COLEMAN, DEIDRA ANDREA. Advances in Significance Testing for Cluster Detection. (Under the direction of Donald E. K. Martin and Brian J. Reich.)

Over the past two decades, much attention has been given to data driven project goals such as the Human Genome Project and the development of syndromic surveillance systems. A major component of these types of projects is analyzing the abundance of data. Detecting clusters within the data can be beneficial as it can lead to the identification of specified sequences of DNA nucleotides that are related to important biological functions or the locations of epidemics such as disease outbreaks or bioterrorism attacks. Cluster detection techniques require efficient and accurate hypothesis testing procedures.

In this dissertation, we improve upon the hypothesis testing procedures for cluster detection by enhancing distributional theory and providing an alternative method for spatial cluster detection using syndromic surveillance data. In Chapter 2, we provide an efficient method to compute the exact distribution of the number and coverage of $h$-clumps of a collection of words. This method involves defining a Markov chain using a minimal deterministic automaton to reduce the number of states needed for computation. We allow words of the collection to contain other words of the collection making the method more general. We use our method to compute the distributions of the number and coverage of $h$-clumps in the Chi motif of *H. influenzae*.

In Chapter 3, we provide an efficient algorithm to compute the exact distribution of multiple window discrete scan statistics for higher-order, multi-state Markovian sequences. This algorithm involves defining a Markov chain to efficiently keep track of probabilities needed to compute p-values of the statistic. We use our algorithm to identify cases where the available approximation does not perform well. We also use our algorithm to detect unusual clusters of made free throw shots by National Basketball Association players during the 2009-2010 regular season.

In Chapter 4, we give a procedure to detect outbreaks using syndromic surveillance data while controlling the Bayesian False Discovery Rate (BFDR). The procedure entails choosing an appropriate Bayesian model that captures the spatial dependency inherent in epidemiological data and considers all days of interest, selecting a test statistic based on a chosen measure that provides the magnitude of the maximal spatial cluster for each day, and identifying a cutoff value that controls the BFDR for rejecting the collective null hypothesis of no outbreak over a collection of days for a specified region. We use our procedure to analyze botulism-like syndrome data collected by the North Carolina Disease Event Tracking and Epidemiologic Collection Tool (NC DETECT).
DEDICATION

To my mother, Gwendolyn W. Ross, who has supported me in every endeavor I have attempted.
BIOGRAPHY

Deidra Andrea Coleman grew up in Savannah, GA. She graduated from Savannah High School in 2000. After completing high school, she completed her undergraduate studies at Shaw University earning a Bachelor of Science in Mathematics in 2003 with the honors distinction of summa cum laude. She joined the Department of Statistics at North Carolina State University in January 2004 and completed her Master of Statistics in May 2006.
I’d like to begin by thanking and acknowledging members of the statistics department. I’d like to thank my co-advisors, Drs. Donald E. K. Martin and Brian J. Reich, for their commitment to guiding me throughout this process. I would also like to thank my committee members, David Dickey and Yichao Wu, for their assistance along the way. Additionally, I’d like to acknowledge current and former faculty members of the department that have supported me in various ways throughout this journey: Drs. William Swallow, Dennis Boos, Sastry Pantula, Kimberly Weems, Roger Woodard, Jacqueline Hughes-Oliver, Montserrat Fuentes, and Renee Moore. I’d also like to acknowledge current and former departmental staff members that have been supportive throughout the process: Adrian Blue, Chris Waddell, Alison McCoy, Lanakila Alexander, and Terry Byron.

Next, I’d like to thank and acknowledge people and organizations outside of the department that have propelled me forward in critical ways and moments. I am thankful for the EDGE community, specifically for EDGE members and affiliates who have consistently checked in on me. I’d like to acknowledge the help that the dissertation grant program provided to me, specifically the writing coach and students that participated in the program along with me. I am grateful for the financial support that I have received both directly and indirectly from the National Science Foundation (NSF) and the National Aeronautics and Space Administration (NASA). I am also appreciative of the introduction to the wonderful people at NC DETECT that my graduate school representative, Dr. Julie Ivy, provided me with. Additionally, I am thankful for the willingness that the NC DETECT team has had in assisting me both with obtaining data and understanding the collection of the data. More recently, I have had the sincere support of the vice president of academic affairs at my current institution, Dr. Hazel Ervin, and other supportive members of the Philander Smith College community.

Lastly but certainly not least, I’d like to thank and acknowledge members of my personal support community. My pastor, Bishop James H. McKenzie, whose spiritual guidance has been a consistent sounding board. My undergraduate mentor that has not wavered in his belief in me, Dr. McLouis Clayton. My mother, Gwendolyn W. Ross, who first taught me the value of hard work and has been willing to support me in every endeavor I attempted. My stepfather, Cordie L. Ross, who has committed to supporting me as if I was his natural daughter. My grandmother, Elizabeth C. Fields, who taught me with her life the true value of higher education. My friend-sister, Michele Johnson, who has proofread countless documents for mechanical and grammatical errors, many of which included content that she might not have understood. My friend, Yinn, whose consistent support in various ways has been priceless. To all of you and the countless others that I have not been able to name here, I say thank you.
# TABLE OF CONTENTS

**LIST OF TABLES** .......................................................... vi

**LIST OF FIGURES** ......................................................... vii

Chapter 1 Introduction ..................................................... 1

Chapter 2 Distribution of $h$-Clump Statistics for a Collection of Words .......................... 4
  2.1 Introduction ............................................................ 4
  2.2 Number and Coverage of $h$-clumps of a Pattern ......................... 5
  2.3 Defining the Markov Chain ............................................. 7
    2.3.1 Describing the Strings of the State Space ....................... 7
    2.3.2 Describing the Transition Matrix ............................... 13
    2.3.3 Describing the Initial Distribution ............................. 13
  2.4 Computational Considerations ........................................ 13
  2.5 Application to the Chi motif of *H. Influenzae* ....................... 14
  2.6 Summary ............................................................... 15

Chapter 3 Exact Distribution of the Multiple Window Discrete Scan Statistic
  for Higher-Order Markovian Sequences ................................. 16
  3.1 Introduction ............................................................ 16
  3.2 Discrete Multiple Window Scan Statistic ............................. 18
  3.3 Defining the Markov Chain ............................................. 19
    3.3.1 Description of the State Space ................................ 19
    3.3.2 Description of the Transition Matrix $\mathcal{T}$ .................. 21
    3.3.3 Description of the Initial Distribution $\tilde{\pi}$ ............. 22
  3.4 Computational Set Up ................................................... 23
    3.4.1 Generating the $m$-tuples .................................... 23
    3.4.2 Generating the State Space .................................... 24
    3.4.3 Set Up of the Transition Matrix $\mathcal{T}$ and the Initial Distribution $\tilde{\pi}$ 27
    3.4.4 Other Computational Considerations .............................. 28
  3.5 Numerical Results ..................................................... 28
    3.5.1 Application to NBA Free Throw Data ............................. 31
  3.6 Summary ............................................................... 36

Chapter 4 A Bayesian False Discovery Approach to Syndromic Surveillance ......................... 37
  4.1 Introduction ............................................................ 37
  4.2 BFDR Procedure for Outbreak Detection ................................ 40
  4.3 Simulation Study ....................................................... 42
  4.4 Application to NC DETECT data ....................................... 43
  4.5 Summary ............................................................... 47

References ................................................................. 50
# LIST OF TABLES

| Table 3.1 | Model fit statistics | 33 |
| Table 3.2 | Tail probabilities of the observed scan statistic values of select NBA players within windows \{35, 40\} | 34 |
| Table 3.3 | Most significant clusters among windows \{30, 35, 40\} for NBA players with free throw percentages in \(\text{[.75,.8)}\) for the 2009-10 regular season | 34 |
| Table 3.4 | Most significant clusters among windows \{24, 26, 28\} for NBA players with free throw percentages in \(\text{[.7,.75)}\) for the 2009-10 regular season | 35 |
| Table 4.1 | False Discovery Rate (FDR) and True Discovery Rate (TDR) results for the BFDR approach and the spatial scan statistic methodology under various outbreak conditions, defined by the constant change in count \(\Delta\) for outbreak counties and the spatial correlation \(\phi\), when the test statistic is the maximum mean first-order neighborhood posterior probability for day \(t\), \(r\) is 5, and the significance level of the spatial scan statistic is 0.002. | 44 |
| Table 4.2 | False Discovery Rate (FDR) and True Discovery Rate (TDR) results for the BFDR approach and the spatial scan statistic methodology under various outbreak conditions, defined by the constant change in count \(\Delta\) for outbreak counties and the spatial correlation \(\phi\), when the test statistic is the maximum mean second-order neighborhood posterior probability for day \(t\), \(r\) is 4.5, and the significance level of the spatial scan statistic is 0.002. | 45 |
LIST OF FIGURES

Figure 2.1 DFA for the Pattern $W = \{1001\}$. .................................................. 10

Figure 3.1 Comparison of ratios of approximate (Glaz and Zhang) to exact (proposed) tail probabilities for the two window combination $\{16,22\}$ and fixed window discrete scan statistic tail probabilities for each window size. The horizontal axis gives the critical value for window size 16. The solid unmarked horizontal line is plotted at ratio 1. .................................................. 29

Figure 3.2 Comparison of ratios of approximate (Glaz and Zhang) to exact (proposed) tail probabilities for the two window combination $\{16,22\}$ and fixed window discrete scan statistic tail probabilities for each window size. The horizontal axis gives the critical value for window size 16. The solid unmarked horizontal line is plotted at probability .0001. .................................................. 30

Figure 3.3 Comparison of ratios of approximate (Glaz and Zhang) to exact (proposed) tail probabilities for the three window combination $\{16,18,22\}$ and fixed window discrete scan statistic tail probabilities for each window size. The horizontal axis gives the critical values for each window size. ............................................. 31

Figure 3.4 Comparison of ratios of approximate (Glaz and Zhang) to exact (proposed) tail probabilities for the two window combination $\{16,18,22\}$ and fixed window discrete scan statistic tail probabilities for each window size. The horizontal axis gives the critical values for each window size. ............................................. 32

Figure 4.1 Map of North Carolina counties with counties in which outbreaks have been injected in the simulation data shaded. .................................................. 43

Figure 4.2 Total daily counts of emergency department (ED) visits classified as exhibiting symptoms associated with the botulism-like syndrome for Dare, Polk, Richmond, and Wake Counties from January 1, 2008 to December 31, 2010. ............................................. 46

Figure 4.3 Map of Average Population Size from 2008-2010 for each NC County based on estimates of the office of state and budget management. Counties are shaded in the following manner: unshaded counties have populations less than 20,000; light gray shaded counties have populations of at least 20,000 but less than 50,000; gray shaded counties have populations of at least 50,000 but less than 100,000; and dark gray shaded counties have populations of at least 100,000. .................................................. 47

Figure 4.4 Posterior probabilities of no outbreak by day from analyzing botulism-like syndrome count data collected by NC DETECT. Red solid (black cross) circles represent outbreak (non-outbreak) days. .................................................. 48

Figure 4.5 Plots of the adjusted counts for the days designated as outbreaks by the BFDR procedure. The upper left corner plot is the plot for July 14, 2009, the upper right corner plot is the plot for July 31, 2009, the lower left corner plot is the plot for July 28, 2010, and the remaining plot is for September 21, 2010. Counties in black (gray) have adjusted count $Y_{it}/E_{it}$ greater than (less than) 2. .................................................. 49
Chapter 1

Introduction

There are many significant events that tend to be associated with clustering within sequence or spatial data. For example, a sequence of symbols called a string is considered to be over-represented in DNA if it occurs excessively within a relatively short span of the sequence. In sports, a basketball player is considered to have the “hot-hand” if he successfully makes an exceptional amount of his attempted shots within some relatively short period of time. The onset of an epidemic is thought to have occurred when there is an increased number of individuals exhibiting a particular set of symptoms within some relatively small region. Developing methods to aid in the identification of truly significant events through the detection of clusters across sequence and spatial data has long been of interest to statisticians.

In this dissertation, we provide methodology that extends the capability of detecting clusters that might lead to the identification of significant events. Specifically, we provide algorithms to compute the exact distributions of statistics that are needed to obtain p-values for significance testing that will signify abnormal activity. We also delineate an improved hypothesis testing procedure for signaling abnormal activity based on syndromic surveillance data.

Distributions of statistics of sequence data such as the number of times a string appears within a random sequence is used to determine the probability that a significant event has happened. Balakrishnan and Koutras [1] provided a comprehensive introduction of distributions of statistics associated with strings of identical symbols (runs) as well as “near perfect” runs (scans). Their introduction included descriptions of the earliest methodology that appeared within the literature to compute distributions for various statistics associated with runs and scans. They also included examples of applications across a variety of fields that rely on the knowledge of distributions of runs such as start-up demonstration testing, reliability testing, and process monitoring. Fu and Lou [2] formally introduced the finite Markov chain imbedding technique and described many of the earliest algorithms introduced in the literature for computing distributions of various statistics of collections of strings that incorporated the development
of a Markov chain for efficiently computing distributions. Robin, Rodolphe, and Schbath [3] provided a comprehensive review of the methodology used to obtain distributions of statistics for strings deemed as exceptional for the field of computational biology. They also provide several procedures to obtain approximations of probabilities in cases where an algorithm for finding the exact probabilities was not yet available.

Methods continue to emerge to compute exact distributions of statistics associated with patterns for many cases for which only approximations were available. Aston and Martin [4] and Martin and Aston [5] provided methodology to compute distributions associated with various collections of strings by defining Markov chains to efficiently perform computations. Stefanov, Robin, and Schbath [6] provided methodology to compute the exact distribution of the number of maximal sets of overlapping occurrences of a single string in a sequence. Bassino, Clément, Fayolle, and Nicodème [7] computed the exact distribution of the number of maximal sets of overlapping occurrences of a pattern in a sequence, where no string of the pattern was allowed to contain a substring of another string of the pattern and the sequence data was assumed to follow a Bernoulli model. In Chapter 2, we extend that work to provide an efficient algorithm to compute the exact distribution of the number of \( h \)-clumps of a pattern in a sequence and the number of symbols that compose the \( h \)-clumps that appear within a sequence. We allow strings of a pattern to be substrings of other strings of the pattern and the sequence data is assumed to follow a Markovian model of an arbitrary order.

Some other advances in the distributional theory of statistics associated with patterns has focused on scans within sequence data. The typical scan statistic of interest for sequence data is the maximum of the set of sums obtained by summing each of the possible substrings composed of consecutive symbols of a fixed length. The fixed length is typically referred to as the window. As a result of the vast amount of developments in methodology surrounding this special class of patterns, several books have emerged to summarize research on scan statistics (see Glaz and Balakrishnan [8], Glaz, Naus, and Wallenstein [9], and Glaz, Pozdnyakov, and Wallenstein [10]).

Recently emerging scan statistic methodology for sequence data has focused on the development of exact results for cases in which only good approximations were previously available and the establishment of methods for cases when the window size is unknown. Fu [11] defined a Markov chain to compute the distribution of the discrete fixed window scan statistic for first-order, bi-state Markovian trials. To offer alternative methodology involving less computational time, Ebneshahrashoob, Gao, and Wu [12] used a probability generating function to compute the distribution of the same statistic for the case where events were assumed to be first-order Markovian trials. Defining a Markov chain to compute p-values, Martin [13] provided the methodology for the case where the data is assumed to be higher-order, multi-state Markovian trials. Glaz and Zhang [14] introduced a multiple window scan statistic and provided an approximate to its distribution for obtaining p-values. In Chapter 3, we provide an
efficient algorithm to compute the exact distribution of the multiple window scan statistic for data assumed to follow a multi-state, higher-order Markovian model.

