder,...
      ArchCCMatchLocs, ArchCCAge, NonArchCCs, NonArchCCFeats,...
      NonArchCCFit, NonArchCCFitComp, NonArchCCOrder,...
      NonArchCCMatchLocs, NonArchCCAge, Param, CCstats]=...
       CCEvolution(ArchCCs,ArchCCFeats,ArchCCFit,ArchCCFitComp,...
          ArchCCOrder,ArchCCMatchLocs,ArchCCAge,...
          NonArchCCs,NonArchCCFeats,NonArchCCFit,NonArchCCFitComp,...
          NonArchCCOrder,NonArchCCMatchLocs,NonArchCCAge,...
          DataBin,NaNMask,TargetClass,Param,CCstats);
   % Clean conjunctive clauses
   [ArchCCs, ArchCCFeats, ArchCCFit, ArchCCFitComp, ArchCCOrder,...
       ArchCCMatchLocs, ArchCCAge, NonArchCCs, NonArchCCFeats,...
       NonArchCCFit, NonArchCCFitComp, NonArchCCOrder,...
       NonArchCCMatchLocs, NonArchCCAge, Param, CCstats]=...
         CCreducepop(ArchCCs,ArchCCFeats,ArchCCFit,ArchCCFitComp,...
            ArchCCOrder,ArchCCMatchLocs,ArchCCAge,NonArchCCs,...
            NonArchCCFeats,NonArchCCFit,NonArchCCFitComp,...
            NonArchCCOrder,NonArchCCMatchLocs,NonArchCCAge,Param,...
```

```
            CCstats);
else
    % Then add a new population and perform  crossover or mutation on
    % population
    % first perform mutation or crossover
    [ArchCCs, ArchCCFeats, ArchCCFit, ArchCCFitComp, ArchCCOrder,...
      ArchCCMatchLocs, ArchCCAge, NonArchCCs, NonArchCCFeats,...
      NonArchCCFit, NonArchCCFitComp, NonArchCCOrder,...
      NonArchCCMatchLocs, NonArchCCAge, Param, CCstats]=...
       CCEvolution(ArchCCs,ArchCCFeats,ArchCCFit,ArchCCFitComp,...
          ArchCCOrder,ArchCCMatchLocs,ArchCCAge,...
          NonArchCCs,NonArchCCFeats,NonArchCCFit,NonArchCCFitComp,...
          NonArchCCOrder,NonArchCCMatchLocs,NonArchCCAge,...
          DataBin,NaNMask,TargetClass,Param,CCstats);
    if ~isempty(ArchCCFit)
       % Calculate a new probability of selecting an observation for
       % the template of a new conjunctive clause. Increase the odds
       % of selecting a target observation that is underrepresented in
       % the archive
       % First sum the total number of times that a target observation
       % is covered in the archive population.
       TotObsArchive=sum(ArchCCMatchLocs(TargetClass,:),2);
```

```
    % Determine the maximum sum
    MaxSum=max(TotObsArchive);
    % Now subtract TotObsArchive from MaxSum-1
    DiffSum=(MaxSum+1)-TotObsArchive;
    clear TotObsArchive
    clear MaxSum
    % Now normalize to get a total probability of 1
    Param.PrObsSel=DiffSum/sum(DiffSum);
    clear DiffSum
end
% Now create a new population of CCs
[NewCCs, NewCCFeats, NewCCFit, NewCCFitComp, NewCCOrder, ...
        NewCCMatchLocs, ArchiveMask, CCstats]=...
            CCPopInit(Param,DataBin,NaNMask,TargetClass,CCstats);
% First set the age of the NewCCs to one
NewCCAge=ones(size(NewCCFit));
% Now combine the new CCs with the population of CCs
ArchCCs=[ArchCCs; NewCCs(ArchiveMask,:)];
ArchCCFeats=[ArchCCFeats; NewCCFeats(ArchiveMask,:)];
ArchCCFit=[ArchCCFit; NewCCFit(ArchiveMask)];
ArchCCFitComp.TotObs=[ArchCCFitComp.TotObs;...
                        NewCCFitComp.TotObs(ArchiveMask)];
```

```
ArchCCFitComp.TotObsT=[ArchCCFitComp.TotObsT;...
                    NewCCFitComp.TotObsT(ArchiveMask)];
ArchCCFitComp.TotCCObs=[ArchCCFitComp.TotCCObs;...
                    NewCCFitComp.TotCCObs(ArchiveMask)];
ArchCCFitComp.TotCCObsT=[ArchCCFitComp.TotCCObsT;...
                    NewCCFitComp.TotCCObsT(ArchiveMask)];
ArchCCOrder=[ArchCCOrder; NewCCOrder(ArchiveMask)];
ArchCCMatchLocs=[ArchCCMatchLocs NewCCMatchLocs(:,ArchiveMask)];
ArchCCAge=[ArchCCAge; NewCCAge(ArchiveMask)];
NonArchCCs=[NonArchCCs; NewCCs(~ArchiveMask,:)];
NonArchCCFeats=[NonArchCCFeats; NewCCFeats(~ArchiveMask,:)];
NonArchCCFit=[NonArchCCFit; NewCCFit(~ArchiveMask)];
NonArchCCFitComp.TotObs=[NonArchCCFitComp.TotObs;...
                    NewCCFitComp.TotObs(~ArchiveMask)];
NonArchCCFitComp.TotObsT=[NonArchCCFitComp.TotObsT;...
                    NewCCFitComp.TotObsT(~ArchiveMask)];
NonArchCCFitComp.TotCCObs=[NonArchCCFitComp.TotCCObs;...
                    NewCCFitComp.TotCCObs(~ArchiveMask)];
NonArchCCFitComp.TotCCObsT=[NonArchCCFitComp.TotCCObsT;...
                    NewCCFitComp.TotCCObsT(~ArchiveMask)];
NonArchCCOrder=[NonArchCCOrder; NewCCOrder(~ArchiveMask)];
NonArchCCMatchLocs=[NonArchCCMatchLocs ...
```

```
                                NewCCMatchLocs(:,~ArchiveMask)];
        NonArchCCAge=[NonArchCCAge; NewCCAge(~ArchiveMask)];
        clear NewCCs
        clear NewCCFeats
        clear NewCCFit
        clear NewCCFitComp
        clear NewCCOrder
        clear NewCCMatchLocs
        clear NewCCAge
        clear ArchiveMask
        % Clean conjunctive clauses
        [ArchCCs, ArchCCFeats, ArchCCFit, ArchCCFitComp, ArchCCOrder,...
           ArchCCMatchLocs, ArchCCAge, NonArchCCs, NonArchCCFeats,...
           NonArchCCFit, NonArchCCFitComp, NonArchCCOrder,...
           NonArchCCMatchLocs, NonArchCCAge, Param, CCstats]=...
             CCreducepop(ArchCCs,ArchCCFeats,ArchCCFit,ArchCCFitComp,...
               ArchCCOrder,ArchCCMatchLocs,ArchCCAge,NonArchCCs,...
               NonArchCCFeats,NonArchCCFit,NonArchCCFitComp,...
               NonArchCCOrder,NonArchCCMatchLocs,NonArchCCAge,Param,...
               CCstats);
    end
end
```

8.1.2.1 Conjunctive Clause Population Initialization (CCPopInit)

```
function [NewCCs, NewCCFeats, NewCCFit, NewCCFitComp, NewCCOrder, ...
            NewCCMatchLocs, ArchiveMask, CCstats]=...
               CCPopInit(Param,DataBin,NaNMask,TargetClass,CCstats)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
% October 14, 2016
% Last updated: October 14, 2016
%
% CCPopInit is a population initialization algorithm for the conjunctive
% clauses. It randomly creates conjunctive clauses of various sizes using
% observations from the dataset that has an output equal to the current
% target class.
%
% Inputs:
% Param = A structure array with many of the parameters for the algorithm.
% DataBin = The data as a binary logical matrix.
% NaNMask = A logical mask of the location of the NaN values in the
%         dataset.
% TargetClass = A logical vector of the observations that have the output
%            class.
```

```
% CCstats = Structure array statistics on the conjunctive clauses.
%
% Outputs:
% NewCCs = The newly created conjunctive clauses.
% NewCCFeats = The active features of the new conjunctive clauses.
% NewCCFit = The fitness of the new conjunctive clauses measured using the
%           hypergeometric PMF.
% NewCCFitComp = The inputs for the calculation of NewCCFit.
% NewCCOrder = The order of the conjunctive clause.
% NewCCMatchLocs = A logical matrix of the observations the conjunctive
%                 clause matches. Each row represents an observation and
%                 each column represents a conjunctive clause.
% ArchiveMask = Is a logical vector of the new conjunctive clauses that are
%              to be archived.
% CCstats = Structure array statistics on the conjunctive clauses.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For efficiency
NewCCs=false(Param.NumNewPop,Param.NumBinCols);
NewCCFeats=false(Param.NumNewPop,Param.NumFeat);
NewCCFit=NaN(Param.NumNewPop,1);
NewCCFitComp.TotObs=NaN(Param.NumNewPop,1);
```

```
NewCCFitComp.TotObsT=NaN(Param.NumNewPop,1);
NewCCFitComp.TotCCObs=NaN(Param.NumNewPop,1);
NewCCFitComp.TotCCObsT=NaN(Param.NumNewPop,1);
NewCCOrder=NaN(Param.NumNewPop,1);
NewCCMatchLocs=false(Param.NumObs,Param.NumNewPop);
KeepMask=false(Param.NumNewPop,1);
ArchiveMask=false(Param.NumNewPop,1);
% Create a vector of the feature indices
PosFeats=1:Param.NumFeat;
% Create the conjunctive clauses for each new offspring
for i=1:Param.NumNewPop
    % Create a logical Index of the observation that will be selected
    SelObs=false(Param.TotTargetClass,1);
    % Randomly determine the number of features present in the conjunctive
    % clause
    CurFeatNum=randi(Param.MaxNumFeat);
    % Now determine which observations have the requisite number of current
    % features
    CurObsMask=Param.TargetNumNotNaN>=CurFeatNum;
    % Now recalculate the Param.PrObsSel based on the observations that
    % were eliminated
    CurPrObsSel=Param.PrObsSel(CurObsMask)*1/...
```

```
                    (1-sum(Param.PrObsSel(~CurObsMask)));
% Turn CurPrObsSel into lower bound and upper bound
UBCurPrObsSel=cumsum(CurPrObsSel);
LBCurPrObsSel=[0; UBCurPrObsSel(1:end-1)];
clear CurPrObsSel
% Select a random number (0,1)
CurRand=rand();
% Now determine which observation is between the LB and UB
CurObsSel=CurRand>LBCurPrObsSel&CurRand<=UBCurPrObsSel;
clear CurRand
clear LBCurPrObsSel
clear UBCurPrObsSel
% Update the selected observation
SelObs(CurObsMask)=CurObsSel;
clear CurObsMask
clear CurObsSel
% Now Extract the Current Observation Data
CurObsData=Param.TargetObs(SelObs,:);
% Now randomly select the features for the rule
CurPosFeats=PosFeats(Param.TargetNotNaNMask(SelObs,:));
clear SelObs
CurSelFeatInd=randperm(length(CurPosFeats),CurFeatNum);
```

```
CurSelFeats=CurPosFeats(CurSelFeatInd);
clear CurPosFeats
clear CurSelFeatInd
% Now Extract the feature indices for the current selected features
CurFeatInds=[Param.FeatInd{CurSelFeats}];
% Insert the data into the New CC
NewCCs(i,CurFeatInds)=CurObsData(1,CurFeatInds);
NewCCFeats(i,CurSelFeats)=true();
clear CurObsData
% Now Determine the fitness of the new rule
% First Determine the Total Observations that have data for the
% selected features
TotObs=sum(sum(~NaNMask(:,CurSelFeats),2)==CurFeatNum);
TotObsT=sum(sum(~NaNMask(TargetClass,CurSelFeats),2)==CurFeatNum);
% Now determine which observations the current Conjunctive clause
% Matches
% First add the conjunctive clause to the data
TwosSum=bsxfun(@plus,NewCCs(i,CurFeatInds),DataBin(:,CurFeatInds));
% Now create a twos mask
TwosMask=TwosSum==2;
clear TwosSum
% Now determine the total number of twos
```

```
TotTwos=sum(TwosMask,2);
clear TwosMask
% Now create an observation match mask
NewCCMatchLocs(:,i)=TotTwos==CurFeatNum;
clear TotTwos
% Determine the total number of observations that match and are target
% class
TotCCObs=sum(NewCCMatchLocs(:,i));
TotCCObsT=sum(NewCCMatchLocs(TargetClass,i));
% If TotCCObsT/TotCCObs > TotObsT/TotObs then evaluate the fitness of
% the Conjunctive clause using hypergeometric PMF
if TotCCObsT/TotCCObs>TotObsT/TotObs
   % Calculate the fitness
   NewCCFit(i)=hygepdf(TotCCObsT,TotObs,TotObsT,TotCCObs);
   % Update the keep mask
   KeepMask(i)=true();
   % Determine if the CC is archivable
   % First extract the order mask
   OrderMask=Param.Thresh(:,1)==CurFeatNum;
   if sum(OrderMask)==0
      % Then set the last OrderMask to true
      OrderMask(end)=true();
```

```
    end
    ArchiveMask(i)=NewCCFit(i)<=Param.Thresh(OrderMask,2);
    if ArchiveMask(i)
        % Then record an archived evaluation
        CCstats.EvalsArchive(Param.CurGen,CurFeatNum)=...
                    CCstats.EvalsArchive(Param.CurGen,CurFeatNum)+1;
    else
        % Record a non-archived evaluation
        CCstats.EvalsNonArchive(Param.CurGen,CurFeatNum)=...
                CCstats.EvalsNonArchive(Param.CurGen,CurFeatNum)+1;
    end
    clear OrderMask
else
    NewCCFit(i)=1;
end
% Record the total number of evaluations for the current order
CCstats.EvalsAll(Param.CurGen,CurFeatNum)=...
                CCstats.EvalsAll(Param.CurGen,CurFeatNum)+1;
% Save the fitness component values
NewCCFitComp.TotObs(i)=TotObs;
NewCCFitComp.TotObsT(i)=TotObsT;
NewCCFitComp.TotCCObs(i)=TotCCObs;
```

```
    NewCCFitComp.TotCCObsT(i)=TotCCObsT;
    NewCCOrder(i)=CurFeatNum;
    clear TotObs
    clear TotObsT
    clear TotCCObs
    clear TotCCObsT
    clear CurFeatNum
    clear CurSelFeats
    clear CurFeatInds
end
clear PosFeats
clear i
% Only keep the values in the keep mask
NewCCs=NewCCs(KeepMask,:);
NewCCFeats=NewCCFeats(KeepMask,:);
NewCCFit=NewCCFit(KeepMask,1);
NewCCFitComp.TotObs=NewCCFitComp.TotObs(KeepMask,1);
NewCCFitComp.TotObsT=NewCCFitComp.TotObsT(KeepMask,1);
NewCCFitComp.TotCCObs=NewCCFitComp.TotCCObs(KeepMask,1);
NewCCFitComp.TotCCObsT=NewCCFitComp.TotCCObsT(KeepMask,1);
NewCCOrder=NewCCOrder(KeepMask,1);
NewCCMatchLocs=NewCCMatchLocs(:,KeepMask);
```

```
ArchiveMask=ArchiveMask(KeepMask,1);

clear KeepMask
```

8.1.2.2 Remove Repeat CCs (CCreducepop)

```
function [ArchCCs, ArchCCFeats, ArchCCFit, ArchCCFitComp, ArchCCOrder,...
        ArchCCMatchLocs, ArchCCAge, NonArchCCs, NonArchCCFeats,...
        NonArchCCFit, NonArchCCFitComp, NonArchCCOrder,...
        NonArchCCMatchLocs, NonArchCCAge, Param, CCstats]=...
         CCreducepop(ArchCCs,ArchCCFeats,ArchCCFit,ArchCCFitComp,...
          ArchCCOrder,ArchCCMatchLocs,ArchCCAge,NonArchCCs,...
          NonArchCCFeats,NonArchCCFit,NonArchCCFitComp,...
          NonArchCCOrder,NonArchCCMatchLocs,NonArchCCAge,Param,...
          CCstats)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
% October 18, 2016
% Last updated: October 18, 2016
%
% CCreducepop will remove any repeat conjunctive clauses and will reduce
% the conjunctive clause populations if they exceed their thresholds
%
% Inputs:
```

```
% ArchCCs = The archived conjunctive clauses.
% ArchCCFeats = The features present in the archived conjunctive clauses.
% ArchCCFit = The fitness of the archived conjunctive clauses.
% ArchCCFitComp = The raw inputs used to calculate the archived fitness.
% ArchCCOrder = The order of the archived conjunctive clause.
% ArchCCMatchLocs = A logical matrix that shows which observations the
%                   archived conjunctive clauses match.
% ArchCCAge = The age of the archived conjunctive clause.
% NonArchCCs = The non-archived conjunctive clauses.
% NonArchCCFeats = The features present in the non-archived conjunctive
%                  clauses.
% NonArchCCFit = The fitness of the non-archived conjunctive clauses.
% NonArchCCFitComp = The raw inputs used to calculate the non-archived
%                    fitness.
% NonArchCCOrder = The order of the non-archived conjunctive clause.
% NonArchCCMatchLocs = A logical matrix that shows which observations the
%                      non-archived conjunctive clauses match.
% NonArchCCAge = The age of the non-archived conjunctive clause.
% Param = A structure array with many of the parameters for the algorithm.
% CCstats = Structure array statistics on the conjunctive clauses.
%
% Outputs:
```

```
% ArchCCs = The archived conjunctive clauses.
% ArchCCFeats = The features present in the archived conjunctive clauses.
% ArchCCFit = The fitness of the archived conjunctive clauses.
% ArchCCFitComp = The raw inputs used to calculate the archived fitness.
% ArchCCOrder = The order of the archived conjunctive clause.
% ArchCCMatchLocs = A logical matrix that shows which observations the
%                   archived conjunctive clauses match.
% NonArchCCAge = The age of the non-archived conjunctive clause.
% NonArchCCs = The non-archived conjunctive clauses.
% NonArchCCFeats = The features present in the non-archived conjunctive
%                  clauses.
% NonArchCCFit = The fitness of the non-archived conjunctive clauses.
% NonArchCCFitComp = The raw inputs used to calculate the non-archived
%                    fitness.
% NonArchCCOrder = The order of the non-archived conjunctive clause.
% NonArchCCMatchLocs = A logical matrix that shows which observations the
%                      non-archived conjunctive clauses match.
% NonArchCCAge = The age of the non-archived conjunctive clause.
% Param = A structure array with many of the parameters for the algorithm.
% CCstats = Structure array statistics on the conjunctive clauses.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Start by reducing the any repeated conjunctive clauses in the Archive
% population
if length(ArchCCAge)>1
    % Then check to see if there are any repeat conjunctive clause
    [ArchCCs, ID]=unique(ArchCCs,'rows');
    % Now save the unique conjunctive clauses
    ArchCCFeats=ArchCCFeats(ID,:);
    ArchCCFit=ArchCCFit(ID);
    ArchCCFitComp.TotObs=ArchCCFitComp.TotObs(ID);
    ArchCCFitComp.TotObsT=ArchCCFitComp.TotObsT(ID);
    ArchCCFitComp.TotCCObs=ArchCCFitComp.TotCCObs(ID);
    ArchCCFitComp.TotCCObsT=ArchCCFitComp.TotCCObsT(ID);
    ArchCCOrder=ArchCCOrder(ID);
    ArchCCMatchLocs=ArchCCMatchLocs(:,ID);
    ArchCCAge=ArchCCAge(ID);
    clear ID
end
if length(ArchCCAge)>1
    % Determine if any of the archive bins are over their limit
    % First create a temporary order so that all Conjunctive clauses
    % greater than the max bin are set to max bin
    TempOrder=ArchCCOrder;
```

```
TempMask=TempOrder>max(Param.Thresh(:,1));
TempOrder(TempMask)=max(Param.Thresh(:,1));
clear TempMask
% Now tabulate the temporary order
TabTempOrder=tabulate(TempOrder);
% Remove any rows that do not have a value
TabTempOrder=TabTempOrder(TabTempOrder(:,2)>0,:);
% Compare the tabulated TempOrder to the associated maximum allowable
% populations
% Test to see if TabTempOrder are the same
if size(TabTempOrder,1)==size(Param.Thresh,1)
   % Then all orders a present
   % Determine how many if any bins are over the limit
   LimitMask=TabTempOrder(:,2)>Param.Thresh(:,4);
else
   % Then not all orders are present so need to determine which orders
   %  are present
   % First set up a logical vector for efficiency
   LimitMask=false(size(Param.Thresh,1),1);
   % for each of the orders present, determine if the limit is
   % surpassed
   for i=1:size(TabTempOrder,1)
```

```
        % Grab the ith order
        CurOrder=TabTempOrder(i,1);
        % Create a mask of the Order
        TempMask=CurOrder==Param.Thresh(:,1);
        clear CurOrder
        % Now check to see if the limit is surpassed
        LimitMask(TempMask)=TabTempOrder(i,2)>Param.Thresh(TempMask,4);
        clear TempMask
    end
    clear i
end
% If the sum of limit mask is greater than 0 then at least one bin is
% over the limit so reduce the population of the bin
if sum(LimitMask)>0
    % Then for each bin over the limit reduce the bin population
    % Fist determine the orders of conjunctive clauses that are over
    % the mask
    OrderOver=Param.Thresh(LimitMask,1);
    % Create a keep mask for efficiency
    KeepMask=true(size(ArchCCAge));
    for i=1:length(OrderOver)
        % Create a mask of the current OrderOver
```

