

A COMPREHENSIVE PROCESS FOR SPATIOTEMPORAL ANALYSIS OF
NETWORK-BASED PHENOMENA

by

David C. Eckley
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Geographic and Cartographic Sciences

Committee:

Kevin M. Curtin

Dr. Kevin Curtin, Chair

Nigel Waters

Dr. Nigel Waters, Committee
Member

Andrew G. Loerch

Dr. Andrew Loerch, Committee
Member

P. Agouris

Dr. Peggy Agouris,
Department Chairperson

Richard Diecchio

Dr. Richard Diecchio, Associate
Dean for Academic and Student
Affairs, College of Science

Vikas Chandhoke

Dr. Vikas Chandhoke, Dean,
College of Science

Date: 3 December 2010

Fall Semester 2010
George Mason University
Fairfax, VA

A Comprehensive Process for Spatiotemporal Analysis of Network-Based Phenomena

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Geographic and Cartographic Sciences at George Mason University

By

David C. Eckley
Bachelor of Science
United States Military Academy, 2000

Director: Dr. Kevin Curtin, Associate Professor
Department of Geography and Geoinformation Science

Fall Semester 2010
George Mason University
Fairfax, VA

Copyright: 2010 David C. Eckley
All Rights Reserved

Dedication

This work is dedicated to the American Soldier. The research enclosed is focused on developing methods that can identify and defeat the network-based attacks of our adversary.

Acknowledgements

“Whatever you do, work at it with all your heart, as working for the Lord, not for men.”
Colossians 3:23

This work is for Him.

Cathryn, what a tremendous sacrifice you have made these past months, and what grace you have demonstrated managing our home, caring for our boys, battling morning sickness, and loving me. None of these words would be on paper without you. I love you.

Ezra, Micaiah, and John, you will never know how much your welcome home greetings mean to me.

Dad and Mom, the discipline and independence you instilled in me continue to drive my pursuit of excellence in the work I’m given to do. Dad, I greatly admire your mastery of the English language and appreciate your help in editing this manuscript.

Dr. Curtin, your mentorship and subtle encouragement helped me set the bar very high in this academic pursuit. Your passion for geography and GIS is contagious and has motivated me to do my best work.

Dr. Waters, Dr. Loerch, thank you for your expertise and validation of this research.

JIEDDO Research Team, your work has amazed and inspired me to pursue this thesis. Thank you for your dedication to the American Service Member.

Nancy, your kindness and professionalism are admirable. Thank you for introducing me to the power of Python.

Dr. Leslie, Adobe Illustrator has opened up a whole new world of opportunities for me.

Several network-based analyses implemented in this study utilized the SANET toolbox developed by Dr. Okabe, Dr. Okunuki and Dr. Shiode. The author is grateful for their provision of these tools and their continuing contributions to network-based analysis.

Table of Contents

	Page
List of Tables	vii
List of Figures	viii
List of Abbreviations/Symbols	x
Abstract	xi
Forward	1
Introduction	4
What Is a Spatiotemporal Cluster?	4
Why Conduct Spatiotemporal Cluster Analysis?	5
The Issue of Spatial and Temporal Measures.....	5
Global vs. Local Statistics	7
The Origin of Spatiotemporal Cluster Analysis	8
The Development of Spatiotemporal Clustering Methods and Current Applications...	10
GIS Applications	11
Significance of this Research	12
Thesis Structure	13
PART 1 A Process for Investigating Spatiotemporal Clustering of Network-Based Phenomena.....	14
Introduction	14
Step 1: Define a Research Problem and Analytical Approach.....	16
Step 2: Acquire and Evaluate the Data.....	17
Step 3: Pre-process the Data.....	19
<i>Format and Map the Data</i>	19
<i>Examine the Data</i>	22
Step 4: Test for Spatial Clustering.....	26
<i>Continuous Space Tests</i>	27
<i>Network Space Tests</i>	30
Step 5: Test for Temporal Clustering	36
Step 6: Test for Spatiotemporal Clustering	40

Step 7: Explain the Results	45
Conclusions	47
PART 2 An Examination of Significance Tests and Critical Parameters for Network- Based Spatiotemporal Cluster Analysis.....	48
Introduction	48
Study Areas and Datasets	50
Significance Tests for the Knox Method.....	52
<i>Chi-square and Poisson Distributions</i>	52
<i>Normal Distribution</i>	55
<i>Monte Carlo Simulations</i>	56
<i>Multiple Testing</i>	58
Knox Method Critical Parameters	60
Conclusions	63
PART 3 <i>SCAN</i> : A Spatiotemporal Analysis Tool for Networks.....	64
Introduction	64
Program Requirements	65
Tool 1: ST Cluster Basic	66
Tool 2: ST Cluster Automatic	68
Tool 3: ST Cluster Table	71
Tool 4: ST Cluster Monte Carlo.....	73
Tool 5: ST Cluster Range Detector	75
Program Limitations	76
Future Program Developments.....	77
Recommendations for Future Research	78
APPENDIX <i>SCAN</i> Python Scripts	81
Tool 1: ST Cluster Basic	81
Tool 2: ST Cluster Automatic	94
Tool 3: ST Cluster Table	110
Tool 4: ST Cluster Monte Carlo.....	117
Tool 5: ST Cluster Range Detector	125
References.....	132

List of Tables

Table	Page
<i>Table 1- 1.</i> Redundancy of attribute labels in traffic collision data for Franklin County, OH, January-March 2009.....	22
<i>Table 1- 2.</i> Continuous and network average nearest neighbor test results for traffic collisions in Franklin County, OH, January-March 2009.....	35
<i>Table 1- 3.</i> Temporal interval analysis for traffic collisions in Franklin County, OH, January-March, 2009.....	38
<i>Table 1- 4.</i> Results for two linear nearest neighbor statistics given traffic fatalities in Fairfax County, VA, 2004-2008.....	39
<i>Table 1- 5.</i> Spatiotemporal clusters, Knox R, for the given spatial and temporal critical distance ranges and associated statistical significance for traffic collisions in Franklin County, OH, January-March, 2009. Highlighted values are significant where $Q \leq$ the Bonferroni correction for $\alpha = 0.05$	43
<i>Table 1- 6.</i> Selected attribute values for traffic collisions contributing to spatiotemporal clusters in Figure 1-9. Weather attribute was derived from www.wunderground.com (2009).....	45
<i>Table 2- 1.</i> Characteristics of datasets used in Part 2 of this study.....	52
<i>Table 2- 2.</i> Comparison of probabilities for the observed Knox statistic given, the chi-square distribution, the normal distribution, and those distributions depicted in Figures 2-3 and 2-4. *The space shuffled distribution is recommended and used in the tool described in Part 3.....	58
<i>Table 2- 3.</i> Spatiotemporal clusters, Knox R, for the given spatial and temporal critical distance ranges and associated statistical significance for traffic collisions in Franklin County, OH, January-March, 2009. Highlighted values are significant where $Q \leq$ the Bonferroni correction for $\alpha = 0.05$	59
<i>Table 2- 4.</i> Results of nearest neighbor distance cluster analysis for both the spatial and temporal dimensions of the given datasets.	61
<i>Table 2- 5.</i> Comparison of the Knox statistic and associated probabilities calculated using the minimum nearest neighbor distance in space and time as the critical parameters. Highlighted values indicate $Q \leq \alpha = 0.05$	62
<i>Table 2- 6.</i> Comparison of the Knox statistic and associated probabilities calculated using the average nearest neighbor distance in space and time as the critical parameters. Highlighted values indicate $Q \leq \alpha = 0.05$	62
<i>Table 2- 7.</i> Comparison of the Knox statistic and associated probabilities calculated using the maximum nearest neighbor distance in space and time as the critical parameters. Highlighted values indicate $Q \leq \alpha = 0.05$	62

List of Figures

Figure	Page
<i>Figure 1.</i> Graphical representation of the distance measurement differences between two points measured in continuous and network space.....	7
<i>Figure 1- 1.</i> Map of 586 injury-causing traffic collisions on major roads in Franklin County, OH from January to March 2009.....	21
<i>Figure 1- 2.</i> Distribution of spatial distance between pairs of traffic collisions in Franklin County, OH, January-March 2009.	23
<i>Figure 1- 3.</i> Distribution of temporal distance in days between pairs of traffic collisions in Franklin County, OH, January-March 2009.....	24
<i>Figure 1- 4.</i> Distribution of temporal distance in hours between pairs of traffic collisions in Franklin County, OH, January-March 2009.....	24
<i>Figure 1- 5.</i> Number of traffic collisions occurring by day of the week in Franklin County, OH, January-March 2009.	25
<i>Figure 1- 6.</i> Number of traffic collisions occurring by day in Franklin County, OH, January-March 2009.....	25
<i>Figure 1- 7.</i> Number of traffic collisions occurring by hour of the day in Franklin County, OH, January-March 2009.....	26
<i>Figure 1- 8.</i> SANET's Network Nearest Neighbor Distance Tool results.	34
<i>Figure 1- 9.</i> Map of traffic collisions contributing to spatiotemporal clusters defined by a spatial critical distance of 400 meters and a temporal critical distance of 0 hours (occurred during the same hour) in Franklin County, OH, January-March, 2009.....	44
<i>Figure 2- 1.</i> Fatality-causing traffic collisions on major roads in Fairfax County, Virginia, 2004-2008.....	51
<i>Figure 2- 2.</i> Comparison of the reference distribution generated by 1000 Monte Carlo simulations of spatiotemporal clusters in traffic collisions in Franklin County, OH, January-March 2009, where $\delta = 400$ meters and $\tau = 2$ hours. The Poisson distribution is generated from the reference distribution mean of 4.24.	54
<i>Figure 2- 3.</i> Difference in reference distributions and probabilities generated by 1000 Monte Carlo simulations of the Knox statistic for traffic collisions in Franklin County, OH, January-March 2009, where $\delta = 400$ meters, $\tau = 2$ hours, and $R = 8$	57
<i>Figure 2- 4.</i> Difference in reference distributions and probabilities generated by 10,000 Monte Carlo simulations of the Knox statistic for traffic collisions	

in E. Fairfax County, VA, 2004-2008 where $\delta = 1214$ meters, $\tau = 7$ days, and $R = 1$	57
<i>Figure 3- 1.</i> User input screen for <i>SCAN</i> 's ST Cluster Basic.....	67
<i>Figure 3- 2.</i> The output dialogue screen for <i>SCAN</i> 's ST Cluster Basic.	68
<i>Figure 3- 3.</i> The user input screen for <i>SCAN</i> 's ST Cluster Automatic.	69
<i>Figure 3- 4.</i> Output dialogue for <i>SCAN</i> 's ST Cluster Automatic.....	70
<i>Figure 3- 5.</i> Input screen for <i>SCAN</i> 's ST Cluster Table.....	71
<i>Figure 3- 6.</i> Output dialogue for <i>SCAN</i> 's ST Cluster Table.	72
<i>Figure 3- 7.</i> Input screen for <i>SCAN</i> 's ST Cluster Monte Carlo.	73
<i>Figure 3- 8.</i> Output dialogue for <i>SCAN</i> 's ST Cluster Monte Carlo.	74
<i>Figure 3- 9.</i> User input screen for <i>SCAN</i> 's ST Cluster Range Detector	75
<i>Figure 3- 10.</i> Output dialogue for <i>SCAN</i> 's ST Cluster Range Detector.....	76

List of Abbreviations/Symbols

α	Probability level at which a statistic is considered significant
δ	Critical spatial distance used in determining the Knox statistic
EDA	Exploratory Data Analysis
ESDA	Exploratory Spatial Data Analysis
GIS	Geographic Information System
GIScience	Geographic Information Science
GUI	Graphic User Interface
IED	Improvised Explosive Device
Q	Probability of observing a statistic according to the normal distribution
R	Knox statistic
R_l	Linear average nearest-neighbor statistic
R_p	Continuous space average nearest-neighbor statistic
$SCAN$	Spatiotemporal Cluster Analysis on a network
SANET	Spatial Analysis on a Network
ST	Spatiotemporal
τ	Critical temporal distance used in determining the Knox statistic

Abstract

A Comprehensive Process for Spatiotemporal Analysis of Network-Based Phenomena

David C. Eckley, M.S.

George Mason University, 2010

Thesis Director: Dr. Kevin Curtin

This thesis describes an efficient and effective process for conducting spatiotemporal cluster analysis of network-based phenomena. While various methods are published which describe spatiotemporal analysis of phenomena in continuous space, the literature is lacking for the application of these methods in network space. Through a step by step process, Part 1 of this thesis establishes the validity of a network application for the spatiotemporal clustering method proposed by Knox (1964). Further, it presents an intuitive technique for determining the critical parameters in space and time for the spatiotemporal test, by examination of minimum and average nearest neighbor distances in both dimensions independently. Through examples, Part 2 explores the significance tests used by the described spatiotemporal clustering methodology and expounds upon the critical parameter determination mentioned in Part 1. Part 3 presents a GIS-based toolbox, *SCAN* (Spatiotemporal Cluster Analysis on a *network*) designed to perform

spatiotemporal cluster analysis of network-based phenomena using the methods presented in Part 1 and Part 2.

Forward

“The greatest value of a picture is when it *forces* us to notice what we never expected to see.” (Tukey 1977)

It is not normally the practice of the U.S. Army to offer incentives to its officer corps for the purpose of retention, but for a period of time between 2007-2009 such an exception was made for company grade officers. Of various incentives offered, the opportunity to attend graduate school full-time for 18 months seemed an especially generous option. With the war-time operational tempo requiring frequent deployments, an exclusively education-focused assignment would afford an officer not only a chance to meet higher educational goals, but also a brief period of stability with focused family time. The author is especially grateful to have been afforded this great privilege and has purposed to make this temporary assignment worth every penny invested by the Army on behalf of the U.S. tax payer.

And so, as a research topic was considered for the subject of this thesis, at the forefront of the author’s mind was how to turn this educational opportunity into a contribution for the ongoing war effort. With a background and interest in Geographic Information Systems (GIS) and Geographic Information Science (GIScience), a geographically-based problem was of particular interest. Working with a Department of Defense (DoD) sponsored research team at the university, the author’s interests were further focused on problems arising from the effort to counter the adversary’s use of

Improvised Explosive Devices (IEDs) in Iraq and Afghanistan. Recognizing that IEDs are generally restricted in space to the transportation network upon which they occur, and that IED incidents have both a spatial and temporal component, a geographic research question began to emerge. Since IED incidents are not the result of a random process, but rather are caused by the agents who use them as weapons against coalition forces, defense research has demonstrated that clustering of these events occur in both space and time, functions of both where and when the adversary is operating; or equally as likely, where and when coalition forces use the road network. Understanding the above, the following specific research question seemed plausible: “What can be learned about the adversary from the analysis of the spatiotemporal clustering of IED incidents?” If spatial clusters reveal operational areas, and temporal clusters reveal operational periods, could spatiotemporal clusters reveal a correlation between the two which could be exploited?

While these were intriguing questions, it quickly became evident that the data required for such research could not be openly published. Therefore, either notional data would need to be derived to replicate the phenomena of IED occurrence or data for phenomena with similar characteristics could be explored. Not wanting to deal with the security concerns involved in accomplishing the former, the author decided to pursue the latter option, recognizing too, that the significance of research lies not only in solving specific problems, but in developing methods that can be repeatedly used to solve many similar problems like the question of interest.

Determining that the research data should represent static events with both a spatial and temporal component and that the events should be restricted to network space,

the author chose to use traffic collision data as the subject of this thesis. The data were easy to obtain given that the body of research on the study topic of traffic collisions is enormous. While the causality of IED incidents and traffic collisions cannot be compared, there were potential factors such as weather, construction, or poor road management that could theoretically lead to significant spatiotemporal clustering of traffic collisions.

It was from these ideas that the following thesis became a reality. While this research will not specifically address any applications for the IED problem discussed above, it does describe and explain a methodology for conducting spatiotemporal analysis of similar network-based phenomena, in a straight-forward, comprehensive way, and presents a GIS-based tool with which to do so.

If this picture helps anyone discover what they didn't expect to see, this effort has been worthwhile.

Introduction

What Is a Spatiotemporal Cluster?