Unlike in the case of sequence data, emerging scan statistic methodology for cluster detection in spatial data has focused on selecting a statistic that adequately captures the information available about a cluster from spatial data and developing efficient hypothesis testing procedures based on such statistics. In his foundational paper, Kulldorff [15] proposed a likelihood-ratio statistic based on circular zones, called a spatial scan statistic, as a test statistic for detecting spatial clusters within independently distributed Bernoulli or Poisson spatial data. He also proposed a hypothesis testing procedure based on randomization testing. Several extensions of the spatial scan statistic have appeared in the literature to suggest adjustments to the test statistic based on the nature of the spatial data. In another direction towards improvements in this area, Sun et al. [16] proposed an oracle test statistic that would capture the spatial dependency within the data and hypothesis testing procedures that would improve on the detection of spatial clusters by extracting as much information as possible from the observed data. Sun et al. [16] demonstrated the efficiency of their methodology by showing via simulation that their method controlled the frequentist False Discovery Rate (FDR). In chapter 4, we select a test statistic that will capture any spatial dependency within the data across a collection of specified days and we provide a procedure for detecting spatial clusters that controls FDR by controlling the Bayesian False Discovery Rate (BFDR).
Chapter 2

Distribution of $h$-Clump Statistics for a Collection of Words

2.1 Introduction

The abundant occurrence of clumps of patterns has proven to be a significant indicator of exceptional patterns in DNA sequences and intrusion detection trace sequences. In addition, the closely related statistic of clump coverage has been used to establish an initial screening criterion for determining candidate tandem repeats for the analysis of DNA sequences [17].

In this chapter, we provide an algorithm to compute marginally the exact distribution of the number and coverage of $h$-clumps of a collection of words within data that is assumed to follow a stationary multi-state Markovian model of an arbitrary order. Here we compute exact distributions by associating a minimal deterministic finite automaton with the collection of words, using a transition matrix (and its powers) to hold probabilities for automaton transitions, keep track of statistic updates, and compute coefficients of a probability generating function. This work extends the excellent results of Stefanov, Robin, and Schbath [6] in two directions by (i) providing exact results that do not depend on the knowledge of any other closely related distribution and (ii) allowing any word of the pattern to be completely contained in another word of the pattern.

The chapter is organized as follows. In Section 2.2, the number and coverage of $h$-clumps is defined. A description of the Markov chain used to compute the distributions of $h$-clump statistics is described in Section 2.3. A computational technique that we use is described in Section 2.4. Section 2.5 contains an application to computing p-values for the observed number of clumps and clump coverage of the Chi motif in $H. Influenzae$. The final section is a summary.
2.2 Number and Coverage of $h$-clumps of a Pattern

A word here is a specified sequence of symbols from a finite alphabet $\Sigma$. The length of the word is the number of symbols of the word. For example, let $\Sigma = \{0, 1\}$, the binary alphabet. Then 10011 is a length 5 word. A prefix of a word is a substring that starts at the beginning of the word and a suffix of a word is a substring that ends at the end of a word. More formally, a suffix of a word $w = w_1 w_2 \ldots w_{|w|}$ is a substring of the form $w_l w_{l+1} \ldots w_{|w|}$, $l \in \{1, 2, \ldots, |w|\}$, where $|w|$ denotes the length of the word, and a prefix of a word $w = w_1 w_2 \ldots w_{|w|}$ is a substring of the form $w_1 w_2 \ldots w_l$, $l \in \{1, 2, \ldots, |w|\}$. Consider again the word 10011 from the binary alphabet. The set $\{1, 10, 100, 1001, 10011\}$ contains all of the prefixes of the word 10011 and the set $\{1, 11, 011, 0011, 10011\}$ contains all of its suffixes. A proper prefix of a word is a prefix of a word of length less than the length of the word and a proper suffix of a word is a suffix of a word of length less than the word. For instance, the proper prefixes of 10011 is the set of prefixes of 10011 excluding the proper suffixes of 10011 and the set of length less than the length of the word and a proper suffix of 10011 is the set of suffixes of 10011 containing all of its suffixes. A proper prefix of a word is a prefix of a word of length less than the length of the word and a proper suffix of a word is a suffix of a word of length less than the word. For instance, the proper prefixes of 10011 is the set of prefixes of 10011 excluding the word 10011 and the proper suffixes of 10011 is the set of suffixes of 10011 excluding the word 10011. A word $w^{(j)}$ is said to overlap a word $w^{(j')}$ if at least one suffix of $w^{(j')}$ is also a proper prefix of $w^{(j)}$. For example, rat overlaps scar as $r$ is a suffix of scar and a proper prefix of rat. Note that it is possible for words to overlap by more than one substring and a word may also overlap itself. Consider the words 10011 and 11011. 11011 overlaps 10011 by the substrings 1 and 11; both words overlap themselves. A word $w^{(j)}$ encloses a word $w^{(j')}$ if $w^{(j')}$ is a factor of $w^{(j)}$ that is not a suffix of $w^{(j)}$, where a factor is any substring of a word. The word 10011 encloses the words 100, 00, 001 and 1001 among others. A pattern here is a finite collection of words, where each word is of length at least 2.

Let $X_{1:n} \equiv X_1, \ldots, X_n$ be a stationary multi-state $m$th order Markovian sequence with observed values denoted by $x_{1:n} \equiv x_1, x_2, \ldots, x_n$, where the state space for each random variable $X_t$, $t \in \{1, 2, \ldots, n\}$ is $\Sigma$. Note that the stationary Markov model has been suggested as a good model for sequences in computational biology [3]. The element of $\Sigma$ observed at time (or position) $t$, $t \in \{1, 2, \ldots, n\}$ is denoted by $x_t$. Let $W \equiv \{w^{(1)}, \ldots, w^{(k)}\}$ be a pattern. A word $w^{(i)} \in W$ occurs at the time (or position) where the word ends. We say that a clump of $W$ has occurred with observation $x_t$ if one of the following conditions is satisfied:

- $x_{t-|w^{(j)}|+1:t} \equiv w^{(j)}$ for some $w^{(j)} \in W$ with $x_{t-|w^{(j)}|+1:t}$ not overlapping a previous occurrence of a word of $W$; or
- there exists a string $x_{l:t}$ that begins and ends with a word of $W$ such that all occurrences of $W$ in the string $x_{l:t}$ except the first word overlaps or encloses the word occurring before it, where $l$ is any positive integer greater than zero and $t > (l+1)$.

Example 2.1 As an example, consider the observed sequence $x_{1:10} = 1100011011$ from the binary alphabet along with the pattern $\{11, 00, 1001\}$. The word 11 $\in W$ occurs at $x_2$. There
are two clumps of $W$ in $x_{1:10}$ given by $x_{1:7}$ and $x_{9:10}$. Notice that $x_{1:7}$ begins and ends with the word 11 and every occurrence of a word of $W$ within the string overlaps or encloses the word occurring before it. The word 1001 encloses 00, which is the word occurring before it in the string.

The number of sequences positions that compose a clump is called the coverage of the clump. Consider again the observed sequence and pattern given in Example 2.1. The coverage of clumps of $W$ in this case is 9. The first clump covers 7 sequence positions and the second clump covers 2.

A positive gap is a string that exists between two adjacent words of a string whose symbols are not considered a part of either word. The size of a positive gap is its number of symbols, denoted by nonnegative integers. A negative gap is a substring of two overlapping words of a string that is a proper suffix of the first word of the string and a proper prefix of the second word of the string. The size of a negative gap is the negative of its number of symbols, denoted by negative integers. An examination of the definition of a clump reveals that there must be at least one symbol within overlapping words in a clump that is a symbol of both words. We refer to this requirement as having a negative gap of maximal size negative one.

Let $h$ be some integer. We say that an $h$-clump has occurred at time (or position) $t$ if one of the following two conditions is satisfied:

- $x_{t-|w(j)|+1:t} \equiv w(j)$ for some $w(j) \in W$ with $x_{t-|w(j)|+1:t}$ not overlapping a previous occurrence of a word of $W$; or

- there exist a string $x_{l:t}$ that begins and ends with a word of $W$ such that all of its substrings that are formed by a pair of occurrences of $W$ contain either a negative gap or a positive gap that does not exceed $h$ symbols, where $l$ is any positive integer greater than zero and $t \geq (l + z - 1)$, where $z$ is the length of the shortest word of $W$.

By definition, every substring of an $h$-clump formed by a pair of occurrences of $W$ must either contain at least one symbol that is a symbol of both words of $W$ or contain a factor of length $h$ or shorter that is not a part of either of the occurring words of $W$. This means that depending on the value of $h$ a negative gap or a positive gap of maximal size $h$ is required for each $h$-clump formed by more than one word of $W$. Hence, a clump is a special case of an $h$-clump, where $h = -1$. In fact, an $h$-clump is a string composed of clumps with gaps allowed between them of length no more than $h$. The number of sequence positions that compose the clumps within an $h$-clump is the coverage of an $h$-clump of a pattern, that is, the positive gaps are not counted as a part of the coverage of an $h$-clump. Consider again Example 2.1; there is a single $h$-clump for $h \geq 1$ given by $x_{1:10}$; the gap between the clump given by $x_{1:7}$ and the clump given by $x_{9:10}$
is a positive gap of 1 because there is only one symbol $x_8 = 0$ between the clumps. There are 9 positions covered in the sequence by the 1-clump.

### 2.3 Defining the Markov Chain

To define a Markov chain for computing the probability of the number of $h$-clumps or the coverage of $h$-clumps, we must clearly define the state space, transition matrix, and initial distribution of the Markov chain. In this section, we delineate the state space by providing the condition that a string must satisfy to be a state. Additionally, we describe the transition matrix by (1) providing the rules based on which transitions between states of the Markov chain occur and (2) explaining the relationship between the transition matrix $T$ and the transition matrix of the Markov chain used for computing the desired probabilities, denoted $\Omega$. Similarly, we delineate the initial distribution by providing the details of the relationship between the initial distribution of $X_{1:n}$ and the initial distribution of the Markov chain for computing the desired probabilities.

#### 2.3.1 Describing the Strings of the State Space

The typical vector-matrix computations associated with Markovian sequences of order $m$ allow us to simultaneously compute the probability of a Markovian sequence of order $m$ ending in any of the possible length $m$ strings made up of symbols of the alphabet $\Sigma$, called $m$-tuples, after $n$ observations. We exploit this feature by identifying a collection of strings to be states of our Markov chain that correspond to prefixes of a state indicating the occurrence of an $h$-clump or the continuation of an $h$-clump. Note that an $h$-clump continues with the occurrence of each new clump that occurs before a gap greater than $h$ symbols occurs.

Strings that signify the continuation of an $h$-clump are needed for the coverage of $h$-clumps. One way for an $h$-clump to be continued is for the word that defined the existence of the $h$-clump to be overlapped on the right by another occurrence of a word of the pattern forming a clump of two words. With this in mind, we succinctly describe a set of strings that signify the continuation of a clump in this manner. Let $W_{ext} = \{u | \exists w^{(i)}, w^{(j)} \in W \text{ with } u = \alpha_iv_{ij}\delta_j, \alpha_iw_{ij} = w^{(i)}, v_{ij}\delta_j = w^{(j)}, |\alpha_i| > 0, |v_{ij}| > 0, |\delta_j| > 0 \}$ be the extended set. Note that $v_{ij}$ is both a proper suffix of $w^{(i)}$ and a proper prefix of $w^{(j)}$. The string $\delta_j$ extends $w^{(i)}$ to the overlapping word occurrence $w^{(j)}$. We include prefix strings of the extended set in our state space for any gap size. For $h$ greater than -1, another way for an $h$-clump to be continued is for a word $w^{(j)} \in W$ to be concatenated on the right of the word $w^{(j)} \in W$ that defined the existence of the $h$-clump with a string of length $h$ or shorter between them. Hence, we include in the state space the collection of strings defined by $W\Sigma^dW, d \in \{0, 1, 2, \ldots, h\}$ for $h \geq 0$. 

7
where \( \Sigma^d \) in general is the collection of all strings of length \( d \) with \( \Sigma^0 = \epsilon \), the empty string. Here \( W_1W_2 \) denotes the concatenation of words of the patterns \( W_1 \) and \( W_2 \). Up to this point, we have considered only the events that signify the first continuation of an \( h \)-clump. There are many strings that signify the continuation of an \( h \)-clump if continuations beyond the first one are considered; however, those strings contain the same information as their prefixes that are included based on considering only the first continuation of an \( h \)-clump. Hence, we need not include states that represent these strings in the state space. In general, we include in the state space strings of the extension set \( W_{ext}^h \equiv W_{ext} \cup (\cup_{d=0}^h \Sigma^d W) \) to signify the continuation of an \( h \)-clump.

Similarly, strings that signify the occurrence of an \( h \)-clump are needed for the number and coverage of \( h \)-clumps. Since every \( h \)-clump by definition begins with a word of \( W \), we include the class of strings concisely defined as the collection of words of \( W \) in the state space. We also include the set of all \( m \)-tuples, denoted \( \Sigma^m \). The \( m \)-tuples are needed because \( X_{1:n} \) is assumed to follow an \( m^\text{th} \) order Markovian model.

In addition to the words of the pattern \( W \), the extension set \( W_{ext}^h \), and the \( m \)-tuples \( \Sigma^m \), we need to include in the state space all proper prefixes of length greater than \( m \) of the union of \( W \) and \( W_{ext}^h \). These strings are needed to keep track of probabilities of a state that will lead to the occurrence or continuation of an \( h \)-clump. We will refer to these strings generally as progress strings because they signify progress towards states that indicate that an \( h \)-clump has occurred or continues.

**Example 2.2** Let \( m = 2 \), \( h = 0 \), \( \Sigma = \{0, 1\} \), and \( W = \{00, 11, 1001\} \). Then \( \Sigma^m = \{00, 01, 10, 11\} \), \( W_{ext} = \{000, 011, 11001, 10011, 1001001\} \), and \( W_{ext}^0 \equiv W_{ext} \cup \{0000, 0011, 1100, 1111, 10011001, 001001, 111001, 1001100, 100111\} \). The progress strings of \( W \cup W_{ext}^0 \) is the set \( \{000, 110, 100, 001, 1100, 1001, 10010, 100100, 0010, 00100, 111, 1110, 11110, 100110, 100111, 1001100, 1001101, 1001110\} \). Note that all of the progress strings are of length 3 or greater.

While the states that we have described so far contain natural indicators for the occurrence of an \( h \)-clump or the continuation of an \( h \)-clump, the states lack information about how the count or coverage of \( h \)-clumps should be adjusted with the occurrence or continuation of an \( h \)-clump. When an \( h \)-clump is continued, the \( h \)-clump coverage increases by the number of symbols that make up the \( h \)-clump continuation. This value will likely vary depending on the particular type of \( h \)-clump continuation that occurs. Also, recall that we allow words of a pattern to be enclosed in other words of a pattern. This introduces the possibility that, when \( h = -1 \), the clump count may need to decrease with the continuation of a clump. For instance, consider the pattern \( \{00, 11, 1001\} \) from Example 2.2 and the observed sequence \( x_{1:5} = 11001 \). With \( h = -1 \), the \( h \)-clump increases to 1 at observation \( x_2 \) and to 2 with observation \( x_4 \); however,
the $h$-clump count must decrease by 1 with the $x_5$ observation since the $h$-clump observed at $x_2$ continues. For these reasons, we add a component to each state to represent the value that the statistic for which the Markov chain is used to compute probabilities changes. With this adjustment, each of the states in the state space is now a 2-dimensional vector that includes a string component and a numerical component.

**Minimizing the Number of States using a DFA**

To eliminate some of the computational burden that is inherent to this problem, we setup an Aho-Corasick automaton [18], which is a type of deterministic finite automaton (DFA), for the set $W \cup W_{ext}^h \cup \Sigma^m$ and then minimize the automaton using a modified version of the Hopcroft algorithm. This provides us with the smallest number of states needed for setting up a Markov chain to efficiently compute probabilities of the number or coverage of $h$-clumps. This is important because a smaller state space means less computer storage is needed and fewer matrix computations are needed.