```
OrderMask=OrderOver(i)==TempOrder;
% sort the fitness of the current order fitness
CurSortFit=sort(ArchCCFit(OrderMask));
% Create a mask for Param.Thresh Table
ThreshMask=OrderOver(i)==Param.Thresh(:,1);
% Find the minimum number for this bin
CurMin=Param.Thresh(ThreshMask,3);
% Now use the CurMin to find the fitness of sorted fitness and
% use this to set the new threshold
Param.Thresh(ThreshMask,2)=CurSortFit(CurMin);
clear CurMin
clear CurSortFit
% Now create a mask of all the archived conjunctive clauses
% with a fitness greater than the new threshold
AboveThreshMask=ArchCCFit>Param.Thresh(ThreshMask,2);
clear ThreshMask
% Now create a mask for Removal
RemoveMask=AboveThreshMask&OrderMask;
clear AboveThreshMask
clear OrderMask
% Now set the RemoveMask locations to false
KeepMask(RemoveMask)=false();
```

```
    clear RemoveMask
end
% Determine if any of the removed Archived conjunctive clauses have
% an age that can be moved to the non-archive population
% Create a mask of the archive population that is young enough to
% fit in the non-archive population
YoungPop=ArchCCAge<=(Param.GENn*Param.ALna);
% Now determine if there are any young popvalues that will be
% removed
Move2NonArch=YoungPop&~KeepMask;
clear YoungPop
if sum(Move2NonArch)>0
    % Then move the selected features to the non-archive population
    NonArchCCs=[NonArchCCs; ArchCCs(Move2NonArch,:)];
    NonArchCCFit=[NonArchCCFit; ArchCCFit(Move2NonArch)];
    NonArchCCFeats=[NonArchCCFeats; ArchCCFeats(Move2NonArch,:)];
    NonArchCCFitComp.TotObs=[NonArchCCFitComp.TotObs;...
                        ArchCCFitComp.TotObs(Move2NonArch)];
    NonArchCCFitComp.TotObsT=[NonArchCCFitComp.TotObsT;...
                        ArchCCFitComp.TotObsT(Move2NonArch)];
    NonArchCCFitComp.TotCCObs=[NonArchCCFitComp.TotCCObs;...
                        ArchCCFitComp.TotCCObs(Move2NonArch)];
```

```
        NonArchCCFitComp.TotCCObsT=[NonArchCCFitComp.TotCCObsT;...
                        ArchCCFitComp.TotCCObsT(Move2NonArch)];
        NonArchCCOrder=[NonArchCCOrder; ArchCCOrder(Move2NonArch)];
        NonArchCCMatchLocs=[NonArchCCMatchLocs ...
                            ArchCCMatchLocs(:,Move2NonArch)];
        NonArchCCAge=[NonArchCCAge; ArchCCAge(Move2NonArch)];
        clear Move2NonArch
    end
    % Keep only those conjunctive clauses that are in the KeepMask
    ArchCCs=ArchCCs(KeepMask,:);
    ArchCCFit=ArchCCFit(KeepMask);
    ArchCCFeats=ArchCCFeats(KeepMask,:);
    ArchCCFitComp.TotObs=ArchCCFitComp.TotObs(KeepMask);
    ArchCCFitComp.TotObsT=ArchCCFitComp.TotObsT(KeepMask);
    ArchCCFitComp.TotCCObs=ArchCCFitComp.TotCCObs(KeepMask);
    ArchCCFitComp.TotCCObsT=ArchCCFitComp.TotCCObsT(KeepMask);
    ArchCCOrder=ArchCCOrder(KeepMask);
    ArchCCMatchLocs=ArchCCMatchLocs(:,KeepMask);
    ArchCCAge=ArchCCAge(KeepMask);
    clear KeepMask
end
clear LimitMask
```

```
    clear TabTempOrder
end
% Remove Any NonArchAge that is now aged out
YoungMask=NonArchCCAge<(Param.GENn*Param.ALna);
% If there are any NonArchCCs to remove then remove them
if sum(~YoungMask)>0
    NonArchCCs=NonArchCCs(YoungMask,:);
    NonArchCCFeats=NonArchCCFeats(YoungMask,:);
    NonArchCCFit=NonArchCCFit(YoungMask);
    NonArchCCFitComp.TotObs=NonArchCCFitComp.TotObs(YoungMask);
    NonArchCCFitComp.TotObsT=NonArchCCFitComp.TotObsT(YoungMask);
    NonArchCCFitComp.TotCCObs=NonArchCCFitComp.TotCCObs(YoungMask);
    NonArchCCFitComp.TotCCObsT=NonArchCCFitComp.TotCCObsT(YoungMask);
    NonArchCCOrder=NonArchCCOrder(YoungMask);
    NonArchCCMatchLocs=NonArchCCMatchLocs(:,YoungMask);
    NonArchCCAge=NonArchCCAge(YoungMask);
end
clear YoungMask
% Now check to see if the Non-archived population is exceeded for each
% layer
if ~isempty(NonArchCCAge)
    % First determine the non-archive age layers for each conjunctive
```

```
    % clause
    NonArchCCAgeLayer=ceil(NonArchCCAge/Param.GENn);
    % Now tabulate the NonArchCCAgeLayers
    TabNonArchLayer=tabulate(NonArchCCAgeLayer);
    % Remove any TabNonArchLayer that does not have a value
    TabNonArchLayer=TabNonArchLayer(TabNonArchLayer(:,2)>0,:);
    % Determine if any of the TabNonArchLayer is greater than the maximum
    % allowed
    LimitMask=TabNonArchLayer(:,2)>Param.NonArchLMax;
else
    % Set the limit mask to false
    LimitMask=false();
end
% If any layer is greater then need to remove individuals from the
% non-archive layer
if sum(LimitMask)>0
    % Then determine which layers need to be reduced in size
    Layers=TabNonArchLayer(LimitMask,1);
    % Determine the number of features that should be present per feature
    NumPerFeat=floor(Param.NonArchLMax/Param.NumFeat);
    % for efficiency create a keep mask
    KeepMask=true(size(NonArchCCFit));
```

```
for i=1:length(Layers)
    % Create vector index of the current layer
    LayerInd=find(Layers(i)==NonArchCCAgeLayer);
    % Sort the fitness of the current layer
    [~, ID]=sort(NonArchCCFit(LayerInd));
    % Create a vector that has the maximum number of features capped at
    % NumPerFeat
    % First sum the features present
    CurFeatMax=sum(NonArchCCFeats(LayerInd,:));
    % Now replace any sum greater than NumPerFeat with NumPerFeat
    CurFeatMax(CurFeatMax>NumPerFeat)=NumPerFeat;
    % Check to see if the most fit in the layer meet the CurFeatMax
    % First Sum the most fit individuals in the layer
    MostFitSum=...
            sum(NonArchCCFeats(LayerInd(ID(1:Param.NonArchLMax)),:));
    % Now compare to CurFeatMax
    if sum(MostFitSum>=CurFeatMax)==Param.NumFeat
        % Then the most fit individuals in the current layer should be
        % saved
        % So set the others to false in the keep mask
        KeepMask(LayerInd(ID(Param.NonArchLMax+1:end)))=false();
    else
```

```
% Then need to try an alternative method
% First determine the Problem features and the number of
% conjunctive clauses needed for each feature
ProbFeats=find(MostFitSum<CurFeatMax);
% Create a RandFeatOrder
RandFeatOrder=randperm(length(ProbFeats),length(ProbFeats));
% Now go through and grab the most fit CCs for each of the
% problem features
% For Efficiency
ProbFeatKeep=false(length(ID),1);
for ii=1:length(ProbFeats)
   % Create a mask for the current feature
   CurFeatMask=NonArchCCFeats(LayerInd(ID),...
                        ProbFeats(RandFeatOrder(ii)));
   % Now Create a cumsum for the number of times the feature
   % is present
   CumSumFeat=cumsum(CurFeatMask);
   % Create a cumsum mask
   CumSumMask=CumSumFeat<=CurFeatMax(ProbFeats(ii));
   % Find where cumsum Mask and feature mask overlap
   SaveMask=CumSumMask&CurFeatMask;
   % Set the SaveMask locations to true
```

```
    ProbFeatKeep(SaveMask)=true();
end
% Now determine how many of the Problem FeatKeep are saved
TotProbFeat=sum(ProbFeatKeep);
% Determine the difference with Param.NonArchLMax
CurDiff=Param.NonArchLMax-TotProbFeat;
% Determine the ID of the Problem Keep Feat
IDnum=ID(ProbFeatKeep);
% Determine if new total meets the requirements
NewFeatTot=sum(NonArchCCFeats(LayerInd([ID(1:CurDiff);...
                                        IDnum]),:));
if sum(NewFeatTot>=CurFeatMax)==Param.NumFeat
    % Then the current population of CCs are to be saved
    % Write the KeepIDnums
    KeepIDnums=[ID(1:CurDiff); IDnum];
    % Determine the difference
    RemoveIDnums=setdiff(ID,KeepIDnums);
    % Now set the remove values to false
    KeepMask(LayerInd(RemoveIDnums))=false();
else
    % Need to more thoroughly search CCs
    % Determine the features that can be kept
```

```
PosKeepFeat=find(CurFeatMax>0);
% First randomly determine a feature order
RandFeatOrd=randperm(length(PosKeepFeat),...
                        length(PosKeepFeat));
% Set a vector for conjunctive clauses that are available
AvailCCs=true(length(ID),1);
% set a list to keep
KeepFeat=false(length(ID),1);
for ii=1:length(PosKeepFeat)
   % Create a mask for the current feature
   CurFeatMask=NonArchCCFeats(LayerInd(ID), ...
                    PosKeepFeat(RandFeatOrd(ii)));
   % Remove any features that are not available
   CurFeatMask=CurFeatMask&AvailCCs;
   % Now Create a cumsum for the number of times the
   % feature is present
   CumSumFeat=cumsum(CurFeatMask);
   % Create a cumsum mask
   CumSumMask=CumSumFeat<=CurFeatMax(PosKeepFeat(ii));
   % Find where cumsum Mask and feature mask overlap
   SaveMask=CumSumMask&CurFeatMask;
   % Set the SaveMask locations to true
```

```
            KeepFeat(SaveMask)=true();
            % Update the available CCs
            AvailCCs(SaveMask)=false();
        end
        % Write the KeepIDnums
        KeepIDnums=ID(KeepFeat);
        % Determine the difference
        RemoveIDnums=setdiff(ID,KeepIDnums);
        % Now set the remove values to false
        KeepMask(LayerInd(RemoveIDnums))=false();
    end
  end
end
% Now keep all the information in the keep mask
NonArchCCs=NonArchCCs(KeepMask,:);
NonArchCCFit=NonArchCCFit(KeepMask);
NonArchCCFeats=NonArchCCFeats(KeepMask,:);
NonArchCCFitComp.TotObs=NonArchCCFitComp.TotObs(KeepMask);
NonArchCCFitComp.TotObsT=NonArchCCFitComp.TotObsT(KeepMask);
NonArchCCFitComp.TotCCObs=NonArchCCFitComp.TotCCObs(KeepMask);
NonArchCCFitComp.TotCCObsT=NonArchCCFitComp.TotCCObsT(KeepMask);
NonArchCCOrder=NonArchCCOrder(KeepMask);
```

```
    NonArchCCMatchLocs=NonArchCCMatchLocs(:,KeepMask);
    NonArchCCAge=NonArchCCAge(KeepMask);
    clear KeepMask
end
clear NonArchCCAgeLayer
clear LimitMask
clear TabNonArchLayer
% If the user wants to record the best fitness of each order then record
if Param.BestFit
    % Determine the best fitness for each order
    TempFit=[ArchCCFit; NonArchCCFit];
    for i=1:Param.MaxNumFeat
        if i~=Param.MaxNumFeat
            % Then mask by current order
            CurOrderMask=[ArchCCOrder; NonArchCCOrder]==i;
            if sum(CurOrderMask)>0
                % Then record the best fitness
                CCstats.BestFit(Param.CurGen,i)=min(TempFit(CurOrderMask));
            end
        else
            % The mask by the current order and any larger order
            CurOrderMask=[ArchCCOrder; NonArchCCOrder]>=i;
```

```
         if sum(CurOrderMask)>0
            % Then record the best fitness
            CCstats.BestFit(Param.CurGen,i)=min(TempFit(CurOrderMask));
         end
      end
      clear CurOrderMask
   end
   clear i
   clear TempFit
end
```

8.1.2.3 Conjunctive Clause Evolution (CCEvolution)

```
function [ArchCCs, ArchCCFeats, ArchCCFit, ArchCCFitComp, ArchCCOrder,...
        ArchCCMatchLocs, ArchCCAge, NonArchCCs, NonArchCCFeats,...
        NonArchCCFit, NonArchCCFitComp, NonArchCCOrder,...
        NonArchCCMatchLocs, NonArchCCAge, Param, CCstats]=...
         CCEvolution(ArchCCs,ArchCCFeats,ArchCCFit,ArchCCFitComp,...
           ArchCCOrder,ArchCCMatchLocs,ArchCCAge,...
           NonArchCCs,NonArchCCFeats,NonArchCCFit,NonArchCCFitComp,...
           NonArchCCOrder,NonArchCCMatchLocs,NonArchCCAge,...
           DataBin,NaNMask,TargetClass,Param,CCstats)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
```

```
%
% October 18, 2016
% Last Updated: October 18, 2016
%
% CCEvolution evolves the population of conjunctive clauses.
%
% Inputs:
% ArchCCs = The archived conjunctive clauses.
% ArchCCFeats = The features present in the archived conjunctive clauses.
% ArchCCFit = The fitness of the archived conjunctive clauses.
% ArchCCFitComp = The raw inputs used to calculate the archived fitness.
% ArchCCOrder = The order of the archived conjunctive clause.
% ArchCCMatchLocs = A logical matrix that shows which observations the
%                   archived conjunctive clauses match.
% ArchCCAge = The age of the archived conjunctive clause.
% NonArchCCs = The non-archived conjunctive clauses.
% NonArchCCFeats = The features present in the non-archived conjunctive
%                  clauses.
% NonArchCCFit = The fitness of the non-archived conjunctive clauses.
% NonArchCCFitComp = The raw inputs used to calculate the non-archived
%                    fitness.
% NonArchCCOrder = The order of the non-archived conjunctive clause.
```

```
% NonArchCCMatchLocs = A logical matrix that shows which observations the
%                     non-archived conjunctive clauses match.
% NonArchCCAge = The age of the non-archived conjunctive clause.
% DataBin = The data as a binary logical matrix.
% NaNMask = A logical mask of the location of the NaN values in the
%         dataset.
% TargetClass = A logical vector of the observations that have the output
%             class.
% Param = A structure array with many of the parameters for the algorithm.
% CCstats = Structure array statistics on the conjunctive clauses.
%
% Outputs:
% ArchCCs = The archived conjunctive clauses.
% ArchCCFeats = The features present in the archived conjunctive clauses.
% ArchCCFit = The fitness of the archived conjunctive clauses.
% ArchCCFitComp = The raw inputs used to calculate the archived fitness.
% ArchCCOrder = The order of the archived conjunctive clause.
% ArchCCMatchLocs = A logical matrix that shows which observations the
%                   archived conjunctive clauses match.
% ArchCCAge = The age of the archived conjunctive clause.
% NonArchCCs = The non-archived conjunctive clauses.
% NonArchCCFeats = The features present in the non-archived conjunctive
```

```
%              clauses.
% NonArchCCFit = The fitness of the non-archived conjunctive clauses.
% NonArchCCFitComp = The raw inputs used to calculate the non-archived
%               fitness.
% NonArchCCOrder = The order of the non-archived conjunctive clause.
% NonArchCCMatchLocs = A logical matrix that shows which observations the
%                 non-archived conjunctive clauses match.
% NonArchCCAge = The age of the non-archived conjunctive clause.
% Param = A structure array with many of the parameters for the algorithm.
% CCstats = Structure array statistics on the conjunctive clauses.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate the age layer of the non-archived population
NonArchCCAgeLayer=ceil(NonArchCCAge/Param.GENn);
% If there is arhive layer, then add an archive age layer that is one more
% than the max age-layer in non-archived population
if ~isempty(ArchCCAge)
   % Check to make sure there is an non-archive population
   if ~isempty(NonArchCCAge)
      ArchCCAgeLayer=ones(size(ArchCCAge))+max(NonArchCCAgeLayer);
   else
      ArchCCAgeLayer=ones(size(ArchCCAge));
```

```
    end
end
% First determine if there is an archive population and how big the
% archive population is
if length(ArchCCFit)>Param.ArchOff
    % Then need to select offspring to undergo mutation
    % Sort the Archived CCs by ArchCCAge
    [~, ID]=sort(ArchCCAge);
    % Determine the number of offspring per feature
    NumOffPerFeat=floor(Param.ArchOff/Param.NumFeat);
    % Determine the number of times the features are present in the archive
    PosFeatTots=sum(ArchCCFeats);
    % Now set any of the feat Totals greater than NumOffPerFeat to
    % NumOffPerFeat
    PosFeatTots(PosFeatTots>NumOffPerFeat)=NumOffPerFeat;
    % Now sum the sorted offsrping for number of times feature is
    % present
    CurFeatTotals=sum(ArchCCFeats(ID(1:Param.ArchOff),:));
    % Determine if the features are present the requisite number of
    % times
    if sum(CurFeatTotals>=PosFeatTots)==Param.NumFeat
        % Create a selected CC vector
```

```
    SelCCVec=false(size(ArchCCFit));
    % Now set 1:Param.ArchOff to true
    SelCCVec(1:Param.ArchOff)=true();
    % set the number of archived offspring
    NumArchOff=sum(SelCCVec);
    % Then the selected CCs have enough diversity
    MateCC=[ArchCCs(ID(SelCCVec),:);...
          ArchCCs(ID(~SelCCVec),:);...
          NonArchCCs];
    MateCCFeats=[ArchCCFeats(ID(SelCCVec),:);...
               ArchCCFeats(ID(~SelCCVec),:);...
               NonArchCCFeats];
    MateCCFit=[ArchCCFit(ID(SelCCVec));...
             ArchCCFit(ID(~SelCCVec));...
             NonArchCCFit];
    MateCCAge=[ArchCCAge(ID(SelCCVec));...
             ArchCCAge(ID(~SelCCVec));...
             NonArchCCAge];
    MateCCAgeLayer=[ArchCCAgeLayer(ID(SelCCVec));...
                  ArchCCAgeLayer(ID(~SelCCVec));...
                  NonArchCCAgeLayer];
else
```

```
% Need to smartly select the features to help with
% diversity
% Find the features that did not meet the requirements
% in the first pop
ProbFeatLocs=find(CurFeatTotals<PosFeatTots);
% For efficiency
ProbFeatLogVec=false(size(ArchCCFit));
% For each problem feature find the least evolved CCs
for i=1:length(ProbFeatLocs)
   % find the CCs with the current feature
   CurFeatCCs=ArchCCFeats(ID,ProbFeatLocs(i));
   % Select the NumOffPerFeat first CCs
   TempMask=CurFeatCCs&cumsum(CurFeatCCs)<= ...
                          PosFeatTots(ProbFeatLocs(i));
   % Now set the TempMask locations ot true
   ProbFeatLogVec(TempMask)=true();
   clear CurFeatCCs
   clear TempMask
end
% Now determine if there is any overlap between the
% offspring origingall selected and the ProbFeatLocs
NumOverlap=sum(ProbFeatLogVec(1:Param.ArchOff));
```

```
% Determine the number selected
NumSel=sum(ProbFeatLogVec);
% Now subtract NumOverlap from NumSel
DiffSelOver=NumSel-NumOverlap;
clear NumOverlap
clear NumSel
% Create a new vector for the selected features
SelCCVec=false(size(ArchCCFit));
SelCCVec(1:Param.ArchOff-DiffSelOver)=true();
SelCCVec(ProbFeatLogVec)=true();
%  Determine if all of the features are now represented
NewFeatTots=sum(ArchCCFeats(ID(SelCCVec),:));
% Determine if there are any NewFeatTots that now have
% features that are not included in the first group.
if sum(NewFeatTots<PosFeatTots)>0
   % Then some of the features that were not problematic
   % before are now problematic so go through based on the
   % most problematic to least to select the CCs
   [~,SortTotID]=sort(NewFeatTots);
   % Remove features that were already selected
   SortTotID=setdiff(SortTotID,ProbFeatLocs);
   for ii=1:length(SortTotID)
```

```
    % find the CCs with the current feature
    CurFeatCCs=ArchCCFeats(ID,SortTotID(ii));
    % Select the NumOffPerFeat first CCs
    TempMask=CurFeatCCs&cumsum(CurFeatCCs)<= ...
                                PosFeatTots(SortTotID(ii));
    % Now set the TempMask locations to true
    ProbFeatLogVec(TempMask)=true();
    clear CurFeatCCs
    clear TempMask
    % Determine if the minimum has been found for each
    % feature remaining in SortTotID
    NewTots=sum(ArchCCFeats(ID(ProbFeatLogVec),SortTotID));
    if sum(NewTots>=PosFeatTots(SortTotID))==length(SortTotID)
        % Then break
        break
    end
end
% Determine the total ProbFeatLogVec
if sum(ProbFeatLogVec)==Param.ArchOff
    % Then set ProbFeatLogVec to the logical index
    % vector
    SelCCVec=false(size(ArchCCFit));
```

```
        SelCCVec(ProbFeatLogVec)=true();
    else
        % Add CCs to get to the total
        % Determine the number of CCs to add
        Num2Add=Param.ArchOff-sum(ProbFeatLogVec);
        % Determine the cumsum of ~ProbFeatLogVec
        CumsumProb=cumsum(~ProbFeatLogVec);
        % Now create a mask of
        TempMask=CumsumProb<=Num2Add;
        SelCCVec=false(size(ArchCCFit));
        SelCCVec(ProbFeatLogVec)=true();
        SelCCVec(TempMask)=true();
    end
end
% set the number of archived offspring
NumArchOff=sum(SelCCVec);
% Create the MatingPop
MateCC=[ArchCCs(ID(SelCCVec),:);...
        ArchCCs(ID(~SelCCVec),:);...
        NonArchCCs];
MateCCFeats=[ArchCCFeats(ID(SelCCVec),:);...
            ArchCCFeats(ID(~SelCCVec),:);...
```