While the term “cluster” can be used to describe patterns represented by various types of phenomena, this research is concerned with those patterns created by individual events limited to finite points in geographic space. In this context, an event has both a spatial and temporal component. More specifically then, a spatial cluster, is a geographic point pattern that is represented by an excess number of events relative to the expected pattern, such as a local aggregation of cancer cases, pockets of crime in blighted urban areas, or high collision rates at congested intersections. Likewise, a temporal cluster is represented by the occurrence of a greater number of events than that expected during a particular portion of a specified time period, such as excess flu cases during the month of April, increased incidents of rape during hours of darkness, or many traffic accidents during a rainstorm following an extended period of dryness. A spatiotemporal cluster exists when an excess number of events that occur within some geographic space are also unexpectedly close in time, such as a significant population of students in a given classroom contracting an infectious disease during the same week, a spike in purse snatching on the National Mall during a holiday weekend, or excessive fender benders during rush hour traffic.

Why Conduct Spatiotemporal Cluster Analysis?

From a purely theoretical standpoint, spatiotemporal cluster analysis is one of many techniques utilized in Exploratory Spatial Data Analysis (ESDA) for geographic pattern recognition (Jacquez 2008). Once spatial patterns have been identified and defined through ESDA, hypotheses may be developed to specify real and testable explanations for the observed patterns (Jacquez 2008). Recognition of these patterns is important as they illuminate underlying space-time processes which are the focus of many geographic studies.

Practically speaking, while spatial and temporal clusters may exist independently, spatiotemporal clusters indicate a correlation between the spatial and temporal dimension for the given phenomenon. Identifying and determining a correlation in spatiotemporal clusters may provide valuable insight beyond the determination of exclusively spatial or temporal clusters. In epidemiology, spatiotemporal clusters may reveal the relationship between origin and onset of disease; in criminology, the relationship between the location and recurrence of a particular criminal activity; in transportation research, the highest risk periods of travel within given portions of the network.

The Issue of Spatial and Temporal Measures

As cluster analysis depends upon a measure of separation between events, choosing an appropriate distance measure and resolution or granularity for the phenomenon under study is important. Using identical distance measures to study differing phenomena may result in misleading conclusions or a loss of information.

When exploring temporal clustering, the granularity of the temporal scale should be considered. If examining traffic collisions during peak traffic periods, more information will likely be gained from considering the number of collisions which occurred by hour as opposed to the number which occurred each week. If peak traffic periods occur daily at specific hours, then knowing the frequency of collisions by week will not help the researcher determine whether temporal clustering of traffic collisions corresponds to daily peak traffic hours. If traffic collisions were measured by hour, then this analysis could be conducted. Similarly, since vehicles are restricted to the network space within which they travel, analyzing clusters of traffic collisions using Euclidean measures could lead to a false discovery of clustering not related to the same spatial process. Consider Figure 1. While a Euclidean measurement might place the two points in a cluster defined by a distance threshold equal to their depicted separation in Figure 1, the same points would not produce a cluster when considering their separation using a network distance measurement. Yamada and Thill (2004) illustrate the pitfall of conducting analysis of network-based phenomena with continuous space measurements in the place of network measures with traffic data from Buffalo, NY. Extensive research such as that just mentioned has demonstrated the validity of using network measures to analyze network-based phenomena and numerous continuous space statistical methods have been extended network space (Okabe et al. 1988; Black 1991; Miller 1999; Okunuki and Okabe 2002; Okabe and Satoh 2006; Yamada and Thill 2007; Yamada and Thill 2010; Shiode 2008; Shiode and Shiode 2009; Okabe, Yomono, and Kitamura 1995). One of the goals of this research is to contribute to this work of extending continuous space statistics to network

space by adapting the Knox test to conduct spatiotemporal analysis of network-based phenomena.

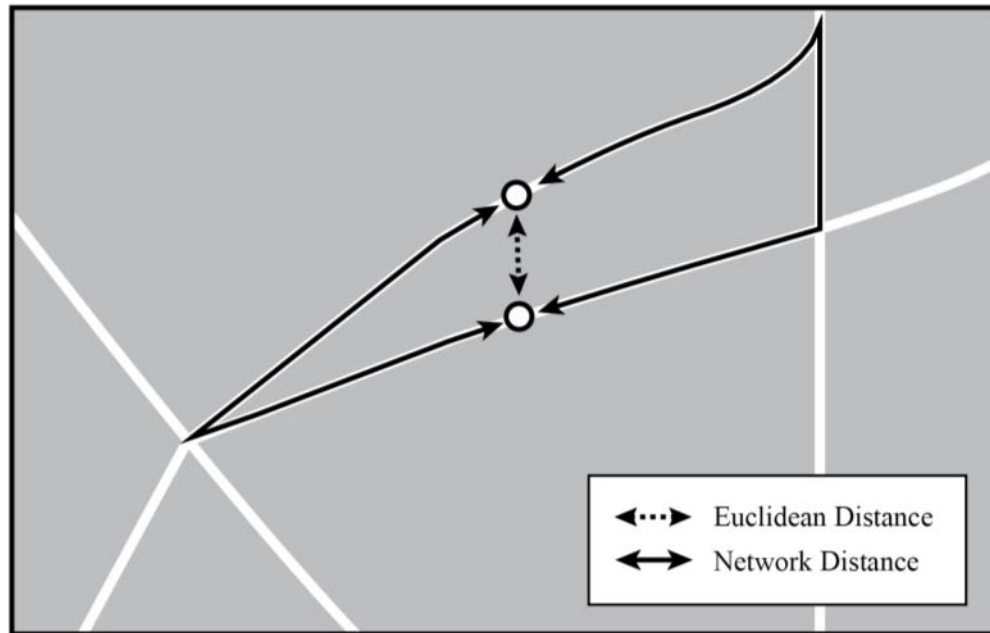


Figure 1. Graphical representation of the distance measurement differences between two points measured in continuous and network space.

Global vs. Local Statistics

A common differentiation between spatial statistical tests is the scale at which a given test is sensitive. In the case of tests for spatial clustering, a global statistic is one that provides a summarized spatial structure for the entire study area without identifying where the specific clusters are located, while local statistics define clustering within small areas which together comprise the larger study area (Jacquez 2008). Many local statistics

like Local Indicators of Spatial Autocorrelation (LISA) (Anselin 1995; Ord and Getis 1995) have global counterparts such as Moran's I (1950).

The Knox test, which is the focus of this research, is unique in that while it is a global statistical test, it is possible to display the pairs of events which contribute to the global spatiotemporal clustering. While it is possible to visualize the cluster(s) defined by the global test, the statistical significance of the test can only be determined for the global case, and not each individual event pair.

The Origin of Spatiotemporal Cluster Analysis

Academic research traces the origin of spatiotemporal clustering techniques to the work of Knox (1964) who was interested in the epidemiology of various cancers. Knox's method involved the comparison of distances between all pairwise cases of disease in both space and time, with the statistic equal to the number of pairs which were both close in space and time, based on some specified measure of closeness. Knox supposed that related cases would be closer together in space and time, while unrelated cases would tend to have a larger separation. A difficulty with his statistic was the determination of significance, as the same case could be used multiple times in determining the adjacencies in space and time. Knox conjectured that his statistic followed a Poisson distribution. David and Barton (1966), proved Knox's conjecture that his statistic was Poisson. Further, they provided the mean and variance of the statistic. Mantel (1967) provided the permutation variance of the Knox statistic, describing how to apply Monte Carlo methods to determine the statistic's significance.

Knox's statistic is both simple and efficient, and remains the most common spatiotemporal clustering method in epidemiology today. However, a major complaint with the statistic is the need to define arbitrary critical parameters measuring closeness in space and time. Many researchers have suggested techniques to improve this perceived shortfall. David and Barton (1966) proposed their own statistic which compares spatial clusters within subsets of time defined by the average time interval between events. Mantel (1967) proposed the use of reciprocal transformations for the actual space and time labels of cases. Klauber (1971) defined a two-sample spatiotemporal clustering test. These and other similar tests are described in William's (1984) thorough review of continuous space spatiotemporal clustering. Later techniques specific to epidemiology include Baker's (1996) modification of Knox's method, where spatiotemporal clusters were detected within a range of acceptable space and time critical parameters, Jacquez (1996) k-nearest neighbor method, which specified spatiotemporal clusters based on which points were neighbor in space and time, and a variety of other researchers who developed scan statistics to enable the detection of emerging spatiotemporal clusters (Kulldorff and Hjalmar 1999; Rogerson 2001; Assuncao and Correa 2009; Mirghani et al. 2010).

Similar to the efforts of those aforementioned, this research introduces another technique by which to determine appropriate spatial and temporal critical distances for defining spatiotemporal clusters when appropriate critical parameters are unknown. It involves a preliminary examination of nearest neighbor distances in both space and time

and uses the test results to define a range of critical spatial and temporal distances which are used as parameters for a Knox-based spatiotemporal analysis.

The Development of Spatiotemporal Clustering Methods and Current Applications

As computer and satellite technology has improved over the past two decades, so has the variety and complexity of geographic spatiotemporal clustering techniques. While the origins of spatiotemporal cluster investigation lie almost exclusively in epidemiology, other applications of the statistic have now been explored. Black (1991) extended spatiotemporal cluster analysis to traffic accidents on a linear stretch of highway. Jacquez (1996) developed a spatiotemporal cluster test which he used to expose the spatiotemporal clustering of forest fires in Canada. Criminal activity-based spatiotemporal clusters have been investigated by Assuncao et al. (2007) and Nakaya and Yano (2010). Lesniak and Isakow (2009) applied spatiotemporal clustering analysis to seismic activity and further spatiotemporal analysis has been conducted with traffic collision data by Mountrakis and Gunson (2009) and Khan et al. (2009).

The most recent spatiotemporal methodological advances include a windowed nearest neighbor approach (Pei et al. 2010), multi-dimensional map algebra (Mennis 2010), visualizing clusters in a space-time cube (Nakaya and Yano 2010), the use of bivariate kernel density estimators (Mountrakis and Gunson 2009), cross k-function analysis (Khan et al. 2009), and stack-based spatiotemporal clustering (Chang, Zeng, and Chen 2008).

With the exception of Black (1991), Mountrakis and Gunson (2009), and Khan et al. (2009), all of the original spatiotemporal clustering techniques and their most recent counterparts rely on Euclidean distance measures. In accordance with the preceding discussion of distance measures, this research will apply this general method to network-based phenomena using network spatial measures.

GIS Applications

Originally designed in the 1960s, Geographic Information Systems (GIS) are the primary tools for the collection, storage, management, query, analysis, and display of spatial data today. The unique data structures employed by GIS technology allows for the statistical analysis of large spatial datasets, a capability which has greatly advanced the ability to perform ESDA and design new spatial statistical techniques. While there are numerous GIS designed for specific applications, such as TransCAD, used by the transportation industry, arguably the most universally employed GIS software today is ESRI's ArcGIS. ArcGIS's Spatial Analyst, Tracking Analyst, and Network Analyst provide statistical tools for assessing spatial, temporal, and network-based data respectively. ArcGIS lacks the capability to perform the spatiotemporal analyses described thus far.

That is not to say that GIS software does not exist to perform spatiotemporal clustering analysis. Many of these applications are stand-alone tools designed for very specific analyses. SaTScan, developed by Martin Kulldorff (2006), searches for spatial, temporal, and spatiotemporal clusters in data but does not have a graphical interface nor

is it integrated with any specific GIS program (Block 2007). Similar to SaTScan, GeoSurveillance provides both retrospective and prospective tests for spatiotemporal clustering. GeoSurveillance improves upon SaTScan as a stand-alone system by incorporating a Graphic User Interface (GUI), but like SaTScan, is not integrated with any mainstream GIS programs (Yamada, Rogerson, and Lee 2009). Another stand-alone GIS, TerraSeer offers various spatiotemporal analysis packages and shares compatibility with files used by ESRI, MapInfo, and ENVI programs (Jacquez 2008).

Unlike these tools described above, but similar to the SANET toolset developed by Okabe and his team (2009), the spatiotemporal cluster analysis toolbox described in this research, called *SCAn* (Spatiotemporal Cluster Analysis on a *network*), integrates directly into ArcGIS ArcMap software and can be run through the ArcToolbox menu display. This tool is free and simple to use and provides a currently non-existent capability in ArcGIS to conduct spatiotemporal cluster analysis using network distance measures.

Significance of this Research

Unlike works preceding this effort, the following thesis presents a step-by-step comprehensive process for investigating the global spatiotemporal clustering of network-based phenomena. A new technique is described for determining an appropriate range of values for the critical distance parameters in space and time when appropriate values are unknown. The determination of statistical significance for these tests is examined and a caution is cited in generating reference distributions from Monte Carlo simulations.

Finally, a user-friendly ArcGIS-based tool (which can perform the analyses described) is presented.

Thesis Structure

This thesis begins in Part 1 with a technical methodology for conducting a spatiotemporal cluster analysis of network-based phenomena, using traffic collision data as the test case. Part 2 explores the statistical relevance and significance of the methodology set forth in Part 1 and provides various illustrations as expository aides. Finally, Part 3 describes the GIS-based tool designed to implement the statistical tests described in Parts 1 and 2 and is intended to serve as a general user's guide for the application.

PART 1

A Process for Investigating Spatiotemporal Clustering of Network-Based Phenomena

Introduction

The specific analysis to be addressed in this study is the spatiotemporal clustering of events occurring on a network. Unlike continuous space, network space is one-dimensional and the events which occur on it are restricted to it. There are many network-based phenomena that are regularly researched, such as pipeline leaks, power failures, carjacking, or as discussed previously, IED attacks and traffic collisions. While it is not unusual to perform spatial or temporal cluster analysis of such network-based phenomena, spatiotemporal analyses are much less common. Temporal cluster analysis has long been performed by the transportation research community, and the temporal characteristics of traffic studies are well known (Lord and Mannering 2010). Spatial clustering, in the continuous case, dates to the early 1900's (Clements 1905; Voronoi 1907), but adaptation of popular continuous space statistics to network space has only begun within the past two decades (Okabe et al. 1988; Black 1991; Miller 1999). In contrast, although continuous space spatiotemporal cluster analysis has been thoroughly investigated in epidemiology (Knox and Bartlett 1964; Smith et al. 1976; Meighan and Knox 1965; Lloyd and Roberts 1973; Roberts, Laurence, and Lloyd 1975; Glass and Mantel 1969), the analytical extension to the network-based case remains uncommon.

The intent of this work is to describe a comprehensive process for conducting spatiotemporal cluster analysis of network-based phenomena where one does not currently exist. The underlying spatiotemporal method employed in this process is the Knox test, and a common critique of this method is the need to designate arbitrary critical spatial and temporal distances as clustering parameters. Original to this research, a simple technique is suggested for determining an acceptable range of critical spatial and temporal distances when they are unknown, which are obtained through the results of spatial and temporal cluster analysis using the average nearest neighbor distance test. The range of acceptable values is then used to search for spatiotemporal clusters using the GIS tool provided here, *SCAN*.

In order to demonstrate this process, the network-based phenomenon of traffic collisions will be examined. Traffic collisions provide a good case study for spatiotemporal cluster analysis as the data are easy to obtain and reporting standards require very specific attribute information about each event, including precise locations (often with at least 10 meter resolution) and precise times (often with minute granularity). In addition to these key attributes, traffic collision reports compiled by law enforcement officials typically contain extensive details about the physical and human conditions present at the time and place of collision. If associated with the traffic collision data table, this information may help explain a discovery of spatiotemporal clustering.

The organization of the proposed comprehensive process is as follows: Step 1 involves defining the spatiotemporal problem of interest; Step 2 addresses concerns for

acquiring and evaluating the data to be analyzed; Step 3 discusses the requirements for pre-processing the data subjected to analysis and presents possible ways to spatially and temporally examine the data; Step 4 assesses the spatial clustering of the data through an examination of nearest neighbors in order to determine an acceptable range of critical spatial distances in determining spatiotemporal clusters; Step 5 tests for temporal clustering in the data for the same purpose; Step 6 evaluates the findings of Steps 4 and 5 by testing for spatiotemporal clusters; Step 7 seeks to explain the results of the preceding steps.

Step 1: Define a Research Problem and Analytical Approach

Fundamental to any scientific research is the need for defining a problem. As described above, the interest here is to answer questions concerning a problem that fits within the specific constraints required for the conduct of network-based spatiotemporal analysis. First, the problem must be geographically limited to network space, and second, the problem must have finite and quantifiable spatial and temporal measures. While there are many problems that fit within this scope, the problem to be addressed as an example here is that of traffic collisions.