A DFA is defined using the following components:

- a set of input symbols;
- a finite set of states;
- a transition function that gives for each state and input symbol the next state;
- an initial state, the designated start state for the automaton; and
- a set of final states, which is a subset of the set of finite states.

A DFA is commonly represented in graphical form such as in Figure 2.1. Each node is a DFA state and the directed line segments represent transitions between states for the given input symbol. The initial state is the dotted node and the final state is the shaded node.

A DFA is a model that accepts or rejects a finite strings of symbols. Let $Q$ denote a finite set of states of a DFA and $\delta$ denote a transition function of a DFA. A string $w_1 w_2 \ldots w_{|w|}$, where $w_i, i \in \{1, 2, \ldots, |w|\}$, are input symbols of a specific DFA, is recognized or accepted by the specified DFA if a sequence of states $q_0, q_1, \ldots, q_{|w|}$, where $q_i \in Q$ for each $i$ satisfies the following conditions:

- $q_0$ is the initial state of the specified DFA;
- $q_{i+1} = \delta(q_i, w_{i+1})$, for $i \in \{0, 1, \ldots, n - 1\}$; and
- $q_{|w|}$ is a finite state of the specified DFA.
An Aho-Corasick automaton is a DFA for a pattern, where the input symbols is the alphabet associated with the pattern \( \Sigma \); the set of finite states is the set of all prefixes of the words of the pattern and the initial state; and the set of final states is the words of the pattern. The DFA in Figure 2.1 is an Aho-Corasick automaton. We set up an Aho-Corasick automaton where the final states is the collection of strings in the set \( W \cup W^{\text{ext}} \cup \Sigma^m \).

A DFA is said to be minimized if it is composed of the smallest number of states needed to correctly recognize the specified pattern. A typical algorithm for minimizing a DFA is the Hopcroft algorithm [19]. The traditional Hopcroft algorithm is implemented by first partitioning the states into two basic classes: non-final states and final states. This partition is chosen because the outcome for any input string that transitions to a final state is different than the outcome for an input string that transitions to a non-final state. Specifically, an input string that transitions to a final state is accepted while an input string that transitions to a non-final state is not. After the Hopcroft algorithm partitions the states into non-final and final states, the algorithm chooses an input symbol of the specified DFA, call it \( x \), and either the set of non-final states or the set of final states, call the chosen set \( A \). Then it splits the non-final states into two subsets; a subset that contains strings that lead to \( A \) on \( x \) if it is not empty and a subset that contains strings that do not lead to \( A \) on \( x \) if it is not empty. This task is repeated to split the final states, if the new subsets are not empty. If any splits lead to non-empty subsets, the new partition is made up of the classes created by that split and the original classes that were not split if they exist. This procedure is then repeated by choosing a class from the new partition and an input symbol to arrive at the next new partition. This process continues until no more non-empty classes can be formed based on splitting the classes of the current partition. This results in a partition that only leaves strings that transition into the same class combined. These equivalence classes are the states of the minimized DFA.
For our purposes, we need more than the typical initial partition. We need to initially partition the automaton states into classes that only allow strings to combine that update the value of the statistic the same. We modify the Hopcroft algorithm by changing the initial partition to reflect the changes in the updates to the value of the statistics. Hence, we first partition the automaton states into the following general classes:

- \( Q_{\text{pre,new}} \) - proper prefixes of words of \( W \) that contain a suffix that is a word of \( W \) that indicates the occurrence of a new \( h \)-clump;
- \( Q_{\text{pre,ext}} \) - proper prefixes of words of \( W \) that contain a suffix that is a word of \( W \);
- \( Q_W \) - states that are words of \( W \) that are not included in \( Q_{\text{pre,new}} \cup Q_{\text{pre,ext}} \) and all \( m \)-tuples that contain a suffix that is a word of \( W \);
- \( Q_{\text{pre}} \) - proper prefixes of \( W \) that do not contain a suffix that is a word of \( W \);
- \( Q_{\text{path}} \) - strings along the path between strings of \( Q_W \) and \( W^h_{\text{ext}} \) that do not have a word of \( W \) as a suffix, where being along the path means that a string is a progress string that indicates that an \( h \)-clump could continue;
- \( Q_{\text{path,new}} \) - strings along the path between strings of \( Q_W \) and \( W^h_{\text{ext}} \) that have a word of \( W \) as a suffix;
- \( Q_{\text{path,ext}} \) - strings along the path between strings of \( Q_W \) and \( W^h_{\text{ext}} \) that have a word of \( W \) as a suffix;
- \( Q_{\text{ext}} \) - strings of \( W^h_{\text{ext}} \) that don’t fall into any of the classes above; and
- \( Q_m \) - strings of \( \Sigma^m \) that do not fall in any of the classes above.

We refer to classes that contain strings that when observed require updating the statistic value as counting classes. The classes \( Q_W \), \( Q_{\text{pre,new}} \), and \( Q_{\text{path,new}} \) serve as counting classes for both number and coverage of \( h \)-clumps whereas \( Q_{\text{pre,ext}} \), \( Q_{\text{path,ext}} \), and \( Q_{\text{ext}} \) serve as counting classes for only the coverage of \( h \)-clumps. When a given pattern does not contain enclosed words, \( Q_{\text{pre,ext}} \), and \( Q_{\text{path,new}} \) are empty whereas there is a possibility that \( Q_{\text{path,ext}} \) may not be empty. Consider the pattern \( W = \{aaaa\} \) in which there are no enclosed words. For this case, \( W^{-1}_{\text{ext}} = \{aaaaa,aaaaaaaa,aaaaaaaaaaaa\} \) so that the strings aaaaa and aaaaaa are in the set \( Q_{\text{path,ext}} \) for \( W \), that is, they are substrings that must occur for aaaa to become aaaaaaaa. However, \( Q_{\text{pre,ext}} \) and \( Q_{\text{path,new}} \) for \( W = \{aaaa\} \) are both empty.

Since the update value of the coverage of \( h \)-clumps is not always the same, we need to further partition the counting classes used to compute its distribution. Thus, we partition the
necessary counting classes based on the number of coverage positions that need to be added to the total coverage at the time of \( h \)-clump occurrence or \( h \)-clump continuation. Similarly, we need to further partition the counting classes used for computing the distribution of the number of \( h \)-clumps. Hence, we partition the appropriate counting classes based on the number of coverage positions that need to be added each possible transition \( q \to q' \) in the original automaton. We denote the states of the minimal automaton by \( Q \). The result is a minimal automaton that has at most the same number of states as the automaton states, we obtain the minimal automaton or coarsest partition [20] of the automaton partition for minimization. Now that we have identified an appropriate initial partition for the completed all of the aforementioned partitioning, we have our basic classes, which is our initial partition for minimization. Now that we have identified an appropriate initial partition for the automaton states, we obtain the minimal automaton or coarsest partition [20] of the automaton states. The result is a minimal automaton that has at most the same number of states as the original automaton. We denote the states of the minimal automaton by \( Q' \).

We denote the number and coverage of \( h \)-clumps in \( X_{1:n} \) by \( \gamma(h) \) and \( \eta(h) \), respectively. For each possible transition \( q_i \to q'_j \) with \( q_i, q'_j \in Q' \), the value by which the statistic is updated is determined using the general classes as follows:

- \((q_i, q'_j) \in [(Q_m \cup Q_{pre}) \times (Q_{pre,new} \cup Q_W)] \cup (Q_{path} \times Q_{path,new}) \cup [(Q_{pre,new} \cup Q_{pre,ext}) \times Q_W],\)
  the clump count increases by \( 1 - \eta_{w(i)} \) and coverage by \(|w^{(j)}| - \gamma_{w(i)}\), where \( w^{(j)} \in W \)
  is the word that has just occurred and \( \eta_{w(i)} \) (\( \gamma_{w(i)} \)) denotes the number (or coverage) of \( h \)-clumps enclosed within \( w^{(j)} \);

- \((q_i, q'_j) \in (Q_{pre} \times Q_{pre,ext}) \cup (Q_{path} \times Q_{path,ext}) \cup [(Q_{pre,new} \cup Q_{pre} \cup Q_{path,new} \cup Q_{path}) \times Q_{ext}],\)
  the \( h \)-clump count decreases by \( \eta_{w,j} \) and coverage increases by \(|w_j| - \gamma_{w,j} \) if \( g < 0 \), but if \( h \geq g \geq 0 \), the \( h \)-clump count decreases by \( \eta_{w,(j)} \) and the coverage increases by \(|w^{(j)}| - \gamma_{w,(j)}\), where \( g \) is the gap between the last two word occurrences;

- \((q_i, q'_j) \in [(Q_{pre,new} \cup Q_{pre,ext}) \times Q_{pre,ext}] \cup [Q_W \times (Q_{pre,ext} \cup Q_{path,ext} \cup Q_{ext})] \cup [(Q_{pre,ext} \cup Q_{path,ext} \cup Q_{ext}) \times (Q_{path,ext} \cup Q_{ext})],\)
  coverage increases by one but the \( h \)-clump count remains the same.

It should be noted that when a state \( q' \in Q' \) contains more than one string \( q \in Q \) we need only use the first string of \( q' \) to determine the updates. At this point, we delete strings \( q' \in (Q' \cap Q_{ext}) \) and map their incoming transitions and corresponding updates to the longest proper suffix of \( q' \) that is still an element of \( Q' \), call it \( q'' \). These strings are deleted because they transition the same as the \( q'' \) strings to which they are assigned with the exception of the update information that is appropriately associated with \( q'' \) strings. States \( q' \in Q' \) that contain a single string \( q \in Q \) are deleted if \(|q| < m \). When states \( q' \in Q' \) contain more than one string \( q \in Q \), the states are deleted if all strings of \( q' \) have length less than \( m \). These states can be deleted since they are never entered for \( t \geq m \).
2.3.2 Describing the Transition Matrix

Now that we have obtained the appropriate DFA, we set up a square transition matrix $\Omega$ to hold the monomials that contain information about the probability and the update associated with a transition from one DFA state to another. Let $T$ be the transition probability matrix for $m$-tuples. The matrix element $\Omega_{ij}$ corresponding to the transition $q_i \rightarrow q_j'$ is the monomial with the coefficient from $T$ that represents the transition from the length $m$ suffix of $q_i$ to the length $m$ suffix of $q_j'$, the variable $\gamma(h)$ for the number of $h$-clumps or $\eta(h)$ for the coverage of $h$-clumps, and an exponent that is the update associated with the transition.

2.3.3 Describing the Initial Distribution

Let $\pi$ be the stationary distribution of the sequence over $m$-tuples that is embedded into a first-order Markov chain, so that $\pi$ also serves as the initial distribution for $\Sigma^m$. We denote an arbitrary element of $\Sigma^m$ by $\tilde{x}_m$. The initial distribution $\tilde{\pi}$ over the states $Q'$ is a vector of monomials, where $\tilde{\pi}(\tilde{x}_m)$ is the monomial with the coefficient $\pi(\tilde{x}_m)$, the variable $\gamma(h)$ for the number of $h$-clumps or $\eta(h)$ for the coverage of $h$-clumps, and an exponent that is the number or coverage of $h$-clumps in $\tilde{x}_m$ depending on the statistic of interest. If $q' \neq \tilde{x}_m$, then $\tilde{\pi}(q') = 0$.

2.4 Computational Considerations

The probability generating function $\psi_{\gamma(h)}$ or $\psi_{\eta(h)}$ is obtained by computing $\tilde{\pi} \Omega^{n-m} 1$, where $1$ is a column vector consisting of ones that is used to sum over the minimal automaton states. A MATLAB program was written to perform computations. In addition to the techniques previously described, we also used a technique to minimize the number of multiplications needed for computing $\Omega^{n-m}$. Rather than compute $\Omega^{n-m}$ by performing $n-m-1$ matrix multiplications, we obtain it by successively doubling and multiplying only the matrices identified through converting $n-m$ to a binary number. By successively doubling, we mean that we compute and store $\Omega^2 = \Omega \times \Omega$, $\Omega^4 = \Omega^2 \times \Omega^2$, $\Omega^8 = \Omega^4 \times \Omega^4$, $\ldots$, $\Omega^L$ with $L = 2^{\log_2(n-m)}$. Then when we convert $n-m$ into its binary representation we multiply all the matrices that correspond to a one in the binary representation of the number. For example, the binary representation of 100 is 1100100$_2$. The ones in the binary representation of 100 correspond to $2^2$, $2^5$, and $2^6$, which means that we compute $\Omega^2, \Omega^4, \Omega^8, \Omega^{16}, \Omega^{32}$, and $\Omega^{64}$ to obtain $\Omega^{100} = \Omega^4 \Omega^{32} \Omega^{64}$. The method of doubling requires only 8 matrix multiplications for $n-m$ equal to 100 whereas the computation without doubling requires 99 matrix multiplications.
2.5 Application to the Chi motif of *H. Influenzae*

We now compute distributions of clump statistics for the Chi motif of *H. Influenzae*, $W = GΣTGGTGG$, where $Σ = \{A, C, G, T\}$. This pattern is important because its presence is needed in processes that prevent a cell from destroying its own DNA each time that it is broken. The pattern is over-represented in *H. Influenzae*, occurring 223 times in 215 clumps in the complete genome of 1,830,140 base pairs [21].

For this pattern, $W_{\text{ext}} = \{WTGG, WTGGTGG, WΣTGGTGG\}$. Whereas the Aho-Corasick automaton has 165 states, many of them are combined in the minimal automaton (one reason being that any letter from $Λ = \{A, C, T\}$ used as the second symbol gives the same progress into $W$), leaving only 31 states in the minimal automaton: $\{A, C, T, G, GG, GA, GC, GT, GGT, GAT, GΣG, WTGGTGG, WTGGTG, WTGGT, GΣTGG, GΣTGGT, GΣTGGTG, W, WTG, WG, WAT, WGT, WΣTG, WΣTGG, WΣTGGT, WΣTGGTG, WTGG, WΣTGGTGG, WT, WA, WC\}$. We assume that the underlying DNA sequence is first-order Markovian $(m = 1)$. States WTGGTGG and WΣTGGTGG are deleted since they are in $Q_{\text{ext}}$. For 0-clumps, $W_{\text{ext}}^0 = W_{\text{ext}} \cup WW$ and the Aho-Corasick automaton has 273 states, whereas its coarsest partition has only 41, of which only 38 are needed to carry out the computation, an indication of the great savings that can take place by forming the coarsest partition.

To compute the distributions of $h$-clump statistics for the Chi motif *H. Influenzae*, we estimate the transition matrix of the DNA sequence using the maximum likelihood estimates based on transitions of symbols $\{A, C, G, T\}$ in the data (see [3, p. 124]). The estimated transition matrix is:

$$
\mathcal{T} = \begin{pmatrix}
0.383 & 0.155 & 0.164 & 0.299 \\
0.343 & 0.187 & 0.216 & 0.254 \\
0.269 & 0.264 & 0.197 & 0.269 \\
0.230 & 0.160 & 0.220 & 0.390
\end{pmatrix}
$$

The initial distribution of $x_1$ is $(\pi_A, \pi_C, \pi_G, \pi_T) = (0.305, 0.184, 0.198, 0.313)$, which is obtained by solving the equation $\pi \mathcal{T} = \pi$ with the additional condition that the entries of $\pi$ must sum to 1. The exact probability of observing 215 or more clumps in 1,830,140 base pairs conditional on the parameter estimates is $1.2515211 \times 10^{-11}$. Figure 2.2(a) shows the distribution of the number of both clumps and 0-clumps for the sequence (the plots overlap), and Figure 2.2(b) has the distribution of clump coverage. The combined run time was approximately 26 minutes for the count of $h$-clumps, $h = -1, 0$. 