```
                NonArchCCFeats];
        MateCCFit=[ArchCCFit(ID(SelCCVec));...
                ArchCCFit(ID(~SelCCVec));...
                NonArchCCFit];
        MateCCAge=[ArchCCAge(ID(SelCCVec));...
                ArchCCAge(ID(~SelCCVec));...
                NonArchCCAge];
        MateCCAgeLayer=[ArchCCAgeLayer(ID(SelCCVec));...
                    ArchCCAgeLayer(ID(~SelCCVec));...
                    NonArchCCAgeLayer];
    end

elseif ~isempty(ArchCCFit)
    % Then all archived offspring will be selected
    MateCC=[ArchCCs; NonArchCCs];
    MateCCFeats=[ArchCCFeats; NonArchCCFeats];
    MateCCFit=[ArchCCFit; NonArchCCFit];
    MateCCAge=[ArchCCAge; NonArchCCAge];
    MateCCAgeLayer=[ArchCCAgeLayer; NonArchCCAgeLayer];
    % Set the SelCCVec to the length of ArchCCFit and to True
    SelCCVec=true(size(ArchCCFit));
    % Set the number of offspring
```

```
    NumArchOff=sum(SelCCVec);
    % set ID
    ID=1:length(ArchCCAge);
else
    % set the number of ArchOffsping to zero
    NumArchOff=0;
    % Then there is no archive population
    % Then the Mating population is simply the non-archive
    % population
    MateCC=NonArchCCs;
    MateCCFeats=NonArchCCFeats;
    MateCCFit=NonArchCCFit;
    MateCCAge=NonArchCCAge;
    MateCCAgeLayer=NonArchCCAgeLayer;
end
% For Effificiency
EvoCC=false(NumArchOff+length(NonArchCCFit),Param.NumBinCols);
EvoCCFeats=false(NumArchOff+length(NonArchCCFit),Param.NumFeat);
EvoCCFit=NaN(NumArchOff+length(NonArchCCFit),1);
EvoCCFitComp.TotObs=NaN(NumArchOff+length(NonArchCCFit),1);
EvoCCFitComp.TotObsT=NaN(NumArchOff+length(NonArchCCFit),1);
EvoCCFitComp.TotCCObs=NaN(NumArchOff+length(NonArchCCFit),1);
```

```
EvoCCFitComp.TotCCObsT=NaN(NumArchOff+length(NonArchCCFit),1);
EvoCCOrder=NaN(NumArchOff+length(NonArchCCFit),1);
EvoCCMatchLocs=false(Param.NumObs,NumArchOff+length(NonArchCCFit));
EvoCCAge=NaN(NumArchOff+length(NonArchCCFit),1);
EvoArchiveMask=false(NumArchOff+length(NonArchCCFit),1);
% If there is an ArchPop then perform one task otherwise another
if NumArchOff>0
   % Then an archive age layer is present
   % Determine the number of age layers
   UniqueLayers=unique(MateCCAgeLayer);
   NumLayers=length(UniqueLayers);
   % Initialize start
   start=1;
   % Run a for loop so that each age layer can undergo either
   % mutation or crossover
   for i=1:NumLayers
      % perform crossover or mutation on the current layer
      CurLayer=UniqueLayers(NumLayers-i+1);
      % Create a mask for the CCs that will evolve
      CurMask=MateCCAgeLayer==CurLayer;
      % Now select the necessary data for mutation or crossover
      CurMateCC=MateCC(CurMask,:);
```

```
CurCCFeats=MateCCFeats(CurMask,:);
CurCCFit=MateCCFit(CurMask);
CurCCAge=MateCCAge(CurMask);
% If this is the 1st loop then CurNumOff=ArchNumOff
if i~=1
   CurNumOff=sum(CurMask);
else
   CurNumOff=NumArchOff;
   % Also add 1 to the age of the selected archived offspring
   CurCCAge(1:NumArchOff)=CurCCAge(1:NumArchOff)+1;
end
clear CurMask
% If the current layer isn't one then add the younger layer
% to mate with
if CurLayer~=1
   % Add a layer to the current layer for mating
   CurMask=MateCCAgeLayer==CurLayer-1;
   CurMateCC=[CurMateCC; MateCC(CurMask,:)];
   CurCCFeats=[CurCCFeats; MateCCFeats(CurMask,:)];
   CurCCFit=[CurCCFit; MateCCFit(CurMask)];
   CurCCAge=[CurCCAge; MateCCAge(CurMask)];
   clear CurMask
```

```
end
% Now perform crossover and or mutation
[OffCC, OffCCFeats, OffCCFit, OffCCFitComp, OffCCOrder,...
   OffCCMatchLocs, OffCCAge, ArchiveMask, CCstats]=...
     CCMutCross(CurMateCC,CurCCFeats,CurCCFit,CurCCAge,...
       CurNumOff,Param,DataBin,NaNMask,TargetClass,CCstats);
clear CurLayer
clear CurMateCC
clear CurCCFeats
clear CurCCAge
clear CurNumOff
% Now save the offspring
% Determine the number of offspring
NumOff=length(ArchiveMask);
EvoCC(start:start+NumOff-1,:)=OffCC;
EvoCCFeats(start:start+NumOff-1,:)=OffCCFeats;
EvoCCFit(start:start+NumOff-1)=OffCCFit;
EvoCCFitComp.TotObs(start:start+NumOff-1)=OffCCFitComp.TotObs;
EvoCCFitComp.TotObsT(start:start+NumOff-1)=OffCCFitComp.TotObsT;
EvoCCFitComp.TotCCObs(start:start+NumOff-1)=OffCCFitComp.TotCCObs;
EvoCCFitComp.TotCCObsT(start:start+NumOff-1)= ...
                                OffCCFitComp.TotCCObsT;
```

```
        EvoCCOrder(start:start+NumOff-1)=OffCCOrder;
        EvoCCMatchLocs(:,start:start+NumOff-1)=OffCCMatchLocs;
        EvoCCAge(start:start+NumOff-1)=OffCCAge;
        EvoArchiveMask(start:start+NumOff-1)=ArchiveMask;
        % update the start
        start=start+NumOff;
        clear NumOff
    end
    % Increase only the age of the archive population that underwent either
    % mutation or crossover
    ArchCCAge(ID(SelCCVec))=ArchCCAge(ID(SelCCVec))+1;
else
    % then age layers do not have an archive layer
    % Determine the number of age layers
    UniqueLayers=unique(MateCCAgeLayer);
    NumLayers=length(UniqueLayers);
    % set a start counter
    start=1;
    % Run a for loop so that each age layer can undergo either
    % mutation or crossover
    for i=1:NumLayers
        % perform crossover or mutation on the current layer
```

```
CurLayer=UniqueLayers(NumLayers-i+1);
% Create a mask for the CCs that will evolve
CurMask=MateCCAgeLayer==CurLayer;
% Now select the necessary data for mutation or crossover
CurMateCC=MateCC(CurMask,:);
CurCCFeats=MateCCFeats(CurMask,:);
CurCCFit=MateCCFit(CurMask);
CurCCAge=MateCCAge(CurMask);
CurNumOff=sum(CurMask);
clear CurMask
% If the current layer isn't one then add the younger layer
% to mate with
if CurLayer~=1
   % Add a layer to the current layer for mating
   CurMask=MateCCAgeLayer==CurLayer-1;
   CurMateCC=[CurMateCC; MateCC(CurMask,:)];
   CurCCFeats=[CurCCFeats; MateCCFeats(CurMask,:)];
   CurCCFit=[CurCCFit; MateCCFit(CurMask)];
   CurCCAge=[CurCCAge; MateCCAge(CurMask)];
   clear CurMask
end
% Now perform crossover and or mutation
```

```
[OffCC, OffCCFeats, OffCCFit, OffCCFitComp, OffCCOrder,...
   OffCCMatchLocs, OffCCAge, ArchiveMask, CCstats]=...
     CCMutCross(CurMateCC,CurCCFeats,CurCCFit,CurCCAge,...
       CurNumOff,Param,DataBin,NaNMask,TargetClass,CCstats);
clear CurLayer
clear CurMateCC
clear CurCCFeats
clear CurCCAge
clear CurNumOff
% Now save the offspring
% Determine the number of offspring
NumOff=length(ArchiveMask);
EvoCC(start:start+NumOff-1,:)=OffCC;
EvoCCFeats(start:start+NumOff-1,:)=OffCCFeats;
EvoCCFit(start:start+NumOff-1)=OffCCFit;
EvoCCFitComp.TotObs(start:start+NumOff-1)=OffCCFitComp.TotObs;
EvoCCFitComp.TotObsT(start:start+NumOff-1)=OffCCFitComp.TotObsT;
EvoCCFitComp.TotCCObs(start:start+NumOff-1)=OffCCFitComp.TotCCObs;
EvoCCFitComp.TotCCObsT(start:start+NumOff-1)=...
                                OffCCFitComp.TotCCObsT;
EvoCCOrder(start:start+NumOff-1)=OffCCOrder;
EvoCCMatchLocs(:,start:start+NumOff-1)=OffCCMatchLocs;
```

```
        EvoCCAge(start:start+NumOff-1)=OffCCAge;
        EvoArchiveMask(start:start+NumOff-1)=ArchiveMask;
        % update the start
        start=start+NumOff;
        clear NumOff
    end
end
% Reduce offspring to only those that were actually kept
EvoCC=EvoCC(1:start-1,:);
EvoCCFeats=EvoCCFeats(1:start-1,:);
EvoCCFit=EvoCCFit(1:start-1);
EvoCCFitComp.TotObs=EvoCCFitComp.TotObs(1:start-1);
EvoCCFitComp.TotObsT=EvoCCFitComp.TotObsT(1:start-1);
EvoCCFitComp.TotCCObs=EvoCCFitComp.TotCCObs(1:start-1);
EvoCCFitComp.TotCCObsT=EvoCCFitComp.TotCCObsT(1:start-1);
EvoCCOrder=EvoCCOrder(1:start-1);
EvoCCMatchLocs=EvoCCMatchLocs(:,1:start-1);
EvoCCAge=EvoCCAge(1:start-1);
EvoArchiveMask=EvoArchiveMask(1:start-1);
clear start
% Now extract the archived population
ArchCCs=[ArchCCs; EvoCC(EvoArchiveMask,:)];
```

```
ArchCCFeats=[ArchCCFeats; EvoCCFeats(EvoArchiveMask,:)];
ArchCCFit=[ArchCCFit; EvoCCFit(EvoArchiveMask)];
ArchCCFitComp.TotObs=[ArchCCFitComp.TotObs; ...
                        EvoCCFitComp.TotObs(EvoArchiveMask)];
ArchCCFitComp.TotObsT=[ArchCCFitComp.TotObsT; ...
                        EvoCCFitComp.TotObsT(EvoArchiveMask)];
ArchCCFitComp.TotCCObs=[ArchCCFitComp.TotCCObs; ...
                        EvoCCFitComp.TotCCObs(EvoArchiveMask)];
ArchCCFitComp.TotCCObsT=[ArchCCFitComp.TotCCObsT; ...
                        EvoCCFitComp.TotCCObsT(EvoArchiveMask)];
ArchCCOrder=[ArchCCOrder; EvoCCOrder(EvoArchiveMask)];
ArchCCMatchLocs=[ArchCCMatchLocs EvoCCMatchLocs(:,EvoArchiveMask)];
ArchCCAge=[ArchCCAge; EvoCCAge(EvoArchiveMask)];
% Now extract the non-archived population
NonArchCCs=[NonArchCCs; EvoCC(~EvoArchiveMask,:)];
NonArchCCFeats=[NonArchCCFeats; EvoCCFeats(~EvoArchiveMask,:)];
NonArchCCFit=[NonArchCCFit; EvoCCFit(~EvoArchiveMask)];
NonArchCCFitComp.TotObs=[NonArchCCFitComp.TotObs; ...
                        EvoCCFitComp.TotObs(~EvoArchiveMask)];
NonArchCCFitComp.TotObsT=[NonArchCCFitComp.TotObsT; ...
                        EvoCCFitComp.TotObsT(~EvoArchiveMask)];
NonArchCCFitComp.TotCCObs=[NonArchCCFitComp.TotCCObs; ...
```

```
                              EvoCCFitComp.TotCCObs(~EvoArchiveMask)];
NonArchCCFitComp.TotCCObsT=[NonArchCCFitComp.TotCCObsT; ...
                              EvoCCFitComp.TotCCObsT(~EvoArchiveMask)];
NonArchCCOrder=[NonArchCCOrder; EvoCCOrder(~EvoArchiveMask)];
NonArchCCMatchLocs=[NonArchCCMatchLocs
EvoCCMatchLocs(:,~EvoArchiveMask)];
NonArchCCAge=[NonArchCCAge; EvoCCAge(~EvoArchiveMask)];
clear EvoArchiveMask
```

8.1.2.4 Conjunctive Clause Mutation/Crossover (CCMutCross)

```
function [OffCC, OffCCFeats, OffCCFit, OffCCFitComp, OffCCOrder,...
          OffCCMatchLocs, OffCCAge, ArchiveMask, CCstats]=...


CCMutCross(MateCC,MateCCFeats,MateCCFit,MateCCAge,NumOff,Param,...
                    DataBin,NaNMask,TargetClass,CCstats)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
% October 17, 2016
% Last updated: October 18, 2016
%
% CCMutCross performs mutation or crossover for my evolutionary algoirthm.
% The crossover is a little different than typical crossover since the
```

```
% tournament selection selects the potential mate with the most features in
% common. Also, the crossover will take active features from both parents.
%
% Inputs:
% MateCC = The mating population of conjunctive clauses.  Only conjunctive
%          clauses 1:NumOff will undergo crossover or mutation
% MateCCFeats = The features present for each of the conjunctive clauses in
%               the mating population.
% MateCCFit = The fitness of the mates.
% MateCCAge = The age of the conjunctive clauses in the mating population.
% NumOff = The number of conjunctive clauses that will undergo either
%          crossover or mutation and produce one offspring each.
% Param = general parameters for the evolutionary algorithm.
% DataBin = The data as a binary logical matrix.
% NaNMask = A logical mask of the location of the NaN values in the
%           dataset.
% TargetClass = A logical vector of the observations that have the output
%               class.
% CCstats = Structure array statistics on the conjunctive clauses.
%
% Outputs:
% OffCC = The conjunctive clauses of the offspring of either crossover or
```

```
%        mutation.
% OffCCFeats = The features that are present in the OffCC.
% OffCCFit = The fitness of the OffCC using the hypergeometric PMF.
% OffCCFitComp = The raw values that are fed into the fitness function.
% OffCCOrder = The order of the conjunctive clauses.
% OffCCMatchLocs = A logical matrix indicating which observations the OffCC
%                matches.
% OffCCAge = The age of the OffCC, calculated as the maximum age of the
%          parents.
% ArchiveMask = A logical mask indicating teh offspring that should be
%              archived.
% CCstats = Structure array statistics on the conjunctive clauses.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First determine the number of CCs in the mating Pop
NumCCs=size(MateCC,1);
% First randomly determine which inidividuals will undergo mutation and
% which will undergo crossover
if NumCCs~=1
   % Then randomly select crossover or mutation
   CrossOver=rand(NumOff,1)<Param.Px;
else
```

```
    CrossOver=false(1);
end
% For efficiency
OffCC=false(NumOff,Param.NumBinCols);
OffCCFeats=false(NumOff,Param.NumFeat);
OffCCFit=NaN(NumOff,1);
OffCCFitComp.TotObs=NaN(NumOff,1);
OffCCFitComp.TotObsT=NaN(NumOff,1);
OffCCFitComp.TotCCObs=NaN(NumOff,1);
OffCCFitComp.TotCCObsT=NaN(NumOff,1);
OffCCOrder=NaN(NumOff,1);
OffCCMatchLocs=false(Param.NumObs,NumOff);
OffCCAge=NaN(NumOff,1);
ArchiveMask=false(NumOff,1);
KeepMask=false(NumOff,1);
% Now run a for loop where each selected individual will either undergo
% mutation or crossover
for i=1:NumOff
    if CrossOver(i)
        % Then the current MateCC will undergo crossover
        % Set up random mate population
        PotMatePop=setdiff(1:NumCCs,i);
```

```
% Now randomly select the potential mates
PotMateInd=randi(NumCCs-1,[Param.TournSize,1]);
PotMates=PotMatePop(PotMateInd);
clear PotMatePop
clear PotMateInd
% Determine the best feat
BestMask=MateCCFit(PotMates)==min(MateCCFit(PotMates));
% If there is more than one mate selected then randomly pick mate
if sum(BestMask)==1
   % Then rthe the mate ID is easy
   MateID=PotMates(BestMask);
else
   % Randomly choose a Mate
   PotMates=PotMates(BestMask);
   RandPick=randi(sum(BestMask),1);
   MateID=PotMates(RandPick);
   clear RandPick
end
clear PotMates
% Set the offspring age
% The age of the offspring is the age of the oldest parent
OffCCAge(i)=max([MateCCAge(i) MateCCAge(MateID)]);
```

```
% Now randomly determine the features from the 1st parent
P1FeatInd=rand(1,Param.NumFeat)<0.5;
% Now Determine the parent 1 and 2 CC columns
P1CCcols=[Param.FeatInd{P1FeatInd}];
P2CCcols=[Param.FeatInd{~P1FeatInd}];
% Insert the data for the offspring
OffCC(i,P1CCcols)=MateCC(i,P1CCcols);
OffCC(i,P2CCcols)=MateCC(MateID,P2CCcols);
% Create the feature matrix
OffCCFeats(i,P1FeatInd)=MateCCFeats(i,P1FeatInd);
OffCCFeats(i,~P1FeatInd)=MateCCFeats(MateID,~P1FeatInd);
% check to see if the offsrping is the same as either parent or
% there are no features selected
if isequal(OffCC(i,:),MateCC(i,:))||...
        isequal(OffCC(i,:),MateCC(MateID,:))||...
        sum(OffCCFeats(i,:))==0
   % Then switch the offspring
   OffCC(i,P2CCcols)=MateCC(i,P2CCcols);
   OffCC(i,P1CCcols)=MateCC(MateID,P1CCcols);
   OffCCFeats(i,~P1FeatInd)=MateCCFeats(i,~P1FeatInd);
   OffCCFeats(i,P1FeatInd)=MateCCFeats(MateID,P1FeatInd);
end
```

```
    clear P1CCcols
    clear P2CCcols
    clear P1FeatInd
    clear MateID
else
    % Then the current MateCC will undergo mutation
    % First randomly choose which features will undergo mutation
    MutLocs=rand(1,Param.NumFeat)<Param.Pm;
    % If no location was selected then randomly change one feature
    if sum(MutLocs)==0;
        MutLocs(randi(Param.NumFeat))=true();
    end
    % Set the offspring equal to the parent
    OffCC(i,:)=MateCC(i,:);
    OffCCFeats(i,:)=MateCCFeats(i,:);
    OffCCAge(i)=MateCCAge(i);
    % Extract the mutation feature indeces
    MutInd=find(MutLocs);
    clear MutLocs
    % Determine if any of the mutation locations should be made
    % inactive
    InActFeat=rand(1,length(MutInd))<Param.Pwc;
```

```
% Check to see that at least one feature will be left active in the
% offspring
if sum(InActFeat)==length(MutInd)&&sum(OffCCFeats(i,MutInd))== ...
                                    sum(OffCCFeats(i,:))
   % Then randomly change one location to false for wildcard
   InActFeat(randi(length(MutInd)))=false();
end
% Run a for loop to change the features at each location
for j=1:length(MutInd)
   % Grab the jth MutInd and determine if the feature is active
   % and if it is active and InActFeat is true then make it
   % inactive
   if OffCCFeats(i,MutInd(j))&&InActFeat(j)
      % Then make the feature inactive
      % First find the index of feature values
      CurInd=[Param.FeatInd{MutInd(j)}];
      % Now set the conjunctive clause here to false
      OffCC(i,CurInd)=false();
      clear CurInd
      % Now set the feature to false
      OffCCFeats(i,MutInd(j))=false();
   elseif ~OffCCFeats(i,MutInd(j))
```

```
% Then activate the feature and randomly fill in values
OffCCFeats(i,MutInd(j))=true();
% Now determine the datatype of the feature
if Param.DataType(MutInd(j))==1
    % Then the data is continuous or discrete
    % Determine the CurIndex in the CCs
    CurInd=[Param.FeatInd{MutInd(j)}];
    % Determine the range of the data
    CurRange=length(CurInd);
    % Now randomly select a range that will be covered
    RandSelRange=randi(CurRange-1,1);
    % Now randomly select a lower bound
    LB=randi(CurRange-RandSelRange+1,1);
    % Now calculate the upperbound
    UB=LB+RandSelRange-1;
    clear RandSelRange
    % Create a logical vector the same size as the CurInd
    NewVals=false(1,CurRange);
    clear CurRange
    % Set the new values LB:UB range as true
    NewVals(LB:UB)=true();
    % Insert the new values
```

```
    OffCC(i,CurInd)=NewVals;
    clear CurInd
    clear LB
    clear UB
    clear NewVals
elseif Param.DataType(MutInd(j))==2
    % Then the data is categorical
    % Determine the CurIndex in the CCs
    CurInd=[Param.FeatInd{MutInd(j)}];
    % Determine the number of categories of the feature
    CurNumCats=length(CurInd);
    % Determine how many categories will be in the CC
    SelNumCats=randi(CurNumCats-1,1);
    % Randomly select the categories
    SelCats=randperm(CurNumCats,SelNumCats);
    clear SelNumCats
    % Create a logical vector the same size as the CurInd
    NewVals=false(1,CurNumCats);
    clear CurNumCats
    % Now set the selected categories to true
    NewVals(SelCats)=true();
    % Insert the new values
```