Once the problem is defined, there are two primary ways to approach the data comprising the problem. First, confirmatory data analysis may be pursued. This deductive approach is widely accepted, and involves the generation of hypotheses in order to seek definite answers to specific research questions. Inferential statistics are utilized which rely on probability models and confidence interval estimations. A

significant drawback of this approach is that it can create a misleading perception of precision, which is especially true when dealing with the complexities of geographic processes that rarely conform to any type of true randomness (Jacquez 2008). The other option is to pursue exploratory data analysis. An inductive approach, exploratory data analysis or as previously described, ESDA, uses descriptive statistics to let the data propose questions and develop hypotheses. ESDA attempts to evaluate the validity of assumptions and relies heavily on graphical displays to support conclusions. The drawback with this analytic technique is that it usually does not provide definitive answers and can generate subjective results (Michaelson 2007).

Although this research presents a specific process by which to conduct spatiotemporal cluster analysis, the data analysis approach taken here is primarily exploratory. Before any hypotheses are generated for spatiotemporal clustering, the data will first be collected, pre-processed, and then analyzed for purely spatial and temporal clustering. In so doing, information learned from spatial and temporal clustering in the data should inform the researcher how to define the variables for the spatiotemporal tests. Without knowledge of the underlying spatial and temporal distributions, a spatiotemporal analysis may not be as meaningful.

Step 2: Acquire and Evaluate the Data

While event-based spatiotemporal datasets have not always been easy to generate or analyze, the advent of powerful computers and GPS technology has made their existence and utility prevalent. Where traffic collisions were once recorded based on

their proximity to a specific intersection or highway mile-marker, today their locations are usually recorded precisely with geographic coordinates. Improved emergency response times and traffic cameras have made it possible to record more accurate time stamps for these events as well. In the case of traffic collisions, many local, county, and state governments publicly provide access to such data.

Because this research is specifically concerned with network-based phenomena, in addition to an event-based spatiotemporal dataset, a network dataset is required, which defines the network space within which the events of interest occur. Again, in the case of traffic collisions, most related road network data is publically available at the local or regional level, but if it is not, U.S. Census Bureau's TIGER files may be utilized for U.S. specific research (www.census.gov).

Prior to utilizing such data for research, however, it is important to evaluate the data for accuracy and completeness so as not to compromise the analytic results. Data quality may be assessed by reviewing the associated metadata or if there is any question, by contacting the publisher directly.

For the test case, traffic collisions along major roadways in Franklin County, Ohio, from January to March, 2009 are considered. Traffic collision data were obtained through the Ohio Department of Public Safety's crash request portal (Kennedy 2010) and the road data through OpenStreetMap (2010). See Figure 1-1 for a map of the data following the pre-processing of Step 3.

Step 3: Pre-process the Data

Format and Map the Data

If the data is assessed to be of acceptable quality and coverage, a key first step in formatting the data is to ensure there are no duplicate entries or entries that are missing either the spatial or temporal attribute. This is important for cluster analysis as duplicate entries could cause false clustering in the data, while missing spatial or temporal attributes could cause automated calculations to fail or report inaccurate results. That is not to say that multiple events may occur at the exact same location but at different times, or alternatively at the same time but at different locations. Depending on the purpose of the study, such spatial or temporal attribute redundancy is acceptable but should be noted as it may lead to unique or unexpected results.

Once duplicate and incomplete entries have been removed, additional formatting may be required. Both the spatial and temporal attribute fields should be audited. Spatial formatting will likely require the transformation of the geographic events and the network dataset onto an acceptable map projection and coordinate system. The selected projection should be appropriate for the analysis conducted. Because network-based spatiotemporal cluster analysis examines distances between point pairs of the phenomenon, a map projection which preserves distance is preferred, although a compromise projection may be suitable, especially for large scale study areas. Similarly, temporal attributes may need to be formatted into a date/time format that can be read by the GIS performing the analysis.

Next, the number of events and extent of the study area (network) must be considered. While technology has greatly enhanced many geographic analyses, trying to compute certain statistical methods with large datasets may take an excessive amount of time or exhaust computer memory. If a large dataset must be subdivided for time-saving reasons, the division of data should be undertaken carefully as it may bias the results of cluster analysis.

Finally, before analysis may be conducted in network space, the line segment dataset must be converted into a network dataset and the event data must be located within the network space. This is a critical pre-analysis step, but because procedures vary between GIS programs, the process will not be described here.

For the Franklin County traffic collision data, the event and network datasets were transformed onto the Lambert Conformal Conic projection with the Southern Ohio State Plane coordinate system. While not preserved, this projection reduces distance and area distortions across the study area. Additionally, the temporal attribute was reformatted from a date-time format into a format which listed the date, weekday, hour, and minute values in separate fields. This was done in order to facilitate temporal analyses of varying granularity.

For this particular study, only a subset of the total collisions and a subset of the road infrastructure are considered. This was done in order to work with a manageable set of data small enough not to require excessive time for computational analyses and large enough to provide meaningful statistical results. From the complete Franklin County collision dataset for 2009, only those collisions involving injuries during the months of

January through March are considered. From the network dataset, only primary roads and highways inside Franklin County were considered. Based on these selections, this reduced the dataset for the study from 29,129 collision events to 586 and from 29,357 road segments to 2034 representing 930 linear kilometers of roadway. See Figure 1-1 for a map of the study area and data.

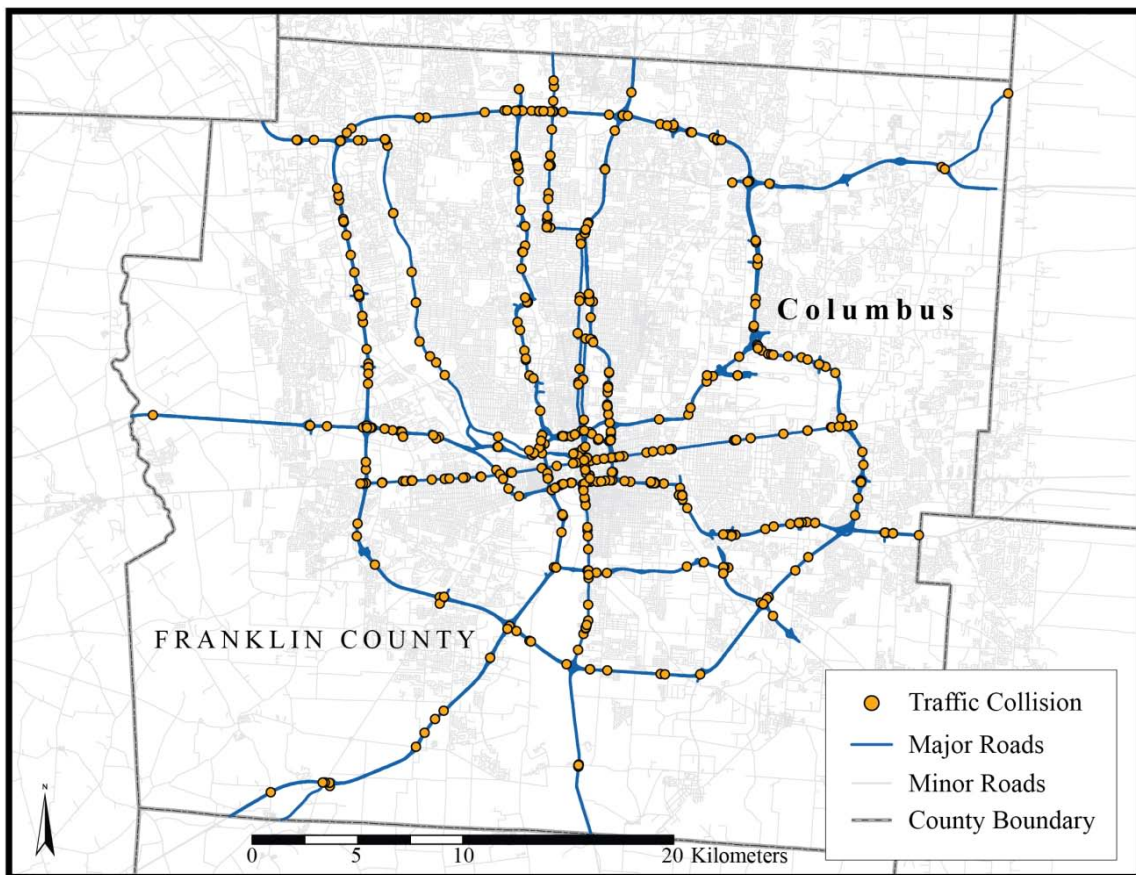


Figure 1- 1. Map of 586 injury-causing traffic collisions on major roads in Franklin County, OH from January to March 2009.

Examine the Data

It is important to become familiar with the test data before performing specific analyses. Understanding the spatial and temporal characteristics of the data under study can help formulate specific research questions and hypotheses, as well as help explain the results obtained from the cluster analysis.

Table 1- 1. Redundancy of attribute labels in traffic collision data for Franklin County, OH, January-March 2009.

	Spatial Attribute	Temporal Attribute
Total Events	586	586
Unique Events	474	581
Attribute shared by 2 Events	50	5
Attribute shared by 3 Events	11	0
Attribute shared by 4 Events	8	0
Attribute shared by 5 Events	1	0
Attribute shared by 11 Events	1	0

As mentioned in the previous section, identifying the existence of duplicate spatial or temporal attributes is of particular interest to cluster analysis. For the Franklin County traffic collision data used in this study, the existence of such duplicates is prevalent, especially in the spatial attribute (see Table 1-1). The spatial duplicates can be explained by the practice of using a street intersection as a point of reference for a traffic collision instead of locating the exact position of the collision. In the data used for this study, of the 586 events, 474 unique locations, and 581 unique time stamps are represented. The effect of the spatial duplicates will be noted later in Step 4.

Because this research is interested in investigating continuous and network distance measures and various temporal granularities in the context of spatiotemporal cluster analysis, several figures are presented here. Figures 1-2 through 1-4 present the distribution of spatial and temporal distances between pairs. For the spatial distribution (Figure 1-2), the minimum distance between collision pairs is 0 meters, the maximum distance is 58,018 meters, and the mean is 16,208 meters. For the temporal distribution in days (Figure 1-3), the minimum distance between collision pairs is 0 days (events

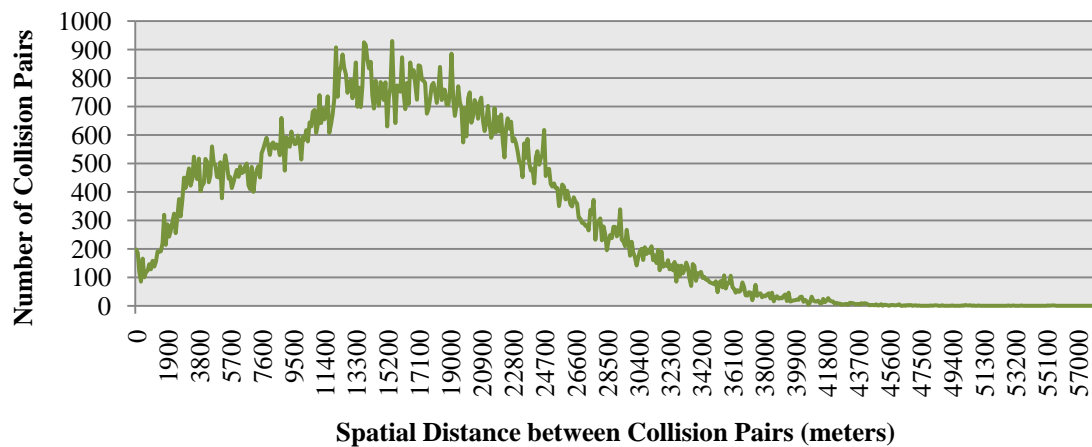


Figure 1- 2. Distribution of spatial distance between pairs of traffic collisions in Franklin County, OH, January-March 2009.

occurred on the same day), the maximum is 89 days, and the mean is 30 days. The temporal distribution in hours (Figure 1-4) is bounded by a minimum of 0 hours (events occurred during the same hour), a maximum of 2149 hours, with a mean of 718 hours. Figures 1-5 through 1-7 show the frequency distributions for various temporal granularities of the data. In Figure 1-5, depicting day of week frequencies, the minimum

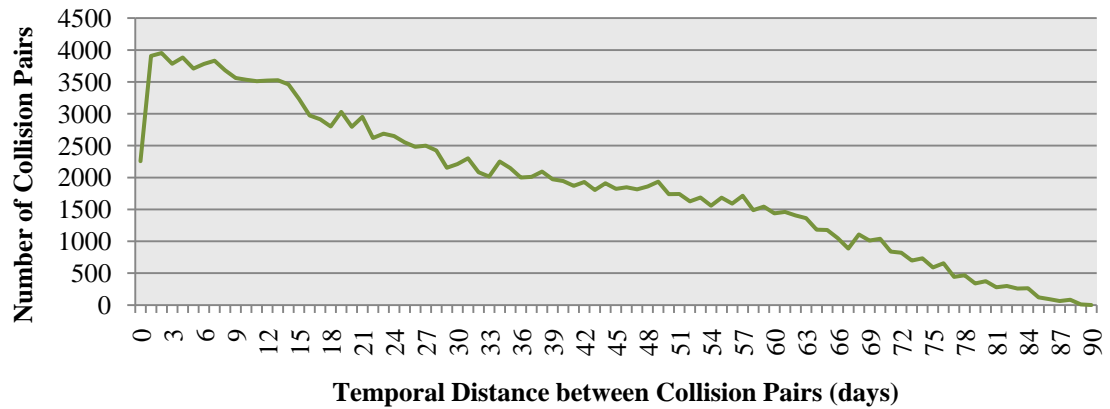


Figure 1- 3. Distribution of temporal distance in days between pairs of traffic collisions in Franklin County, OH, January-March 2009.

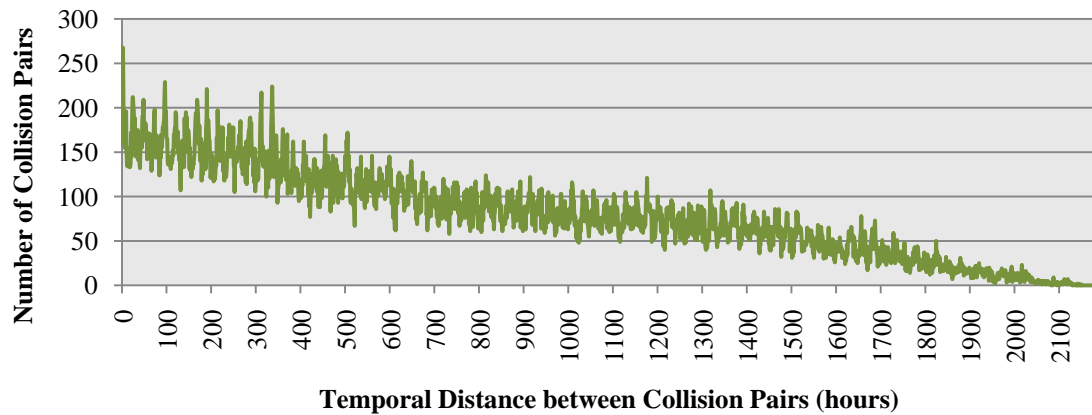


Figure 1- 4. Distribution of temporal distance in hours between pairs of traffic collisions in Franklin County, OH, January-March 2009.

number of collisions occurred on Saturdays, with 45, the maximum on Fridays, with 105, and the average weekday occurrence was 84 collisions. Looking at collisions by day (Figure 1-6), the minimum tally was 0 events on March 2nd, and the maximum number was 19 on January 16th, with a daily average of 6.5 collisions. Finally, in collisions by hour of the day (Figure 1-7), the minimum number of events occurred at 0400, with 3,

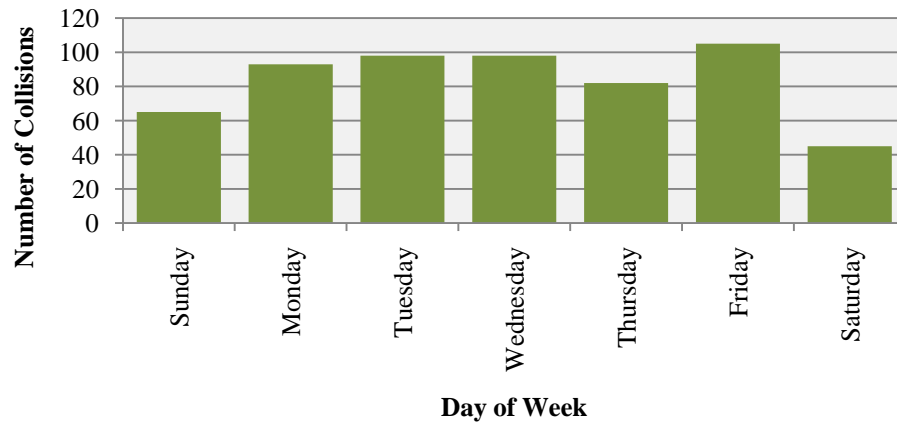


Figure 1- 5. Number of traffic collisions occurring by day of the week in Franklin County, OH, January-March 2009.