14
Figure 2.2: Distributions of clump and 0-clump count (a) and coverage (b) for the pattern $W = G\Sigma TGGTGG$ ($\Sigma = \{A,C,G,T\}$), with $n = 1.83014 \times 10^6$ and $m = 1$. Initial and transition probabilities for the data are as given in the text.

2.6 Summary

In this chapter, we present an efficient method for computing the distributions of the number and coverage of $h$-clumps of a pattern. We describe how to minimize the state space needed for computing the distributions using an automaton and a modified version of the Hopcroft algorithm. The method is applied to examining the Chi motif of *H. Influenzae*.

Computing the distributions of statistics of a pattern with minimal restrictions on the relationships between words of the pattern is computationally intensive. Results that use the development of a Markov chain with a minimal state space reduce the computational complexity of the problem. In future work, we will develop an optimal state space for computing the distribution of statistics associated with words of a pattern, where both a minimal and maximal gap is required between word occurrences.
Chapter 3

Exact Distribution of the Multiple Window Discrete Scan Statistic for Higher-Order Markovian Sequences

3.1 Introduction

Reliable methods for identifying clusters of abnormal activity are needed in many fields. For example, public health officials regularly monitor hospital admissions for an unusual number of patients reporting particular combinations of symptoms. In bioinformatics, the genome is scanned to identify clusters with increased frequency of a particular pattern. When clusters like these occur, it is crucial to ascertain the likelihood that the cluster occurred by random chance. Appropriate and efficient means to detect truly unusual clusters will avoid spurious conclusions and allow time, money, and personnel to be used more efficiently.

Scan statistics are commonly used to test for abnormal clusters in space or time (see [8], [9], and [10]). A scan statistic is a measure obtained by scanning for the maximum number of events that occur in a window (the scanning window) of the size of the cluster of interest. Distributions of scan statistics are complicated, but essential for determining the likelihood that a cluster occurred by random chance.

The earliest techniques for computing distributions of scan statistics were based on a combinatorial approach. Naus ([22], [23], [24]) provided the methodology to compute the distribution of the fixed window continuous scan statistic for uniformly distributed events ([22], [23]) and the fixed window discrete scan statistic for Bernoulli trials [24]. These methods require a computationally-intensive summation of many determinants. Despite this computational limitation, the results of Naus proved to be extremely useful for the development of good approximations and bounds on scan-statistic probabilities (see [25], [26], and [27]).
More recently, some of the developments in the distributional theory of scan statistics have been based on computing distributions through defining an appropriate Markov chain for computing the necessary probabilities. Fu [11] computed the distribution of the discrete fixed window scan statistic for first-order, bi-state Markovian trials by defining a Markov chain. His work includes as a special case the distribution of the same statistic for Bernoulli trials and it represented a significant computational advancement. Using a probability generating function method through the setup of a Markov chain, Ebneshrashoob, Gao, and Wu [12] computed the distribution for the case where events were assumed to be first-order Markovian trials. They assert that this approach led to a reduction in computational time. Martin [13] extended the distributional theory of scan statistics further and computed the distribution of the fixed discrete window scan statistic for higher-order, multi-state Markovian trials by defining a Markov chain. The approach of defining a Markov chain to compute the probabilities has proven very efficient and overcomes many of the long-standing computational limitations.

In most cases, the size of the scanning window needed to detect unusual clusters is unknown. Glaz and Zhang [14] show that in these cases, choosing an arbitrary window size is ineffective because detection power diminishes greatly based on the difference between the arbitrarily chosen window size and the true cluster size. One alternative approach might be to perform several tests, one for each possible window size. This approach would result in the loss of power due to conservative adjustments that would be necessary to control Type I errors in the presence of multiple testing. To minimize loss of power to detect unusual clusters associated with an unknown cluster size, variable (multiple) window scan statistics have been introduced for several probability models (see [15], [28], [29], [30]). However, to date, the only exact result for multiple window scan statistics is the methodology provided by Wu, Glaz, and Fu [31] to compute the distribution of the discrete multiple window scan statistic for Bernoulli and first-order, bi-state trials.

In this chapter, we provide an algorithm to compute the exact distribution of the multiple window scan statistic for discrete data modeled by higher-order, multi-state Markovian trials. Here we compute the exact distribution by extending the methodology used in Martin [13] in which Markov chains are defined to compute \( p \)-values for the fixed discrete window scan statistic for higher-order, multi-state Markovian trials. The methodology of Martin [13] hinges on identifying suitable state spaces for the Markov chains used for computing the exact distribution. Each state space is defined by defining a criterion for strings to be included in the state space. The criterion used allows a string to be included in the state space if it is either:

(a) an \( m \)-tuple that does not correspond to obtaining or exceeding a particular scan statistic value or

(b) a string that is a proper prefix of at least one string that corresponds to obtaining or
exceeding a particular scan statistic value with a string length of at least \( m + 1 \) but no longer than the window size minus one.

We extend the methodology of Martin [13] to the multiple window case by defining a criterion for numerical strings to be included in the state space. Identifying the criterion is a complex task in this case because there is a collection of scan statistic values and each string needs only to correspond to obtaining or exceeding at least one of those values to be excluded from the state space. We overcome this complexity by defining a criterion for strings to be included in the state space that is based on both the maximum possible sum of a string for each window size and the current sums of a string for each window size.

The multiple window discrete scan statistic for higher-order, multi-state trials is defined in Section 3.2. A description of the Markov chain developed to compute the distribution of the discrete multiple window scan statistic is provided in Section 3.3. We include a discussion of how to efficiently generate strings needed to setup the Markov chain in Section 3.4. This section also includes a discussion of an implication that we exploit to speed up the computation in that special case. Some numerical results are given in Section 3.5, which includes the application of this methodology to National Basketball Association (NBA) free throw data. We study data from the NBA because basketball enthusiasts often marvel at a player on a hot streak that suggests a deviation from their career performance. A summary is provided in the final section.

3.2 Discrete Multiple Window Scan Statistic

Let \( X \equiv \{X_1, X_2, \ldots, X_n\} \) be a discrete time multi-state stationary Markov chain of order \( m \), where the state space for each random variable \( X_t, t \in \{1, 2, \ldots, n\} \) is a finite subset of the integers denoted \( \Sigma \). Without loss of generality, we let \( \Sigma = \{0, 1, \ldots, k - 1\} \). The fixed window discrete scan statistic for a specified window size \( \omega \), \( S_\omega \), is the maximum sum of the set of all sums of \( \omega \) consecutive trials, that is,

\[
S_\omega = \max_{1 \leq j \leq n - \omega + 1} \left\{ \sum_{t=j}^{j+\omega-1} X_t \right\}.
\]

Let \( T \) denote the transition matrix of \( X \). Typically, \( S_\omega \) is used to test the null hypothesis \( H_{0S_\omega} : \exists \) a matrix \( T \) such that \( T \) is the transition matrix \( X \) \( \forall t \) against the alternative hypothesis \( H_{AS_\omega} : \exists \) at least one window of size \( \omega \) in which the transition matrix is not \( T \). When a specific window size is unknown but a plausible set of window sizes are available, there is a need to perform a single test based on multiple window sizes.

Let \( 2 \leq \omega_1 < \ldots < \omega_k \leq n - 1 \) be a specified sequence of window sizes. The discrete multiple window scan statistic \( S_\omega \) is the collection of fixed window scan statistics that correspond to the
specified window values, that is, the multivariate statistic

\[ S_\omega \equiv \{ S_{\omega_1}, \ldots, S_{\omega_\kappa} \} \].

Let \( \Omega \) denote the set of window sizes \( \{ \omega_1, \omega_2, \ldots, \omega_\kappa \} \). The discrete multiple window scan statistic is used to test the null hypothesis \( H_0 \) against the alternative hypothesis that there exists at least one window of any of the window sizes \( \omega_i \in \Omega \) in which the transition matrix is not \( T \). Let \( s_{\omega_i} \) denote the observed value of the fixed window scan statistic for \( \omega_i \). The null hypothesis is rejected if there is at least one window \( \omega_i \in \Omega \) with a large sum, that is, if \( P\{ \bigcup S_{\omega_i} \geq s_{\omega_i} \} \) is small. Critical values \( s_{\omega_i, \text{crit}} \) for determining if \( P\{ \bigcup S_{\omega_i} \geq s_{\omega_i} \} \) is small are obtained by computing tail probabilities \( P\{ \bigcup S_{\omega_i} \geq s_{\omega_i, \text{crit}} \} \), which is the contribution of this work.

### 3.3 Defining the Markov Chain

To define a Markov chain for computing the probability that there is at least one window of one of the specified sizes with a large sum, we must clearly describe the state space, transition matrix, and initial distribution of the Markov chain. We will describe the state space by providing our criteria for a specified sequence of symbols, called a string, to be a state. In addition, we will delineate the transition matrix by (1) providing the rules based on which transitions between states of our Markov chain occur and (2) illuminating the relationship between the transition matrix \( T \) and the transition matrix of the Markov chain used for computing the desired probabilities, denoted \( T' \). We similarly describe the initial distribution by providing the specifics of the relationship between the initial distribution of \( X \) and the initial distribution of the Markov chain used for computing the desired probabilities, denoted by \( \tilde{\pi} \).

#### 3.3.1 Description of the State Space

Computing tail probabilities \( P\{ \bigcup S_{\omega_i} \geq s_{\omega_i, \text{crit}} \} \) for \( X \) amounts to computing the probability of observing within \( X \) at least one string that satisfies \( [S_{\omega_i} \geq s_{\omega_i, \text{crit}}] \) for at least one of the specified window sizes \( \omega \in \Omega \). By a string satisfying \( [S_{\omega_i} \geq s_{\omega_i, \text{crit}}] \) for one of the specified window sizes \( \omega \), we mean that within that string there are \( \omega \) or fewer consecutive symbols with a sum that is greater than or equal to \( s_{\omega_i, \text{crit}} \). As an example, let \( \Sigma = \{0, 1, 2\}, \omega_1 = 5, \omega_2 = 8, \omega_3 = 13, s_{5, \text{crit}} = 3, s_{8, \text{crit}} = 6, s_{13, \text{crit}} = 7 \), and suppose that the string 1001010000112 occurs within \( X \). The string satisfies \( [S_5 \geq 3] \) since the sum of the last three consecutive symbols is 4. The string also satisfies \( [S_{13} \geq 7] \) as the entire string sums to 7; however, the string does not satisfy \( [S_8 \geq 6] \).

For a given \( X \), set of specified window values, and associated set of critical values, there
may be many strings that satisfy \([S_\omega \geq s_{\omega,\text{crit}}]\) for at least one of the specified window sizes. We limit our interest to strings that do not begin with zero as each string that begins with a zero contains the same information as its suffix that does not begin with zero. Hereafter when we refer to a string that satisfies \([S_\omega \geq s_{\omega,\text{crit}}]\) for some \(\omega_i\) we will be referring to only those strings that do not begin with zero.

The typical vector-matrix computations associated with \(m^{th}\) order Markov chains permit us to simultaneously compute the probability of an \(m^{th}\) order Markovian sequence ending in any of the possible strings of length \(m\) made up of symbols of the state space, called \(m\)-tuples, after \(n\) observations. Since we are not interested in which string has been observed but rather only that a string satisfying \([S_\omega \geq s_{\omega,\text{crit}}]\) for some \(\omega_i\) has been observed, we include in our state space a single state, denoted by \(A\), that represents the collection of all strings satisfying \([S_\omega \geq s_{\omega,\text{crit}}]\) for one of the window sizes. We also include in our state space all of the \(m\)-tuples that do not satisfy \([S_\omega \geq s_{\omega,\text{crit}}]\) for some \(\omega_i\). These \(m\)-tuples are needed because \(X\) is assumed to be \(m^{th}\) order Markovian.

In addition to the state \(A\) and the \(m\)-tuples not covered by state \(A\), we also need to include in our state space all proper substrings of lengths \(m+1, m+2, \ldots, \omega_\kappa - 1\) represented by \(A\). These strings are needed to keep track of the probabilities of being in a state that will lead to a string corresponding to \(A\). Consider observing the data sequence \(x_1, \ldots, x_5 = 11202\) and let \(m = 2\). Suppose that 11202 is in \(A\). Then, in addition to the state \(A\) and the appropriate \(m\)-tuples we would include 112 and 1120. By doing this, we are able to obtain the probability of the Markov chain ending in 112 and 1120 at any time step. In this case, we would have the probability from the product of \(\pi(11)\) and \(P(2|11)\) at time step 3 when we need to multiply it by \(P(0|12)\) to compute the probability of 1120 at time step 4 as progress towards the string 11202. We refer to the strings that we include in our state space to track progress towards a string represented by \(A\) as progress strings. Our state space is made up of the state \(A\), the set of all \(m\)-tuples that do not contain a substring that satisfies \([S_\omega \geq s_{\omega,\text{crit}}]\) for one of the window sizes \(\omega_i\), and the set of progress strings.

**Example 3.1** Let \(m = 2, \omega_1 = 3, \omega_2 = 6\). The state space of \(X\) is \(\Sigma = \{0, 1\}\). Now consider the state space for the Markov chain to compute \(P[S_3 \geq 3 \cup S_6 \geq 4]\), which is the set \(\{00, 01, 10, 11, 100, 101, 110, 1001, 1010, 1011, 1100, 1101, 10011, 10101, 10110, 11001, 11010\}\) and the state \(A\). Note that all of the \(m\)-tuples \(\{00, 01, 10, 11\}\) are included because none of them contain a substring that satisfies \([S_\omega \geq s_{\omega,\text{crit}}]\) for one of the window sizes. In fact, the string sums are 0, 1, 1, and 2, respectively. All other strings in the state space are of lengths ranging from 3 to 5. None of these strings begin with zero as is the case with the \(m\)-tuple 01. These strings are included because they indicate progress towards strings corresponding to \(A\). For instance, the string 100 is included because it indicates progress towards 100111, which is
a string represented by $A$ because it corresponds to $S_3 = 3$ and $S_6 = 4$. Similarly, the string 101 is included because it indicates progress towards 10111 or 101011 or 101101.

The conditions provided in this subsection completely characterize the state space; however, generating these strings can be a cumbersome task for large window values. We provide an efficient way to generate the strings in Subsection 3.4.2.

### 3.3.2 Description of the Transition Matrix $T$

In the traditional $m^{th}$ order Markov chain setting, every row of the transition matrix represents an $m$-tuple and the probabilities within the row signify the probability of transitioning into an $m$-tuple after observing a particular symbol. For instance, let $\Sigma = \{0, 1\}$, $m = 2$, and the transition matrix

$$
T = \begin{pmatrix}
0.9 & 0.1 & 0 & 0 \\
0 & 0 & 0.3 & 0.7 \\
0.1 & 0.9 & 0 & 0 \\
0 & 0 & 0.5 & 0.5
\end{pmatrix},
$$

where the order of the 2-tuples is $\{00, 01, 10, 11\}$. The $T(2, 3)$ element represents the probability of transitioning from 01 to 10 after observing a 0. Similarly, the $T(2, 4)$ element represents the probability of transitioning from 01 to 11 after observing a 1. Note that all other values in the row are 0 because transitioning into those 2-tuples is not possible. The transitions of 01 are more explicitly described in the following manner: 01 transitions to the length 2 suffix of the string generated by concatenating the observed symbol to the right of 01. As an illustration of this description, suppose that the observed symbol is 0. Then the string generated by concatenating 0 on the right of 01 is 010 and the length 2 suffix of 010 is 10.