```
    OffCC(i,CurInd)=NewVals;
    clear CurInd
    clear SelCats
    clear NewVals
elseif Param.DataType(MutInd(j))==3
    % Then the data is unique categorical or binary
    % Determine the CurIndex in the CCs
    CurInd=[Param.FeatInd{MutInd(j)}];
    % Determine the number of categories of the feature
    CurNumCats=length(CurInd);
    % Randomly select a category to put into the CC
    SelCat=randi(CurNumCats,1);
    % Create a logical vector the same size as the CurInd
    NewVals=false(1,CurNumCats);
    clear CurNumCats
    % Now set the selected categories to true
    NewVals(SelCat)=true();
    % Insert the new values
    OffCC(i,CurInd)=NewVals;
    clear CurInd
    clear SelCat
    clear NewVals
```

```
    end
else
    % Then the feature is active and should have the values
    % changed
    if Param.DataType(MutInd(j))==1
        % Then the data is continuous or discrete
        % Determine the CurIndex in the CCs
        CurInd=[Param.FeatInd{MutInd(j)}];
        % Determine the number of values for the feature
        CurNumVals=length(CurInd);
        % Determine the values that are currently active
        CurVals=find(OffCC(i,CurInd));
        % Determine the current lower and upper bounds
        LB=min(CurVals);
        UB=max(CurVals);
        % Determine if the lower bound is at the min and if the
        % upper bound is at the max
        if LB==1
            % Then ensure that the upper bound is not changed
            % to the max
            % Randomly choose to change the upper or lower
            % bound unless LB==UB then the UB will be changed
```

```
if LB==UB
    % Then the upperbound will be changed
    % Randomly pick a new upper bound
    UB=randi(CurNumVals-2,1)+1;
elseif randi(2,1)==1
    % then the lower bound will be increased
    % Randomly pick a new lower bound
    LB=randi(UB-1,1)+1;
else
    % then the upper bound will be changed
    % Determine the values that the upperbound can
    % take
    PosUB=1:CurNumVals-1;
    % Remove the current upper bound from the set
    % of values
    PosUB=setdiff(PosUB,UB);
    % Randomly pick a new upper bound index
    UBInd=randi(length(PosUB),1);
    % Now extract the new upper bound
    UB=PosUB(UBInd);
    clear PosUB
    clear UBInd
```

```
    end
elseif UB==CurNumVals
    % Then ensure that the lower bound is not changed
    % to the min
    if LB==UB
        % Then the lower bound will be decreased
        % Randomly pick a new lower bound
        LB=randi(CurNumVals-2,1)+1;
    elseif randi(2,1)==1
        % Then the lower bound will be changed
        % Determine the values the lower bound can take
        PosLB=2:CurNumVals;
        % Remove thecurrent lower bound from the set of
        % values
        PosLB=setdiff(PosLB,LB);
        % Randomly pick a new lower bound index
        LBInd=randi(length(PosLB),1);
        % Now extract the new lower bound
        LB=PosLB(LBInd);
        clear PosLB
        clear LBInd
    else
```

```
        % Then the upper bound will be decreased
        % Determine the values the upper bound can take
        PosUB=LB:UB-1;
        % Now randomly pick a new UB index
        UBInd=randi(length(PosUB),1);
        % Now extract the new upper bound
        UB=PosUB(UBInd);
        clear PosUB
        clear UBInd
    end
else
    % Then no need to worry about the LB being the min
    % nor the UB being the max
    % randomly choose if the lower or upper bound will
    % change
    if randi(2,1)==1
        % Then the lower bound will change
        % Determine the possible lower bound values
        PosLB=1:UB;
        % Remove the current lower bound from the set
        % of values
        PosLB=setdiff(PosLB,LB);
```

```
        % Randomly pick a new LB index
        LBInd=randi(length(PosLB),1);
        % Now extract the new lower bound
        LB=PosLB(LBInd);
        clear PosLB
        clear LBInd
    else
        % Then the upper bound will change
        % Determine the possible upper bound values
        PosUB=LB:CurNumVals;
        % Remove the current upper bound from the set
        % of values
        PosUB=setdiff(PosUB,UB);
        % Randomly pick a new upper bound index
        UBInd=randi(length(PosUB),1);
        % Now extract the new upper bound
        UB=PosUB(UBInd);
        clear PosUB
        clear UBInd
    end
end
% Now with the new upper or lower bounds create the new
```

```
    % values
    % Create a logical vector the same size as the CurInd
    NewVals=false(1,CurNumVals);
    % Insert true between LB and UB
    NewVals(LB:UB)=true();
    % Insert the new values
    OffCC(i,CurInd)=NewVals;
    clear LB
    clear UB
    clear CurInd
    clear NewVals
    clear CurNumVals
    clear CurVals
elseif Param.DataType(MutInd(j))==2
    % Then the data is categorical
    % Determine the CurIndex in the CCs
    CurInd=[Param.FeatInd{MutInd(j)}];
    % Determine the number of categories of the feature
    CurNumCats=length(CurInd);
    % Determine CurLogic vector of values
    NewVals=OffCC(i,CurInd);
    % Determine the categories that are currently active
```

```
CurCats=find(NewVals);
if length(CurCats)==1
    % Then the category can either be changed or
    % another category can be addded
    % Randomly choose which one will be selected
    RandChoice=randi(2,1);
    if RandChoice==1
        % Then a category will be randomly changed
        % Determine the categories to choose from
        PosCats=setdiff(1:CurNumCats,CurCats);
        % Randomly choose a category to add
        AddInd=randi(length(PosCats),1);
        % Randomly choose a category to remove
        RemoveInd=randi(length(CurCats),1);
        % Now add the category that is meant to be
        % added and remove the category that is meant
        % to be removed
        NewVals(PosCats(AddInd))=true();
        NewVals(CurCats(RemoveInd))=false();
        clear PosCats
        clear AddInd
        clear RemoveInd
```

```
        % Insert the new values
        OffCC(i,CurInd)=NewVals;
    else
        % Then a category will be randomly added
        % Determine the categories to choose from
        PosCats=setdiff(1:CurNumCats,CurCats);
        % Randomly choose a category to add
        AddInd=randi(length(PosCats),1);
        % Now add the category that is meant to be
        % added and remove the category that is meant
        % to be removed
        NewVals(PosCats(AddInd))=true();
        clear AddInd
        clear PosCats
        % Insert the new values
        OffCC(i,CurInd)=NewVals;
    end
elseif length(CurCats)==CurNumCats-1
    % Then a category can either be changed or a
    % category can be removed
    % Randomly choose which one will be selected
    RandChoice=randi(2,1);
```

```
if RandChoice==1
    % Then a category will be randomly changed
    % Determine the categories to choose from
    PosCats=setdiff(1:CurNumCats,CurCats);
    % Randomly choose a category to add
    AddInd=randi(length(PosCats),1);
    % Randomly choose a category to remove
    RemoveInd=randi(length(CurCats),1);
    % Now add the category that is meant to be
    % added and remove the category that is meant
    % to be removed
    NewVals(PosCats(AddInd))=true();
    NewVals(CurCats(RemoveInd))=false();
    clear PosCats
    clear AddInd
    clear RemoveInd
    % Insert the new values
    OffCC(i,CurInd)=NewVals;
else
    % Then a category will be randomly removed
    % Randomly choose a category to remove
    RemoveInd=randi(length(CurCats),1);
```

```
        % Now add the category that is meant to be
        % added and remove the category that is meant
        % to be removed
        NewVals(CurCats(RemoveInd))=false();
        clear RemoveInd
        % Insert the new values
        OffCC(i,CurInd)=NewVals;
    end
else
    % Then either a category can be chaged a category
    % can be added or a category can be deleted
    % Randomly choose which one will be selected
    RandChoice=randi(3,1);
    if RandChoice==1
        % Then a category will be randomly changed
        % Determine the categories to choose from
        PosCats=setdiff(1:CurNumCats,CurCats);
        % Randomly choose a category to add
        AddInd=randi(length(PosCats),1);
        % Randomly choose a category to remove
        RemoveInd=randi(length(CurCats),1);
        % Now add the category that is meant to be
```

```
    % added and remove the category that is meant
    % to be removed
    NewVals(PosCats(AddInd))=true();
    NewVals(CurCats(RemoveInd))=false();
    clear PosCats
    clear AddInd
    clear RemoveInd
    % Insert the new values
    OffCC(i,CurInd)=NewVals;
elseif RandChoice==2
    % Then a category will be randomly added
    % Determine the categories to choose from
    PosCats=setdiff(1:CurNumCats,CurCats);
    % Randomly choose a category to add
    AddInd=randi(length(PosCats),1);
    % Now add the category that is meant to be
    % added and remove the category that is meant
    % to be removed
    NewVals(PosCats(AddInd))=true();
    clear AddInd
    clear PosCats
    % Insert the new values
```

```
            OffCC(i,CurInd)=NewVals;
        else
            % Then a category will be randomly removed
            % Randomly choose a category to remove
            RemoveInd=randi(length(CurCats),1);
            % Now add the category that is meant to be
            % added and remove the category that is meant
            % to be removed
            NewVals(CurCats(RemoveInd))=false();
            clear RemoveInd
            % Insert the new values
            OffCC(i,CurInd)=NewVals;
        end
    end
    clear CurNumCats
    clear CurCats
    clear RandChoice
    clear CurInd
    clear NewVals
elseif Param.DataType(MutInd(j))==3
    % Then the data is unique categorical or binary
    % Determine the CurIndex in the CCs
```

```
CurInd=[Param.FeatInd{MutInd(j)}];
% Determine the number of categories of the feature
CurNumCats=length(CurInd);
% Determine the category that is currently active
CurCat=find(OffCC(i,CurInd));
% Create a list of possible categories that can be
% activated
PosCats=setdiff(1:CurNumCats,CurCat);
clear CurCat
% Now randomly select a category index
SelCatInd=randi(CurNumCats-1,1);
% Grab the selected category
SelCat=PosCats(SelCatInd);
clear PosCats
clear SelCatInd
% Create a logical vector the same size as the CurInd
NewVals=false(1,CurNumCats);
clear CurNumCats
% Now set the selected categories to true
NewVals(SelCat)=true();
% Insert the new values
OffCC(i,CurInd)=NewVals;
```

```
            clear CurInd
            clear SelCat
            clear NewVals
        end
    end
  end
end
% Calculate the fitness of the newly created offspring
% First calculate the order of the offspring
OffCCOrder(i)=sum(OffCCFeats(i,:),2);
% Create a temporary Order to handle CCs with too many features
if OffCCOrder(i)<=Param.Thresh(end,1)
  % Then the order is unchanged
  TempOrder=OffCCOrder(i);
else
  TempOrder=Param.Thresh(end,1);
end
% Now Determine the fitness of the new rule
% First Determine the Total Observations that have data for the
% selected features
TotObs=sum(sum(~NaNMask(:,OffCCFeats(i,:)),2)==OffCCOrder(i));
TotObsT=sum(sum(~NaNMask(TargetClass,OffCCFeats(i,:)),2)== ...
```

```
                                        OffCCOrder(i));
% Now determine which observations the current Conjunctive clause
% Matches
% Need to index the columns of the features that are selected
CurInd=[Param.FeatInd{OffCCFeats(i,:)}];
% First add the conjunctive clause to the data
TwosSum=bsxfun(@plus,OffCC(i,CurInd),DataBin(:,CurInd));
clear CurInd
% Now create a twos mask
TwosMask=TwosSum==2;
clear TwosSum
% Now determine the total number of twos
TotTwos=sum(TwosMask,2);
clear TwosMask
% Now create an observation match mask
OffCCMatchLocs(:,i)=TotTwos==OffCCOrder(i);
clear TotTwos
% Determine the total number of observations that match and are target
% class
TotCCObs=sum(OffCCMatchLocs(:,i));
TotCCObsT=sum(OffCCMatchLocs(TargetClass,i));
% If TotCCObsT/TotCCObs > TotObsT/TotObs then evaluate the fitness of
```

```
% the Conjunctive clause using hypergeometric PMF
if TotCCObsT/TotCCObs>TotObsT/TotObs
   % Calculate the fitness
   OffCCFit(i)=hygepdf(TotCCObsT,TotObs,TotObsT,TotCCObs);
   % Update the keep mask
   KeepMask(i)=true();
   % Determine if the CC is archivable
   % First extract the order mask
   OrderMask=Param.Thresh(:,1)==TempOrder;
   if sum(OrderMask)==0
      % Then set the last OrderMask to true
      OrderMask(end)=true();
   end
   ArchiveMask(i)=OffCCFit(i)<=Param.Thresh(OrderMask,2);
   if ArchiveMask(i)
      % Then record an archived evaluation
      CCstats.EvalsArchive(Param.CurGen,TempOrder)=...
                  CCstats.EvalsArchive(Param.CurGen,TempOrder)+1;
   else
      % Record a non-archived evaluation
      CCstats.EvalsNonArchive(Param.CurGen,TempOrder)=...
               CCstats.EvalsNonArchive(Param.CurGen,TempOrder)+1;
```

```
        end
        clear OrderMask
    else
        OffCCFit(i)=1;
    end
    % Record the total number of evaluations for the current order
    CCstats.EvalsAll(Param.CurGen,TempOrder)=...
                        CCstats.EvalsAll(Param.CurGen,TempOrder)+1;
    % Save the fitness component values
    OffCCFitComp.TotObs(i)=TotObs;
    OffCCFitComp.TotObsT(i)=TotObsT;
    OffCCFitComp.TotCCObs(i)=TotCCObs;
    OffCCFitComp.TotCCObsT(i)=TotCCObsT;
    clear TotObs
    clear TotObsT
    clear TotCCObs
    clear TotCCObsT
    clear TempOrder
end
% Only keep the values in the keep mask
OffCC=OffCC(KeepMask,:);
OffCCFeats=OffCCFeats(KeepMask,:);
```

```
OffCCFit=OffCCFit(KeepMask,1);
OffCCFitComp.TotObs=OffCCFitComp.TotObs(KeepMask,1);
OffCCFitComp.TotObsT=OffCCFitComp.TotObsT(KeepMask,1);
OffCCFitComp.TotCCObs=OffCCFitComp.TotCCObs(KeepMask,1);
OffCCFitComp.TotCCObsT=OffCCFitComp.TotCCObsT(KeepMask,1);
OffCCOrder=OffCCOrder(KeepMask,1);
OffCCMatchLocs=OffCCMatchLocs(:,KeepMask);
OffCCAge=OffCCAge(KeepMask);
ArchiveMask=ArchiveMask(KeepMask,1);
clear KeepMask
```

8.1.3 Disjunctive Normal Form EA (DNFEA)

```
 function [ArchDNF, ArchDNFMatchLocs, ArchDNFfit, ArchDNFacc,...
        ArchDNFcov, ArchDNFage, NonArchDNF, NonArchDNFMatchLocs,...
        NonArchDNFfit, NonArchDNFacc, NonArchDNFcov, ...
        NonArchDNFage, ParamDNF, DNFstats]=...
        DNFEA(CCMatchLocs,TargetClass,ParamDNF)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
% October 24, 2016
% Last Updated: October 25, 2016
%
```

```
% DNFEA is the evolutionary algorithm designed to find the disjunctive
% normal form (DNF) of the conjunctive clauses found using
% the CCEA function.
%
% Inputs:
% CCMatchLocs = A logical matrix where each row represents an observation
%               and each column represents a conjunctive clause (CC).
% TargetClass = A logical vector indicating the observations that are part
%               of the target class.
% ParamDNF = A structure array with the parameters needed to run the
%            algorithm.
%            .ALna = Number of non-archived age-layers.
%            .GENn = The number of generations until a novel population is
%                    introduced.
%            .POPn = Number of DNFs created in novel population.
%            .MAXcc = The maximum number of CCs allowed in novel population
%            .TotGens = Total number of generations to run the algorithm.
%            .Prx = The probability of crossover.
%            .Pm = The probability of loci mutation.
%            .Pbf = The probability that mutation is standard bit flip vs
%                   targeted mutation.
%            .Pxf = The Probability that mate is selected based on best
```

```
%           fitness vs other metrics.
%       .PxAlt = The cumulative sum for alternative mate selection.
%       .PmAlt = The probabilities for targeted mutation (must sum to
%           one).
%       .TournSize = The size of the tournament for mate selection
%               with replacement.
%       .NonArchLMax = The maximum number of non-archived DNFs
%               allowed in each non-archive layer.
%       .ArchOff = The maximum number of archived offsrping that will
%               undergo mutation or crossover.
%       .Thresh = A matrix with the initial threshold settings. The
%               matrix has 4 columns with the first column
%               containing all of the orders of the DNFs that the
%               user is interested in. For instance if the user
%               wants to explore DNFs of orders 1 - 6, then each
%               row represents the order. The second column is the
%               initial probability threshold [0, 1]. The third
%               column is the minimum number of DNFs the user wants
%               to save for each order. The fourth and final column
%               is the maximum number of DNFs the user wants to
%               archive for the given order. If the maximum is
%               exceeded then the threshold for the given order is
```

```
%                   replaced.
%          .BestFit = A logical indicator if the user wants to record the
%                    the best fit each generation for each order.
%
% Outputs:
% ArchDNF = The archived disjunctive normal form (DNF). Each
%          column represents a DNF and each row represents a conjunctive
%          clause.
% ArchDNFMatchLocs = A logical matrix where each row represents an
%                  observation and each column represents an ArchDNF.
% ArchDNFfit = The fitness (using the hypergeometric PMF) of the ArchDNF.
% ArchDNFacc = The accuracy of the ArchDNF.
% ArchDNFcov = The coverage of the ArchDNF.
% ArchDNFage = The age of the ArchDNF.
% NonArchDNF = The non-archived disjunctive normal form (DNF).
%             Each column represents a DNF and each row represents a
%             conjunctive clause.
% NonArchDNFMatchLocs = A logical matrix where each row represents an
%                    observation and each column represents an
%                    NonArchDNF.
% NonArchDNFfit = The fitness (using the hypergeometric PMF) of the
%               NonArchDNF.
```

```
% NonArchDNFacc = The accuracy of the NonArchDNF.
% NonArchDNFcov = The coverage of the NonArchDNF.
% NonArchDNFage = The age of the NonArchDNF.
% ParamDNF = A structure array with the parameters needed to run the
%           algorithm.
% DNFstats = A struture array with various statistics on the DNF for each
%           generation and each order. The main statistics are the number
%           of fitness evaluations and the best fitness each generation
%           for each order DNF.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For efficiency
DNFstats.EvalsAll=zeros(ParamDNF.TotGens, ParamDNF.MAXcc);
DNFstats.EvalsArchive=zeros(ParamDNF.TotGens, ParamDNF.MAXcc);
DNFstats.EvalsNonArchive=zeros(ParamDNF.TotGens, ParamDNF.MAXcc);
if ParamDNF.BestFit
   % Then the best fitness for each order for each generation will be
   % recorded
   DNFstats.BestFit=NaN(ParamDNF.TotGens, ParamDNF.MAXcc);
end
% Determine the current number of observations and number of target
% observations
```

```
ParamDNF.NumObs=length(TargetClass);
ParamDNF.TotTarC=sum(TargetClass);
% Determine the number of conjunctive clauses
ParamDNF.NumCCs=size(CCMatchLocs,2);
% Set the current generation
ParamDNF.CurGen=1;
% first randomly create DNFs
% Maybe create a probability that will pick conjunctive clauses that cover
% target observations that are not well covered in the archiveDNF
[NewDNF, NewDNFMatchLocs, NewDNFfit, NewDNFacc, NewDNFcov,...
    ArchiveMask, DNFstats]=DNFPopInit(CCMatchLocs,TargetClass,...
    ParamDNF,DNFstats);
% Separate into archive and non-archive conjunctive clauses
ArchDNF=NewDNF(ArchiveMask,:);
ArchDNFMatchLocs=NewDNFMatchLocs(:,ArchiveMask);
ArchDNFfit=NewDNFfit(ArchiveMask);
ArchDNFacc=NewDNFacc(ArchiveMask);
ArchDNFcov=NewDNFcov(ArchiveMask);
ArchDNFage=ones(size(ArchDNFfit));
NonArchDNF=NewDNF(~ArchiveMask,:);
NonArchDNFMatchLocs=NewDNFMatchLocs(:,~ArchiveMask);
NonArchDNFfit=NewDNFfit(~ArchiveMask);
```

```
NonArchDNFacc=NewDNFacc(~ArchiveMask);
NonArchDNFcov=NewDNFcov(~ArchiveMask);
NonArchDNFage=ones(size(NonArchDNFfit));
clear NewDNF
clear NewDNFMatchLocs
clear NewDNFfit
clear NewDNFacc
clear NewDNFcov
clear NewDNFage
clear ArchiveMask
% Clean the DNFs
[ArchDNF, ArchDNFMatchLocs, ArchDNFfit, ArchDNFacc, ArchDNFcov,...
   ArchDNFage, NonArchDNF, NonArchDNFMatchLocs, NonArchDNFfit,...
   NonArchDNFacc, NonArchDNFcov, NonArchDNFage, ParamDNF, ...
   DNFstats]=...
   DNFreducepop(ArchDNF,ArchDNFMatchLocs,ArchDNFfit,ArchDNFacc,...
   ArchDNFcov,ArchDNFage,NonArchDNF,NonArchDNFMatchLocs,...
   NonArchDNFfit,NonArchDNFacc,NonArchDNFcov,NonArchDNFage,...
   ParamDNF,DNFstats);
for gen=2:ParamDNF.TotGens
   % set the current generation
   ParamDNF.CurGen=gen;
```