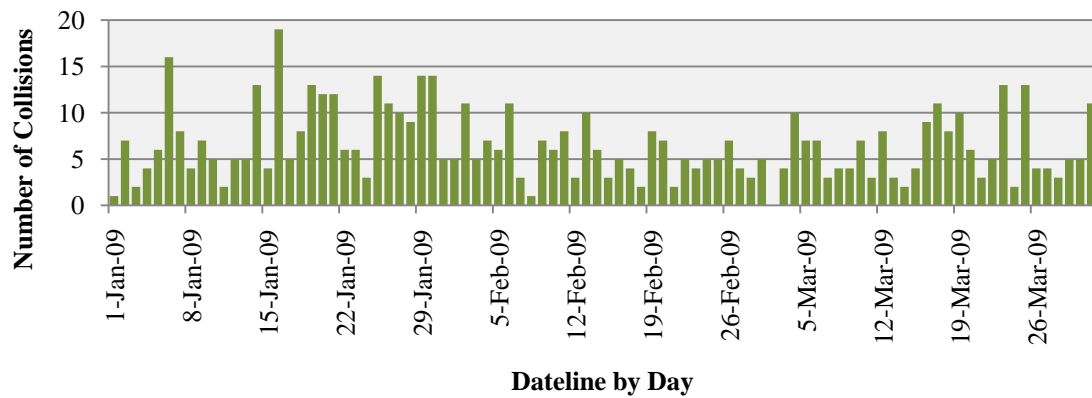


Figure 1- 6. Number of traffic collisions occurring by day in Franklin County, OH, January-March 2009.

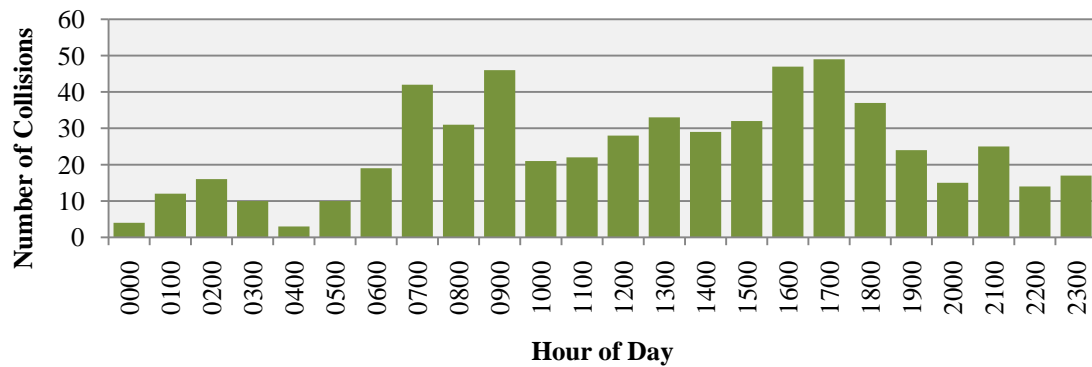


Figure 1- 7. Number of traffic collisions occurring by hour of the day in Franklin County, OH, January-March 2009.

the maximum at 1700, with 49, and an average number of collisions by hour of the day at 24.4.

While these figures and values are not necessarily meaningful at this point in the spatiotemporal analysis process, they may provide insight for results from Steps 4 through 7.

Step 4: Test for Spatial Clustering

In review, a spatial cluster is a geographic point pattern that is represented by an excess number of points relative to the expected pattern. While the focus of this research is the investigation of spatiotemporal clusters, examination of the underlying spatial distribution should not be overlooked. Understanding the spatial pattern of the phenomenon under study may help formulate questions and hypotheses about possible spatiotemporal correlation.

Another reason to first investigate the spatial distribution is to determine an appropriate resolution at which to search for spatiotemporal clusters in the geographic space. If the events studied have an average separation in kilometers, then it probably does not make sense to search for clusters at a resolution of 1 meter.

As previously discussed, this research is concerned primarily with network-based phenomena, however, in order to compare and contrast the differences between continuous and network tests for spatial clustering, both are presented here.

Continuous Space Tests

There are a variety of continuous space spatial clustering tests used to describe point distributions. While there are both global and local spatial clustering tests, only the global variety are considered here. Some of the more popular tests include quadrat analysis (Clements 1905), Voronoi diagrams (Voronoi 1907), Moran's I (Moran 1950), average nearest neighbor (Clark and Evans 1954), kernel density analysis (Parzen 1962), and Ripley's K (Ripley 1981). Perhaps the most commonly used of those above is the average nearest neighbor test. Because this test will provide a sense of the spatial scale at which points are distributed, it is a good choice for this step.

The average nearest neighbor distance test proposed by Clark and Evans (1954) examines the distance between each point in a distribution and its nearest neighbor. The distance between each point and its closest neighbor is summed and the average is taken. In the following equation quantifying this method, \bar{d}_O is the observed average distance, n is the number of points in the distribution, and $Min(d_{ij})$ is the shortest distance between

point i and its neighbor point j in continuous space:

$$\bar{d}_o = \frac{\sum_{i=1}^n \text{Min}(d_{ij})}{n} \quad (1.1)$$

In order to determine whether the observed point distribution is clustered or not, the average nearest neighbor distance is compared to the expected value of the average nearest neighbor when points are distributed according to the Poisson point process. The expected average nearest neighbor distance is given by the equation:

$$\bar{d}_E = \frac{0.5}{\sqrt{n/A}} \quad (1.2)$$

where \bar{d}_E is the expected value of the average nearest neighbor distance, n is the number of points in the distribution, and A is the geographic area bounding the points in the distribution. It should be noted that this is the basic formula for the test and it does not take into consideration irregularly shaped areas or boundary edge effects. Formally, the average nearest neighbor statistic is annotated:

$$R_p = \frac{\bar{d}_o}{\bar{d}_E} \quad (1.3)$$

If R_P is greater than 1, meaning that the observed average nearest neighbor distance is greater than the expected nearest neighbor distance for a random distribution, then the observed distribution is dispersed. If R_P is less than 1, the observed distribution is clustered. In order to test for the significance of R_P , a formula for the z-score is provided:

$$z = \frac{\bar{d}_O - \bar{d}_E}{SE} \quad (1.4)$$

where SE is the standard error of the variance given by:

$$SE = \frac{0.26136}{\sqrt{n^2/A}} \quad (1.5)$$

An important assumption made by the calculations for the average nearest neighbor distance statistic and its associated test for significance is that for the expected random distribution, points are free to locate anywhere within the area described by the numerical value given. In the case of network-based phenomena, this assumption is invalid, as point distributions are constrained to network space. This is not an appropriate test to use for network-based phenomena, but it is provided here as a means for comparison.

For the test case of traffic collisions in Franklin County, measured in continuous space, the minimum nearest neighbor distance between all the given points in the 586-

point distribution is 0 meters, while the maximum nearest neighbor distance is 7470 meters. Based on the formulae given above, the observed average nearest neighbor distance is 187 meters, the expected distance is 748 meters, and the average nearest neighbor statistic is 0.25. Because the statistic value is less than 1, the point distribution in the test case is clustered. For significance testing, a statistic is typically considered significant if the probability of its value occurring is less than 5% given the expected distribution of the statistic under the case of spatial randomness. For a normal distribution, a probability of 5% is equivalent to a z-score of ± 1.96 . In order to be significant, the z-score of the observed statistic should be greater than 1.96 or less than -1.96. The z-score for the test case is -34.74 indicating that the observed clustering is significant.

Network Space Tests

Because this research is focused on network-based phenomena, it is important to consider spatial clustering tests which deal specifically with network space. Over the past twenty years, all of the popular continuous space statistics described in the preceding section have been extended to network space: network quadrat (Shiode 2008), network Voronoi diagrams (Okabe et al. 2000), network Moran's I (Black 1992), network average nearest neighbor (Okabe, Yomono, and Kitamura 1995), network kernel density analysis (Flahaut et al. 2003), and network Ripley's K (Okabe and Yamada 2001). For the sake of comparison with the above section, the network average nearest neighbor distance test will be discussed here.

Conceptually, the network and continuous average nearest neighbor distance statistics are the same. The difference lies in how the distance to the nearest neighbor is measured and how the distribution of the expected nearest neighbor distances is derived. The formulas described in equations 1.6 through 1.11 can be found the work by Okabe et al. (1995). The network average nearest neighbor distance is:

$$\bar{t}_O = \frac{\sum_{i=1}^n \text{Min}(t_{ij})}{n} \quad (1.6)$$

where \bar{t}_O is the value of the observed average nearest neighbor distance, n is the number of points in the distribution, and $\text{Min}(t_{ij})$ is the shortest path distance between point i and its neighbor point j in network space.

In the case where the “network” within which the point distribution exists is represented by a single line, the equation for the expected value of the average nearest neighbor distance is:

$$\bar{t}_E = \frac{(n + 2)l}{2n(n + 1)} \quad (1.7)$$

where \bar{t}_E is the value of the expected average nearest neighbor distance when points are distributed on the line according to the Poisson point process, n is the number of points in the distribution, and l is the length of the line. In the case where the network is made up

of multiple line segments, the expected distance is given by a probability distribution function:

$$F(\bar{t}) = \sum_{j=1}^m \frac{l_j}{l_T} F_j(\bar{t}) \quad (1.8)$$

where l_j/l_T is the probability of a point being placed line segment j in network T , and $F_j(\bar{t})$ is the probability distribution function of the average nearest neighbor distance on line segment j . The derivations for both of these equations are complex and can be found in Okabe et al. (1995).

The network average nearest neighbor statistic is now defined as:

$$R_L = \frac{\bar{t}_O}{\bar{t}_E} \quad (1.9)$$

The expected value of R_L given a Poisson point process on a line is $E(R_L) = 1$. Similar to the continuous test, if R_L is greater than 1, meaning that the observed network average nearest neighbor distance is greater than the expected network average nearest neighbor distance for a random point distribution on a line, then the observed distribution is dispersed. If R_L is less than 1, the observed distribution is clustered. When testing for significance in the case of points distributed across a single

line, the z-value can be found by:

$$z = \frac{R_L - E(R_L)}{\sqrt{Var(R_L)}} \quad (1.10)$$

where $Var(R_L) = Var(\bar{t})/n$ with the variance of \bar{t} given by:

$$Var(\bar{t}) = \frac{(n^3 + 8n^2 - 8)l^2}{4n^2(n + 1)^2(n + 2)} \quad (1.11)$$

In order to test for significance in the case of a network comprised of many line segments, Monte Carlo methods are employed using Okabe et al.'s SANET software (2009).

In the test case of 586 traffic collisions distributed across the major road network of Franklin County, by network space measures, the minimum shortest path distance between any two neighbors is 0 meters while the maximum nearest neighbor distance is 7470 meters. While uncommon, the identical maximum distance between nearest neighbors in both continuous and network space for this dataset can be explained by the fact that these two events are located on a unidirectional segment of road. Based on the equations above, the network average nearest neighbor distance is 399 meters. This result is more than double the value obtained when calculating the average nearest neighbor distance in continuous space of 187 meters, indicating that the density of points is greater in continuous space than network space.

In order to calculate the actual network average nearest neighbor statistic, the SANET toolbox for ArcGIS was implemented (Okabe, Okunuki, and SANET Team 2009). Figure 1-8 displays the results which can be interpreted in similar fashion to Ripley's K-function graph (1981). The first observation to note in the results is that the cumulative points do not add up to the expected value of 586. This is because SANET does not account for the redundancy of the spatial attribute in the dataset. Events occurring at identical locations are considered to be the same point. In the chart, the expected point distribution and the distributions for significant clustering and dispersion at the 1% and 5% confidence intervals were derived through 10,000 Monte Carlo simulations of the probability distribution function given previously. While Figure 1-8 does not give a specific value for the network average nearest neighbor statistic, it can be determined that significant clustering occurs for those events whose nearest neighbor is located within 1300 meters.

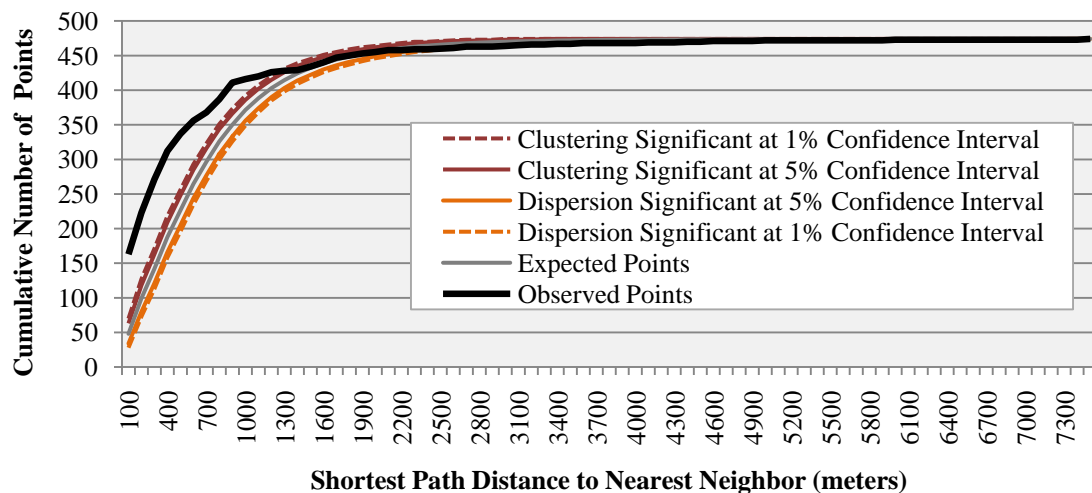


Figure 1- 8. SANET's Network Nearest Neighbor Distance Tool results.

This spatial clustering analysis offers several insights into the nature of the spatial distribution under study (see Table 1-2). The first observation is that the continuous and network distribution of traffic fatalities in Franklin County are quite different. While the minimum, maximum, and range of nearest neighbor distance values are identical for both continuous and network measures (this is rare), the difference in the averages is considerable. Based on the tests performed above, significant global clustering does occur in both continuous and network space, but the z-value of the continuous test, when compared to the output of the network test, suggests that the clustering observed in continuous space is rarer than that observed in network space. These differences illustrate the importance of using appropriate spatial measures for the phenomenon under study. It is clear that using Euclidean measures for traffic collisions restricted to network space may lead to false conclusions when attempting to explain their distribution. Network measures should be used for network-based phenomena.

Table 1- 2. Continuous and network average nearest neighbor test results for traffic collisions in Franklin County, OH, January-March 2009.

Average Nearest Neighbor Test Results	Continuous Space	Network Space
Minimum Nearest Neighbor Distance	0 m	0 m
Maximum Nearest Neighbor Distance	7470 m	7470 m
Nearest Neighbor Distance Range	7470 m	7470 m
Average Nearest Neighbor Distance	187 m	399 m
Expected Nearest Neighbor Distance	748 m	See Figure 1-8
Average Nearest Neighbor Statistic	0.25	See Figure 1-8
Clustered/Random/Dispersed	Clustered	Clustered
z-score	-34.74	N/A
Probability (Q)	0	See Figure 1-8

Secondly, it is now possible to determine a sense for the scale at which the 586 traffic collisions are spatially dispersed within the Franklin County major road network. The closest neighbors are collocated, the most distant neighbors are 7470 meters apart, and the average distance between nearest neighbors is 399 meters within the network. Overall, the spatial distribution of traffic collisions in the study area demonstrates significant clustering. These findings now provide basis for making decisions about appropriate spatial critical distances when testing for spatiotemporal clusters. Since we know some collisions are collocated and that significant clustering exists up to 1300 meters, an acceptable range of appropriate spatial critical distances might exist between 0 and 1300 meters. A somewhat more conservative range is represented between the minimum nearest neighbor distance and the mean, or in this case, 0 and 399 meters. The latter range will be tested in Step 6.