The transition matrix $T$ is setup to keep track of probabilities of progress towards a string corresponding to $A$ using the probabilities from $T$. Based on the string generated by concatenating the observed symbol on the right of the state, a state of our Markov chain transitions to the next state in one of the following ways:

(a) it goes into state $A$ if the generated string satisfies $[S_{\omega_i} \geq s_{\omega_i, crit}]$ for some $\omega_i$, or

(b) it goes into a state other than $A$ that corresponds to the longest suffix of the generated string that is also a state in our state space.
Consider again the case described in Example 3.1, and specifically, the state 10110. Based on transition rule (a), the state 10110 transitions to state $A$ after the symbol 1 is observed. This follows from the fact that 101101 has a sum of 4 so that it satisfies $[S_6 \geq 4]$. The state 10110 transitions to 1100 after the symbol 0 is observed based on transition rule (b). The string generated by concatenating the state 10110 on the right by 0 is 101100. Its set of suffixes is \{101100, 01100, 1100, 100, 00, 0\}. The longest suffix of 101100 that is a state of our state space is 1100.

Establishing the rule under which transitions between states occur is only one part of setting up the transition matrix. We also need to describe how the probabilities are assigned. By definition every state is length $m$ or longer. The transition probabilities are assigned by considering the $m$-tuples of the states between which transitions occur. The probability of any state other than $A$, say $\tilde{\sigma}$, transitioning into another state of our Markov chain, call it $\tilde{\sigma}'$, is the probability that the ending $m$-tuple of $\tilde{\sigma}$ transitions into the ending $m$-tuple of $\tilde{\sigma}'$. For instance, consider again the string 10110 from Example 3.1 that transitions into 1100 after the symbol 0 is observed. The probability of our Markov chain going from state 10110 to 1100 is the probability that 10 transitions to 00. State $A$ is an absorbing state so state $A$ transitions to itself with probability 1.

### 3.3.3 Description of the Initial Distribution $\tilde{\pi}$

Typically, the initial distribution of an $m$th order Markov chain, denoted $\pi$, is a vector with the same number of elements as there are $m$-tuples. Each element of the vector represents the probability that a realization of the Markov chain begins with the $m$-tuple that corresponds to it. For instance, consider a 2nd order Markov chain with $\Sigma = \{0, 1\}$ and an initial distribution $\pi = (0.25, 0.3, 0.2, 0.25)$, where the order of the 2-tuples is $\{00, 01, 10, 11\}$. The $\pi(2)$ element represents the probability that a realization of the Markov chain begins with 01. Similarly, the $\pi(4)$ element represents the probability that a realization of the Markov chain begins with 11. Note that all the elements of $\pi$ sum to 1 as the chain must begin with one of the 2-tuples.

As discussed in the description of the state space, every $m$-tuple that does not satisfy $[S_{\omega_i} \geq s_{\omega_i,\text{crit}}]$ for some $\omega_i$ is included in the state space. These $m$-tuples are assigned the same probabilities in $\tilde{\pi}$ as given in the initial distribution of $X$. Every $m$-tuple that satisfies $[S_{\omega_i} \geq s_{\omega_i,\text{crit}}]$ for some $\omega_i$ is represented by state $A$ in $\tilde{\pi}$. Hence, we assign the sum of their probabilities to state $A$ in $\tilde{\pi}$. All other states of our Markov chain are assigned zero probabilities in $\tilde{\pi}$.
3.4 Computational Set Up

To implement the Markov chain defined in Section 3.3,

- window sizes, corresponding critical values, the state space $\Sigma$, transition matrix $T$, and initial distribution $\pi$ of $X$ must be provided;
- the $m$-tuples and strings of the state space must be generated;
- the transition matrix $T$ and initial distribution $\tilde{\pi}$ must be set up; and
- the appropriate vector-matrix computations must be performed.

3.4.1 Generating the $m$-tuples

Given $\Sigma$, the $m$-tuples are efficiently generated through the conversion of values $0, 1, \ldots, k^m - 1$ to base $k$ numbers of length $m$. A base $k$ number of length $m$ is a base $k$ number that is extended by concatenating the necessary number of zeros on the left of the base $k$ number so the number contains $m$ digits. For instance, the conversion of 5 to a base 3 number of length 2 is 12; however, to make 5 a length 4 base 3 number, two zeros need to be concatenated on the left of 12 to get 0012. The algorithm used to generate $m$-tuples is given as Algorithm 3.1. Each row of the $m$-tuple matrix generated by the algorithm represents one of the $m$-tuples. The (1,1) element is the first symbol of the first $m$-tuple, the (1,2) element is the second symbol of the first $m$-tuple, and so on. Similarly, the (2,1) element is the first symbol of the second $m$-tuple, the (2,2) element is the second symbol of the second $m$-tuple, and so on.

**Algorithm 3.1: Generating $m$-tuples**

**Input:** $k, m$

**Output:** A $k^m \times m$ matrix

for $\ell \leftarrow 0$ to $k^m - 1$ do
  $u \leftarrow \ell$
  for $v \leftarrow m$ to 1 by -1 do
    $mtuples(\ell + 1, m + 1 - v) \leftarrow \lfloor u/k^{v-1} \rfloor$
    $u \leftarrow u \mod k^{v-1}$
  end
end

Let $m = 2, k = 2$, and $\Sigma = \{0, 1\}$. Using the algorithm, the values 0, 1, 2, and 3 are converted to 00, 01, 10, and 11, respectively. The base 2 representations of 0, 1, 2, and 3 are 0, 1, 10, and 11, respectively; however, a zero is concatenated on the left of the base 2 representations of 0 and 1 to obtain length 2 strings 00 and 01. The concatenation of zeros on the left is accomplished in
Algorithm 3.1 by dividing the value to be converted to base \( k \) by descending powers of \( k \) from \( m - 1 \) to 1 and keeping only the integer part of the solution.

3.4.2 Generating the State Space

To determine if an \( m \)-tuple is a string of the state space, the window value of the \( m \)-tuple for each window size \( \omega_i \) must be obtained. Given a window size \( \omega \) and a string of length \( \omega \) or shorter, the window value of the string is the sum of all the symbols of the string. On the other hand, given a window size \( \omega \) and a string of length greater than \( \omega \), the window value of the string is the maximum of the set of sums obtained by summing all of the sets of \( \omega \) consecutive symbols. For instance, the window value of the 6-tuple 110101 for window size 7 is 4 whereas the window value of 110101 for window size 4 is 3. Let \( \tilde{\sigma}_j \) denote an arbitrary string of length \( j \) made up of symbols of \( \Sigma \) and \( W_{\omega_i, \tilde{\sigma}_j} \) denote the window value of \( \tilde{\sigma}_j \) for window size \( \omega_i \). The string \( \tilde{\sigma}_j \) does not satisfy \( [S_{\omega_i} \geq s_{\omega_i, \text{crit}}] \) for \( \omega_i \) if \( W_{\omega_i, \tilde{\sigma}_j} \leq s_{\omega_i, \text{crit}} - 1 \) \( \forall \omega_i \), then \( \tilde{\sigma}_j \) does not satisfy \( [S_{\omega_i} \geq s_{\omega_i, \text{crit}}] \) for any \( i \). Hence, an \( m \)-tuple is included in the state space if \( W_{\omega_i, \tilde{\sigma}_m} \leq s_{\omega_i, \text{crit}} - 1 \) \( \forall \omega_i \).

By the definition of a progress string, every string in the state space that is not an \( m \)-tuple is an extension of another string in the state space. Hence, the search for length \( j = m + 1, \ldots, \omega_{\kappa} - 1 \) progress strings is efficiently implemented by considering the extension of the length \( j - 1 \) progress strings by a symbol of \( \Sigma \). For instance, the length 4 strings \{1001, 1010, 1011, 1100, 1101\} of Example 3.1 for which \( \Sigma = \{0, 1\} \) would lead us to consider the strings \{10010, 10011, 10100, 10101, 10110, 10111, 11000, 11001, 11010, 11011\} as potential length 5 progress strings. However, the other 22 length 5 strings need not be considered because they are not extensions of the length 4 progress strings. When searching for progress strings of length \( m + 1 \), only extensions of \( m \)-tuples included in the state space that do not begin with zero are considered.

A progress string is by definition a string that does not satisfy \( [S_{\omega} \geq s_{\omega, \text{crit}}] \) for any \( \omega_i \) in the set of specified window sizes but that can be extended so that it satisfies \( [S_{\omega} \geq s_{\omega, \text{crit}}] \) for some \( \omega_i \) in the set of specified window sizes. Therefore, we need to check each potential progress string to determine if it satisfies \( [S_{\omega} \geq s_{\omega, \text{crit}}] \) for some \( \omega_i \) and to determine if it can be extended so that it satisfies \( [S_{\omega} \geq s_{\omega, \text{crit}}] \) for some \( \omega_i \). Consider again the potential length 5 progress string 11001 associated with Example 3.1. To compute the window value of 11001 for window size 3, three sums need to be computed: the sum of 110, the sum of 100, and the sum of 001. However, the string 11001 is an extension of 1100, a length 4 progress string of Example 3.1, which means that the sum of the first three consecutive symbols and the sum of the second three consecutive symbols of the string have already been determined not to exceed the \( s_{3, \text{crit}} \) value. In this case, we need only consider the sum of the last three consecutive symbols to
determine if 11001 satisfies \([S_3 \geq s_{3,crit}]\). In general, given a window size \(\omega\) and a string \(\tilde{\sigma}_j\), where \(\omega \leq j\), the current window value of \(\tilde{\sigma}_j\) for window size \(\omega\) is used to determine if \(\tilde{\sigma}_j\) satisfies \([S_\omega \geq s_{\omega,crit}]\) for \(\omega\) in order to aid in minimizing the number of computations needed for setting up the state space. The current window value of \(\tilde{\sigma}_j\) for window size \(\omega\), where \(\omega \leq j\), is defined as the sum of the last \(\omega\) symbols of \(\tilde{\sigma}_j\). The current window value \(\tilde{\sigma}_j\) for window size \(\omega\), where \(\omega > j\), is its window value. Let \(W_{\omega,\tilde{\sigma}_j}\) denote the current window value of \(\tilde{\sigma}_j\) for window size \(\omega\). The string \(\tilde{\sigma}_j\) does not satisfy \([S_\omega \geq s_{\omega,crit}]\) for \(\omega\) if \(W_{\omega,\tilde{\sigma}_j} \leq s_{\omega,crit} - 1\). If \(W_{\omega,\tilde{\sigma}_j} \leq s_{\omega,crit} - 1\) \(\forall \omega_i\), then \(\tilde{\sigma}_j\) does not satisfy \([S_\omega \geq s_{\omega,crit}]\) for any \(\omega_i\). For computational purposes, we keep track of current window values.

To determine if a string can be extended so that it satisfies \([S_\omega \geq s_{\omega,crit}]\) for some \(\omega_i\), it is necessary to introduce attainable window values. An attainable window value of a string for a window of \(\omega \in \Omega\) is a window value possible by concatenating on the right of the string at most enough symbols to make the total string length equal to the largest window of the set of specified window sizes. A progress string has an attainable window value of at least \(s_{\omega,crit}\) for some \(\omega \in \Omega\). Given a string \(\tilde{\sigma}_j = \sigma_1 \sigma_2 \ldots \sigma_j\), where \(\sigma_d \in \Sigma\) for \(d \in \{1, 2, \ldots, j\}\), and a set of specified window sizes \(\Omega\), the maximum attainable window value for a window size \(\omega_i\) is \((k - 1)\omega_i\) if \(\omega_i\) is less than or equal to \(\omega_k - j\) and \((\omega_k - j)(k - 1) + \sum_{d=\omega_k-\omega_i+1}^{j} \sigma_d\) otherwise. For instance, given \(\Sigma = \{0, 1\}\) and window sizes \(\{\omega_1, \omega_2, \omega_3\} = \{2, 4, 7\}\), the string 10010 has maximum attainable window values of 2, 3, and 4 for window sizes 2, 4, and 7, respectively. The maximum attainable window value of 10010 for window size 2 is computed by summing the last two symbols of 1001011. On the other hand, the maximum attainable window value of 10010 for window size 4 is computed by summing the last four symbols of 1001011. In the case of the window size 2, the maximum attainable window value was computed using only the symbols concatenated to make 10010 a length 7 string. In contrast, in the case of the window size 4, the maximum attainable window value was computed using both the symbols concatenated to make 10010 a length 7 string and the last two symbols of 10010. Let \(Z_{\omega_i, \omega_k, \tilde{\sigma}_j}\) denote the maximum attainable window value of \(\tilde{\sigma}_j\) for window size \(\omega_i\) when \(\omega_k\) is the largest specified window size. The string \(\tilde{\sigma}_j\) has a maximum attainable window value of \(s_{\omega,crit}\) for \(\omega\) if \(Z_{\omega, \omega_k, \tilde{\sigma}_j} \geq s_{\omega,crit}\) for \(\omega\). If \(Z_{\omega_i, \omega_k, \tilde{\sigma}_j} \geq s_{\omega_i,crit}\) for some \(\omega_i\), then \(\tilde{\sigma}_j\) can be extended so that it satisfies \([S_\omega \geq s_{\omega,crit}]\) for \(\omega_i\). For computational purposes, we keep track of maximum attainable window values.

The algorithm used to identify \(m\)-tuples to include in the state space and to efficiently generate progress strings of the state space that are not \(m\)-tuples using current and attainable window values is given as Algorithm 3.2. For each \(m\)-tuple and each window size, the window value is computed. An \(m\)-tuple is included in the state space if all of its window values are less than their corresponding critical values and the \(m\)-tuples current window values are stored. If an \(m\)-tuple is included in the state space and it does not begin with a zero, its maximum attainable window values are computed. Strings of length \(m + 1\) that are obtained by extending
the \(m\)-tuples that do not begin with zero are considered by computing their current window values and maximum attainable window values. Current window values of strings of length \(j = m + 1, \ldots, \omega_\kappa - 1\) for window sizes larger than \(j\) are computed by the single operation of adding the current window value of the length \(j - 1\) string being extended to the value of the

\begin{algorithm}
\textbf{Algorithm 3.2: Generating the State Space}

\begin{verbatim}
foreach \(m\)-tuple do
    foreach window size \(\omega_i\) do
        Compute the window value \(W_{\omega_i, \tilde{\sigma}_m}\)
    end
    if \(W_{\omega_i, \tilde{\sigma}_m} \leq s_{\omega_i, \text{crit}} - 1 \forall \omega_i\) then
        • Include \(\tilde{\sigma}_m\) in the state space
        • Keep the current window value \(W_{\omega_i, \tilde{\sigma}_m}\)
    end
end
foreach \(m\)-tuple included in the state space that does not begin with zero do
    foreach window size \(\omega_i\) do
        Compute the maximum attainable window value
    end
end
for \(\ell \leftarrow m + 1\) to \(\omega_\kappa - 1\) do
    foreach length \(\ell - 1\) string \(\hat{\sigma}_{\ell-1}\) in the state space that does not begin with 0 do
        foreach each symbol \(\sigma \in \Sigma\) do
            foreach window size \(\omega_i\) do
                • Compute the current window value of the string that results from concatenating \(\sigma\) on the right of \(\hat{\sigma}_{\ell-1}\), i.e., \(W_{\omega_i, \hat{\sigma}_{\ell-1}\sigma}\)
                • Compute the maximum attainable window value of \(\hat{\sigma}_{\ell-1}\sigma\)
            end
            if \(W_{\omega_i, \hat{\sigma}_{\ell-1}\sigma} \leq s_{\omega_i, \text{crit}} - 1 \forall \omega_i\) and \(Z_{\omega_i, \omega_\kappa, \hat{\sigma}_{\ell-1}\sigma} \geq s_{\omega_i, \text{crit}}\) for some \(\omega_i\) then
                Include \(\hat{\sigma}_{\ell-1}\sigma\) in the state space
            end
        end
    end
end
\end{verbatim}
\end{algorithm}
symbol that is extending it. Maximum attainable window values of strings of length \( j = m + 1, \ldots, \omega_\kappa - 1 \) for window sizes larger than \( \omega_\kappa - j \) are computed by obtaining the sum of the maximum attainable window value of the length \( j - 1 \) string being extended and the difference between the largest symbol of \( \Sigma \) and the symbol that is extending the length \( j - 1 \) string. A string of length \( j = m + 1, \ldots, \omega_\kappa - 1 \) that is considered as a potential progress string is included in the state space if all of its current window values are less than their corresponding critical values and if at least one of its maximum attainable window values is greater than or equal to its corresponding critical value.