```
% Increase the age of the non-archived population
NonArchDNFage=NonArchDNFage+1;
% Determine if a new population should be added
if mod(gen,ParamDNF.GENn)~=0
   % Then just perform crossover or mutation on population
   [ArchDNF, ArchDNFMatchLocs, ArchDNFfit, ArchDNFacc, ...
      ArchDNFcov, ArchDNFage, NonArchDNF, NonArchDNFMatchLocs,...
      NonArchDNFfit, NonArchDNFacc, NonArchDNFcov, ...
      NonArchDNFage, DNFstats]=...
      DNFEvolution(ArchDNF,ArchDNFMatchLocs,ArchDNFfit, ...
      ArchDNFacc,ArchDNFcov,ArchDNFage,NonArchDNF, ...
      NonArchDNFMatchLocs,NonArchDNFfit,NonArchDNFacc, ...
      NonArchDNFcov,NonArchDNFage,CCMatchLocs,TargetClass,...
      ParamDNF,DNFstats);
   % Clean the DNFs
   [ArchDNF, ArchDNFMatchLocs, ArchDNFfit, ArchDNFacc, ...
      ArchDNFcov, ArchDNFage, NonArchDNF, NonArchDNFMatchLocs,...
      NonArchDNFfit,NonArchDNFacc, NonArchDNFcov, ...
      NonArchDNFage, ParamDNF, DNFstats]=...
      DNFreducepop(ArchDNF,ArchDNFMatchLocs,ArchDNFfit,...
      ArchDNFacc,ArchDNFcov,ArchDNFage,NonArchDNF,...
      NonArchDNFMatchLocs,NonArchDNFfit,NonArchDNFacc,...
```

```
        NonArchDNFcov,NonArchDNFage,ParamDNF,DNFstats);
else
    % Then add a new population and perform  crossover or mutation on
    % population
    % first perform mutation or crossover
    % Then just perform crossover or mutation on population
    [ArchDNF, ArchDNFMatchLocs, ArchDNFfit, ArchDNFacc, ...
        ArchDNFcov, ArchDNFage, NonArchDNF, NonArchDNFMatchLocs,...
        NonArchDNFfit, NonArchDNFacc, NonArchDNFcov, ...
        NonArchDNFage, DNFstats]=...
        DNFEvolution(ArchDNF,ArchDNFMatchLocs,ArchDNFfit,...
        ArchDNFacc,ArchDNFcov,ArchDNFage,NonArchDNF,...
        NonArchDNFMatchLocs,NonArchDNFfit,NonArchDNFacc,...
        NonArchDNFcov,NonArchDNFage,CCMatchLocs,TargetClass,...
        ParamDNF,DNFstats);
    % Now create a new population of DNFs
    [NewDNF, NewDNFMatchLocs, NewDNFfit, NewDNFacc, NewDNFcov,...
        ArchiveMask, DNFstats]=DNFPopInit(CCMatchLocs,TargetClass,...
        ParamDNF,DNFstats);
    % First set the age of the NewDNF
    NewDNFage=ones(size(NewDNFfit));
    % Now combine the new DNFs with the existing population
```

```
ArchDNF=[ArchDNF; NewDNF(ArchiveMask,:)];
ArchDNFMatchLocs=[ArchDNFMatchLocs ...
                    NewDNFMatchLocs(:,ArchiveMask)];
ArchDNFfit=[ArchDNFfit; NewDNFfit(ArchiveMask)];
ArchDNFacc=[ArchDNFacc; NewDNFacc(ArchiveMask)];
ArchDNFcov=[ArchDNFcov; NewDNFcov(ArchiveMask)];
ArchDNFage=[ArchDNFage; NewDNFage(ArchiveMask)];
NonArchDNF=[NonArchDNF; NewDNF(~ArchiveMask,:)];
NonArchDNFMatchLocs=[NonArchDNFMatchLocs...
                    NewDNFMatchLocs(:,~ArchiveMask)];
NonArchDNFfit=[NonArchDNFfit; NewDNFfit(~ArchiveMask)];
NonArchDNFacc=[NonArchDNFacc; NewDNFacc(~ArchiveMask)];
NonArchDNFcov=[NonArchDNFcov; NewDNFcov(~ArchiveMask)];
NonArchDNFage=[NonArchDNFage; NewDNFage(~ArchiveMask)];
clear NewDNF
clear NewDNFMatchLocs
clear NewDNFfit
clear NewDNFacc
clear NewDNFcov
clear NewDNFage
clear ArchiveMask
% Clean the DNFs
```

```
        [ArchDNF, ArchDNFMatchLocs, ArchDNFfit, ArchDNFacc, ...
            ArchDNFcov, ArchDNFage, NonArchDNF, NonArchDNFMatchLocs,...
            NonArchDNFfit,NonArchDNFacc, NonArchDNFcov, ...
            NonArchDNFage, ParamDNF, DNFstats]=...
            DNFreducepop(ArchDNF,ArchDNFMatchLocs,ArchDNFfit,...
            ArchDNFacc,ArchDNFcov,ArchDNFage,NonArchDNF,...
            NonArchDNFMatchLocs,NonArchDNFfit,NonArchDNFacc,...
            NonArchDNFcov,NonArchDNFage,ParamDNF,DNFstats);
    end
end
```

8.1.3.1 Disjunction of CC Population Initialization (DNFPopInit)

```
function [NewDNF, NewDNFMatchLocs, NewDNFfit, NewDNFacc, NewDNFcov,...
             ArchiveMask, DNFstats]=...
             DNFPopInit(CCMatchLocs,TargetClass,ParamDNF,DNFstats)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
% October 20, 2016
% Last Updated: October 20, 2016
%
% DNFPopInit is the population initialization algorithm for the DNF EA.
%
```

```
% Inputs:
% CCMatchLocs = A logical matrix with each column representing a
%               conjunctive clause and each row representing an
%               observation.
% TargetClass = A logical vector indicating the observations that are in
%               the target class.
% ParamDNF = A structure array with the parameters needed to run DNFEA.
% DNFstats = Statistics on the DNF evolution.
%
% Outputs:
% NewDNF = The disjunctive normal form index. Each row
%          represents a DNF and each column indexes a conjunctive clause.
% NewDNFMatchLocs = Each column represents a DNF and each row represents
%                   anobservation. NewDNFMatchLocs indexes the
%                   observations the DNF matches.
% NewDNFfit = The fitness of the DNF using the hypergeometric PMF
% NewDNFacc = The accuracy (% True Positives) of the DNF.
% NewDNFcov = The coverage of the DNF (i.e., % coverage of target
%             observations).
% ArchiveMask = A mask indicating the DNFs that should be archived.
% DNFstats = Statistics on the DNF evolution.
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% first randomly create DNFs
% Maybe create a probability that will pick conjunctive clauses that cover
% target observations that are not well covered in the archiveDNF
NewDNF=false(ParamDNF.POPn,ParamDNF.NumCCs);
NewDNFMatchLocs=false(ParamDNF.NumObs,ParamDNF.POPn);
NewDNFfit=NaN(ParamDNF.POPn,1);
NewDNFacc=NaN(ParamDNF.POPn,1);
NewDNFcov=NaN(ParamDNF.POPn,1);
KeepMask=false(ParamDNF.POPn,1);
ArchiveMask=false(ParamDNF.POPn,1);
for i=1:ParamDNF.POPn
   % First randomly determine the number of conjunctive clauses
   NumCCs=randi(ParamDNF.MAXcc,1);
   % Now randomly determine the conjunctive clauses that will be in DNF
   CCind=randperm(ParamDNF.NumCCs,NumCCs);
   % Set the new DNF
   NewDNF(i,CCind)=true();
   % Extract the CCs to create the DNF
   CurCCs=CCMatchLocs(:,CCind);
   clear CCind
   % Now determine the DNFMatchLocs
```

```
CurCCsSum=sum(CurCCs,2);
clear CurCCs
% DNFMatchLocs is simply a mask of CurCCsSum>0
NewDNFMatchLocs(:,i)=CurCCsSum>0;
clear CurCCsSum
% Now determine the fitness
ntot=sum(NewDNFMatchLocs(:,i));
xmatch=sum(NewDNFMatchLocs(TargetClass,i));
% Ensure the order will be recorded
if NumCCs>ParamDNF.Thresh(end,1)
   NumCCs=ParamDNF.Thresh(end,1);
end
% Determine if the DNF is a target class DNF
if xmatch/ntot>ParamDNF.TotTarC/ParamDNF.NumObs
   % calculate the fitness function
   NewDNFfit(i)=hygepdf(xmatch,ParamDNF.NumObs,ParamDNF.TotTarC,...
                                                ntot);
   % Calculate the accuracy and coverage
   NewDNFacc(i)=xmatch/ntot*100;
   NewDNFcov(i)=xmatch/ParamDNF.TotTarC*100;
   % Set the KeepMask to true
   KeepMask(i)=true();
```

```
        % Determine if the new DNF should be archived
        Tmask=ParamDNF.Thresh(:,1)==NumCCs;
        if NewDNFfit(i)<=ParamDNF.Thresh(Tmask,2)
            % Then it is archiveable
            ArchiveMask(i)=true();
            DNFstats.EvalsArchive(ParamDNF.CurGen,NumCCs)=...
                        DNFstats.EvalsArchive(ParamDNF.CurGen,NumCCs)+1;
        else
            % then it is non-archiveable
            DNFstats.EvalsNonArchive(ParamDNF.CurGen,NumCCs)=...
                    DNFstats.EvalsNonArchive(ParamDNF.CurGen,NumCCs)+1;
        end
        clear Tmask
    end
    % Record the an evaluation for the current order
    DNFstats.EvalsAll(ParamDNF.CurGen,NumCCs)=...
                        DNFstats.EvalsAll(ParamDNF.CurGen,NumCCs)+1;
    clear NumCCs
    clear ntot
    clear xmatch
end
% Keep only the DNFs in the DNF mask
```

```
NewDNF=NewDNF(KeepMask,:);
NewDNFMatchLocs=NewDNFMatchLocs(:,KeepMask);
NewDNFfit=NewDNFfit(KeepMask,1);
NewDNFacc=NewDNFacc(KeepMask,1);
NewDNFcov=NewDNFcov(KeepMask,1);
ArchiveMask=ArchiveMask(KeepMask,1);
clear KeepMask
```

8.1.3.2 Remove Repeat DNFs (DNFreducepop)

```
function [ArchDNF, ArchDNFMatchLocs, ArchDNFfit, ArchDNFacc, ...
        ArchDNFcov, ArchDNFage, NonArchDNF, NonArchDNFMatchLocs,...
        NonArchDNFfit, NonArchDNFacc, NonArchDNFcov, ...
        NonArchDNFage, ParamDNF, DNFstats]=...
        DNFreducepop(ArchDNF,ArchDNFMatchLocs,ArchDNFfit,...
        ArchDNFacc,ArchDNFcov,ArchDNFage,NonArchDNF,...
        NonArchDNFMatchLocs,NonArchDNFfit,NonArchDNFacc,...
        NonArchDNFcov,NonArchDNFage,ParamDNF,DNFstats)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
% October 24, 2016
% Last Updated: October 24, 2016
%
```

```
% DNFreducepop will remove any repeat disjunctive normal forms
% (DNFS) and will reduce the DNFs populations if they exceed their
% thresholds.
%
% Inputs:
% ArchDNF = The archived disjunctive normal form (DNF). Each
%           column represents a DNF and each row represents a conjunctive
%           clause.
% ArchDNFMatchLocs = A logical matrix where each row represents an
%                    observation and each column represents an ArchDNF.
% ArchDNFfit = The fitness (using the hypergeometric PMF) of the ArchDNF.
% ArchDNFacc = The accuracy of the ArchDNF.
% ArchDNFcov = The coverage of the ArchDNF.
% ArchDNFage = The age of the ArchDNF.
% NonArchDNF = The non-archived disjunctive normal form (DNF).
%              Each column represents a DNF and each row represents a
%              conjunctive clause.
% NonArchDNFMatchLocs = A logical matrix where each row represents an
%                       observation and each column represents an
%                       NonArchDNF.
% NonArchDNFfit = The fitness (using the hypergeometric PMF) of the
%                 NonArchDNF.
```

```
% NonArchDNFacc = The accuracy of the NonArchDNF.
% NonArchDNFcov = The coverage of the NonArchDNF.
% NonArchDNFage = The age of the NonArchDNF.
% ParamDNF = A structure array with the parameters needed to run the
%          algorithm.
% DNFstats = Statistics on the DNF evolution.
%
% Outputs:
% ArchDNF = The archived disjunctive normal form (DNF). Each
%          column represents a DNF and each row represents a conjunctive
%          clause.
% ArchDNFMatchLocs = A logical matrix where each row represents an
%                  observation and each column represents an ArchDNF.
% ArchDNFfit = The fitness (using the hypergeometric PMF) of the ArchDNF.
% ArchDNFacc = The accuracy of the ArchDNF.
% ArchDNFcov = The coverage of the ArchDNF.
% ArchDNFage = The age of the ArchDNF.
% NonArchDNF = The non-archived disjunctive normal form (DNF).
%            Each column represents a DNF and each row represents a
%            conjunctive clause.
% NonArchDNFMatchLocs = A logical matrix where each row represents an
%                    observation and each column represents an
```

```
%                     NonArchDNF.
% NonArchDNFfit = The fitness (using the hypergeometric PMF) of the
%                NonArchDNF.
% NonArchDNFacc = The accuracy of the NonArchDNF.
% NonArchDNFcov = The coverage of the NonArchDNF.
% NonArchDNFage = The age of the NonArchDNF.
% ParamDNF = A structure array with the parameters needed to run the
%           algorithm.
% DNFstats = Statistics on the DNF evolution.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start by reducing the any repeated DNFs in the Archive population
if length(ArchDNFage)>1
   % Then check to see if there are any repeat DNFs
   [ArchDNF, ID]=unique(ArchDNF,'rows');
   % Now save the unique conjunctive clauses
   ArchDNFMatchLocs=ArchDNFMatchLocs(:,ID);
   ArchDNFfit=ArchDNFfit(ID);
   ArchDNFacc=ArchDNFacc(ID);
   ArchDNFcov=ArchDNFcov(ID);
   ArchDNFage=ArchDNFage(ID);
   clear ID
```

```
end
if length(ArchDNFage)>1
    % Determine if any of the archive bins are over their limit
    % First create a temporary order so that all conjunctive clauses
    % greater than the max bin are set to max bin
    TempOrder=sum(ArchDNF,2);
    % Now tabulate the temporary order
    TabTempOrder=tabulate(TempOrder);
    % Remove any rows that do not have a value
    TabTempOrder=TabTempOrder(TabTempOrder(:,2)>0,:);
    % Compare the tabulated TempOrder to the associated maximum allowable
    % populations
    % Test to see if TabTempOrder are the same
    if size(TabTempOrder,1)==size(ParamDNF.Thresh,1)
        % Then all orders are present
        % Determine how many if any bins are over the limit
        LimitMask=TabTempOrder(:,2)>ParamDNF.Thresh(:,4);
    else
        % Then not all orders are present so need to determine which orders
        % are present
        % First set up a logical vector for efficiency
        LimitMask=false(size(ParamDNF.Thresh,1),1);
```

```
    % for each of the orders present, determine if the limit is
    % surpassed
    for i=1:size(TabTempOrder,1)
        % Grab the ith order
        CurOrder=TabTempOrder(i,1);
        % Create a mask of the Order
        TempMask=CurOrder==ParamDNF.Thresh(:,1);
        clear CurOrder
        % Now check to see if the limit is surpassed
        LimitMask(TempMask)=TabTempOrder(i,2)> ...
                                ParamDNF.Thresh(TempMask,4);
        clear TempMask
    end
    clear i
end
% If the sum of limit mask is greater than 0 then at least one bin is
% over the limit so reduce the population of the bin
if sum(LimitMask)>0
    % Then for each bin over the limit reduce the bin population
    % Fist determine the orders of conjunctive clauses that are over
    % the mask
    OrderOver=ParamDNF.Thresh(LimitMask,1);
```

```
% Create a keep mask for efficiency
KeepMask=true(size(ArchDNFage));
for i=1:length(OrderOver)
    % Create a mask of the current OrderOver
    OrderMask=OrderOver(i)==TempOrder;
    % Create a mask for ParamDNF.Thresh Table
    ThreshMask=OrderOver(i)==ParamDNF.Thresh(:,1);
    % Determine if the current threshold is 0
    if OrderOver(i)==1
        % Then randomly choose the order 1 DNFs to keep. This
        % ensures that there is diversity in order 1 since we
        % already know the order 1 population.
        % First determine the number in the mask
        TotOrd1=sum(OrderMask);
        % Now find the minimum allowed
        CurMin=ParamDNF.Thresh(ThreshMask,3);
        % Now find the number to remove
        Num2Remove=TotOrd1-CurMin;
        clear TotOrd1
        % Now find the locations of the Order 1 DNFs
        Ord1Locs=find(OrderMask);
        % Now randomly select the order 1 to remove
```

```
    Ord1ID=randperm(length(Ord1Locs),Num2Remove);
    clear Num2Remove
    % Create a remove mask
    RemoveMask=false(size(KeepMask));
    % Set the randomly chosen locations to true
    RemoveMask(Ord1Locs(Ord1ID))=true();
    clear Ord1Locs
    clear Ord1ID
    % Now set the RemoveMask locs to false in the KeepMask
    KeepMask(RemoveMask)=false();
elseif ParamDNF.Thresh(ThreshMask,2)~=0
    % Then need to sort based on fitness
    % sort the fitness of the current order fitness
    CurSortFit=sort(ArchDNFfit(OrderMask));
    % Find the minimum number for this bin
    CurMin=ParamDNF.Thresh(ThreshMask,3);
    % Now use the CurMin to find the fitness of sorted fitness
    % and use this to set the new threshold
    ParamDNF.Thresh(ThreshMask,2)=CurSortFit(CurMin);
    clear CurSortFit
    % Create a Mask with all the DNFs that are below or equal
    % to the threshold
```

```
BelowThreshMask=ArchDNFfit<= ...
                    ParamDNF.Thresh(ThreshMask,2);
% Now determine if the number that will be saved is greater
% than the max
if sum(BelowThreshMask&OrderMask)<=...
                    ParamDNF.Thresh(ThreshMask,4)
   % Now create a mask of all the archived conjunctive
   % clauses with a fitness greater than the new threshold
   AboveThreshMask=ArchDNFfit> ...
                    ParamDNF.Thresh(ThreshMask,2);
   % Now create a mask for Removal
   RemoveMask=AboveThreshMask&OrderMask;
   clear AboveThreshMask
   % Now set the RemoveMask locations to false
   KeepMask(RemoveMask)=false();
else
   % There are too many to save
   % First determine the number in the current order that
   % are equal to the current threshold
   EqualThreshMask=ArchDNFfit== ...
                    ParamDNF.Thresh(ThreshMask,2);
   NumEqual=sum(EqualThreshMask&OrderMask);
```

```
% Determine the number in the current order that are
% below the current threshold
BelowThreshMask=ArchDNFfit< ...
                    ParamDNF.Thresh(ThreshMask,2);
NumBelow=sum(BelowThreshMask&OrderMask);
% Determine locations of the ones that are equal
EqualDNFs=find(EqualThreshMask&OrderMask);
clear EqualThreshMask
% Determine the number to remove from the above
Num2Remove=(NumEqual+NumBelow)-CurMin;
% Now randomly select the ones above to remove
RandID=randperm(NumEqual,Num2Remove);
clear Num2Remove
clear NumBelow
clear NumEqual
% Then create a mask of all the archived DNFs with an
% ArchDNFsum less than the SumThresh
AboveThreshMask=ArchDNFfit> ...
                    ParamDNF.Thresh(ThreshMask,2);
% Now create a mask for Removal
RemoveMask=AboveThreshMask&OrderMask;
clear AboveThreshMask
```

```
        % Now set the BelowDNFs that were randomly selected to
        % true
        RemoveMask(EqualDNFs(RandID))=true();
        clear EqualDNFs
        clear RandID
        % Now set the RemoveMask locations to false
        KeepMask(RemoveMask)=false();
    end
    clear BelowThreshMask
else
    % Then need to sort based on sum of acc + cov
    % Determine the sum of acc and cov
    ArchDNFsum=ArchDNFacc+ArchDNFcov;
    % sort the sum of Acc and Cov of the current order
    CurSortFit=sort(ArchDNFsum(OrderMask),'descend');
    % Find the minimum number for this bin
    CurMin=ParamDNF.Thresh(ThreshMask,3);
    % Now determine the sum Thresh
    SumThresh=CurSortFit(CurMin);
    clear CurSortFit
    % Now create a mask of all the archived DNFs with an
    % ArchDNFsum greater than or equal to the SumThresh
```

```
AboveThreshMask=ArchDNFsum>=SumThresh;
% Determine if the number that will be saved in the current
% order is greater than the maximum allowed to be saved
if sum(AboveThreshMask&OrderMask)<=...
                    ParamDNF.Thresh(ThreshMask,4)
   % Then create a mask of all the archived DNFs with an
   % ArchDNFsum less than the SumThresh
   BelowThreshMask=ArchDNFsum<SumThresh;
   % Now create a mask for Removal
   RemoveMask=BelowThreshMask&OrderMask;
   clear BelowThreshMask
   % Now set the RemoveMask locations to false
   KeepMask(RemoveMask)=false();
else
   % Then need to randomly choose from the DNFs above the
   % thresholdhold mask
   % First determine the number in the current orer that
   % are equal to the current threshold
   EqualThreshMask=ArchDNFsum==SumThresh;
   NumEqual=sum(EqualThreshMask&OrderMask);
   % Determine the number in the current order that
   % are abovel to the current threshold
```

```
AboveThreshMask=ArchDNFsum>SumThresh;
NumAbove=sum(AboveThreshMask&OrderMask);
% Determine locations of the ones above
EqualDNFs=find(EqualThreshMask&OrderMask);
clear EqualThreshMask
% Determine the number to remove from the above
Num2Remove=(NumAbove+NumEqual)-CurMin;
% Now randomly select the ones above to remove
RandID=randperm(NumEqual,Num2Remove);
clear Num2Remove
clear NumAbove
clear NumEqual
% Then create a mask of all the archived DNFs with an
% ArchDNFsum less than the SumThresh
BelowThreshMask=ArchDNFsum<SumThresh;
% Now create a mask for Removal
RemoveMask=BelowThreshMask&OrderMask;
clear BelowThreshMask
% Now set the AboveDNFs that were randomly selected to
% true
RemoveMask(EqualDNFs(RandID))=true();
clear EqualDNFs
```