Step 5: Test for Temporal Clustering

A temporal cluster is represented by the occurrence of a greater number of events than that expected during a particular portion of a specified time period. A simple way to test for clustering in time is by considering the time period as a single line in space, and then performing a linear (or network) nearest neighbor distance test according to a specified unit interval. Various linear nearest neighbor tests have been proposed, including those by von Neumann (1941), Pinder and Witherick (1973), Young (1982), and Okabe et al. (1995). For ease of evaluation, Young's statistic is chosen here, and the

results will be compared with those from Okabe et al.'s statistic above (Equations 1-6 through 1-11).

Young's statistic is given by:

$$M = \frac{\sum_{i=1}^n M_i}{L} \quad (1.12)$$

For points X_1, \dots, X_n along a line of length L , where $D_1 = X_1$ and $D_i = X_i - X_{i-1}$, and where M_i is the minimum value of D_i, D_{i+1} . A value of M close to 0 indicates clustering of the data points, while a value of M close to $n/(n + 1)$ indicates dispersion of the data points.

The expected value and variance of M follow:

$$E(M) = \frac{n}{2(n + 1)}, \quad Var(M) = \frac{2n - 1}{12(n + 1)^2} \quad (1.13, 1.14)$$

The z-value can be calculated:

$$z = \frac{M - E(M)}{\sqrt{Var(M)}} \quad (1.15)$$

Before performing the linear nearest neighbor test to assess temporal clustering within the test data, an appropriate temporal interval must be chosen. A simple technique

to determine a meaningful interval is to divide the total number of events in the dataset by the sum of the possible temporal intervals in the study period. This is demonstrated in Table 1-3. Redundancy of events within a temporal interval may cause a loss of information when searching for temporal clustering, while choosing an excessively small temporal interval may not add any information. Based on the results of Table 1-3, hours will be used as the temporal interval for this study.

Table 1- 3. Temporal interval analysis for traffic collisions in Franklin County, OH, January-March, 2009.

Time Interval	Number of Intervals in Study Period (January-March 2009)	Number of Events per Interval (586 Collisions)
Month	3	195.33
Week	12.9	45.43
Day	90	6.51
Hour	2160	0.27
Minute	129600	0.005

A comparison of results for Young and Okabe et al.'s statistic for the test case are given in Table 1-4. While the two tests use different standardization techniques, the results are nearly identical. Young's test standardizes the statistic by the time period of study, in this case 2160 hours (90 days), while Okabe et al.'s statistic uses the total number of points in the test distribution, 586. Consequently, the derivations of the first two moments differ for each test. Both tests, however, suggest that the temporal distribution of traffic collisions in Franklin County between January and March 2009 is

slightly more clustered than the expected random temporal distribution and statistically significant.

While the tests do provide very similar results, the benefit of using Okabe et al.'s statistic is the ability to observe the average and expected nearest neighbor distance as a function of the temporal interval. As was done in Step 4, from the results here, a basis can be established from which to determine an appropriate range of temporal critical distance values when testing for spatiotemporal clusters in Step 6. Again, using the smallest nearest neighbor distance between traffic collisions as a minimum, and the average nearest neighbor distance as the maximum, an acceptable range of temporal critical distance values is between 0 and 1.65 hours. This range will be tested in the next step.

Table 1- 4. Results for two linear nearest neighbor statistics given traffic fatalities in Fairfax County, VA, 2004-2008.

Linear Nearest Neighbor Test	Okabe et al.'s Test	Young's Test
Minimum Nearest Neighbor Distance	0 hours	0 hours
Maximum Nearest Neighbor Distance	13 hours	13 hours
Nearest Neighbor Distance Range	13 hours	13 hours
Average Nearest Neighbor Distance	1.65 hours	N/A
Expected Nearest Neighbor Distance	1.87 hours	N/A
Average Nearest Neighbor Statistic	0.88	0.44
Clustered/Random/Dispersed	Clustered	Clustered
z-value	-1.46	-3.4
Probability (Q)	0.072	0.0003

Step 6: Test for Spatiotemporal Clustering

The object of this penultimate step is to test for spatiotemporal clustering in the phenomena under study, that is whether or not there are an excess number of events occurring within some geographic space that are also unexpectedly close in time. While the literature discusses a variety of spatiotemporal clustering methods (as shown above), the Knox method is still perhaps the most straight-forward and widely used today, a network-based application which is the focus of this study. A detailed explanation of the general method can be found in Cliff and Ord (1981) and with examples in Upton and Fingleton (1985). The derivations for the expected value and variance of the statistic were first described in David and Barton's (1966) assessment of Knox's work and are provided here for reference as well.

The Knox method involves the construction of two event proximity matrices with the dimensions of $n \times n$ for n events. The first matrix defines spatial proximity where a 1 is included in some cell X_{ij} if event i occurred within some critical spatial distance δ of event j and 0 otherwise. The second matrix defines temporal proximity where a 1 is included in some cell Y_{ij} if event i occurred within some critical temporal distance τ of event j and 0 otherwise. For both matrices, if $i = j$, then the entry is 0. The Knox statistic is then obtained by the cross-product:

$$R_{\delta\tau} = \sum_{i=1}^n \sum_{j < i} X_{ij} Y_{ij} \quad (1.16)$$

If the events are completely independent spatially and temporally, then there is no space-time interaction and $R_{\delta\tau} = 0$. For rendering simplicity, $R_{\delta\tau}$ is hereafter written as R .

The expected value for R can be found by:

$$E(R) = \frac{S_0 T_0}{n(n-1)} \quad (1.17)$$

where $S_0 = \sum_i \sum_j X_{ij}$ ($i \neq j$), $T_0 = \sum_i \sum_j Y_{ij}$ ($i \neq j$), and n is the number of events.

The variance of R is:

$$\begin{aligned} Var(R) = & \frac{S_1 T_1}{2n^{(2)}} + \frac{(S_2 - 2S_1)(T_2 - 2T_1)}{4n^{(3)}} \\ & + \frac{(S_0^2 + S_1 - S_2)(T_0^2 + T_1 - T_2)}{n^{(4)}} - \{E(R)\}^2 \end{aligned} \quad (1.18)$$

where $S_1 = \frac{1}{2} \sum_i \sum_j (X_{ij} + X_{ji})^2$ ($i \neq j$); $S_2 = \sum_i (X_{i0} + X_{0i})^2$; $X_{i0} = \sum_j X_{ij}$

$X_{0i} = \sum_j X_{ji}$; $n^{(2)} = n(n-1)$; $n^{(3)} = n(n-1)(n-2)$;

$n^{(4)} = n(n-1)(n-2)(n-3)$.

where the formulae for T_0 , T_1 , T_2 are identical in form to those for S_0 , S_1 , S_2 with the exception that X is replaced by Y wherever it occurs in the formulae.

When the number of events is large (greater than 30), the normal approximation may be assumed, however studies by Mielke (1978) and Siemiatycki (1978) identify potential exceptions which emphasize the approximate nature of this assumption.

In order to calculate the value of the test statistic for a normal approximation, follow:

$$z = \frac{|R - E(R)| - 1}{\sqrt{Var(R)}} \quad (1.19)$$

In order to implement this method for the data under study, the critical parameters defining the spatiotemporal clusters for the test must be determined. Unlike spatiotemporal studies in epidemiology, where critical parameters can be defined by the known etiology of disease, for the case of traffic collisions, critical parameters are difficult to discern and relative to the spatial and temporal processes involved at the area and time of study. Instead of methodically implementing every possible combination of spatial and temporal critical distances for the study data, a technique for determining an acceptable range of critical spatial and temporal distances has been presented in Steps 4 and 5 of this process. Based on Step 4, a conservative range of acceptable spatial critical distances was defined between 0 and 399 meters. Likewise, in Step 5, a range of acceptable temporal critical distances was defined between 0 and 1.65 hours. The results of testing for spatiotemporal clusters using the GIS tool, *SCAn* (the focus of Part 3 of this

research), are presented in Table 1-5, where the spatial critical distance range was tested at 100 meter intervals and the temporal critical distance range at 1 hour intervals.

While it is commonly accepted that a statistic is significant when its probability of occurrence is less than 5%, or $\alpha = 0.05$, in this case a Bonferroni correction is implemented because multiple tests (hypotheses) are being performed on the same statistic. The adjusted Bonferroni correction used here was suggested by Simes (1986) and the procedure is described further in Part 2 of this work. For Table 1-5, only the highlighted Knox R values meet the significance criteria of $Q \leq$ the Bonferroni correction for $\alpha = 0.05$.

Table 1- 5. Spatiotemporal clusters, Knox R, for the given spatial and temporal critical distance ranges and associated statistical significance for traffic collisions in Franklin County, OH, January-March, 2009. Highlighted values are significant where $Q \leq$ the Bonferroni correction for $\alpha = 0.05$.

Critical Spatial Distance (meters)	Critical Temporal Distance (hours)	Knox-R Value	Probability (Q)	Bonferroni Correction for $\alpha = 0.05$
0	0	0	0.233	0.020
	1	1	0.498	0.050
	2	2	0.216	0.013
100	0	1	0.408	0.030
	1	2	0.291	0.023
	2	3	0.222	0.017
200	0	1	0.483	0.043
	1	2	0.413	0.033
	2	3	0.368	0.027
300	0	1	0.473	0.040
	1	2	0.489	0.047
	2	3	0.464	0.037
400	0	5	0.000003	0.003
	1	6	0.004	0.007
	2	8	0.005	0.010

Using the GIS tool *SCAN*, presented in Part 3 of this thesis, it is possible to map the spatiotemporal clusters that contribute to the observed Knox R values. In order to illustrate this capability, the traffic collisions contributing to the significant spatiotemporal clustering at a critical spatial distance of 400 meters and a critical temporal distance of 0 hours (meaning the events occurred within the same hour) are presented in Figure 1-9. The attributes associated with the mapped traffic collisions representing spatiotemporal clusters in Figure 1-9 are presented in Table 1-6.

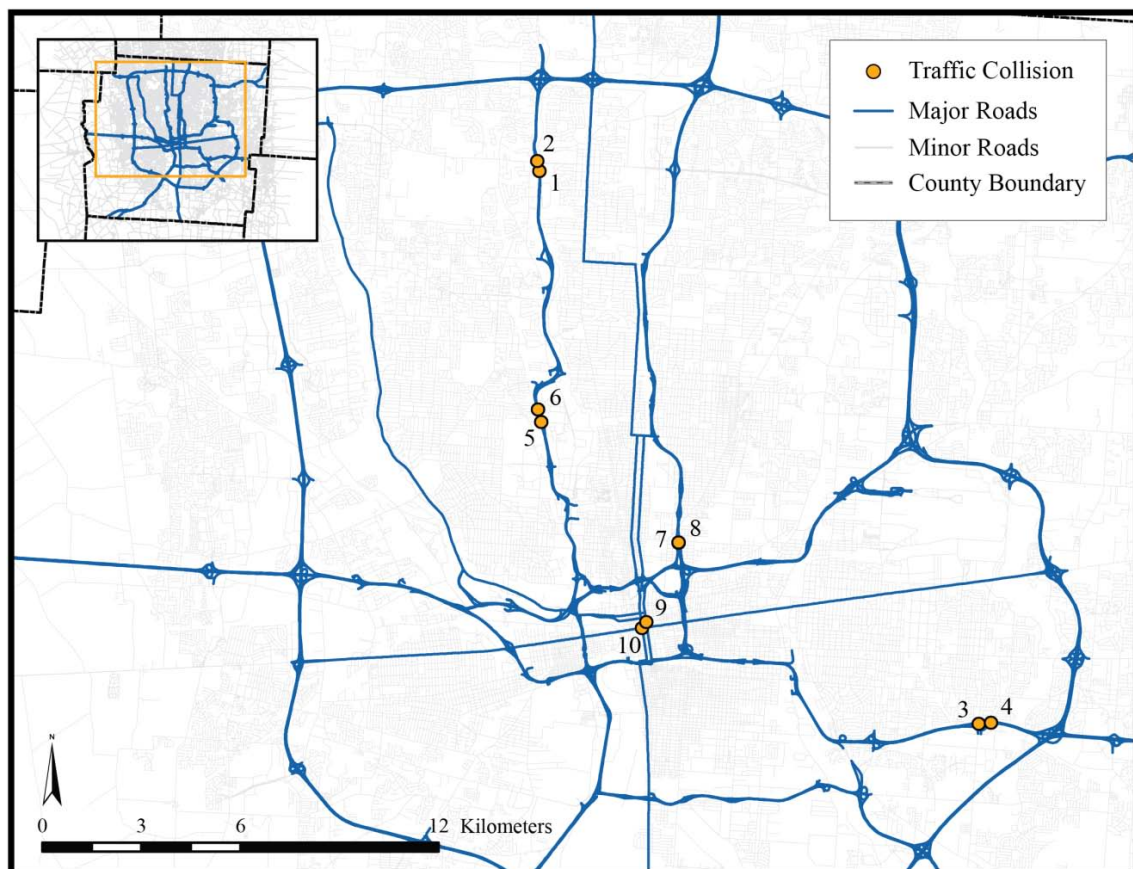


Figure 1- 9. Map of traffic collisions contributing to spatiotemporal clusters defined by a spatial critical distance of 400 meters and a temporal critical distance of 0 hours (occurred during the same hour) in Franklin County, OH, January-March, 2009.

Table 1- 6. Selected attribute values for traffic collisions contributing to spatiotemporal clusters in Figure 1-9. Weather attribute was derived from www.wunderground.com(2009).

ID	Date	Day of Week	Time	Weather
1	1/16/2009	Friday	4:35:00 PM	Record low temp (-14 degrees)
2	1/16/2009	Friday	4:15:00 PM	Record low temp (-14 degrees)
3	1/20/2009	Tuesday	9:00:00 AM	Record low temp (-1 degree)
4	1/20/2009	Tuesday	9:05:00 AM	Record low temp (-1 degree)
5	1/26/2009	Monday	12:06:00 PM	Snow
6	1/26/2009	Monday	12:27:00 PM	Snow
7	1/30/2009	Friday	6:44:00 PM	Snow
8	1/30/2009	Friday	6:44:00 PM	Snow
9	2/26/2009	Thursday	6:45:00 PM	Rain (0.05 inches)
10	2/26/2009	Thursday	6:13:00 PM	Rain (0.05 inches)

An examination of Figure 1-9 and Table 1-6 reveals likely spatial and temporal processes contributing to the existence of the observed spatiotemporal clusters. In the case of Figure 1-9, the spatiotemporal clusters depicted on the map are located in the vicinity of major intersections or access/exit ramps to multi-lane highways or freeways, locations where vehicles are abruptly changing travel speed and/or lanes. Table 1-6 indicates that the spatiotemporal clusters occurred during periods of extreme weather in every case, and during weekday rush hour traffic in four out of five clusters.

Step 7: Explain the Results

An important final step of this comprehensive process involves using any available resources to explain the results observed in the preceding steps. There may be obvious conclusions to be made or new questions may emerge requiring further investigation.

In the test case presented here, the fact that significant clustering was observed in both the spatial and temporal dimensions indicates that there are underlying spatial and temporal processes effecting the distribution of traffic collisions both in the network space and during the temporal period of study. A further examination of clustering in the spatial dimension should reveal the general areas or specific locations on the network where the greatest number of collisions have occurred, possible factors being restricted traffic flow, an intersection of traffic lanes, or poor surface maintenance. Similarly, a further investigation into the strictly temporal clustering observed in the test data will reveal the temporal periods with the greatest number of traffic collisions, likely the result of peak traffic flow, reduced visibility, or temporally-based environmental changes, such as extreme weather. The added level of analysis provided by testing for spatiotemporal clusters reveals those specific locations where traffic collisions occur in rapid succession. The results presented above suggest that periods of extreme weather create conditions within specific portions of the network where traffic collisions are prone to happen, which in this study are areas at which traffic merges or intersects. This result may not be surprising, but it illuminates a risk that can be further investigated. While the temporal aspect of extreme weather cannot be controlled, the contributing spatial factors can be examined and may lead to spatial solutions that can mitigate the temporal risk, such as improving the road surface, erecting signage, or reducing traffic flow to the area. It is apparent from these findings that a spatiotemporal cluster analysis of traffic collisions could be implemented as a tool in prioritizing research, maintenance, and development on a given transportation network. Conversely, a spatiotemporal cluster analysis may be

useful in assessing the relative safety of a transportation network by observing an absence of overall spatiotemporal clustering or by observing those portions of the network where clustering has not been exhibited.