**Algorithm 3.3: Set Up of \( T \)**

```text
count ← 0

foreach \( \tilde{\sigma}_j \in Q \setminus \{A\} \) do
  foreach \( \sigma \in \Sigma \) do
    sprow(count+1) ← the number of the state \( \tilde{\sigma}_j \)
    Generate \( \tilde{\sigma}_j \sigma \)
  foreach \( \omega_i \) do
    Compute \( W_{\omega_i, \tilde{\sigma}_j \sigma} \)
    if \( W_{\omega_i, \tilde{\sigma}_j \sigma} \geq s_{\omega_i, crit} \) then
      spcol(count+1) ← |Q|
    else
      Identify the longest suffix of \( \tilde{\sigma}_j \sigma \) that is also a state of \( Q \)
      spcol(count+1) ← the number of the identified state
    end
  end

  Identify the \( m \)-tuple that is a suffix of \( \tilde{\sigma}_j \), call it \( \tilde{\sigma}_m \)
  Identify the \( m \)-tuple that is a suffix of \( \tilde{\sigma}_j \sigma \), call it \( \tilde{\sigma}'_m \)
  spprob(count+1) ← \( T(\tilde{\sigma}_m, \tilde{\sigma}'_m) \)
end

sprow(count+1) ← spcol(count+1) ← |Q|

spprob(count+1) ← 1
```

### 3.4.3 Set Up of the Transition Matrix \( T \) and the Initial Distribution \( \tilde{\pi} \)

Let \( Q \) denote the state space of the Markov chain that we have defined to compute tail probabilities of scan statistics and \( |Q| \) denote the number of states in \( Q \). The transition matrix \( T \) is a \( |Q| \times |Q| \) sparse matrix with only \( k \) nonzero elements in each row. Hence, we set up \( T \) in a sparse fashion. The algorithm used to set up \( T \) is given as Algorithm 3.3. In Algorithm 3.3, spprob is the column vector of the probabilities of \( T \), sprow is the column vector with indicators of
which row each probability belongs in, and spcol is the column vector with indicators of which column each probability belongs in. The initial distribution $\tilde{\pi}$ is a $1 \times |Q|$ row vector.

### 3.4.4 Other Computational Considerations

Given $T$ and $\tilde{\pi}$, $P(\cup S_{\omega_i} \geq s_{\omega_i,\text{crit}})$ is the last element of the vector obtained by computing the product $\tilde{\pi}T^{n-m}$. While the Markov chain defined in Section 3.3 is useful for computing all tail probabilities of the discrete multiple window scan statistic, there are instances in which $P(\cup S_{\omega_i} \geq s_{\omega_i,\text{crit}})$ can be more efficiently computed. Let $s_{\omega_1,\text{crit}} \geq \ldots \geq s_{\omega_k,\text{crit}}$. Then $\omega_i < \omega_i+1$ and $s_{\omega_i,\text{crit}} \geq s_{\omega_i+1,\text{crit}}$ for $i \in \{1,2,\ldots,k-1\}$ implies that $[S_{\omega_i} \geq s_{\omega_i,\text{crit}}] \subset [S_{\omega_i+1} \geq s_{\omega_i+1,\text{crit}}]$ for $i \in \{1,2,\ldots,k-1\}$ so that $P(\cup S_{\omega_i} \geq s_{\omega_i,\text{crit}}) = P([S_{\omega_k} \geq s_{\omega_k,\text{crit}}])$. In general, if there exists a subset of window sizes $\{\omega_1,\omega_2,\ldots,\omega_j\} \in \Omega$, $j \leq k$ for which $s_{\omega_1} \geq s_{\omega_2} \geq \ldots \geq s_{\omega_j}$, then $P(\cup_{i=1}^{k}[S_{\omega_i} \geq s_{\omega_i,\text{crit}}]) = P(\cup_{i=j}^{k}[S_{\omega_i} \geq s_{\omega_i,\text{crit}}])$. A FORTRAN program was written to implement the algorithms described in this section.

### 3.5 Numerical Results

As an illustration of the usefulness of our approach, we examined the strength of the approximations of tail probabilities of the discrete multiple window scan statistic for Bernoulli trials provided by Glaz and Zhang [14] by comparing Glaz and Zhang approximations to the exact probabilities obtained by our approach for various cases. We computed probabilities for Bernoulli trials with our algorithm by assuming a first-order model with equal success probabilities, that is, we let $m = 1$ and $P(1|0) = P(1|1) = p$. The approximation of Glaz and Zhang [14] is:

$$
P\{\cup_{i=1}^{k}[S_{\omega_i} \geq s_{\omega_i,\text{crit}}]\} \approx \sum_{i=1}^{k} P[S_{\omega_i} \geq s_{\omega_i,\text{crit}}]
- \sum_{i=1}^{k-1} \left\{P[S_{\omega_i+1} \geq s_{\omega_i+1,\text{crit}}] \times \left( P\{[S_{\omega_i} \geq s_{\omega_i,\text{crit}}] \sum_{t=1}^{\omega_i+1} X_t = s_{\omega_i+1,\text{crit}} - 1\}
+ H(s_{\omega_i,\text{crit}} - 1|\omega_i+1) - 1, \omega_i - 1, s_{\omega_i+1,\text{crit}} - 1),
+ P\{[S_{\omega_i} \geq s_{\omega_i,\text{crit}}] \in \{X_1,\ldots,X_{n-\omega_i+1}\}\}\right\},
$$

where

$$
H(y|N,M,K) = \frac{M_y}{(N)_y} \frac{(N-M)}{(K-y)}
$$

is the hypergeometric probability function.

The Glaz and Zhang approximation is derived from the upper bound for the probability of the union provided by Hunter [32]. The result given by Hunter shows that

$$
P\{\cup_{i=1}^{k}[S_{\omega_i} \geq s_{\omega_i,\text{crit}}]\} \leq \sum_{i=1}^{k} P[S_{\omega_i} \geq s_{\omega_i,\text{crit}}]
- \sum_{i=1}^{k-1} P\{[S_{\omega_i} \geq s_{\omega_i,\text{crit}}] \cap [S_{\omega_i+1} \geq s_{\omega_i+1,\text{crit}}]\}.
$$
The Hunter upper bound is close to the exact probability when:

\[ \sum_{1 \leq j < i, j < \ell \leq K} \lambda \leq P(\cap \{ S_{\omega_i} \geq s_{\omega_i,\text{crit}} \} \cap \{ S_{\omega_j} \geq s_{\omega_j,\text{crit}} \} \cap \{ S_{\omega\ell} \geq s_{\omega\ell,\text{crit}} \}) + \ldots + (-1)^{K-1} P(\cap \{ S_{\omega_i} \geq s_{\omega_i,\text{crit}} \}) \]

is negligible. The fixed window probabilities of the Glaz and Zhang approximation are obtained using the approximation of Naus [33] given as Theorem 2.

Following Naus and Wallenstein [29], Glaz and Zhang consider that \( P\{ \{ S_{\omega_i} \geq s_{\omega_i,\text{crit}} \} \cap \{ S_{\omega_i+1} \geq s_{\omega_i+1,\text{crit}} \} \} = P\{ \{ S_{\omega_i} \geq s_{\omega_i,\text{crit}} \} \mid \{ S_{\omega_i+1} \geq s_{\omega_i+1,\text{crit}} \} \} P\{ \{ S_{\omega_i+1} \geq s_{\omega_i+1,\text{crit}} \} \} \) and approximate \( P\{ \{ S_{\omega_i} \geq s_{\omega_i,\text{crit}} \} \mid \{ S_{\omega_i+1} \geq s_{\omega_i+1,\text{crit}} \} \} \) by treating the following events conditional on observing a string that satisfies \( \{ S_{\omega_i+1} \geq s_{\omega_i+1,\text{crit}} \} \) as if they are mutually exclusive:

- the event that a string that satisfies \( S_{\omega_i} \geq s_{\omega_i,\text{crit}} \) is simultaneously observed with a string that satisfies \( S_{\omega_i+1} \geq s_{\omega_i+1,\text{crit}} \);
- the event that a string that satisfies \( S_{\omega_i} \geq s_{\omega_i,\text{crit}} \) is observed within the window where the string that satisfies \( S_{\omega_i+1-1} \geq s_{\omega_i+1,\text{crit}} - 1 \) is observed; and
- the event that a string that satisfies \( S_{\omega_i} \geq s_{\omega_i,\text{crit}} \) is observed outside of the window where the string that satisfies \( S_{\omega_i+1} \geq s_{\omega_i+1,\text{crit}} \) is observed.

![Figure 3.1: Comparison of ratios of approximate (Glaz and Zhang) to exact (proposed) tail probabilities for the two window combination \( \{16,22\} \) and fixed window discrete scan statistic tail probabilities for each window size. The horizontal axis gives the critical value for window size 16. The solid unmarked horizontal line is plotted at ratio 1.](image)

Figure 3.1: Comparison of ratios of approximate (Glaz and Zhang) to exact (proposed) tail probabilities for the two window combination \( \{16,22\} \) and fixed window discrete scan statistic tail probabilities for each window size. The horizontal axis gives the critical value for window size 16. The solid unmarked horizontal line is plotted at ratio 1.
The reasoning for the approximation provided by Naus and Wallenstein [29] suggests that approximating $P\{S_{\omega_i} \geq s_{\omega_i, crit}|S_{\omega_i+1} \geq s_{\omega_i+1, crit}\}$ in this manner will work well when $P\{S_{\omega_i} \geq s_{\omega_i, crit}\}$ and $P\{S_{\omega_i+1} \geq s_{\omega_i+1, crit}\}$ are both small. The conditional probability $P\{S_{\omega_i} \geq s_{\omega_i, crit}| \sum_{t=1}^{\omega_i+1} X_t = s_{\omega_i+1, crit} - 1\}$ is obtained using the result given in Naus [24], which is the exact conditional probability when $s_{\omega_i, crit} > (s_{\omega_{i+1}, crit} - 1)/2$.

Figure 3.1 is a comparison plot of the ratios of approximate to the exact tail probabilities for the two window combination \{16,22\} and the tail probabilities of the fixed window discrete scan statistic for each of the window sizes. As the tail probabilities of the fixed window discrete scan statistic for window size 16 and 22 decrease, the ratios of approximate to exact tail probabilities are closer to 1. A closer look at the critical values for which the ratios of approximate to exact tail probabilities are close to one for the two window combination \{16,22\} shows that the approximation performs best in this case when the tail probabilities for the corresponding single window cases are less than .0001 (Figure 3.2).

Figure 3.3 is a comparison plot of the ratios of approximate to the exact tail probabilities for the three window combination \{16,18,22\} and the tail probabilities of the fixed window discrete scan statistic for each of the window sizes. The ratios of approximate to exact tail probabilities
are nearly 1 when all of the single window tail probabilities are extremely close to 0. A look specifically at the critical values for which the ratios of approximate to exact tail probabilities are close to one for the three window combination \( \{16,18,22\} \) shows that the approximation performs best in this case when the tail probabilities for the corresponding single window cases are less than \( 3 \times 10^{-9} \) (see Figure 3.4).

Overall the Glaz and Zhang approximation performs well when tail probabilities \( P\{S_{\omega_i} \geq s_{\omega_i,\text{crit}}\} \) for the associated window sizes and critical values are all very small. The window sizes and the probability of success \( p \) affects how small the tail probabilities must be. When the tail probabilities are all large, the performance of the approximation is unpredictable. We have observed cases like those given in Figure 3.1 where the approximation performs poorly as well as cases in which the approximations perform well (not provided).

### 3.5.1 Application to NBA Free Throw Data

Streaks of successful free throw attempts among NBA players have long garnered the attention of basketball enthusiasts. As early as the 1950-51 season of the NBA, Joe Fulks was hailed as a deadly free throw shooter for having made 49 consecutive free throw shots [34]. During the 1992-93 NBA season, Micheal Williams, then of the Minnesota Timberwolves, established
Figure 3.4: Comparison of ratios of approximate (Glaz and Zhang) to exact (proposed) tail probabilities for the two window combination \{16,18,22\} and fixed window discrete scan statistic tail probabilities for each window size. The horizontal axis gives the critical values for each window size.
the current all-time NBA record of 97 consecutive made free throws ([35], [36]). More recently, basketball enthusiasts marveled as Dirk Nowitzki made his 24th consecutive free throw breaking the previously established NBA playoff record for most consecutive free throws. Dirk went on to set a new record by continuing to make consecutive free throws for a total of 38 during that playoff season [37]. While these occurrences are noteworthy, it is often of interest to determine the probability that such occurrences could have happened by random chance.

We focused on the analysis of the 2009-10 NBA season free throw data. The data was obtained from the website www.basketballgeek.com/data. The probabilities of observing certain multiple window scan values for Kobe Bryant, Tim Duncan, and Dirk Nowitzki were computed, and the most significant clusters of made free throw shots for players with free throw percentages in specified intervals are identified. Kobe Bryant, Tim Duncan, and Dirk Nowitzki were selected based on the attention that their free throw records have received over recent years.

To determine the appropriate order of the Markov chains for analyzing the NBA free throw data, the Minimum Akaike’s Information Criterion estimate (MAICE) as proposed by Tong [38] was used. MAICE is defined as the model order \( m \) that minimizes \( R(m) = -2\{\log\left(\frac{L(\hat{\theta}_m|X)}{L(\hat{\theta}_M|X)}\right) - \left[(k^{m+1} - k^m) - (k^M+1 - k^M)\right]\}, \) where \( L(\hat{\theta}_m|X) \) is the likelihood function of the Markov chain of order \( m \), \( \hat{\theta}_m \) is the vector of maximum likelihood estimates of the \( m \)th order Markov chain, and \( M \) is the largest model order considered. For each of the selected NBA players, Table 3.1 contains \( R(m) \) for each \( m < M = 4 \). The preferred model order for each player is the one that is MAICE. Thus, the independent model is used for all of the selected players. We also used the independent model for computing probabilities to identify the most significant clusters because it is MAICE for approximately 95% of the players with free throw percentages in each of the intervals considered. Career free throw percentages were obtained from the website www.nba.com and used as the probability of making a free throw for the analyses of selected players. For all other analyses, the probability of making a free throw was estimated from the data for each player using the proportion of total made free throw attempts. Hence, the exact results given here are conditional on the estimated free throw percentages.

Table 3.1: Model fit statistics

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim Duncan</td>
<td>-7.962</td>
<td>-3.240</td>
<td>-5.595</td>
<td>-4.695</td>
</tr>
</tbody>
</table>
Table 3.2: Tail probabilities of the observed scan statistic values of select NBA players within windows \{35, 40\}

<table>
<thead>
<tr>
<th>NBA Player</th>
<th>( \hat{p} )</th>
<th>n</th>
<th>{s_{\omega_2}, s_{\omega_2}}</th>
<th>( P{\cup [S_{\omega_i} \geq s_{\omega_j}] } )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kobe Bryant</td>
<td>0.838</td>
<td>510</td>
<td>{34,39}</td>
<td>0.621</td>
</tr>
<tr>
<td>Tim Duncan</td>
<td>0.694</td>
<td>365</td>
<td>{34,37}</td>
<td>0.029</td>
</tr>
<tr>
<td>Dirk Nowitzki</td>
<td>0.879</td>
<td>581</td>
<td>{35,40}</td>
<td>0.537</td>
</tr>
</tbody>
</table>

Table 3.2 contains the career free throw percentages for each player, the number of free throw attempts by each player during the 2009-2010 NBA season, the observed scan statistic values for window sizes 35 and 40 from the data, and the corresponding tail probabilities. Although the other players selected have observed clusters with more made free throws within a window of 35 or 40, Tim Duncan is the only player with a cluster of shots that can be considered statistically significant at a level of .05 or smaller. Even the streak of 40 consecutive made free throws within a window of size 40 by Dirk Nowitski is not statistically significant because of the player’s free throw shooting ability.