```
            clear RandID
            % Now set the RemoveMask locations to false
            KeepMask(RemoveMask)=false();
        end
        clear AboveThreshMask
        clear ArchDNFsum
        clear SumThresh
    end
    clear CurMin
    clear ThreshMask
    clear OrderMask
    clear RemoveMask
end
% Determine if any of the removed Archived conjunctive clauses have
% an age that can be moved to the non-archive population
% Create a mask of the archive population that is young enough to
% fit in the non-archive population
YoungPop=ArchDNFage<=(ParamDNF.GENn*ParamDNF.ALna);
% Now determine if there are any young popvalues that will be
% removed
Move2NonArch=YoungPop&~KeepMask;
clear YoungPop
```

```
    if sum(Move2NonArch)>0
        % Then move the selected features to the non-archive population
        NonArchDNF=[NonArchDNF; ArchDNF(Move2NonArch,:)];
        NonArchDNFMatchLocs=[NonArchDNFMatchLocs...
                                ArchDNFMatchLocs(:,Move2NonArch)];
        NonArchDNFfit=[NonArchDNFfit; ArchDNFfit(Move2NonArch)];
        NonArchDNFacc=[NonArchDNFacc; ArchDNFacc(Move2NonArch)];
        NonArchDNFcov=[NonArchDNFcov; ArchDNFcov(Move2NonArch)];
        NonArchDNFage=[NonArchDNFage; ArchDNFage(Move2NonArch)];
        clear Move2NonArch
    end
    % Keep only those conjunctive clauses that are in the KeepMask
    ArchDNF=ArchDNF(KeepMask,:);
    ArchDNFMatchLocs=ArchDNFMatchLocs(:,KeepMask);
    ArchDNFfit=ArchDNFfit(KeepMask);
    ArchDNFacc=ArchDNFacc(KeepMask);
    ArchDNFcov=ArchDNFcov(KeepMask);
    ArchDNFage=ArchDNFage(KeepMask);
    clear KeepMask
end
clear LimitMask
clear TabTempOrder
```

```
end
% Remove Any NonArchAge that is now aged out
YoungMask=NonArchDNFage<(ParamDNF.GENn*ParamDNF.ALna);
% If there are any NonArchCCs to remove then remove them
if sum(~YoungMask)>0
   NonArchDNF=NonArchDNF(YoungMask,:);
   NonArchDNFMatchLocs=NonArchDNFMatchLocs(:,YoungMask);
   NonArchDNFfit=NonArchDNFfit(YoungMask);
   NonArchDNFacc=NonArchDNFacc(YoungMask);
   NonArchDNFcov=NonArchDNFcov(YoungMask);
   NonArchDNFage=NonArchDNFage(YoungMask);
end
clear YoungMask
% Now check to see if the Non-archived population is exceeded for each
% layer
if ~isempty(NonArchDNFage)
   % First determine the non-archive age layers for each conjunctive
   % clause
   NonArchDNFageLayer=ceil(NonArchDNFage/ParamDNF.GENn);
   % Now tabulate the NonArchCCAgeLayers
   TabNonArchLayer=tabulate(NonArchDNFageLayer);
   % Remove any TabNonArchLayer that does not have a value
```

```
    TabNonArchLayer=TabNonArchLayer(TabNonArchLayer(:,2)>0,:);
    % Determine if any of the TabNonArchLayer is greater than the maximum
    % allowed
    LimitMask=TabNonArchLayer(:,2)>ParamDNF.NonArchLMax;
else
    % Set the limit mask to false
    LimitMask=false();
end
% If any layer is greater then need to remove individuals from the
% non-archive layer
if sum(LimitMask)>0
    % Then determine which layers need to be reduced in size
    Layers=TabNonArchLayer(LimitMask,1);
    % for efficiency create a keep mask
    KeepMask=true(size(NonArchDNFfit));
    for i=1:length(Layers)
        % for each age layer select the most fit
        % First create a layer mask
        LayerMask=NonArchDNFageLayer==Layers(i);
        % Now sort by fitness
        CurSortFit=sort(NonArchDNFfit(LayerMask));
        % set a temporary threshold based on the CurSortFit
```

```
        TempThresh=CurSortFit(ParamDNF.NonArchLMax);
        clear CurSortFit
        % Now create a mask baed on threshold
        ThreshMask=NonArchDNFfit>TempThresh;
        clear TempThresh
        % Now create a remove mask
        RemoveMask=ThreshMask&LayerMask;
        clear LayerMask
        clear ThreshMask
        % Set all of the RemoveMask locations to false
        KeepMask(RemoveMask)=false();
        clear RemoveMask
    end
    % Now keep all the information in the keep mask
    NonArchDNF=NonArchDNF(KeepMask,:);
    NonArchDNFMatchLocs=NonArchDNFMatchLocs(:,KeepMask);
    NonArchDNFfit=NonArchDNFfit(KeepMask);
    NonArchDNFacc=NonArchDNFacc(KeepMask);
    NonArchDNFcov=NonArchDNFcov(KeepMask);
    NonArchDNFage=NonArchDNFage(KeepMask);
    clear KeepMask
end
```

```
clear NonArchDNFageLayer
clear LimitMask
clear TabNonArchLayer
% If the user wants to record the best fitness of each order then record
if ParamDNF.BestFit
   % Determine the best fitness for each order
   TempFit=[ArchDNFfit; NonArchDNFfit];
   % Determine the
   for i=1:ParamDNF.MAXcc
      if i~=ParamDNF.MAXcc
         % Then mask by current order
         CurOrderMask=[ArchCCOrder; NonArchCCOrder]==i;
         if sum(CurOrderMask)>0
            % Then record the best fitness
            DNFstats.BestFit(ParamDNF.CurGen,i)= ...
                                    min(TempFit(CurOrderMask));
         end
      else
         % The mask by the current order and any larger order
         CurOrderMask=[ArchCCOrder; NonArchCCOrder]>=i;
         if sum(CurOrderMask)>0
            % Then record the best fitness
```

```
            DNFstats.BestFit(ParamDNF.CurGen,i)= ...
                                    min(TempFit(CurOrderMask));
          end
        end
        clear CurOrderMask
    end
    clear i
    clear TempFit
end
```

8.1.3.3 Disjunctive Normal Form Evolution (DNFEvolution)

```
function [ArchDNF, ArchDNFMatchLocs, ArchDNFfit, ArchDNFacc, ...
        ArchDNFcov, ArchDNFage, NonArchDNF, NonArchDNFMatchLocs,...
        NonArchDNFfit, NonArchDNFacc, NonArchDNFcov, ...
        NonArchDNFage, DNFstats]=...
        DNFEvolution(ArchDNF,ArchDNFMatchLocs,ArchDNFfit,...
        ArchDNFacc,ArchDNFcov,ArchDNFage,NonArchDNF,...
        NonArchDNFMatchLocs,NonArchDNFfit,NonArchDNFacc,...
        NonArchDNFcov,NonArchDNFage,CCMatchLocs,TargetClass,...
        ParamDNF,DNFstats)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
```

```
% October 24, 2016
% Last Updated: October 24, 2016
%
% DNFEvolution evolves the population of disjunction of conjunctive
% clauses.
%
% Inputs:
% ArchDNF = The archived disjunctive normal forms (DNFs). Each
%           column represents a DNF and each row represents a conjunctive
%           clause.
% ArchDNFMatchLocs = A logical matrix where each row represents an
%                    observation and each column represents an ArchDNF.
% ArchDNFfit = The fitness (using the hypergeometric PMF) of the ArchDNF.
% ArchDNFacc = The accuracy of the ArchDNF.
% ArchDNFcov = The coverage of the ArchDNF.
% ArchDNFage = The age of the ArchDNF.
% NonArchDNF = The non-archived disjunctive normal form
%              (DNF). Each column represents a DNF and each row
%              represents a conjunctive  clause.
% NonArchDNFMatchLocs = A logical matrix where each row represents an
%                       observation and each column represents an
%                       NonArchDNF.
```

```
% NonArchDNFfit = The fitness (using the hypergeometric PMF) of the
%                NonArchDNF.
% NonArchDNFacc = The accuracy of the NonArchDNF.
% NonArchDNFcov = The coverage of the NonArchDNF.
% NonArchDNFage = The age of the NonArchDNF.
% CCMatchLocs = A logical matrix where each row represents an observation
%               and each column represents a conjunctive clause.
% TargetClass = A logical vector indicating which observations are in the
%               target class.
% ParamDNF = A structure array with the parameters needed to run the
%            algorithm.
% DNFstats = Statistics on the DNF evolution.
%
% Outputs:
% ArchDNF = The archived disjunctive normal form (DNF). Each
%           column represents a DNF and each row represents a conjunctive
%           clause.
% ArchDNFMatchLocs = A logical matrix where each row represents an
%                    observation and each column represents an ArchDNF.
% ArchDNFfit = The fitness (using the hypergeometric PMF) of the ArchDNF.
% ArchDNFacc = The accuracy of the ArchDNF.
% ArchDNFcov = The coverage of the ArchDNF.
```

```
% ArchDNFage = The age of the ArchDNF.
% NonArchDNF = The non-archived disjunctive normal form (DNF).
%             Each column represents a DNF and each row represents a
%             conjunctive clause.
% NonArchDNFMatchLocs = A logical matrix where each row represents an
%                   observation and each column represents an
%                   NonArchDNF.
% NonArchDNFfit = The fitness (using the hypergeometric PMF) of the
%               NonArchDNF.
% NonArchDNFacc = The accuracy of the NonArchDNF.
% NonArchDNFcov = The coverage of the NonArchDNF.
% NonArchDNFage = The age of the NonArchDNF.
% DNFstats = Statistics on the DNF evolution.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate the age layer of the non-archived population
NonArchDNFageLayer=ceil(NonArchDNFage/ParamDNF.GENn);
% If there is arhive layer, then add an archive age layer that is one more
% than the max age-layer in non-archived population
if ~isempty(ArchDNFage)
   % Check to make sure there is an non-archive population
   if ~isempty(NonArchDNFage)
```

```
        ArchDNFageLayer=ones(size(ArchDNFage))+max(NonArchDNFageLayer);
    else
        ArchDNFageLayer=ones(size(ArchDNFage));
    end
end
% First determine if there is an archive population and how big the
% archive population is
if length(ArchDNFfit)>ParamDNF.ArchOff
    % Then need to select offspring to undergo mutation
    % Sort the Archived DNFs by ArchDNFage
    [~, ID]=sort(ArchDNFage);
    % Create a selected DNF vector
    SelDNFVec=false(size(ArchDNFfit));
    % Now set 1:ParamDNF.ArchOff to true
    SelDNFVec(1:ParamDNF.ArchOff)=true();
    % set the number of archived offspring
    NumArchOff=sum(SelDNFVec);
    % Then the selected DNFs have enough diversity
    MateDNF=[ArchDNF(ID(SelDNFVec),:);...
            ArchDNF(ID(~SelDNFVec),:);...
            NonArchDNF];
    MateDNFfit=[ArchDNFfit(ID(SelDNFVec));...
```

```
            ArchDNFfit(ID(~SelDNFVec));...
            NonArchDNFfit];
    MateDNFage=[ArchDNFage(ID(SelDNFVec));...
            ArchDNFage(ID(~SelDNFVec));...
            NonArchDNFage];
    MateDNFMatchLocs=[ArchDNFMatchLocs(:,ID(SelDNFVec))...
                ArchDNFMatchLocs(:,ID(~SelDNFVec))...
                NonArchDNFMatchLocs];
    MateDNFageLayer=[ArchDNFageLayer(ID(SelDNFVec));...
                ArchDNFageLayer(ID(~SelDNFVec));...
                NonArchDNFageLayer];
elseif ~isempty(ArchDNFfit)
    % Then all archived offspring will be selected
    MateDNF=[ArchDNF; NonArchDNF];
    MateDNFfit=[ArchDNFfit; NonArchDNFfit];
    MateDNFage=[ArchDNFage; NonArchDNFage];
    MateDNFMatchLocs=[ArchDNFMatchLocs NonArchDNFMatchLocs];
    MateDNFageLayer=[ArchDNFageLayer; NonArchDNFageLayer];
    % Set the SelDNFVec to the lenght of ArchDNFfit and to True
    SelDNFVec=true(size(ArchDNFfit));
    % Set the number of offspring
    NumArchOff=sum(SelDNFVec);
```

```
    % set ID
    ID=1:length(ArchDNFage);
else
    % set the number of ArchOffsping to zero
    NumArchOff=0;
    % Then there is no archive population
    % Then the Mating population is simply the nonarchive
    % population
    MateDNF=NonArchDNF;
    MateDNFfit=NonArchDNFfit;
    MateDNFage=NonArchDNFage;
    MateDNFMatchLocs=NonArchDNFMatchLocs;
    MateDNFageLayer=NonArchDNFageLayer;
end
% For Effificiency
EvoDNF=false(NumArchOff+length(NonArchDNFfit),ParamDNF.NumCCs);
EvoDNFMatchLocs=false(ParamDNF.NumObs,NumArchOff+length(NonArchDNFfit
));
EvoDNFfit=NaN(NumArchOff+length(NonArchDNFfit),1);
EvoDNFacc=NaN(NumArchOff+length(NonArchDNFfit),1);
EvoDNFcov=NaN(NumArchOff+length(NonArchDNFfit),1);
EvoDNFage=NaN(NumArchOff+length(NonArchDNFfit),1);
```

```
EvoArchiveMask=false(NumArchOff+length(NonArchDNFfit),1);
% If there is an ArchPop then perform one task otherwise another
if NumArchOff>0
    % Then an archive age layer is present
    % Determine the number of age layers
    UniqueLayers=unique(MateDNFageLayer);
    NumLayers=length(UniqueLayers);
    % Initialize start
    start=1;
    % Run a for loop so that each age layer can undergo either
    % mutation or crossover
    for i=1:NumLayers
        % perform crossover or mutation on the current layer
        CurLayer=UniqueLayers(NumLayers-i+1);
        % Create a mask for the DNFs that will evolve
        CurMask=MateDNFageLayer==CurLayer;
        % Now select the necessary data for mutation or crossover
        CurMateDNF=MateDNF(CurMask,:);
        CurMateDNFfit=MateDNFfit(CurMask);
        CurMateDNFage=MateDNFage(CurMask);
        CurDNFMatchLocs=MateDNFMatchLocs(:,CurMask);
        % If this is the 1st loop then CurNumOff=ArchNumOff
```

```
if i~=1
   CurNumOff=sum(CurMask);
else
   CurNumOff=NumArchOff;
   % Also add 1 to the age of the selected archived offspring
   CurMateDNFage(1:NumArchOff)=CurMateDNFage(1:NumArchOff)+1;
end
clear CurMask
% If the current layer isn't one then add the younger layer
% to mate with
if CurLayer~=1
   % Add a layer to the current layer for mating
   CurMask=MateDNFageLayer==CurLayer-1;
   CurMateDNF=[CurMateDNF; MateDNF(CurMask,:)];
   CurMateDNFfit=[CurMateDNFfit; MateDNFfit(CurMask)];
   CurMateDNFage=[CurMateDNFage; MateDNFage(CurMask)];
   CurDNFMatchLocs=[CurDNFMatchLocs ...
                        MateDNFMatchLocs(:,CurMask)];
   clear CurMask
end
% Now perform crossover and or mutation
[OffDNF, OffDNFMatchLocs, OffDNFfit, OffDNFacc, OffDNFcov,...
```

```
    OffDNFage, ArchiveMask, DNFstats]=...
    DNFMutCross(CurMateDNF,CurMateDNFfit,CurMateDNFage,...
    CurDNFMatchLocs,CCMatchLocs,TargetClass,CurNumOff,...
    ParamDNF,DNFstats);
clear CurLayer
clear CurMateDNF
clear CurMateDNFfit
clear CurMateDNFage
clear CurDNFMatchLocs
clear CurNumOff
% Now save the offspring
% Determine the number of offspring
NumOff=length(ArchiveMask);
EvoDNF(start:start+NumOff-1,:)=OffDNF;
EvoDNFMatchLocs(:,start:start+NumOff-1)=OffDNFMatchLocs;
EvoDNFfit(start:start+NumOff-1)=OffDNFfit;
EvoDNFacc(start:start+NumOff-1)=OffDNFacc;
EvoDNFcov(start:start+NumOff-1)=OffDNFcov;
EvoDNFage(start:start+NumOff-1)=OffDNFage;
EvoArchiveMask(start:start+NumOff-1)=ArchiveMask;
% update the start
start=start+NumOff;
```

```
        clear NumOff
    end
    % Increase only the age of the archive population that underwent either
    % mutation or crossover
    ArchDNFage(ID(SelDNFVec))=ArchDNFage(ID(SelDNFVec))+1;
else
    % then age layers do not have an archive layer
    % Determine the number of age layers
    UniqueLayers=unique(MateDNFageLayer);
    NumLayers=length(UniqueLayers);
    % set a start counter
    start=1;
    % Run a for loop so that each age layer can undergo either
    % mutation or crossover
    for i=1:NumLayers
        % perform crossover or mutation on the current layer
        CurLayer=UniqueLayers(NumLayers-i+1);
        % Create a mask for the DNFs that will evolve
        CurMask=MateDNFageLayer==CurLayer;
        % Now select the necessary data for mutation or crossover
        CurMateDNF=MateDNF(CurMask,:);
        CurMateDNFfit=MateDNFfit(CurMask);
```

```
CurMateDNFage=MateDNFage(CurMask);
CurDNFMatchLocs=MateDNFMatchLocs(:,CurMask);
% Set the number of offspring
CurNumOff=sum(CurMask);
% If the current layer isn't one then add the younger layer
% to mate with
if CurLayer~=1
   % Add a layer to the current layer for mating
   CurMask=MateDNFageLayer==CurLayer-1;
   CurMateDNF=[CurMateDNF; MateDNF(CurMask,:)];
   CurMateDNFfit=[CurMateDNFfit; MateDNFfit(CurMask)];
   CurMateDNFage=[CurMateDNFage; MateDNFage(CurMask)];
   CurDNFMatchLocs=[CurDNFMatchLocs ...
                        MateDNFMatchLocs(:,CurMask)];
   clear CurMask
end
% Now perform crossover and or mutation
[OffDNF, OffDNFMatchLocs, OffDNFfit, OffDNFacc, OffDNFcov,...
   OffDNFage, ArchiveMask, DNFstats]=...
   DNFMutCross(CurMateDNF,CurMateDNFfit,CurMateDNFage,...
   CurDNFMatchLocs,CCMatchLocs,TargetClass,CurNumOff,...
   ParamDNF,DNFstats);
```

```
        clear CurLayer
        clear CurMateDNF
        clear CurMateDNFfit
        clear CurMateDNFage
        clear CurDNFMatchLocs
        clear CurNumOff
        % Now save the offspring
        % Determine the number of offspring
        NumOff=length(ArchiveMask);
        EvoDNF(start:start+NumOff-1,:)=OffDNF;
        EvoDNFMatchLocs(:,start:start+NumOff-1)=OffDNFMatchLocs;
        EvoDNFfit(start:start+NumOff-1)=OffDNFfit;
        EvoDNFacc(start:start+NumOff-1)=OffDNFacc;
        EvoDNFcov(start:start+NumOff-1)=OffDNFcov;
        EvoDNFage(start:start+NumOff-1)=OffDNFage;
        EvoArchiveMask(start:start+NumOff-1)=ArchiveMask;
        % update the start
        start=start+NumOff;
        clear NumOff
    end
end
% Reduce offspring to only those that were actually kept
```

```
EvoDNF=EvoDNF(1:start-1,:);
EvoDNFMatchLocs=EvoDNFMatchLocs(:,1:start-1);
EvoDNFfit=EvoDNFfit(1:start-1);
EvoDNFacc=EvoDNFacc(1:start-1);
EvoDNFcov=EvoDNFcov(1:start-1);
EvoDNFage=EvoDNFage(1:start-1);
EvoArchiveMask=EvoArchiveMask(1:start-1);
clear start
% Now extract the archived population
ArchDNF=[ArchDNF; EvoDNF(EvoArchiveMask,:)];
ArchDNFMatchLocs=[ArchDNFMatchLocs EvoDNFMatchLocs(:,EvoArchiveMask)];
ArchDNFfit=[ArchDNFfit; EvoDNFfit(EvoArchiveMask)];
ArchDNFacc=[ArchDNFacc; EvoDNFacc(EvoArchiveMask)];
ArchDNFcov=[ArchDNFcov; EvoDNFcov(EvoArchiveMask)];
ArchDNFage=[ArchDNFage; EvoDNFage(EvoArchiveMask)];
% Now extract the non-archived population
NonArchDNF=[NonArchDNF; EvoDNF(~EvoArchiveMask,:)];
NonArchDNFMatchLocs=[NonArchDNFMatchLocs ...
                    EvoDNFMatchLocs(:,~EvoArchiveMask)];
NonArchDNFfit=[NonArchDNFfit; EvoDNFfit(~EvoArchiveMask)];
NonArchDNFacc=[NonArchDNFacc; EvoDNFacc(~EvoArchiveMask)];
NonArchDNFcov=[NonArchDNFcov; EvoDNFcov(~EvoArchiveMask)];
```

```
NonArchDNFage=[NonArchDNFage; EvoDNFage(~EvoArchiveMask)];
clear EvoArchiveMask
```

8.1.3.4 DNF Mutation/Crossover (DNFMutCross)