Conclusions

This research has presented a comprehensive process for the spatiotemporal clustering analysis of network-based phenomena. A thorough implementation of the process described will provide a greater understanding of the spatial, temporal, and spatiotemporal distribution of the phenomenon under study. While the determination of spatial and temporal critical distance parameters for the Knox test may be intuitive in some studies, when it is not, an effective technique has been described here that identifies spatiotemporal clustering based on a range of nearest neighbor distance values in both space and time. This process did not address the statistical theory of the methods described, however the ensuing Part 2 will highlight a few specific concerns. Neither did this process describe the GIS tools used to implement the process described. These details are addressed in Part 3. Through the example of the traffic collision phenomenon examined here, a compelling case for the usefulness of a network-based extension of Knox method has been presented, and the potential benefit of such spatiotemporal analysis has been explained.

PART 2

An Examination of Significance Tests and Critical Parameters for Network-Based Spatiotemporal Cluster Analysis

Introduction

In statistical testing, perhaps the most important element and greatest challenge is determining the significance of the statistical finding. Generally speaking, statistical significance is based on how closely the observed result compares to the distribution of the expected result. If the observed result is uncommon when compared to the expected distribution, it is said to be significant. The distribution of the expected result typically follows some type of stochastic model describing the process under examination. Often, the normal distribution is used as the sampling distribution, although it is not always an appropriate assumption, especially when sample sizes are very small.

This challenge of determining significance is relevant to the Knox method used in this study. While significance tests for the Knox method have been developed for the chi-square distribution, the normal distribution, and Monte Carlo methods, the normal distribution has been primarily used for this research partly based on the assumption that GIS users often work with large sample sizes. Large sample sizes tend to conform to the normal distribution, making a significance test that conforms to a normal approximation acceptable here. While Monte Carlo simulations may create the best reference distribution for significance testing, because the assumption of large sample sizes is

made, the generation of Monte Carlo simulations can become quite time consuming, and are therefore not used as the primary significance test in this study. For a reference purpose, however, each significance test is described here, and the results of all three tests are presented for comparison (see Table 2-2). The ability to perform each test is provided within the *SCAn* tool described in Part 3 of this work, so that a decision on which significance test to be implemented can be based on the desired accuracy and time available for performing the test. Regardless of the significance test selected for the Knox method, if multiple iterations of the test are performed on the same dataset, then the probability levels determining significance should be adjusted. This is commonly accomplished through Bonferroni methods which will be discussed here.

Additionally, as part of the comprehensive process described in Part 1 of this research, a technique was presented for determining spatial and temporal critical distances in the execution of spatiotemporal testing. Provided here are the significance test results for multiple ranges of critical parameters derived from the nearest neighbor distance test in space and time. These findings determined the recommendation to use a range of critical parameter values between the minimum and the average nearest neighbor distance for both the spatial and temporal domains.

The organization of Part 2 follows: first, the datasets used as a basis for comparison in this part of the study are described; next, three significance testing methods used for the Knox test are presented, with a discussion on adjusting significance levels when performing multiple tests; finally, an examination of the process for

determining a range of critical parameters for spatiotemporal cluster testing using nearest neighbor distance values is provided.

Study Areas and Datasets

For the purposes of comparing the varying results of the statistical significance tests described hereafter, multiple datasets are examined. There are a total of seven datasets representing two different study areas used in this portion of the thesis. The two study areas represented are the major road network of Franklin County, Ohio, depicted previously in Figure 1-1, and the major road network of Eastern Fairfax County, Virginia, depicted in Figure 2-1. Within the Franklin County study area, four datasets representing actual injury-causing traffic collisions are examined, one for each successive three-month period during the year of 2009. An additional randomly generated dataset is presented for this study area, created using SANET's random point generator tool (Okabe, Okunuki, and SANET Team 2009) to define the event locations on the network, and a random number generator to define event time stamps.

Within the Fairfax County study area, two datasets are presented. One contains the fatality-causing traffic collisions during the five-year period between 2004 and 2008 and one contains randomly generated values within the same spatial and temporal constraints as the observed data. Table 2-1 lists characteristics of each dataset successively. Note the difference in point density between the two study areas.

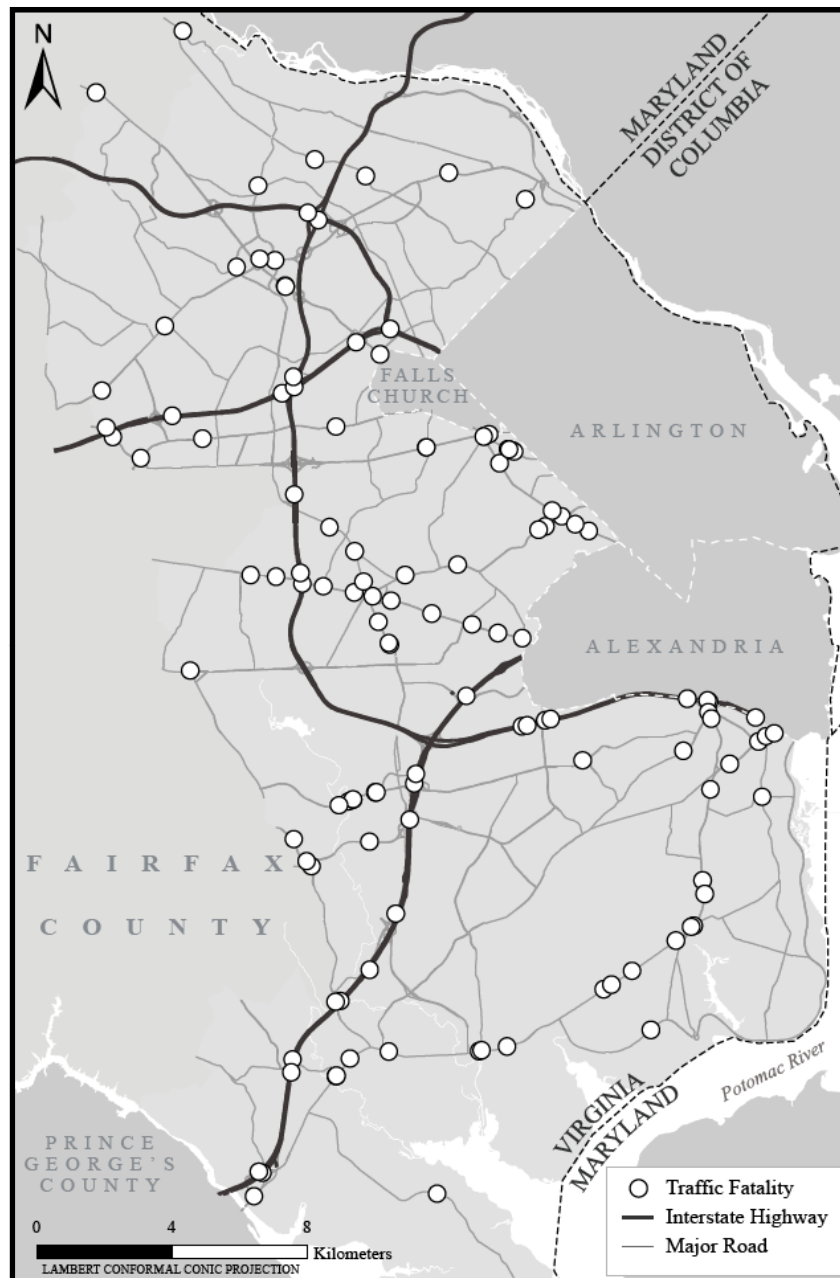


Figure 2- 1. Fatality-causing traffic collisions on major roads in Fairfax County, Virginia, 2004-2008.

Table 2- 1. Characteristics of datasets used in Part 2 of this study.

Minimum Nearest Neighbor Distance	Network Distance (km)	Temporal Period (days)	Number of Events	Spatial/Temporal Event Density
Franklin Co. Collisions (Jan-Mar '09)	930	90	586	0.63 / 6.51
Franklin Co. Collisions (Apr-Jun '09)	930	91	671	0.72 / 7.37
Franklin Co. Collisions (Jul-Sep '09)	930	92	653	0.70 / 7.10
Franklin Co. Collisions (Oct-Dec '09)	930	92	698	0.75 / 7.59
Random Set (Franklin Co. Network)	930	90	698	0.75 / 7.76
E. Fairfax Co. Fatalities ('04-'08)	950	1827	125	0.13 / 0.07
Random Set (E. Fairfax Co. Network)	950	1827	125	0.13 / 0.07

Significance Tests for the Knox Method

Chi-square and Poisson Distributions

Because the pairings derived through the Knox method can be summarized in a two by two contingency table, the chi-square test has been suggested as a means of testing for the significance of the statistic (Knox 1964; Jacquez 1996). The contingency table is established such that:

		Space	
		$\leq \delta$	$> \delta$
Time	$\leq \tau$	a	b
	$> \tau$	c	d

where δ is the critical spatial distance, τ is the critical temporal distance, a is the value of spatiotemporal pairs, b is the value of temporal pairs, c is the value of spatial pairs, and d

is the value of all other pairs. In the contingency table above, the value of a is the Knox statistic, R .

Chi-square (χ^2) is then calculated by:

$$\chi^2 = \sum_i \sum_j \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \quad (2.1)$$

where O_{ij} is the observed value in cells a through d of the contingency table and E_{ij} is:

$$E_{ij} = \frac{R_i \times C_j}{N} \quad (2.2)$$

with R_i the row sum for the observed value O_{ij} , C_j the column sum for the observed value O_{ij} , and N the grand total number of observations, $a + b + c + d$.

In this instance the probability of χ^2 may be determined from a chi-square distribution table with one degree of freedom. Generally speaking, the higher the χ^2 value, the more rare the result. A large χ^2 value indicates that somewhere in the contingency table, the observed frequencies for a given cell differ markedly from the expected values, although the χ^2 value does not indicate which cell (or cells) are contribute to the observed effect (Anon. 2010a). Baker (1996) notes that because a majority of the terms in the χ^2 contingency table will result from the squared differences

between observed and predicted numbers of close pairs over distances much larger than the specified critical distances, the power of the χ^2 test is reduced.

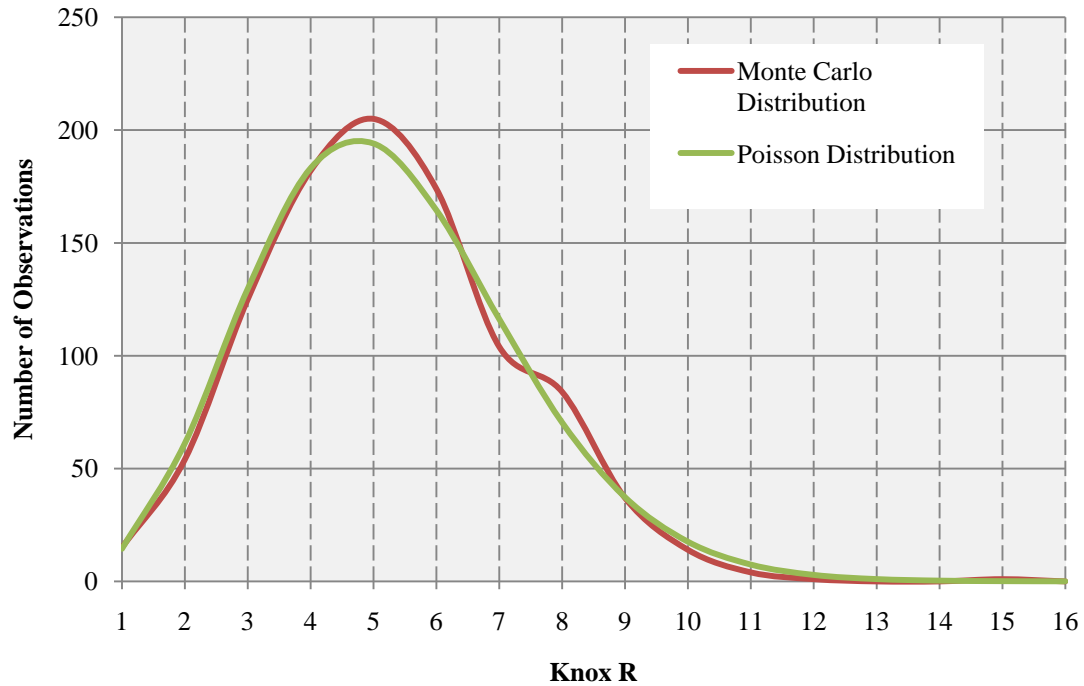


Figure 2- 2. Comparison of the reference distribution generated by 1000 Monte Carlo simulations of spatiotemporal clusters in traffic collisions in Franklin County, OH, January-March 2009, where $\delta = 400$ meters and $\tau = 2$ hours. The Poisson distribution is generated from the reference distribution mean of 4.24.

If the value of a , or the Knox statistic, R , is very small, then it has been demonstrated that the significance of the value may be directly calculated using a single-tailed Poisson distribution where the mean is equal to E_{ij} above (Knox and Gilman 1992; David and Barton 1966). The Poisson tendency of the Knox statistic distribution is characterized by the data of this study as well (see Figure 2-2). Table 2-2 shows how the probability of the Knox statistic based on the χ^2 distribution, and the Poisson distribution compare to the other significance tests presented in this study. It should be noted that

although the Poisson distribution and the Monte Carlo distribution generated in Figure 2-2 are nearly identical, the derived probabilities of the observed Knox statistic based on these distributions is different, as reported in Table 2-2. This unexpected difference is due to how the probabilities are calculated in each case.

Normal Distribution

Developed for the Knox test initially by David and Barton (1966), the formulae for significance testing according to a normal approximation have been previously presented in Equations 1.17 – 1.19. Details of these calculations are also provided in Upton and Fingleton (1985). When sample sizes are large (generally greater than 60 events) then this is an appropriate distribution for significance testing. The probability of observing the Knox statistic according to this normal approximation (Q) is used throughout this study. As previously mentioned, comparison of the probabilities of the Knox statistic based on the various distributions discussed here can be found in Table 2-2. From this table, it is apparent that the Knox statistic probability based on the normal approximation most closely resembles the probability based on the Monte Carlo generated reference distribution (where space labels are shuffled). As the Monte Carlo distribution provides the best representation of possible Knox values for a given test, the fact that the normal and Monte Carlo probabilities are similar is further evidence in support of using the normal approximation when an expedient significance test is required.

Monte Carlo Simulations

While originally suggested by Knox (1964), Mantel (1967) provided details for generating a reference distribution for the Knox statistic using Monte Carlo simulations. The process involves the repeated randomization of event labels, while calculating the Knox statistic for each iteration, until enough values have been generated to build an empiric distribution adequate for significance testing purposes. While there is no standard number of iterations required to create an “adequate” reference distribution, the literature suggests anywhere between 1000 to 10,000 repetitions to be sufficient (Mantel 1967; Baker 1996). In order to determine the probability value of the observed Knox statistic, the proportion of the right hand tail of the reference distribution whose simulated Knox values are equal to or greater than the original statistic is calculated.

In the execution of Monte Carlo simulations for the Knox test, the literature suggests that which labels are shuffled is immaterial (Mantel 1967; Baker 1996; Jacquez 1996). Through the examination of this technique using the data under study, it is apparent that shuffling the time labels while the space labels remain fixed, or shuffling the space labels while the time labels remain fixed, provide very similar results. However, if both time and space labels are shuffled concurrently, a very different reference distribution is generated which increases the probability of the observed statistic (see Figures 2-3 and 2-4). The theoretical explanation for this effect will not be explained here and is a subject for future research. For the GIS tool described in Part 3, the Monte Carlo test implemented there holds the time labels constant while shuffling the space labels.