Table 3.3: Most significant clusters among windows \{30, 35, 40\} for NBA players with free throw percentages in [.75,.8) for the 2009-10 regular season

<table>
<thead>
<tr>
<th>NBA Player</th>
<th>( \hat{p} )</th>
<th>n</th>
<th>{s_{\omega_1}, s_{\omega_2}, s_{\omega_2}}</th>
<th>( P{\cup [S_{\omega_i} \geq s_{\omega_j}] } )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Luis Scola</td>
<td>0.779</td>
<td>267</td>
<td>{30, 35, 39}</td>
<td>0.04</td>
</tr>
<tr>
<td>Stephen Jackson</td>
<td>0.781</td>
<td>425</td>
<td>{30, 35, 39}</td>
<td>0.03</td>
</tr>
<tr>
<td>Allen Iverson</td>
<td>0.794</td>
<td>131</td>
<td>{30, 35, 38}</td>
<td>0.19</td>
</tr>
</tbody>
</table>

We also considered the entire pool of NBA players with average free throw percentages and identified the most significant clusters made by members of this group. First, we considered all players with free throw percentages less than 80% but greater than or equal to 75%. Table 3.3 contains the estimated success probabilities, the number of free throw attempts during the 2009-2010 NBA season, the observed scan statistic values corresponding to the window combination \{30, 35, 40\}, and the corresponding tail probabilities for the most statistically significant clusters of these players with free throw percentages in the interval [.75,.8). All other players with free throw percentages in the same range had observed values for the window combination for which the tail probabilities exceeded .20. We find that only Luis Scola and
Stephen Jackson had clusters with p-values less than .05, and these are not significant after a multiple comparisons (multiple players) adjustment. We use the Bonferroni adjustment to obtain the significance level of $\alpha/56$ since there are 56 players with free throw percentages in the interval [.75,.8).

Table 3.4: Most significant clusters among windows {24, 26, 28} for NBA players with free throw percentages in [.7,.75) for the 2009-10 regular season

<table>
<thead>
<tr>
<th>NBA Player</th>
<th>$\hat{p}$</th>
<th>n</th>
<th>${s_{\omega_1}, s_{\omega_2}, s_{\omega_3}}$</th>
<th>$P{\cup [S_{\omega_i} \geq s_{\omega_k}]}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim Duncan</td>
<td>0.723</td>
<td>365</td>
<td>{24, 26, 28}</td>
<td>0.039</td>
</tr>
<tr>
<td>Michael Redd</td>
<td>0.712</td>
<td>66</td>
<td>{22, 24, 26}</td>
<td>0.129</td>
</tr>
<tr>
<td>Tyler Hansbrough</td>
<td>0.743</td>
<td>113</td>
<td>{23, 25, 27}</td>
<td>0.131</td>
</tr>
</tbody>
</table>

Then we examined the clusters of all players with free throw percentages that were at least 70% but did not exceed 75%. Table 3.4 contains the estimated success probabilities, the number of free throw attempts during the 2009-2010 NBA season, the observed scan statistic values corresponding to the window combination {28, 32, 36}, and the corresponding tail probabilities for the most statistically significant clusters of players with free throw percentages in the interval [.70,.75). All other players with free throw percentages in the same range had observed values for the window combination for which the tail probabilities exceeded .14. Only Tim Duncan has a cluster with a p-value less than .05.

Our analyses of the NBA free throw data suggest that there was not significant clustering of successful free throw shot attempts during the 2009-2010 season. MAICE was predominately order 0 corresponding to the independent model, and most clusters were not significant. However, our analyses does suggest that some noteworthy streaks could have happened by random chance.

It is tempting to conclude that these results imply that the controversial hot-hand phenomenon does not exist for free throw shooting in the NBA. However, we refrain from making such a conclusion because we have not conducted a formal test of this hypothesis. To reject the general notion that there is a hot-hand phenomenon in free throw shooting in the NBA is too broad of a conclusion based on such a small case; however, our method is useful for analyzing the specific cases of determining whether a specific cluster is significant or not.
3.6 Summary

In this chapter, we present an efficient method for computing the distribution of the discrete multiple window scan statistic for higher-order, multi-state Markovian trials. We describe how to generate the state space for computing the tail probabilities of the statistic. This description is also useful in the special case of the discrete multiple window scan statistic for bi-state, first-order Markovian trials and Bernoulli trials.

Glaz and Zhang [14] provided very good approximations for the statistic in the case of Bernoulli trials. However, our work has demonstrated that these approximations do not perform as well when the probabilities of the associated single window cases are not very small. The method provided in this work proves useful both in cases where the single window probabilities are small and in cases where they are not. The method is applied to examining the statistical significance of NBA free throw streaks.

Computing the distribution of the multiple window scan statistic for higher-order, multi-state Markovian trials is a computationally intensive task. Results that use the development of a Markov chain while minimizing the size of the state space reduce the inherent computational burden associated with this problem. In future work, we will develop an optimal state space for computing the distribution by finding the minimal deterministic finite automaton associated with the problem.
Chapter 4

A Bayesian False Discovery Approach to Syndromic Surveillance

4.1 Introduction

Syndromic surveillance systems are important for public health because they aid in the early identification of an increase in individuals exhibiting a particular set of symptoms such as nausea, fever, and rash within a particular sub-region (see [39]). When such an increase is greater than would have occurred by random chance, the early detection of that increase could result in the saving of lives and natural resources. Some current syndromic surveillance systems are the Real-time Outbreak and Disease Surveillance (RODS) system, BioSense, and the North Carolina Disease Event Tracking and Epidemiologic Collection Tool (NC DETECT) [40]. A major challenge in the development of syndromic surveillance systems is maximizing the ability to quickly detect abnormal increases in illness among individuals within a particular sub-region, called outbreaks, while minimizing the number of false alarms produced. Public health officials are interested in early detection to minimize the effects of an outbreak on the general public. However, they are also interested in maintaining a low false-positive rate to maximize the use of monetary and human resources.

Spatial scan statistics are commonly used to detect outbreaks (see [41]). A spatial scan statistic is a likelihood ratio test statistic for testing the null hypothesis of a uniform disease rate over an entire spatial region of interest, $\mathcal{G}$, versus the alternative hypothesis that there exists a sub-region of $\mathcal{G}$ for which the disease rate is higher inside that sub-region than anywhere else within the region $\mathcal{G}$. In practice, the spatial scan statistic is computed by computing the value of the statistic for each subregion of $\mathcal{G}$ with the limitation that no subregion larger than half the size of $\mathcal{G}$ is considered. The maximum of the collection of statistics obtained is the spatial scan statistic on which testing is based. Neill [42] noted that the process of searching for the
value of the statistic can be computationally expensive requiring many days to run a single analysis. Statistical significance testing based on the spatial scan statistic is conducted using randomization testing because the closed form of the distribution of the spatial scan statistic cannot be computed. Typically, the subregions of $G$ to be considered are chosen based on a predetermined shape, such as a circle, in which a specified identifier for a location like a centroid must belong in order to be included in the subregion. In his foundational paper, Kulldorff [15] proposed a spatial scan statistic for independently distributed Bernoulli or Poisson data using circular subregions. He established that the test based on the statistic is most powerful with respect to specific partitions of the critical region given that the data followed either model. However, Loh and Zhu [43] showed that using the spatial scan statistic proposed in Kulldorff [15] leads to increased false alarms as the correlation between data increases.

A variety of extensions of the spatial scan statistic have been proposed. Many of these extensions have been to include methodology for data assumed to follow various models. For instance, Huang, Kulldorff, and Gregorio [44] proposed the spatial scan statistic for survival data; Jung, Kulldorff, and Klassen [45] extended the spatial scan statistic to ordinal data; Huang et. al. [46] proposed the spatial scan statistic for weighted normally distributed data; and Jung, Kulldorff, and Richard [47] extended the spatial scan statistic to multinomial data. Some other extensions have been to expand on the type of clusters that could be detected. For example, Kulldorff et. al. [48] extended the spatial scan statistic to the space-time domain; Kulldorff et. al. [49] proposed an elliptic spatial scan statistic to allow clusters of an elliptical shape to be detected; and Tango and Takahashi [50] provided methodology to extend the spatial scan statistic to non-elliptical regions. Nearly all of the extensions of the spatial scan statistic require searching over many subregions for the value of the test statistic and then using randomization testing to determine its significance. However, Neill, Moore, and Cooper [30] proposed a Bayesian spatial scan statistic that used a more efficient search algorithm for the test statistic and did not require the use of randomization testing to implement outbreak detection. Their extension of the spatial scan statistic like others is for data that is assumed to be independently distributed under a specified model. To date, the only extension of the spatial scan statistic that embodies relaxing the independence assumption is the methodology provided by Loh and Zhu [43]. They proposed a modified spatial scan statistic that accounts for spatial correlation by adding to the original spatial scan statistic an additional level of randomization testing based on a correlated model. While this approach decreased the number of false alarms, it contains an increase in computational intensity. As an alternative to randomization testing, we could perform several tests, one for each subregion of $G$; however, this approach would result in a loss of power that is typically associated with the conservative adjustments that are necessary to control Type I errors in multiple testing.

Multiple testing is a crucial issue in surveillance because tests are conducted for many regions
and many time points. Much has been done to control the errors associated with multiple testing while maintaining suitable detection power. Early methods to control multiple testing error focused on controlling the familywise error rate (FWER), which is the probability that at least one of the null hypotheses is incorrectly rejected. Although methods used in multiple testing to control the FWER are typically simple to implement, they are implemented at the expense of a substantial loss of power as compared to performing a single test for each null hypothesis (see Farcomeni [51]).

To overcome this limitation, Benjamini and Hochberg [52] introduced an alternative error measure for multiple testing called the false discovery rate (FDR), which is the expected ratio of incorrectly rejected null hypotheses to rejected null hypotheses. Benjamini and Hochberg [52] provided a simple procedure to perform multiple testing that would control the FDR if the test statistics are independent. Sarkar [53] and Benjamini and Yekutieli [54] proved that the procedure of Benjamini and Hochberg [52] also controlled the FDR under various dependency structures for the test statistics. Sun and Cai [55] proposed an alternative procedure to control FDR under the hidden Markovian dependence structure. Their approach involved modeling the correlation within the data and computing a statistic that exploited the correlation within the data for multiple testing rather than p-values. Sun et. al. [16] extended the procedure of Sun and Cai [55] to control FDR under spatial dependence. Their approach involved using the Bayesian modeling framework to model the spatial dependency within the data and using a statistic that exploited the correlation within the data for multiple testing.

Recently, Bayesian methods have been proposed to control FDR. Storey [56] noted that the FDR does not include any information that the data may provide about the number of null hypotheses that are actually true. Efron and Tibshirani [57] proposed the Bayesian false discovery rate (BFDR), which depends on the posterior distribution of the null hypothesis given the data. Bogdan et. al. [58] showed that methods that controlled the BFDR also controlled the FDR when the number of true null hypotheses was small relative to the number of hypothesis tests performed.

In this chapter, we propose a BFDR approach to outbreak detection using syndromic surveillance data. Here we provide a procedure for detecting outbreak days based on syndromic surveillance data by extending the methodology proposed by Sun et. al. [16] in which a Bayesian modeling framework is used to improve on the computation of a test statistic for rejecting the collective null hypothesis of no outbreak over a region. The methodology of Sun et. al. [16] hinges on using MCMC samples, generated from the posterior distribution of an appropriate Bayesian model in which the spatial dependency of the data is captured, to carry out multiple testing. Their procedure involves selecting the cutoff value for rejection based on controlling the BFDR. However, they show that their methods also control the FDR and maintains suitable power. We extend the methodology of Sun et. al. [16] to outbreak detection using syndromic
surveillance data by obtaining posterior samples from an appropriate Bayesian model that captures the spatial dependency of the data and considers all days of interest, computing a test statistic based on a measure of the magnitude of the maximum cluster for each day, and selecting a cutoff value that controls the BFDR for rejecting the collective null hypothesis of no outbreak over a collection of days for a specified region.

The BFDR approach to syndromic surveillance is described in Section 4.2. Numerical results that compare the performance of our approach to the spatial scan statistic methodology are given in Section 4.3. In Section 4.4, we apply this methodology to syndromic surveillance data collected via NC DETECT. A summary is provided in the final section.

4.2 BFDR Procedure for Outbreak Detection

In general, our BFDR procedure for detecting outbreaks involves the following steps: (i) selecting an appropriate Bayesian model that captures the spatial dependency inherent to the data; (ii) choosing a measure of the magnitude of the maximum spatial cluster on each day; (iii) selecting a test statistic based on our daily maximum cluster measure; and (iv) identifying the largest cutoff value for the test statistic that corresponds to a BFDR near some pre-specified value $\alpha$.

We begin by describing the Bayesian model. Then we define the test statistic that we use as a measure of the magnitude of the maximum spatial cluster for each day. Finally, we provide the general format of our test statistic, decision rule, and the BFDR in terms of our test statistic and decision rule.

An inherent feature of the count data typically used for detecting outbreaks is its spatial dependence, which is based on the spatial proximity of the areas within which the counts are obtained. Many of the current methods for Bayesian hierarchical models for disease data are summarized in the extensive text of Lawson [59] and references therein. Let $\{A_1, A_2, \ldots, A_n\}$ be a partition of the region $G$ (e.g., the counties of North Carolina) and $Y_{it}$ be the number of cases in sub-region $A_i$ on day $t$. Typically, aggregated disease count data is assumed to follow a Poisson distribution if the disease count is relatively low and the population is relatively large in a small area [59]. Let $Y_{it} \sim \text{Poisson}(E_{it}\theta_{it})$, where $E_{it}$ is the expected count for $A_i$ on day $t$ and $\theta_{it}$ is relative risk for $A_i$ on day $t$. For example, $E_{it}$ may be the population size or the mean estimated from historical data. A common prior for $\gamma_{it} = \log(\theta_{it})$ used to capture spatial dependency in disease mapping is the multivariate normal model with some specified correlation matrix structure (see Lawson [59]). Let the collection of all log relative risks on day $t$ be modeled as $\gamma_t = (\gamma_{1t}, \gamma_{2t}, \ldots, \gamma_{nt})^T \sim MVN(\mu_t, \mathbf{1}\sigma_t^2\Sigma)$, where $\mu_t \sim N(\mu, \tau)$, $\mathbf{1}$ is a vector of ones, $\sigma_t^2 \sim \text{Inv-Gamma}(a,b)$, $\Sigma$ is the covariance matrix of $\gamma_t$ with the $\text{Cov}(\gamma_{it}, \gamma_{jt}) = \exp(-\phi D_{ij})$, $\mu \sim N(0,10)$, $\tau \sim \text{Inv-Gamma}(0.1,0.1)$, $a \sim U(0,25)$, $b \sim \text{Gamma}(0.1,0.1)$, $D_{ij}$ represents the distance between region $i$ and region $j$, and the spatial parameter $\phi$ represents the rate of
decline of correlation with distance. For the hyperparameters, the values are chosen so that the variances are large.