```
function [OffDNF, OffDNFMatchLocs, OffDNFfit, OffDNFacc, OffDNFcov,...
        OffDNFage, ArchiveMask, DNFstats]=...
        DNFMutCross(MateDNF,MateDNFfit,MateDNFage,...
        MateDNFMatchLocs,CCMatchLocs,TargetClass,NumOff,ParamDNF,...
        DNFstats)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
% October 21, 2016
% Last updated: October 24, 2016
%
% DNFMutCross performs either mutation or crossover on the disjunction
% of conjunctive clauses (DNF) of the classifiers.
%
% Inputs:
% MateDNF = The population of DNFs that will be involved in mating. Each
%       row represents a DNF and each column points to a conjunctive
%       clause used in the DNF. Only the 1:NumOff will undergo either
%       crossover or mutation.
```

```
% MateDNFfit = The fitness of the MateDNFs.
% MateDNFage = The age of the MateDNFs.
% MateDNFMatchLocs = A logical matrix where each from represents an
%                    observation and each column represents a DNF.
% CCMatchLocs = A logical matrix where each row represents an observation
%               and each column represents a conjunctive clause.
% TargetClass = A logical vector indicating the observations that are part
%               of the target class.
% NumOff = The number of offspring that will be produced through either
%          crossover or mutation.
% ParamDNF = A structure array with the usr defined parameters to run the
%            algorithm.
% DNFstats = Statistics on the DNF evolution.
%
% Ouputs:
% OffDNF = The DNF of the offsrping that are classifiers for the current
%          target class.
% OffDNFMatchLocs = A logical matrix associated with the OffDNF. Each row
%                   represents an observation and each column represents a
%                   DNF.
% OffDNFfit = The fitness of the OffDNF using the hypergeometric PMF
%             distribution.
```

```
% OffDNFacc = The accuracy of the OffDNF.
% OffDNFcov = The coverage of the OffDNF.
% OffDNFage = The age of the OffDNF.
% ArchiveMask = A logical vector indicating the OffDNF that belong in the
%               archive.
% DNFstats = Statistics on the DNF evolution.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine the number of DNFs in the mating population
NumDNFs=size(MateDNF,1);
% First randomly determine which inidividuals will undergo mutation and
% which will undergo crossover
if NumDNFs~=1
   % Then randomly select crossover or mutation
   CrossOver=rand(NumOff,1)<ParamDNF.Prx;
else
   CrossOver=false(1);
end
% For efficiency
OffDNF=false(NumOff,ParamDNF.NumCCs);
OffDNFMatchLocs=false(ParamDNF.NumObs,NumOff);
OffDNFfit=NaN(NumOff,1);
```

```
OffDNFacc=NaN(NumOff,1);
OffDNFcov=NaN(NumOff,1);
OffDNFage=NaN(NumOff,1);
KeepMask=false(NumOff,1);
ArchiveMask=false(NumOff,1);
for i=1:NumOff
   if CrossOver(i)
      % Then perform crossover
      % Set up the random mating population
      PotMatePop=setdiff(1:NumDNFs,i);
      % Now randomly select the potential mates
      PotMateInd=randi(NumDNFs-1,[ParamDNF.TournSize,1]);
      PotMates=PotMatePop(PotMateInd);
      clear PotMatePop
      clear PotMateInd
      % Determine whether the crossover should be based on fitness or
      % based on other metrics
      if rand(1)<=ParamDNF.Pxf
         % then the mate is the most fit in the tournament
         BestMask=MateDNFfit(PotMates)==min(MateDNFfit(PotMates));
      else
         % Run another metric to determine the mate
```

```
% There are three ways to determine best mate for crossover:
% 1) The DNF that covers the most target observations not
% covered by the current DNF; 2) the DNF that covers the most
% target observations that are not covered by current DNF
% while minimizing the number of new non-target observations
% covered; 3) the DNF with the least number of non-target
% observations that are not covered by the current DNF
% First randomly select a number to determine how the mate will
% be selected
RandNum=rand(1);
if RandNum<=ParamDNF.PxAlt(1)
   % Then mate based on the most target observations not
   % covered by the current DNF
   % Create a mask of the current target observations that are
   % not covered by current mate
   Tmask=TargetClass&~MateDNFMatchLocs(:,i);
   % Now find the sum of the Tmask for the potential Mates
   MateSum=sum(MateDNFMatchLocs(Tmask,PotMates));
   clear Tmask
   % The BestMask is the max of the MateSum
   BestMask=MateSum==max(MateSum);
   clear MateSum
```

```
elseif RandNum<=ParamDNF.PxAlt(2)
    % Then the mate selection is based on the most target
    % observation not covered by the current DNF and least
    % non-target observations not covered by the current DNF
    % not covered by current mate
    Tmask=TargetClass&~MateDNFMatchLocs(:,i);
    % Now find the sum of the Tmask for the potential Mates
    MateTSum=sum(MateDNFMatchLocs(Tmask,PotMates));
    clear Tmask
    % Now create a mask of the non-target observations that are
    % not covered by the current mate
    Tmask=~TargetClass&~MateDNFMatchLocs(:,i);
    % Now sum the Tmask for potential mates
    MateNTSum=sum(MateDNFMatchLocs(Tmask,PotMates));
    clear Tmask
    % Now subtract MatNTSum from MateTSum
    MateSum=MateTSum-MateNTSum;
    clear MateTSum
    clear MateNTSum
    % The BestMask is the max of the MateSum
    BestMask=MateSum==max(MateSum);
    clear MateSum
```

```
    else
        % Then the mate selection is based on the mate that has the
        % least number of non-target observations not covered by
        % the first parent
        % Now create a mask of the non-target observations that are
        % not covered by the current mate
        Tmask=~TargetClass&~MateDNFMatchLocs(:,i);
        % Now sum the Tmask for potential mates
        MateSum=sum(MateDNFMatchLocs(Tmask,PotMates));
        clear Tmask
        % The BestMask is the min of the MateSum
        BestMask=MateSum==min(MateSum);
        clear MateSum
    end
end
% If there is more than one mate selected then randomly pick mate
if sum(BestMask)==1
    % Then rthe the mate ID is easy
    MateID=PotMates(BestMask);
else
    % Randomly choose a Mate
    PotMates=PotMates(BestMask);
```

```
        RandPick=randi(sum(BestMask),1);
        MateID=PotMates(RandPick);
        clear RandPick
    end
    clear PotMates
    % Set the age of the OffDNF to the age of the oldest parent
    OffDNFage(i)=max([MateDNFage(i) MateDNFage(MateID)]);
    % Randomly determine the bits from the first parent
    P1bits=rand(1,ParamDNF.NumCCs)<0.5;
    % Now insert the bits into the current offspring
    OffDNF(i,P1bits)=MateDNF(i,P1bits);
    OffDNF(i,~P1bits)=MateDNF(MateID,~P1bits);
    % Test to see if the Off spring DNF is the same as either parent,
    % or if there are no DNFs selected; if so swap the bits
    if isequal(OffDNF(i,:),MateDNF(i,:))||...
            isequal(OffDNF(i,:),MateDNF(MateID,:))||...
            sum(OffDNF(i,:))==0
        % then switch the bits around
        OffDNF(i,~P1bits)=MateDNF(i,~P1bits);
        OffDNF(i,P1bits)=MateDNF(MateID,P1bits);
    end
else
```

```
% Perform mutation
% Set the offspring DNF equal to the current mate
OffDNF(i,:)=MateDNF(i,:);
OffDNFage(i)=MateDNFage(i);
% Determine if random bit flip mutation will occur or if a more
% targeted mutation will occur
if rand(1)<ParamDNF.Pbf
   % Then perform standard bit flip mutation
   % Randomly determine which bits will be flipped
   FlipBitLocs=rand(1,ParamDNF.NumCCs)<ParamDNF.Pm;
   % Ensure that at least one bit is flipped
   if sum(FlipBitLocs)==0
      FlipBitLocs(randi(1))=true();
   end
   % Determine where the bits will be turned off
   BitOff=FlipBitLocs&MateDNF(i,:);
   % set all of the Flip Bit Locations to true
   OffDNF(i,FlipBitLocs)=true();
   % Now set all of the Bit off locations to false
   OffDNF(i,BitOff)=false();
   % if there are no bits on then randomly choose bits to keep
   if sum(OffDNF(i,:))==0
```

```
% Reset the offspring to the mate
OffDNF(i,:)=MateDNF(i,:);
% Determine the number of bits
if sum(MateDNF(i,:))<=2
    % Then add a bit
    PosBits=...
          setdiff(1:ParamDNF.NumCCs,find(MateDNF(i,:)));
    % Randomly choose a bit to add
    PosBitID=randi(length(PosBits),1);
    % Set the bit to true
    OffDNF(i,PosBits(PosBitID))=true();
    clear PosBits
    clear PosBitID
else
    % Randomly delet a bit
    PosBits=find(MateDNF(i,:));
    % Randomly choose bit to delete
    PosBitID=randi(length(PosBits),1);
    % Now delete the selected bit
    OffDNF(i,PosBits(PosBitID))=false();
    clear PosBits
    clear PosBitID
```

```
        end
    end
else
    % Perform a more targeted mutation there are 4 types of
    % targeted mutation: 1) select the CC that covers the
    % most target observations not covered by the DNF; 2) select
    % the CC that covers the most target observation while
    % minimizing the number of new non target observations covered;
    % 3) remove the CC that covers the least unique target
    % observations; 4) remove the CC that has most non-target
    % observations that are only covered by the CC
    % First determine if the mate has 100% accuracy or 100%
    % coverage, because if either of these is true then this will
    % affect which types of targeted mutation are possible
    CurAcc=sum(MateDNFMatchLocs(TargetClass,i))/...
                                sum(MateDNFMatchLocs(:,i));
    CurCov=sum(MateDNFMatchLocs(TargetClass,i))/sum(TargetClass);
    % Extract the current probabilities
    CurProbs=ParamDNF.PmAlt;
    if CurAcc==1&&CurCov==1
        % Then need to replace the first two probabilities with 0
        CurProbs(1:2)=0;
```

```
    % Now renormalize
    CurProbs=CurProbs/sum(CurProbs);
elseif CurCov==1
    % Then need to repalce the first probability with 0
    CurProbs(1)=0;
    % Now renormalize
    CurProbs=CurProbs/sum(CurProbs);
    if sum(OffDNF(i,:))==1
        % Remove the last two probabilities
        CurProbs(3:4)=0;
    end
    % Now renormalize
    CurProbs=CurProbs/sum(CurProbs);
elseif sum(OffDNF(i,:))==1
    % Remove the last two probabilites
    CurProbs(3:4)=0;
    % Now renormalize
    CurProbs=CurProbs/sum(CurProbs);
end
% Now perform a cumsum on the probabilities
ProbCumSum=cumsum(CurProbs);
clear CurProbs
```

```
% given the new probabilities determine the type of target
% mutation
% Randomly select a number
RandNum=rand(1);
if RandNum<=ProbCumSum(1)
   % Then select the CC that covers the most target
   % observations not covered by the DNF
   Tmask=TargetClass&~MateDNFMatchLocs(:,i);
   % sum the CCMatchLocs of TMask
   MutSum=sum(CCMatchLocs(Tmask,:));
   clear Tmask
   % Create mask of the maximum MutSum
   MaxMask=MutSum==max(MutSum);
   clear MutSum
   % If the MaxMask has more than one CC then randomly choose
   % one
   if sum(MaxMask)>1
      % Then randomly choose a max
      MaxOpts=find(MaxMask);
      % Now randomly draw a potential max
      MaxID=randi(length(MaxOpts),1);
      % Now set MaxMask to false
```

```
        MaxMask=false(size(MaxMask));
        % Insert the select location as true
        MaxMask(MaxOpts(MaxID))=true();
        clear MaxOpts
        clear MaxID
    end
    % Now insert the selected CC into the DNF
    OffDNF(i,MaxMask)=true();
    clear MaxMask
elseif RandNum<=ProbCumSum(2)
    % Select the CC that covers the most target observations
    % and least non-target observations
    % target observations not covered by the DNF
    Tmask=TargetClass&~MateDNFMatchLocs(:,i);
    % sum the CCMatchLocs of TMask
    MutSumT=sum(CCMatchLocs(Tmask,:));
    clear Tmask
    % non-target observations not covered by the DNF
    Tmask=~TargetClass&~MateDNFMatchLocs(:,i);
    % sum the CCMatchLocs of the Tmask
    MutSumNT=sum(CCMatchLocs(Tmask,:));
    clear Tmask
```

```
% Subtract the MutSumNT from MutSumT
MutSum=MutSumT-MutSumNT;
clear MutSumT
clear MutSumNT
% Create mask of the maximum MutSum
MaxMask=MutSum==max(MutSum);
clear MutSum
% If the MaxMask has more than one CC then randomly choose
% one
if sum(MaxMask)>1
   % Then randomly choose a max
   MaxOpts=find(MaxMask);
   % Now randomly draw a potential max
   MaxID=randi(length(MaxOpts),1);
   % Now set MaxMask to false
   MaxMask=false(size(MaxMask));
   % Insert the select location as true
   MaxMask(MaxOpts(MaxID))=true();
   clear MaxOpts
   clear MaxID
end
% Now insert the selected CC into the DNF
```

```
    OffDNF(i,MaxMask)=true();
    clear MaxMask
elseif RandNum<=ProbCumSum(3)
    % remove the CC that covers the least unique target
    % observations
    % First determine the total times each target observation
    % is covered
    ObsSums=sum(CCMatchLocs(:,OffDNF(i,:)),2);
    % Now determine the observations that are covered by only
    % one observation
    OneTargetMask=TargetClass&ObsSums==1;
    clear ObsSums
    % Now determine the total times a CC is the only CC to
    % cover a target observation
    NumCCUniq=sum(CCMatchLocs(OneTargetMask,OffDNF(i,:)));
    clear OneTargetMask
    % Now determine the minimum CC to remove
    MinMask=NumCCUniq==min(NumCCUniq);
    clear NumCCUniq
    % Determine if there is more than one minimum
    if sum(MinMask)>1
        % Then need to randomly choose a CC
```

```
        MinOpts=find(MinMask);
        % Now randomly draw a min
        MinID=randi(length(MinOpts),1);
        % Now set Min Mask to false
        MinMask=false(size(MinMask));
        % Set the selected min mask to true
        MinMask(MinOpts(MinID))=true();
        clear MinOpts
        clear MinID
    end
    % Now find the OffDNF ccs
    OffCCs=find(OffDNF(i,:));
    % Now set the current selected CC to mask
    OffDNF(i,OffCCs(MinMask))=false();
    clear OffCCs
    clear MinMask
else
    % Then remove the CC with the most unique non-target
    % obesrvations
    ObsSums=sum(CCMatchLocs(:,OffDNF(i,:)),2);
    % Now determine the observations that are covered by only
    % one observation
```

```
OneTargetMask=~TargetClass&ObsSums==1;
clear ObsSums
% Now determine the total times a CC is the only CC to
% cover a target observation
NumCCUniq=sum(CCMatchLocs(OneTargetMask,OffDNF(i,:)));
clear OneTargetMask
% Now determine the maximum CC to remove
MaxMask=NumCCUniq==max(NumCCUniq);
clear NumCCUniq
% Determine if there is more than one minimum
if sum(MaxMask)>1
   % Then need to randomly choose a CC
   MaxOpts=find(MaxMask);
   % Now randomly draw a min
   MaxID=randi(length(MaxOpts),1);
   % Now set Max Mask to false
   MaxMask=false(size(MaxMask));
   % Set the selected min mask to true
   MaxMask(MaxOpts(MaxID))=true();
   clear MaxOpts
   clear MaxID
end
```

```
            % Now find the OffDNF ccs
            OffCCs=find(OffDNF(i,:));
            % Now set the current selected CC to mask
            OffDNF(i,OffCCs(MaxMask))=false();
            clear OffCCs
            clear MaxMask
        end
    end
end
% Now determine the fitness of the new DNF
% First extract teh current CC match locations
CurCCs=CCMatchLocs(:,OffDNF(i,:));
% Now determine the DNFMatchLocs
CurCCsSum=sum(CurCCs,2);
clear CurCCs
% DNFMatchLocs is simply a mask of CurCCsSum>0
OffDNFMatchLocs(:,i)=CurCCsSum>0;
clear CurCCsSum
% Now determine the fitness
ntot=sum(OffDNFMatchLocs(:,i));
xmatch=sum(OffDNFMatchLocs(TargetClass,i));
% Determine the number of CCs
```

```
NumCCs=sum(OffDNF(i,:));
% Ensure the order will be recorded
if NumCCs>ParamDNF.Thresh(end,1)
   NumCCs=ParamDNF.Thresh(end,1);
end
% Determine if the DNF is a target class DNF
if xmatch/ntot>ParamDNF.TotTarC/ParamDNF.NumObs
   % calculate the fitness function
   OffDNFfit(i)=...
            hygepdf(xmatch,ParamDNF.NumObs,ParamDNF.TotTarC,ntot);
   % Calculate the accuracy and coverage
   OffDNFacc(i)=xmatch/ntot*100;
   OffDNFcov(i)=xmatch/ParamDNF.TotTarC*100;
   % Set the KeepMask to true
   KeepMask(i)=true();
   % Determine if the new DNF should be archived
   Tmask=ParamDNF.Thresh(:,1)==sum(OffDNF(i,:));
   if OffDNFfit(i)<=ParamDNF.Thresh(Tmask,2)
      % Then it is archiveable
      ArchiveMask(i)=true();
      DNFstats.EvalsArchive(ParamDNF.CurGen,NumCCs)=...
               DNFstats.EvalsArchive(ParamDNF.CurGen,NumCCs)+1;
```

```
        else
            % then it is non-archiveable
            DNFstats.EvalsNonArchive(ParamDNF.CurGen,NumCCs)=...
                    DNFstats.EvalsNonArchive(ParamDNF.CurGen,NumCCs)+1;
        end
        clear Tmask
    end
    % Record the an evaluation for the current order
    DNFstats.EvalsAll(ParamDNF.CurGen,NumCCs)=...
                        DNFstats.EvalsAll(ParamDNF.CurGen,NumCCs)+1;
    clear NumCCs
    clear ntot
    clear xmatch
end
% Only keep the values in the KeepMask
OffDNF=OffDNF(KeepMask,:);
OffDNFMatchLocs=OffDNFMatchLocs(:,KeepMask);
OffDNFfit=OffDNFfit(KeepMask);
OffDNFacc=OffDNFacc(KeepMask);
OffDNFcov=OffDNFcov(KeepMask);
OffDNFage=OffDNFage(KeepMask);
ArchiveMask=ArchiveMask(KeepMask);
```

```
clear KeepMask
```

8.1.4 Smouse and Peakall (1999) Genetic Distance (GeneticDistance)

```
function [RawDist, NormDist, TotNonNaNs, TotSame]=...
                              GeneticDistance(SNPsMat,Combos,GenDist)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
% January 31, 2017
% Last Updated: January 31, 2017
%
% Calculates the genetic distance using the Smouse and Peakall (1999)
% genetic distance equation or a hamming distance. The user needs to enter
% the genetic distance for every combination of SNPs.
%
% Inputs:
% SNPsMat = A matrix where every row is an observation (e.g., person,
%           animal, plant) and each column is a SNP. The values in the
%           matrix should be enetered as numbers (ideally enter the ascii
%           number of the letter using the double() function; also set all
%           Ns to NaN.
% Combos = A two column matrix where each row is the numerical equivalent
%          of one of the possible combinations of SNPs (e.g., row 1 could
```

```
%         be AC = [65 67]; row 45 could be WY = [87 89]).
% GenDist = A vector of the genetic distance that corresponds to the
%          Combos.
%
% Outputs:
% RawDist = A vector of the raw genetic distance between every observation.
% NormDist =  The normalized genetic distance based on the number of SNPs
%            looked at (i.e., RawDist/TotNonNaNs).
% TotNonNaNs = The total number of SNPs where both of the observations had
%             a non-NaN value.
% TotSame = A count of the total number of SNPs that are the same between
%          two observations.
%
% Work Cited:
% Smouse, Peter E, and Rod Peakall. 1999. “Spatial Autocorrelation Analysis
%   of Individual Multiallele and Multilocus Genetic Structure.” Heredity
%   82 (5): 561–73. doi:10.1038/sj.hdy.6885180.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine the number of observations and SNPs
[NumObs, ~]=size(SNPsMat);
% Determine the total number of Pairs
```

```
NumPairs=(NumObs*(NumObs-1))/2;
TotNonNaNs=NaN(NumPairs,1);
TotSame=NaN(NumPairs,1);
RawDist=NaN(NumPairs,1);
NormDist=NaN(NumPairs,1);
count=0;
for i=1:NumObs-1
   % Grab the current observation SNPs
   Obs1=SNPsMat(i,:);
   % Determine the NaN locations
   Obs1NaNs=isnan(Obs1);
   for j=i+1:NumObs
      count=count+1;
      % Grab the current observation SNPs
      Obs2=SNPsMat(j,:);
      % Determine the NaN locations
      Obs2NaNs=isnan(Obs2);
            % Determine the SNPs that do not have NaN values
      NonNaNs=~Obs1NaNs&~Obs2NaNs;
      % Record the number of NonNans
      TotNonNaNs(count)=sum(NonNaNs);
      % Now Grab only the columns with NonNans
```

```
CurObs1=Obs1(NonNaNs);
CurObs2=Obs2(NonNaNs);
% Now Create a mask for all of the columns that are the same
SameMask=CurObs1==CurObs2;
% Determine the number of obs that are the same
TotSame(count)=sum(SameMask);
% Now grab only the columns that are not the same
CurObs1=CurObs1(~SameMask);
CurObs2=CurObs2(~SameMask);
% Now Reorganize the SNPs into columns with the minimum in the
% first column
MinVals=min([CurObs1; CurObs2]);
MaxVals=max([CurObs1; CurObs2]);
% Combine the Mins and Maxs
CombObs=[MinVals' MaxVals'];
[UniRows, ~, IDs]=unique(CombObs,'rows');
% Tabulate the IDs
TabIDs=tabulate(IDs);
TotDist=0;
% For each of the unique rows determine the distance
for r=1:size(UniRows,1)
    % Grab the Current unique row
```