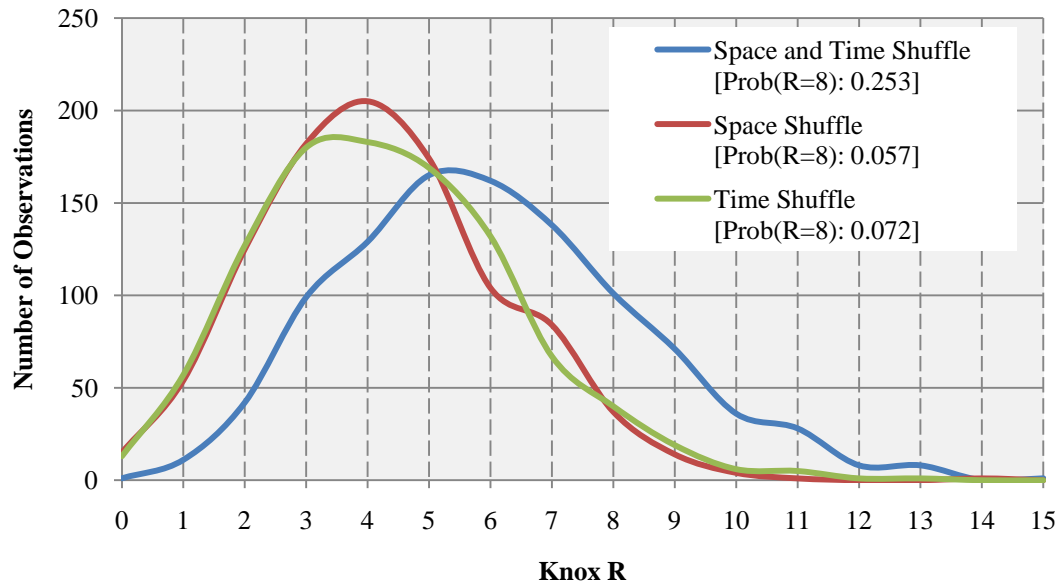


Figure 2- 3. Difference in reference distributions and probabilities generated by 1000 Monte Carlo simulations of the Knox statistic for traffic collisions in Franklin County, OH, January-March 2009, where $\delta = 400$ meters, $\tau = 2$ hours, and $R = 8$.

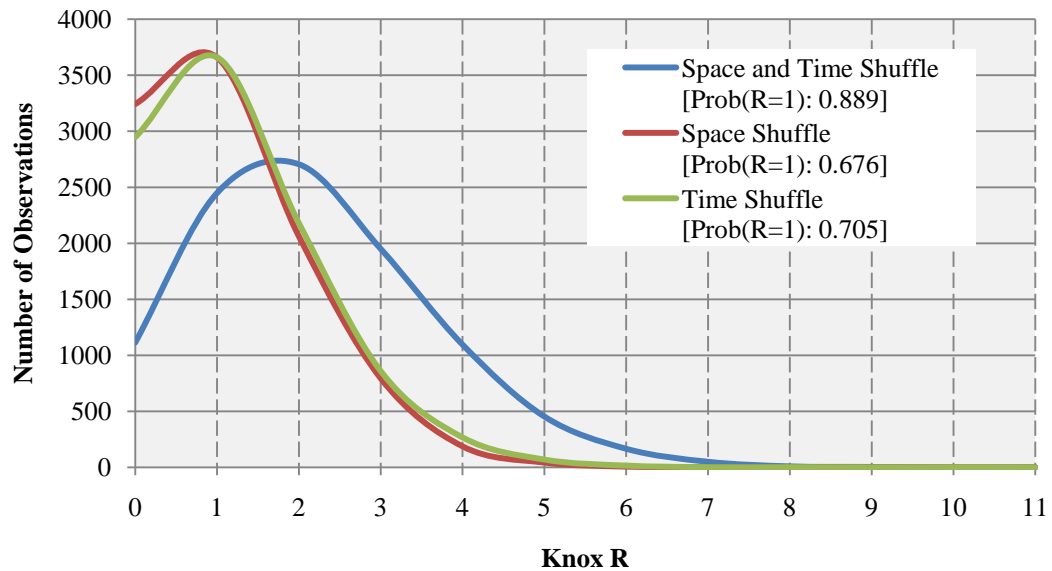


Figure 2- 4. Difference in reference distributions and probabilities generated by 10,000 Monte Carlo simulations of the Knox statistic for traffic collisions in E. Fairfax County, VA, 2004-2008 where $\delta = 1214$ meters, $\tau = 7$ days, and $R = 1$.

Table 2- 2. Comparison of probabilities for the observed Knox statistic given, the chi-square distribution, the normal distribution, and those distributions depicted in Figures 2-3 and 2-4.

*The space shuffled distribution is recommended and used in the tool described in Part 3.

Distribution	Franklin Co. (Jan-Mar '09) $\delta = 399 \text{ m}, \tau = 2 \text{ hours}; R = 8$	E. Fairfax Co. ('04-'08) $\delta = 1214 \text{ m}, \tau = 7 \text{ days}; R = 1$
Chi-square	0.022	0.254
Chi-square (Poisson)	0.147	0.331
Normal	0.030	0.444
Space Shuffled	0.057	0.676
Time Shuffled	0.072	0.705
Space and Time Shuffled	0.253	0.889

Multiple Testing

In order to maintain statistical rigor in the query for significant test results performed as described by the methodology in Part 1, methods for adjusting the level at which the observed statistic is determined significant should be considered. In the case of ESDA, Jacquez (2008) suggests that Bonferroni (Sidak 1967; Simes 1986), Holmes (Holland and Copenhave 1987) or Hochberg (1988) methods may be implemented. This research deals with the Bonferroni techniques. While the traditional Bonferroni adjustment involves dividing the desired α by the number of tests performed, or α/n , this correction can be seen as excessively conservative and improved methods have been suggested. Simes (1986) provides justification for a Bonferroni modification where the probability values of all iterations of the performed test are ordered, $P_{(1)}, \dots, P_{(n)}$. The

null hypothesis for a given iteration is rejected if:

$$P_{(j)} \leq \frac{j\alpha}{n} \quad (2.3)$$

where $j = 1, \dots, n$.

This method was employed in Table 1-5 of Part 1 which is reprinted here in Table 2-3 so that the traditional and improved Bonferroni adjustments can be compared. The strictness of the traditional Bonferroni adjustment is readily apparent when compared to the results of the modified adjustment.

Table 2- 3. Spatiotemporal clusters, Knox R, for the given spatial and temporal critical distance ranges and associated statistical significance for traffic collisions in Franklin County, OH, January-March, 2009. Highlighted values are significant where $Q \leq$ the Bonferroni correction for $\alpha = 0.05$.

Critical Spatial Distance (meters)	Critical Temporal Distance (hours)	Knox-R Value	Probability (Q)	Modified Bonferroni Correction for $\alpha = 0.05$	Traditional Bonferroni Correction for $\alpha = 0.05$
0	0	0	0.233	0.020	0.003
	1	1	0.498	0.050	0.003
	2	2	0.216	0.013	0.003
100	0	1	0.408	0.030	0.003
	1	2	0.291	0.023	0.003
	2	3	0.222	0.017	0.003
200	0	1	0.483	0.043	0.003
	1	2	0.413	0.033	0.003
	2	3	0.368	0.027	0.003
300	0	1	0.473	0.040	0.003
	1	2	0.489	0.047	0.003
	2	3	0.464	0.037	0.003
400	0	5	0.000003	0.003	0.003
	1	6	0.004	0.007	0.003
	2	8	0.005	0.010	0.003

Knox Method Critical Parameters

An assumption made in network-based spatiotemporal clustering, is that the closer events are together in space and time, the more meaningful their relationship. It is therefore desirable to identify the smallest critical parameters possible that reveal a significant result. In order to accomplish this in an objective manner, the distribution of events in relationship to their nearest neighbors is considered. Since the Knox method employed is concerned with the distance between points in space and time, considering critical parameters for space and time based on an examination of nearest neighbor distances in space and time seems logical, especially when acceptable critical parameters for the spatiotemporal test are unknown. The objective here is not to determine a specific individual value for the critical parameters in space and time, but rather to determine an acceptable range of critical parameters to apply to the spatiotemporal test. While methods have been published for identifying the most significant result within a given range of spatiotemporal parameters (Baker 1996), methods for determining an acceptable range of values when they are unknown have not.

In order to evaluate this methodology, nearest neighbor distance calculations were performed on the spatial and temporal attributes of the datasets under study. The lower bound for the parameter range is intuitively the minimum nearest neighbor distance, as no pair of events can be closer together than this distance. The value in question, however, is the upper bound. Both the average nearest neighbor distance and the maximum nearest neighbor distance are considered. Since the eventual spatiotemporal testing will involve multiple tests requiring a Bonferroni adjustment, it is desirable for the critical parameter

range to remain as small as possible in order to reduce the total number of tests performed.

Table 2-4 reveals the results of tests for clustering in the spatial and temporal dimensions as described in Step 4 and Step 5 of Part 1. Tables 2-5 through 2-7 show the results of spatiotemporal tests where the minimum, average, and maximum nearest neighbor distances were used as critical parameters for both space and time. In almost every case, when the average nearest neighbor distance was used as the critical parameter for space and time, the spatiotemporal statistic was significant (Table 2-5). Therefore, it seems appropriate to use the range of values between the minimum and average nearest neighbor distances as an initial range of inputs for spatiotemporal critical parameters, based on the assumption that if the upper bound of the range is significant, then it is possible that values below it will be significant as well. If this range does not produce a significant test result, then the next logical range to consider is between the average nearest neighbor distance (lower bound) and maximum nearest neighbor distance (upper bound). In this way, the impact of the Bonferroni correction is minimized.

Table 2- 4. Results of nearest neighbor distance cluster analysis for both the spatial and temporal dimensions of the given datasets.

Nearest Neighbor Distance Results	Spatial Dimension	Temporal Dimension
Franklin County Collisions (Jan-Mar '09)	Clustered	Clustered
Franklin County Collisions (Apr-Jun '09)	Clustered	Clustered
Franklin County Collisions (Jul-Sep '09)	Clustered	Clustered
Franklin County Collisions (Oct-Dec '09)	Clustered	Clustered
Random Set (Franklin County Network)	Random	Random
Eastern Fairfax County Fatalities ('04-'08)	Clustered	Random
Random Set (Fairfax County Network)	Random	Random

Table 2- 5. Comparison of the Knox statistic and associated probabilities calculated using the minimum nearest neighbor distance in space and time as the critical parameters. Highlighted values indicate $Q \leq \alpha = 0.05$.

Minimum Nearest Neighbor Distance	Spatial Measure	Temporal Measure	Knox R	Probability (Q)
Franklin County Collisions (Jan-Mar '09)	0 m	0 h	0	—
Franklin County Collisions (Apr-Jun '09)	0 m	0 h	0	—
Franklin County Collisions (Jul-Sep '09)	0 m	0 h	2	0.000
Franklin County Collisions (Oct-Dec '09)	0 m	0 h	2	0.012
Random Set (Franklin County Network)	12 m	0 h	0	—
Eastern Fairfax County Fatalities ('04-'08)	12 m	0 d	0	—
Random Set (Fairfax County Network)	112 m	0 d	0	—

Table 2- 6. Comparison of the Knox statistic and associated probabilities calculated using the average nearest neighbor distance in space and time as the critical parameters. Highlighted values indicate $Q \leq \alpha = 0.05$.

Average Nearest Neighbor Distance	Spatial Measure	Temporal Measure	Knox R	Probability (Q)
Franklin County Collisions (Jan-Mar '09)	399 m	1.65 h	7	0.030
Franklin County Collisions (Apr-Jun '09)	354 m	1.44 h	12	0.000
Franklin County Collisions (Jul-Sep '09)	344 m	1.47 h	9	0.005
Franklin County Collisions (Oct-Dec '09)	320 m	1.34 h	11	0.000
Random Set (Franklin County Network)	495 m	1.56 h	0	—
Eastern Fairfax County Fatalities ('04-'08)	1214 m	7 d	1	0.444
Random Set (Fairfax County Network)	1318 m	7.27 d	0	—

Table 2- 7. Comparison of the Knox statistic and associated probabilities calculated using the maximum nearest neighbor distance in space and time as the critical parameters. Highlighted values indicate $Q \leq \alpha = 0.05$.

Maximum Nearest Neighbor Distance	Spatial Measure	Temporal Measure	Knox R	Probability (Q)
Franklin County Collisions (Jan-Mar '09)	7470 m	13 h	449	0.062
Franklin County Collisions (Apr-Jun '09)	5516 m	15 h	408	0.192
Franklin County Collisions (Jul-Sep '09)	5396 m	21 h	551	0.311
Franklin County Collisions (Oct-Dec '09)	5224 m	20 h	749	0.001
Random Set (Franklin County Network)	3236 m	16 h	96	0.285
Eastern Fairfax County Fatalities ('04-'08)	8447 m	40 d	83	0.048
Random Set (Fairfax County Network)	8685 m	29 d	55	0.359

Note that, as expected, the randomly generated datasets do not demonstrate spatial or temporal clustering in Table 2-4. Nor do they demonstrate any significant spatiotemporal clustering in Tables 2-5 through 2-7. While this may be obvious, it is worth noting and reinforces the supposition that independent clustering in space and time is related to spatiotemporal clustering. This also seems to reinforce the importance of testing for clustering in space and time independently prior to proceeding with a spatiotemporal analysis. Although significant spatiotemporal clusters may exist when spatial and temporal clusters do not, a researcher may not wish to invest the time in a spatiotemporal cluster analysis if there is an absence of clustering in both space and time.

Conclusions

This statistical discussion has described various possibilities for significance testing of the Knox statistic. The Knox statistic probabilities based on these tests have been compared for two datasets in Table 2-2. While the Monte Carlo method provides the most accurate reference distribution, it becomes exceedingly time consuming for large datasets, and therefore the normal approximation seems to be an acceptable default. In generating a reference distribution from Monte Carlo simulations, it has been demonstrated that only the space labels or only the time labels should be shuffled, but not both simultaneously. Finally, further explanation has been provided for the determination of an appropriate range of critical parameters for network-based spatiotemporal cluster analysis. If clustering is present in the spatial and temporal dimensions of the dataset, using a range of critical parameters based on the minimum and average nearest neighbor distance for both space and time seems acceptable.

PART 3

***SCAn*: A Spatiotemporal Analysis Tool for Networks**

Introduction

This portion of the thesis outlines the tools of a GIS-based toolbox called *SCAn* (Spatiotemporal Cluster Analysis on a *network*) designed to perform spatiotemporal cluster analysis of network-based phenomena using the methods presented in Part 1 and Part 2. While other programs have been designed to analyze spatiotemporal clustering, few interface directly with ArcGIS and none employ network-based analyses. The intent of *SCAn* is to provide a simple network-based spatiotemporal cluster analysis tool that can be easily used by ArcGIS ArcMap users.

While *SCAn* is still under development, it currently provides the following tools:

Tool 1: ST Cluster Basic

Tool 2: ST Cluster Automatic

Tool 3: ST Cluster Table

Tool 4: ST Cluster Monte Carlo

Tool 5: ST Cluster Range Detector

The order of Part 3 follows: first, the basic requirements for using *nbSTAT* will be discussed; next, each tool will be described in turn; finally, program limitations and future developments will be addressed.

Program Requirements

The scripting language used to develop *SCAn* is Python 2.5. While most of the programming uses the ArcObject 9 geoprocessor and internal Python modules, there are two external Python modules utilized by *SCAn* tools and that must be downloaded to the user's desktop or server in order for the tools to function properly. These modules are *numpy* and *scipy* and both can be downloaded for free from the internet.

SCAn is designed for compatibility with ArcGIS 9 versions and higher. While *SCAn* will work with any license, it does depend on the Network Analyst extension which should be activated prior to using any of the described tools. A basic working knowledge of ArcGIS is required in order to make use of *SCAn*. All of the provided tools are accessed and run from within ArcToolbox. In order to utilize any of the tools described, four basic steps are required:

- 1) In ArcCatalog, create a personal geodatabase.
- 2) Within the geodatabase, create a feature dataset.
- 3) Within the feature dataset, create a network dataset from the shapefile(s) representing the network to be analyzed.
- 4) Within the feature dataset, create a feature class from the event data to be analyzed which must be located on the network. The feature class must have a field of type "date" which holds the dates of the features. If an hourly analysis is

desired, then a field must also be created within the feature class which holds a one or two digit value in 24-hour format pertaining to the event described (e.g. if the event occurred at 7:19 AM, then field should only include a “7”; if the event occurred at 7:19 PM, then the field should read “19”).