We choose a test statistic that exploits the natural neighborhood structure between areas. Let $A_i$ and $A_j$ be two distinct subregions of $G$. Sub-region $A_i$ is a first-order neighbor of sub-region $A_j$ if $A_i$ and $A_j$ share a border. Given that sub-region $A_i$ is not a first-order neighbor of sub-region $A_j$, we say that sub-region $A_i$ is a second-order neighbor of sub-region $A_j$ if there exists a first-order neighbor of $A_i$ that is also a first-order neighbor of $A_j$. In general, $A_i$ is a $k^{th}$-order neighbor of $A_j$ if they are not $(k-1)$-order neighbors and they share a neighbor of the $(k-1)$-order. We define the $k^{th}$-order neighborhood of $A_i$ to be the sub-region $A_i$ and all of its neighbors of orders 1 up to $k$. We measure the magnitude of the cluster centered on $A_i$ by a weighted mean relative risk defined as follows:

$$
\theta^k_{it} = \sum_{j \in G^k_i} \frac{\theta_{jt}}{|G^k_i|},
$$

where $G^k_i$ is the $k^{th}$-order neighborhood of $A_i$ and $|G^k_i|$ is the number of elements in $G^k_i$. The magnitude of the maximum spatial cluster on day $t$ is the maximum of the collection of weighted means over all sub-regions, that is, $M_t = \max\{\theta^1_{1t}, \theta^2_{2t}, \ldots, \theta^k_{nt}\}$. If $M_t$ exceeds a threshold of $r$, then we say that an outbreak has occurred on day $t$. For each day, we test the null hypothesis $H_0 : M_t < r$ versus the alternative $H_A : M_t > r$. Our test statistic is the posterior probability of the null hypothesis, $P_t = P(M_t < r|Y)$, where $Y$ is all observed data. Ideally, $r$ should be chosen to represent a meaningful outbreak size. An expert would choose $r$ based on their previous knowledge of the syndrome on which the outbreak detection is based.

Our decision rule is based on thresholding the posterior probability $P_t$. Let $T$ denote the threshold. We define a decision rule $\delta_t$ that denotes our declaration of whether an outbreak has occurred or not on day $t$ as follows:

$$
\delta_t = \begin{cases} 1 & P_t < T, \\ 0 & \text{otherwise}, \end{cases}
$$

where $\delta_t = 1$ denotes an outbreak on day $t$ and $\delta_t = 0$ denotes no outbreak on day $t$. We select $T$ so that

$$
\text{BFDR} = \frac{\sum \delta_t P_t}{\sum \delta_t} \approx \alpha. \quad (4.1)
$$

The denominator of (4.1) is the number of discoveries declared by the procedure; the numerator is the posterior mean number of these discoveries that are truly null. Therefore, (4.1) provides an intuitive Bayesian estimate of the FDR for the decision rule determined by $T$. 

41
4.3 Simulation Study

To illustrate the usefulness of our proposed approach, we simulated syndromic surveillance data with injected outbreaks and compared the performance of our approach to the performance of the popular spatial scan statistic. As measures of performance, we report the FDR and the true discovery rate (TDR), i.e., the expected proportion of the outbreak days that are flagged, for each method under the various conditions. We used the counties of North Carolina as regions.

We generated data according to the model $Y_{it} \sim \text{Poisson}(E_{it} \Delta_{it} \theta_{it})$, where $E_{it}$ is the expected count for county $i$ on day $t$, $\Delta_{it}$ is the change in the count that characterizes an outbreak day for county $i$ on day $t$, and $\theta_{it}$ is relative risk for county $i$ on day $t$, to represent syndromic surveillance counts for the 100 counties of North Carolina over a 30-day period. To obtain realistic estimates of expected daily syndromic count data, we use the total number of botulism-like syndrome cases reported by participating emergency departments in North Carolina via NC DETECT from January 1, 2008 to December 31, 2009 and the population estimates for each county in 2008 and 2009 obtained from the website of the North Carolina office of state and budget management [60]. Let $E_{iT} = n_{iT} Y_{++}$, where $n_{iT}$ is the population estimate for county $i$ in year $T$, $Y_{++}$ is the sum of the number of botulism-like syndrome cases obtained by NC DETECT in 2008 and 2009, and $n_{++}$ is the sum of 366 times the sum of the population estimates of all counties in 2008 and 365 times the sum of the population estimates of all counties in 2009. We fix the estimate of $E_{it}$ for each county across all days at the value obtained by averaging $E_{iT}$ for 2008 and 2009.

We consider a seven-day outbreak condition over a set of selected counties and vary the value of the constant change in count, $\Delta$. For each simulated dataset, the seven-day outbreak begins on day 19 and the chosen outbreak counties are Alamance, Alexander, Alleghany, Anson, Ashe, Avery, Beaufort, Bertie, and Bladen. The outbreak counties are shaded in Figure 4.1. These counties represent increases in a single county that can lead to increases in surrounding counties. Typically, we are interested in detecting changes before clusters involve more than two neighboring counties. The constant change outbreak is simulated by setting a fixed value of $\Delta$ for all outbreak counties over the specified outbreak days. We choose count change values of 3, 5, and 7 to represent small, medium, and large constant outbreaks, respectively. We vary the spatial parameter $\phi$ from 0.1 to 0.3. This allows the range of correlations to be between 0.07 and 0.41 at the minimum distance and approximately 0 at the maximum distance.

We use the Metropolis-Hastings algorithm for posterior sampling. For each dataset, we generate 5000 posterior samples and discard the first 999 as burn-in. All computations are done in R. We choose the typical significance level of 0.1 at which to control the BFDR.

For each combination of constant change in count and value of $\phi$, we simulated 100 data sets. Table 4.1 contains the FDR and TDR results for the BFDR approach and the scan statistic.
methodology for the various outbreak conditions when the test statistic is the maximum mean first-order neighborhood posterior probability for day $t$ and the significance level of the spatial scan statistic is 0.002. The significance level of 0.002 is used rather than the typical 0.05 because it is the level at which the FDR is controlled at 0.1. In Table 4.1, we see that while both methods control FDR at significance level 0.1 well, our approach outperforms the spatial scan statistic methodology in detection power. Here choosing the threshold $T$ so that BFDR is as close to 0.1 as possible without exceeding it leads to FDR values that are much lower than 0.1. Hence, the BFDR approach is conservative. The TDR for BFDR approach is a little more than double the TDR for the spatial scan statistic methodology for the largest change in count over outbreak counties, $\Delta = 5$. Note that the detection power increases as the outbreak signal increases for both methods as expected.

Table 4.2 contains the FDR and TDR results for the BFDR approach and the scan statistic methodology for the various outbreak conditions when the test statistic is the maximum mean second-order neighborhood posterior probability for day $t$ and the significance level of the spatial scan statistic is 0.002. The performance of our approach under this neighborhood structure is also good. For the cases where the constant change in count over outbreak counties is medium and large, our method outperforms the scan statistic methodology in detection power.

4.4 Application to NC DETECT data

Public health surveillance in North Carolina is conducted by the North Carolina Department of Human and Health Services/Division of Public Health (NC DHHS/DPH) NC DETECT
Table 4.1: False Discovery Rate (FDR) and True Discovery Rate (TDR) results for the BFDR approach and the spatial scan statistic methodology under various outbreak conditions, defined by the constant change in count \( \Delta \) for outbreak counties and the spatial correlation \( \phi \), when the test statistic is the maximum mean first-order neighborhood posterior probability for day \( t \), \( r \) is 5, and the significance level of the spatial scan statistic is 0.002.

<table>
<thead>
<tr>
<th>( \Delta )</th>
<th>( \phi )</th>
<th>FDR</th>
<th>TDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>BFDR</td>
<td>0.023</td>
<td>0.015</td>
</tr>
<tr>
<td></td>
<td>Scan Statistic</td>
<td>0.132</td>
<td>0.117</td>
</tr>
<tr>
<td>4</td>
<td>BFDR</td>
<td>0.043</td>
<td>0.033</td>
</tr>
<tr>
<td></td>
<td>Scan Statistic</td>
<td>0.101</td>
<td>0.102</td>
</tr>
<tr>
<td>5</td>
<td>BFDR</td>
<td>0.068</td>
<td>0.051</td>
</tr>
<tr>
<td></td>
<td>Scan Statistic</td>
<td>0.093</td>
<td>0.081</td>
</tr>
</tbody>
</table>

system. NC DETECT is an advanced, statewide public health surveillance system. NC DETECT is funded with federal funds by North Carolina Division of Public Health (NC DPH), Public Health Emergency Preparedness Grant (PHEP), and managed through a collaboration between NC DPH and the University of North Carolina at Chapel Hill Department of Emergency Medicine’s Carolina Center for Health Informatics (UNC CCHI). The NC DETECT Data Oversight Committee (DOC) does not take responsibility for the scientific validity or accuracy of methodology, results, statistical analyses, or conclusions presented. NC DETECT DOC includes representatives from the NC DPH, UNC NC DETECT Team and NC Hospital Association. With NC DETECT, CCHI monitors 10 syndromes: botulism-like, Center for Disease Control Influenza-like illness (CDC ILI), Emergency Medical Services Transport 65 and over (EMS Transport 65+), Fever Rash, Gastrointestinal-all (GI), Gastrointestinal-severe (GI Severe), Influenza-like illness (ILI), Meningoencephalitis (MenEnc), Respiratory-all, and Respiratory. Each syndrome is characterized by a set of symptoms.

For each patient visiting an emergency department (ED) of a participating hospital, both demographic and medical data are obtained [61]. A specified subset of NC DETECT data is available by request after the approval of the NC DETECT DOC. The demographic data obtained include elements such as age, gender, city, zip code, county, arrival date, and arrival time. Medical data collected includes hospital, insurance coverage, chief complaints, triage notes, and initial temperatures. A patient is recognized as presenting with a particular syndrome if the symptoms captured in the patient’s chief complaints, triage notes, or initial body temperatures correspond with the symptoms of that syndrome [62]. A patient may be classified as presenting with multiple syndromes each of which is coded as a binary variable (yes/no).
Table 4.2: False Discovery Rate (FDR) and True Discovery Rate (TDR) results for the BFDR approach and the spatial scan statistic methodology under various outbreak conditions, defined by the constant change in count $\Delta$ for outbreak counties and the spatial correlation $\phi$, when the test statistic is the maximum mean second-order neighborhood posterior probability for day $t$, $r$ is 4.5, and the significance level of the spatial scan statistic is 0.002.

<table>
<thead>
<tr>
<th>$\Delta$</th>
<th>$\phi$</th>
<th>FDR</th>
<th>TDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.013</td>
<td>0.021</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.117</td>
<td>0.023</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.117</td>
<td>0.026</td>
</tr>
<tr>
<td></td>
<td>BFDR</td>
<td>0.015</td>
<td>0.017</td>
</tr>
<tr>
<td></td>
<td>Scan Statistic</td>
<td>0.117</td>
<td>0.027</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.016</td>
<td>0.140</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.102</td>
<td>0.104</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.102</td>
<td>0.084</td>
</tr>
<tr>
<td></td>
<td>BFDR</td>
<td>0.006</td>
<td>0.104</td>
</tr>
<tr>
<td></td>
<td>Scan Statistic</td>
<td>0.102</td>
<td>0.081</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.030</td>
<td>0.374</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.081</td>
<td>0.354</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.081</td>
<td>0.361</td>
</tr>
<tr>
<td></td>
<td>BFDR</td>
<td>0.029</td>
<td>0.214</td>
</tr>
<tr>
<td></td>
<td>Scan Statistic</td>
<td>0.081</td>
<td>0.200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.201</td>
<td>0.201</td>
</tr>
</tbody>
</table>

As an application of our procedure, we analyze daily botulism-like syndrome ED count data collected by NC DETECT from January 1, 2008 to December 31, 2010. According to Kaydos-Daniels et. al. [62], the botulism-like syndrome is characterized by the appearance of symptoms such as vomiting, blurred vision, and headaches. The data provided by NC DETECT consisted of gender, age group, county, date, time block, and a (0/1) indicator of syndrome classifications for each patient visiting a participating ED during the specified time period. The data did not include any identifying information. We extracted county, date, time block, and (0/1) botulism-like classification indicators. After summing over the 4 hour time blocks that started at midnight for each day by county, the data set used in our analysis consist of a total of 33,785 cases of botulism-like syndrome partitioned by county and day of occurrence. Plots of total counts by county and day for Dare, Polk, Richmond, and Wake Counties are given in Figure 4.2. These counties are selected to demonstrate counts in counties of varying size and in various locations of the state. As expected, botulism-like syndrome counts are higher in more highly populated counties like Wake County. Population estimates for all NC counties for the years 2008-2010 were obtained from the website of the office of state and budget management. The average of the population size estimates for each county from 2008-2010 are used (see Figure 4.3).

For analysis, we use the same model, priors, and algorithm as in the simulation study. Here we let $r = 2$, which corresponds to an outbreak being detected as a two-fold increase in RR compared to the background. In Figure 4.4, we provide a plot of the posterior probabilities of no outbreak by day. The outbreak days are signified by a red solid circle. There are four outbreak days: July 14, 2009 (day 560), July 31, 2009 (day 577), July 28, 2010 (day 938), and September 21, 2010 (day 994). These days have posterior probabilities of no outbreak of 0.026, 0.133, 0.08,
Figure 4.2: Total daily counts of emergency department (ED) visits classified as exhibiting symptoms associated with the botulism-like syndrome for Dare, Polk, Richmond, and Wake Counties from January 1, 2008 to December 31, 2010.
and 0.134, respectively.

We also considered two smaller values of $r$, 1.25 and 1.5. The number of outbreak days increased as the value of $r$ decreased. For values of $r$ greater than 2, there were no outbreak days. The spatial scan statistic detects 27 outbreaks days that have a p-value of .002 or less. None of the days identified by the spatial scan statistic coincided with those identified by the BFDR approach although plots of the adjusted count values for the outbreaks days identified by the BFDR approach suggest that they should have been (see Figure 4.5).

### 4.5 Summary

In this chapter, we present a BFDR approach to detecting outbreaks. We describe a multiple testing procedure that controls the BFDR over a spatial region for a collection of days while maintaining suitable power for detecting outbreak locations. We show via simulation that the procedure is able to control FDR even in the presence of strong spatial dependence between counts. The simulation study also shows that the method has more power than the spatial scan statistic in the presence of dependent data.

The spatial scan statistic originally proposed in Kulldorff [15] and various extensions of it are commonly used to detect outbreaks. These methods perform well when the observed counts
Figure 4.4: Posterior probabilities of no outbreak by day from analyzing botulism-like syndrome count data collected by NC DETECT. Red solid (black cross) circles represent outbreak (non-outbreak) days.
Figure 4.5: Plots of the adjusted counts for the days designated as outbreaks by the BFDR procedure. The upper left corner plot is the plot for July 14, 2009, the upper right corner plot is the plot for July 31, 2009, the lower left corner plot is the plot for July 28, 2010, and the remaining plot is for September 21, 2010. Counties in black (gray) have adjusted count $Y_{it}/E_{it}$ greater than (less than) 2.

at various locations can be assumed to be uncorrelated; however, this is typically not the case for syndromic data. In our work, we have demonstrated that our methodology outperforms the spatial scan statistic when there is spatial correlation in the data. Our procedure is applied to analyzing botulism-like syndrome count data collected by NC DETECT.

The early detection of outbreaks is not a trivial task and inherently involves multiple testing. Public health officials are interested in methods that lead to the early detection of outbreaks and reduce the number of false alarms. In future work, we will investigate the performance of the BFDR procedure for an oracle model to determine the impact estimated parameters may have on FDR. We will also add temporal correlation to the model. The same BFDR post-processing procedure would still apply, however, the computational burden would be far greater.
REFERENCES