```
            CurUniRow=UniRows(r,:);
            % Repeat the Current unique row
            RepCurUniRow=repmat(CurUniRow,[size(Combos,1),1]);
            % now compare to combos
            CmpComb=RepCurUniRow==Combos;
            % Create a mask to get the current genetic distance
            DistMask=sum(CmpComb,2)==2;
            % Grab the current genetic distance
            CurDist=GenDist(DistMask);
            % Now multiply the current distance by the number of times it
            % is present
            CurTotDist=CurDist*TabIDs(r,2);
            % Add the current total distance to the total Distance
            TotDist=TotDist+CurTotDist;
        end
        % Insert the raw genetic distance
        RawDist(count)=TotDist;
        % Calculate the normalized distance
        NormDist(count)=TotDist/TotNonNaNs(count);
    end
end
```

8.1.5 Box Plots (boxplotJH)

```
function boxplotJH(X,Y,Param)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by John Hanley
%
% October 26, 2016
% Last Updated: November 9, 2016
%
% boxplotJH is my version of boxplot that enables the user to change the
% x-axis setting.
%
% Inputs:
% X = The X-data entered either as a vector or matrix. If entered as a
%    matrix, then set Param.DoubleBox to true.
% Y = The Y-data entered as a matrix with each column representing a
%    different group.
% Param = A structure array with various parameters needed to run the
%        function.
%        .BoxColor = The color of the IQR for the boxplot.
%        .MedColor = The color of the median for the boxplot.
%        .WhiskColor = The color of the whiskers for the boxplot.
%        .OutColor = The color of the outliers for the boxplot.
```

```
%       .LineWidth = The line width for the boxplot.
%       .Axis = The type of plot the user wants. Enter either
%             'cartesian', 'semilogx', 'semilogy', or 'loglog'
%       .BoxWidth = The desired width of the box in the boxplot.
%       .WhWidth = The desired width of the whisker cap.
%       .DoubleBox = Enter either true() or false(). True means that the
%                 boxplot will be plotted in both x and y directions.
%
% Outputs:
% A boxplot of the data.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine the number of boxes
Param.NumBox=size(Y,2);
% First determine the 25th percentile, median and IQR
IQRY=iqr(Y);
P25Y=prctile(Y,25);
P50Y=prctile(Y,50);
P75Y=prctile(Y,75);
% For efficiency
WhLBY=NaN(1,Param.NumBox);
WhUBY=NaN(1,Param.NumBox);
```

```
% Now determine the whisker bounds
for i=1:Param.NumBox
    % For each column determine the possible and actual lower bound
    % First calculate a possible lower bound
    PosLB=P25Y(i)-1.5*IQRY(i);
    % Now create a mask for all values that fall in this range
    LBmask=Y(:,i)>=PosLB&Y(:,i)<P25Y(i);
    clear PosLB
    % If a value falls in this range set the WhLB to the minimum value in
    % the range
    if sum(LBmask)>0
        WhLBY(i)=min(Y(LBmask,i));
    end
    clear LBmask
    % Now calculate a possible upper bound
    PosUB=P75Y(i)+1.5*IQRY(i);
    % Now create a mask for all values that fall in this range
    UBmask=Y(:,i)<=PosUB&Y(:,i)>P75Y(i);
    clear PosUB
    % If a value falls in this range set the WhUB to the maximum value in
    % the range
    if sum(UBmask)>0
```

```
        WhUBY(i)=max(Y(UBmask,i));
    end
    clear UBmask
end
% check to see if there is a doublebox
if Param.DoubleBox
    % Then need to calculate the X 25th percentile, median and IQR
    IQRX=iqr(X);
    P25X=prctile(X,25);
    P50X=prctile(X,50);
    P75X=prctile(X,75);
    % For efficiency
    WhLBX=NaN(1,Param.NumBox);
    WhUBX=NaN(1,Param.NumBox);
    % Now determine the whisker bounds
    for i=1:Param.NumBox
        % For each column determine the possible and actual lower bound
        % First calculate a possible lower bound
        PosLB=P25X(i)-1.5*IQRX(i);
        % Now create a mask for all values that fall in this range
        LBmask=X(:,i)>=PosLB&X(:,i)<P25X(i);
        clear PosLB
```

```
    % If a value falls in this range set the WhLB to the minimum value
    % in the range
    if sum(LBmask)>0
        WhLBX(i)=min(X(LBmask,i));
    end
    clear LBmask
    % Now calculate a possible upper bound
    PosUB=P75X(i)+1.5*IQRX(i);
    % Now create a mask for all values that fall in this range
    UBmask=X(:,i)<=PosUB&X(:,i)>P75X(i);
    clear PosUB
    % If a value falls in this range set the WhUB to the maximum value
    % in the range
    if sum(UBmask)>0
        WhUBX(i)=max(X(UBmask,i));
    end
    clear UBmask
end
switch Param.Axis
    case 'cartesian'
        % The plot is cartesian
        % First plot the whiskers
```

```
plot(repmat(P50X,[2,1]),[P25Y; WhLBY],'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
hold on
plot(repmat(P50X,[2,1]),[P75Y; WhUBY],'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
plot([P25X; WhLBX],repmat(P50Y,[2,1]),'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
plot([P75X; WhUBX],repmat(P50Y,[2,1]),'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
% Set the whisker lower bounds and upper bounds in x direction
LBX=P50X-Param.WhWidth;
UBX=P50X+Param.WhWidth;
LBY=P50Y-Param.WhWidth;
UBY=P50Y+Param.WhWidth;
% Now plot the cap of the whisker
plot([LBX; UBX],repmat(WhLBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
plot([LBX; UBX],repmat(WhUBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
plot(repmat(WhLBX,[2,1]),[LBY; UBY],'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
plot(repmat(WhUBX,[2,1]),[LBY; UBY],'-',...
```

```
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
clear LBX
clear UBX
clear LBY
clear UBY
% Now plot the box
for i=1:Param.NumBox
   % plot the IQR using rectangle
   rectangle('Position',[P25X(i) P25Y(i) IQRX(i) IQRY(i)],...
            'EdgeColor',Param.BoxColor,'LineWidth',...
            Param.LineWidth)
end
% Now plot the median
plot([P25X; P75X],repmat(P50Y,[2,1]),'-',...
          'Color',Param.MedColor,'LineWidth',Param.LineWidth)
plot(repmat(P50X,[2,1]),[P25Y; P75Y],'-',...
          'Color',Param.MedColor,'LineWidth',Param.LineWidth)
% Now plot the outliers
for i=1:Param.NumBox
   % Find Any Outliers
   OutMaskY=Y(:,i)<WhLBY(i)|Y(:,i)>WhUBY(i);
   OutMaskX=X(:,i)<WhLBX(i)|X(:,i)>WhUBX(i);
```

```
        % If there are outliers plot them
        if sum(OutMaskY)>0
            plot(repmat(P50X(i),[sum(OutMaskY),1]),...
                Y(OutMaskY,i),'+','Color',Param.OutColor,...
                'LineWidth',Param.LineWidth)
        end
        if sum(OutMaskX)>0
            plot(X(OutMaskX,i),repmat(P50Y(i),...
                 [sum(OutMaskX),1]),'+','Color',...
                 Param.OutColor,'LineWidth',Param.LineWidth)
        end
    end
case 'semilogx'
    % The plot is semilogx
    % First plot the whiskers
    semilogx(repmat(P50X,[2,1]),[P25Y; WhLBY],'--',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    hold on
    semilogx(repmat(P50X,[2,1]),[P75Y; WhUBY],'--',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    semilogx([P25X; WhLBX],repmat(P50Y,[2,1]),'--',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
```

```
semilogx([P75X; WhUBX],repmat(P50Y,[2,1]),'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
% Set the whisker lower bounds and upper bounds in x direction
LBX=10.^(log10(P50X)-Param.WhWidth);
UBX=10.^(log10(P50X)+Param.WhWidth);
LBY=10.^(log10(P50Y)-Param.WhWidth);
UBY=10.^(log10(P50Y)+Param.WhWidth);
% Now plot the cap of the whisker
semilogx([LBX; UBX],repmat(WhLBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
semilogx([LBX; UBX],repmat(WhUBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
semilogx(repmat(WhLBX,[2,1]),[LBY; UBY],'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
semilogx(repmat(WhUBX,[2,1]),[LBY; UBY],'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
clear LBX
clear UBX
clear LBY
clear UBY
% Now plot the box
for i=1:Param.NumBox
```

```
    % plot the IQR using rectangle
    rectangle('Position',[P25X(i) P25Y(i) IQRX(i) IQRY(i)],...
              'EdgeColor',Param.BoxColor,'LineWidth',...
              Param.LineWidth)
end
% Now plot the median
semilogx([P25X; P75X],repmat(P50Y,[2,1]),'-',...
          'Color',Param.MedColor,'LineWidth',Param.LineWidth)
semilogx(repmat(P50X,[2,1]),[P25Y; P75Y],'-',...
          'Color',Param.MedColor,'LineWidth',Param.LineWidth)
% Now plot the outliers
for i=1:Param.NumBox
    % Find Any Outliers
    OutMaskY=Y(:,i)<WhLBY(i)|Y(:,i)>WhUBY(i);
    OutMaskX=X(:,i)<WhLBX(i)|X(:,i)>WhUBX(i);
    % If there are outliers plot them
    if sum(OutMaskY)>0
        semilogx(repmat(P50X(i),[sum(OutMaskY),1]),...
                 Y(OutMaskY,i),'+','Color',Param.OutColor,...
                 'LineWidth',Param.LineWidth)
    end
    if sum(OutMaskX)>0
```

```
            semilogx(X(OutMaskX,i),repmat(P50Y(i),...
                    [sum(OutMaskX),1]),'+','Color',...
                    Param.OutColor,'LineWidth',Param.LineWidth)
        end
    end
case 'semilogy'
    % The plot is semilogy
    % First plot the whiskers
    semilogy(repmat(P50X,[2,1]),[P25Y; WhLBY],'--',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    hold on
    semilogy(repmat(P50X,[2,1]),[P75Y; WhUBY],'--',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    semilogy([P25X; WhLBX],repmat(P50Y,[2,1]),'--',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    semilogy([P75X; WhUBX],repmat(P50Y,[2,1]),'--',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    % Set the whisker lower bounds and upper bounds in x direction
    LBX=P50X-Param.WhWidth;
    UBX=P50X+Param.WhWidth;
    LBY=P50Y-Param.WhWidth;
    UBY=P50Y+Param.WhWidth;
```

```
% Now plot the cap of the whisker
semilogy([LBX; UBX],repmat(WhLBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
semilogy([LBX; UBX],repmat(WhUBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
semilogy(repmat(WhLBX,[2,1]),[LBY; UBY],'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
semilogy(repmat(WhUBX,[2,1]),[LBY; UBY],'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
clear LBX
clear UBX
clear LBY
clear UBY
% Now plot the box
for i=1:Param.NumBox
   % plot the IQR using rectangle
   rectangle('Position',[P25X(i) P25Y(i) IQRX(i) IQRY(i)],...
            'EdgeColor',Param.BoxColor,'LineWidth',...
            Param.LineWidth)
end
% Now plot the median
semilogy([P25X; P75X],repmat(P50Y,[2,1]),'-',...
```

```
            'Color',Param.MedColor,'LineWidth',Param.LineWidth)
    semilogy(repmat(P50X,[2,1]),[P25Y; P75Y],'-',...
            'Color',Param.MedColor,'LineWidth',Param.LineWidth)
    % Now plot the outliers
    for i=1:Param.NumBox
        % Find Any Outliers
        OutMaskY=Y(:,i)<WhLBY(i)|Y(:,i)>WhUBY(i);
        OutMaskX=X(:,i)<WhLBX(i)|X(:,i)>WhUBX(i);
        % If there are outliers plot them
        if sum(OutMaskY)>0
            semilogy(repmat(P50X(i),[sum(OutMaskY),1]),...
                Y(OutMaskY,i),'+','Color',Param.OutColor,...
                'LineWidth',Param.LineWidth)
        end
        if sum(OutMaskX)>0
            semilogy(X(OutMaskX,i),repmat(P50Y(i),...
                [sum(OutMaskX),1]),'+','Color',...
                Param.OutColor,'LineWidth',Param.LineWidth)
        end
    end
case 'loglog'
    % The plot is loglog
```

```
% First plot the whiskers
loglog(repmat(P50X,[2,1]),[P25Y; WhLBY],'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
hold on
loglog(repmat(P50X,[2,1]),[P75Y; WhUBY],'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
loglog([P25X; WhLBX],repmat(P50Y,[2,1]),'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
loglog([P75X; WhUBX],repmat(P50Y,[2,1]),'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
% Set the whisker lower bounds and upper bounds in x direction
LBX=10.^(log10(P50X)-Param.WhWidth);
UBX=10.^(log10(P50X)+Param.WhWidth);
LBY=10.^(log10(P50Y)-Param.WhWidth);
UBY=10.^(log10(P50Y)+Param.WhWidth);
% Now plot the cap of the whisker
loglog([LBX; UBX],repmat(WhLBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
loglog([LBX; UBX],repmat(WhUBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
loglog(repmat(WhLBX,[2,1]),[LBY; UBY],'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
```

```
loglog(repmat(WhUBX,[2,1]),[LBY; UBY],'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
clear LBX
clear UBX
clear LBY
clear UBY
% Now plot the box
for i=1:Param.NumBox
   % plot the IQR using rectangle
   rectangle('Position',[P25X(i) P25Y(i) IQRX(i) IQRY(i)],...
            'EdgeColor',Param.BoxColor,'LineWidth',...
            Param.LineWidth)
end
% Now plot the median
loglog([P25X; P75X],repmat(P50Y,[2,1]),'-',...
          'Color',Param.MedColor,'LineWidth',Param.LineWidth)
loglog(repmat(P50X,[2,1]),[P25Y; P75Y],'-',...
          'Color',Param.MedColor,'LineWidth',Param.LineWidth)
% Now plot the outliers
for i=1:Param.NumBox
   % Find Any Outliers
   OutMaskY=Y(:,i)<WhLBY(i)|Y(:,i)>WhUBY(i);
```

```
            OutMaskX=X(:,i)<WhLBX(i)|X(:,i)>WhUBX(i);
            % If there are outliers plot them
            if sum(OutMaskY)>0
               loglog(repmat(P50X(i),[sum(OutMaskY),1]),...
                     Y(OutMaskY,i),'+','Color',Param.OutColor,...
                     'LineWidth',Param.LineWidth)
            end
            if sum(OutMaskX)>0
               loglog(X(OutMaskX,i),repmat(P50Y(i),...
                     [sum(OutMaskX),1]),'+','Color',...
                     Param.OutColor,'LineWidth',Param.LineWidth)
            end
         end
   end
else
   % Take the median of X
   X=prctile(X,50,1);
   switch Param.Axis
      case 'cartesian'
         % Cartesian plot
         % First plot the whiskers
         plot(repmat(X,[2,1]),[P25Y; WhLBY],'--',...
```

```
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
hold on
plot(repmat(X,[2,1]),[P75Y; WhUBY],'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
% Set the whisker lower bounds and upper bounds in x direction
LBX=X-Param.WhWidth;
UBX=X+Param.WhWidth;
% Now plot the cap of the whisker
plot([LBX; UBX],repmat(WhLBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
plot([LBX; UBX],repmat(WhUBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
clear LBX
clear UBX
% Now plot the box
% First calculate the box lower and upper bounds in the x
% direction
LBX=X-Param.BoxWidth;
UBX=X+Param.BoxWidth;
for i=1:Param.NumBox
   % plot the IQR using rectangle
   rectangle('Position',...
```

```
            [LBX(i) P25Y(i) UBX(i)-LBX(i) IQRY(i)],...
            'EdgeColor',Param.BoxColor,'LineWidth',...
            Param.LineWidth)
    end
    % Now plot the median
    plot([LBX; UBX],repmat(P50Y,[2,1]),'-',...
          'Color',Param.MedColor,'LineWidth',Param.LineWidth)
    clear LBX
    clear UBX
    % Now plot the outliers
    for i=1:Param.NumBox
      % Find Any Outliers
      OutMask=Y(:,i)<WhLBY(i)|Y(:,i)>WhUBY(i);
      % If there are outliers plot them
      if sum(OutMask)>0
        semilogy(repmat(X(i),[sum(OutMask),1]),Y(OutMask,i),...
              '+','Color',Param.OutColor,'LineWidth',...
              Param.LineWidth)
      end
    end
  case 'semilogx'
    % The plot is semilogx
```

```
% First plot the whiskers
semilogx(repmat(X,[2,1]),[P25Y; WhLBY],'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
hold on
semilogx(repmat(X,[2,1]),[P75Y; WhUBY],'--',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
% Set the whisker lower bounds and upper bounds in x direction
LBX=10.^(log10(X)-Param.WhWidth);
UBX=10.^(log10(X)+Param.WhWidth);
% Now plot the cap of the whisker
semilogx([LBX; UBX],repmat(WhLBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
semilogx([LBX; UBX],repmat(WhUBY,[2,1]),'-',...
        'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
clear LBX
clear UBX
% Now plot the box
% First calculate the box lower and upper bounds in the x
% direction
LBX=10.^(log10(X)-Param.BoxWidth);
UBX=10.^(log10(X)+Param.BoxWidth);
for i=1:Param.NumBox
```

```
    % plot the IQR using rectangle
    rectangle('Position',...
            [LBX(i) P25Y(i) UBX(i)-LBX(i) IQRY(i)],...
            'EdgeColor',Param.BoxColor,'LineWidth',...
            Param.LineWidth)
end
% Now plot the median
semilogx([LBX; UBX],repmat(P50Y,[2,1]),'-',...
          'Color',Param.MedColor,'LineWidth',Param.LineWidth)
clear LBX
clear UBX
% Now plot the outliers
for i=1:Param.NumBox
    % Find Any Outliers
    OutMask=Y(:,i)<WhLBY(i)|Y(:,i)>WhUBY(i);
    % If there are outliers plot them
    if sum(OutMask)>0
        loglog(repmat(X(i),[sum(OutMask),1]),Y(OutMask,i),...
             '+','Color',Param.OutColor,'LineWidth',...
             Param.LineWidth)
    end
end
```

```
case 'semilogy'
    % The plot is semilogy
    % First plot the whiskers
    semilogy(repmat(X,[2,1]),[P25Y; WhLBY],'--',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    hold on
    semilogy(repmat(X,[2,1]),[P75Y; WhUBY],'--',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    % Set the whisker lower bounds and upper bounds in x direction
    LBX=X-Param.WhWidth;
    UBX=X+Param.WhWidth;
    % Now plot the cap of the whisker
    semilogy([LBX; UBX],repmat(WhLBY,[2,1]),'-',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    semilogy([LBX; UBX],repmat(WhUBY,[2,1]),'-',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    clear LBX
    clear UBX
    % Now plot the box
    % First calculate the box lower and upper bounds in the x
    % direction
    LBX=X-Param.BoxWidth;
```

```
UBX=X+Param.BoxWidth;
for i=1:Param.NumBox
   % plot the IQR using rectangle
   rectangle('Position',...
            [LBX(i) P25Y(i) UBX(i)-LBX(i) IQRY(i)],...
            'EdgeColor',Param.BoxColor,'LineWidth',...
            Param.LineWidth)
end
% Now plot the median
semilogy([LBX; UBX],repmat(P50Y,[2,1]),'-',...
          'Color',Param.MedColor,'LineWidth',Param.LineWidth)
clear LBX
clear UBX
% Now plot the outliers
for i=1:Param.NumBox
   % Find Any Outliers
   OutMask=Y(:,i)<WhLBY(i)|Y(:,i)>WhUBY(i);
   % If there are outliers plot them
   if sum(OutMask)>0
      semilogy(repmat(X(i),[sum(OutMask),1]),Y(OutMask,i),...
               '+','Color',Param.OutColor,'LineWidth',...
               Param.LineWidth)
```

```
        end
    end
case 'loglog'
    % The plot is a loglog
    % First plot the whiskers
    loglog(repmat(X,[2,1]),[P25Y; WhLBY],'--',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    hold on
    loglog(repmat(X,[2,1]),[P75Y; WhUBY],'--',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    % Set the whisker lower bounds and upper bounds in x direction
    LBX=10.^(log10(X)-Param.WhWidth);
    UBX=10.^(log10(X)+Param.WhWidth);
    % Now plot the cap of the whisker
    loglog([LBX; UBX],repmat(WhLBY,[2,1]),'-',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    loglog([LBX; UBX],repmat(WhUBY,[2,1]),'-',...
            'Color',Param.WhiskColor,'LineWidth',Param.LineWidth)
    clear LBX
    clear UBX
    % Now plot the box
    % First calculate the box lower and upper bounds in the x
```

```
% direction
LBX=10.^(log10(X)-Param.BoxWidth);
UBX=10.^(log10(X)+Param.BoxWidth);
for i=1:Param.NumBox
   % plot the IQR using rectangle
   rectangle('Position',...
            [LBX(i) P25Y(i) UBX(i)-LBX(i) IQRY(i)],...
            'EdgeColor',Param.BoxColor,'LineWidth',...
            Param.LineWidth)
end
% Now plot the median
loglog([LBX; UBX],repmat(P50Y,[2,1]),'-',...
          'Color',Param.MedColor,'LineWidth',Param.LineWidth)
clear LBX
clear UBX
% Now plot the outliers
for i=1:Param.NumBox
   % Find Any Outliers
   OutMask=Y(:,i)<WhLBY(i)|Y(:,i)>WhUBY(i);
   % If there are outliers plot them
   if sum(OutMask)>0
      loglog(repmat(X(i),[sum(OutMask),1]),Y(OutMask,i),...
```

```
                '+','Color',Param.OutColor,'LineWidth',...
                Param.LineWidth)
            end
        end
    end
end
```