Tool 1: ST Cluster Basic

ST Cluster Basic performs a spatiotemporal cluster analysis using the Knox method described in Part 1 (see Figure 3-1). It requires the following user inputs:

- 1) The network dataset upon which analysis will occur.
- 2) The impedance attribute of the network. This is a user-defined attribute when creating a network dataset and can be found in the network dataset’s properties dialogue.
- 3) The feature class containing events to be analyzed.
- 4) The temporal granularity desired for the analysis (day or hour).
- 5) A selection of the feature class fields containing the date and hour information.
- 6) Critical spatial parameter as an integer in the units of the network dataset.
- 7) Critical temporal parameter as an integer in either days or hours.
- 8) The workspace in which to store the output feature class containing the events contributing to observed spatiotemporal clustering.

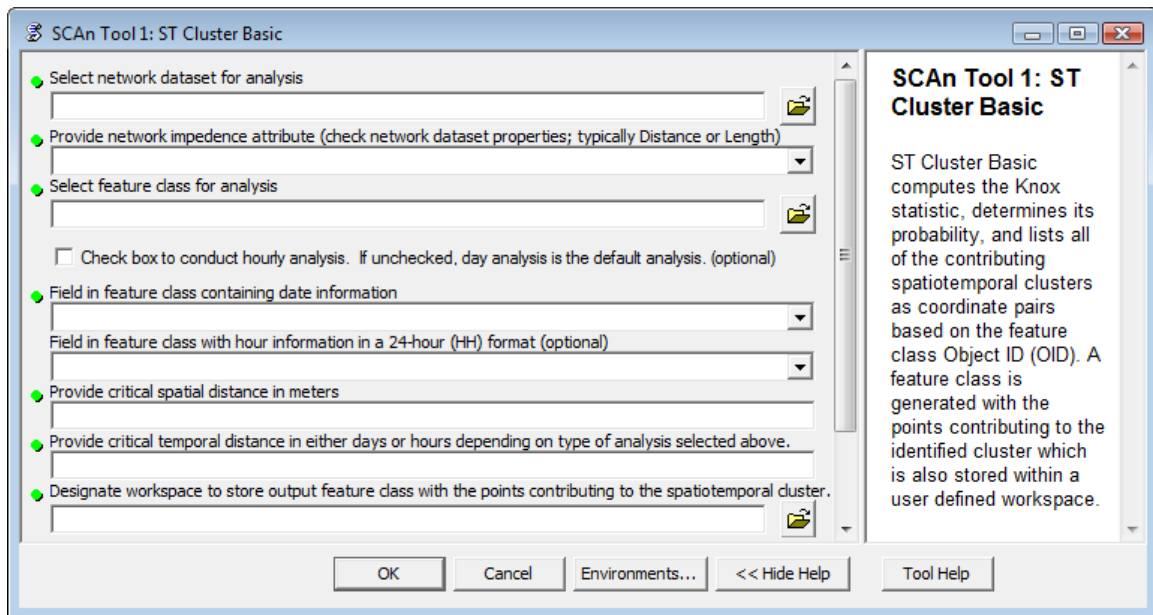


Figure 3- 1. User input screen for SCAn’s ST Cluster Basic.

Using the above inputs, ST Cluster Basic computes the Knox statistic, determines its probability according a chi squared and normal distribution, and prints all of the contributing spatiotemporal clusters as coordinate pairs based on the feature class Object ID (OID) to the tool’s dialogue screen (see Figure 3-2). This screen can subsequently be copied and pasted into a text or other file for future reference. Additionally, the events contributing to the observed clustering are saved in a feature class with the name “PointsInCluster_x_y,” where “x” is the given spatial parameter value and “y” is the given temporal parameter value. The feature class may subsequently be added to an ArcMap display for viewing or used for additional analyses.

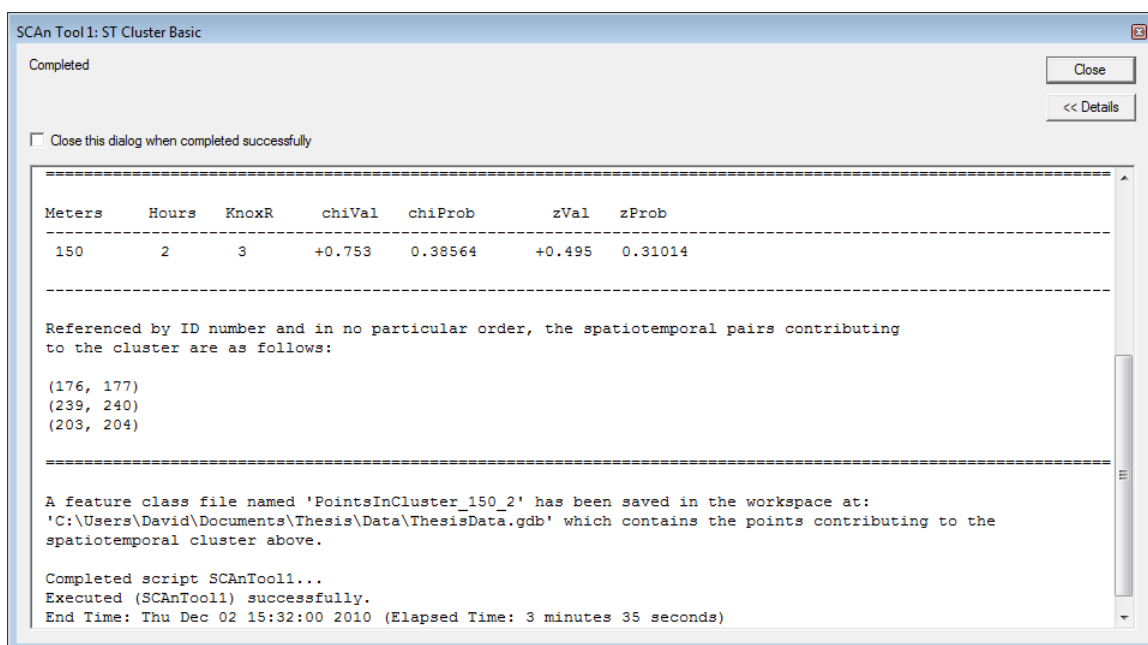


Figure 3- 2. The output dialogue screen for SCAn’s ST Cluster Basic.

Tool 2: ST Cluster Automatic

ST Cluster Automatic does not require the user specification of critical parameters (see Figure 3-3). This tool does require the following inputs:

- 1) The network dataset upon which analysis will occur.
- 2) The impedance attribute of the network. This is a user-defined attribute when creating a network dataset and can be found in the network dataset’s properties dialogue.
- 3) The feature class containing events to be analyzed.
- 4) The temporal granularity desired for the analysis (day or hour).
- 5) A selection of the feature class fields containing the date and hour information.

- 6) The desired significance level, α , by which to select a significant spatiotemporal cluster during the analysis.

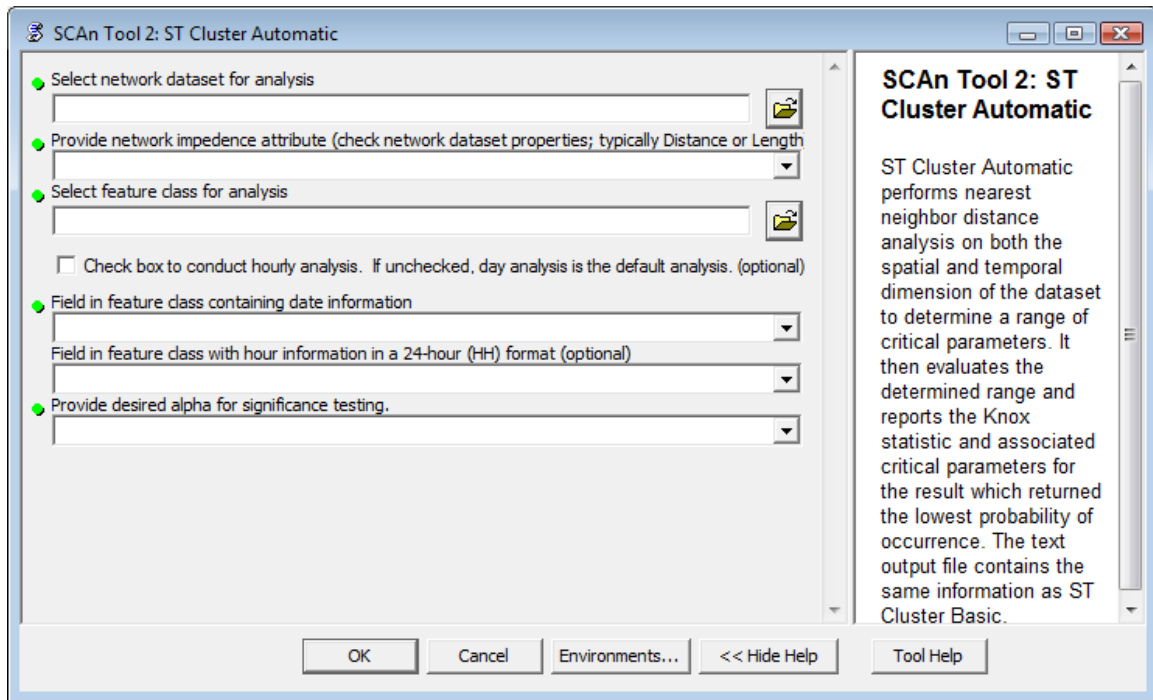


Figure 3- 3. The user input screen for SCAn's ST Cluster Automatic.

Using these inputs, ST Cluster Automatic performs nearest neighbor distance analysis on both the spatial and temporal dimension of the dataset to determine a range of critical parameters. It then evaluates the determined range and reports the Knox statistic and associated critical parameters for the result which returned the lowest probability of occurrence taking into consideration a Bonferroni correction based on the user provided significance level. The results are printed to the tool's output screen (see Figure 3-4).

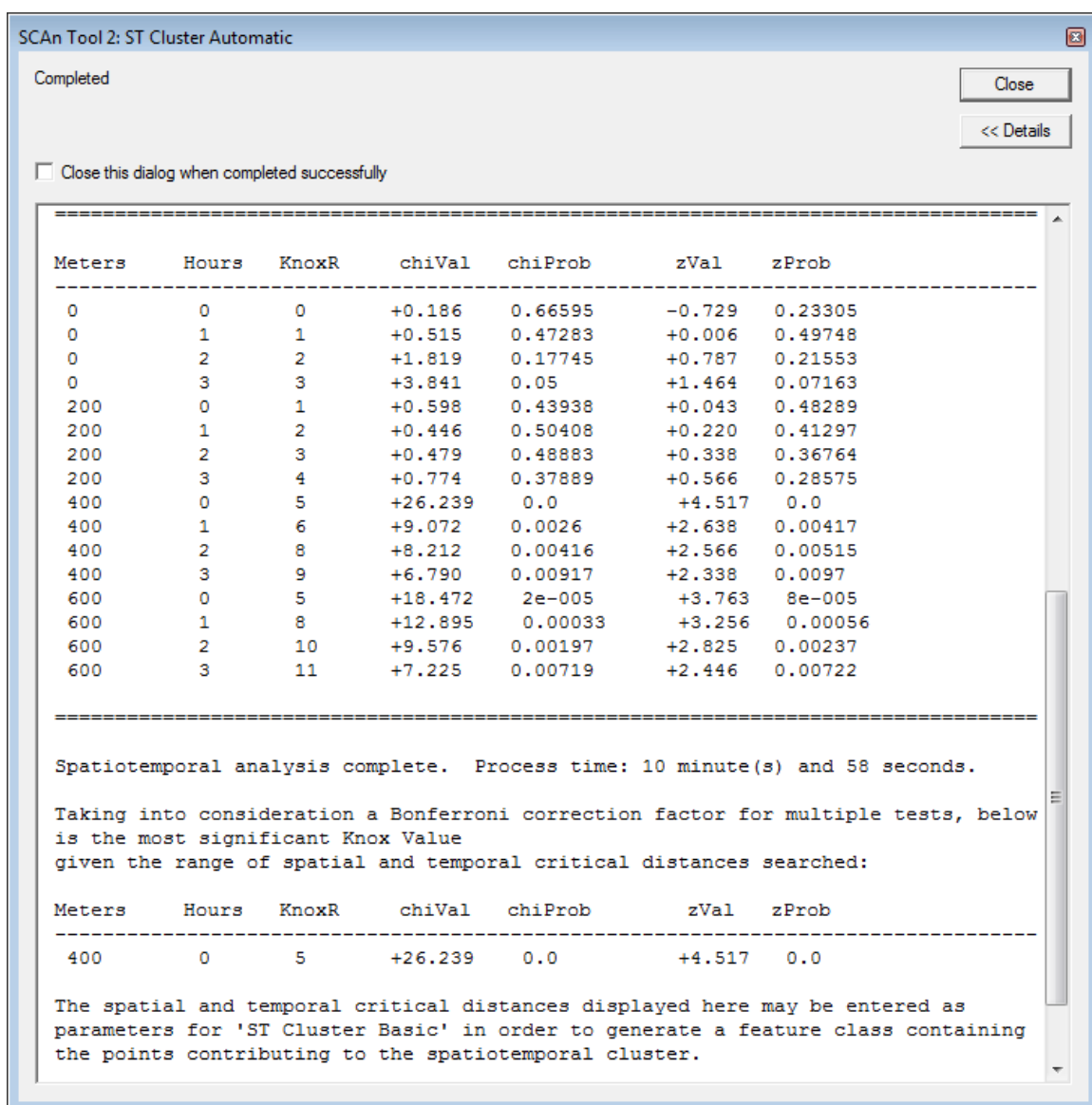


Figure 3- 4. Output dialogue for SCAn's ST Cluster Automatic.

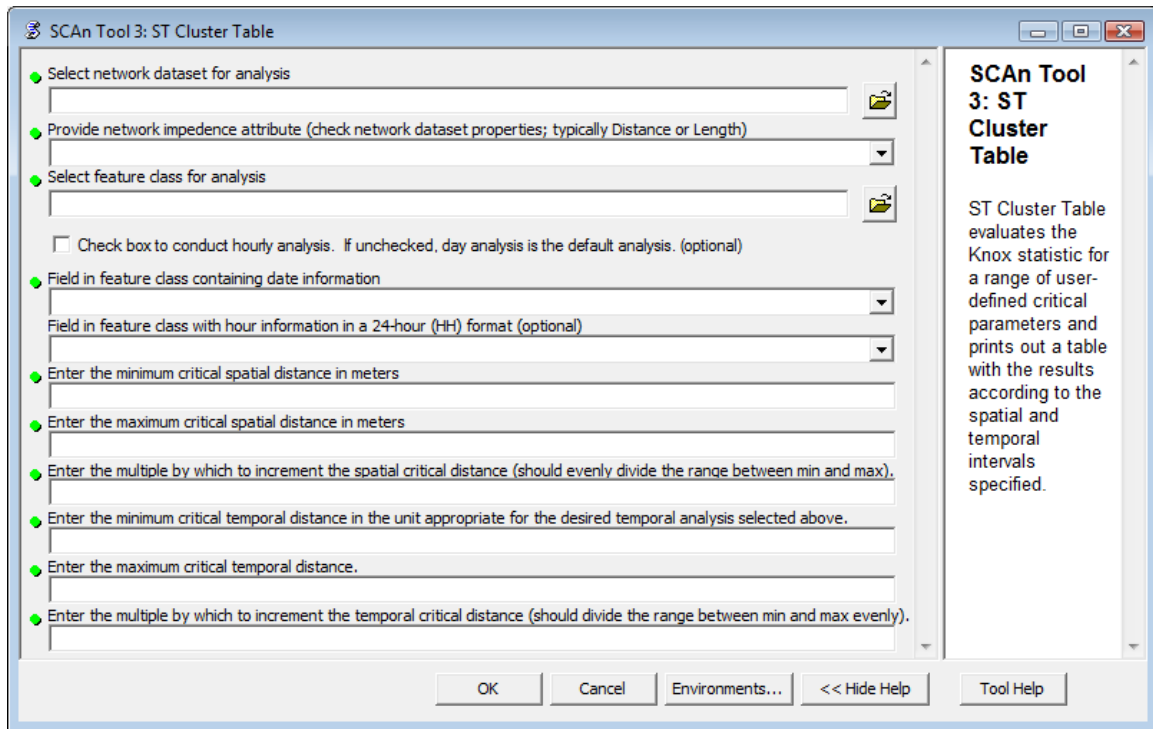


Figure 3- 5. Input screen for SCAn’s ST Cluster Table.

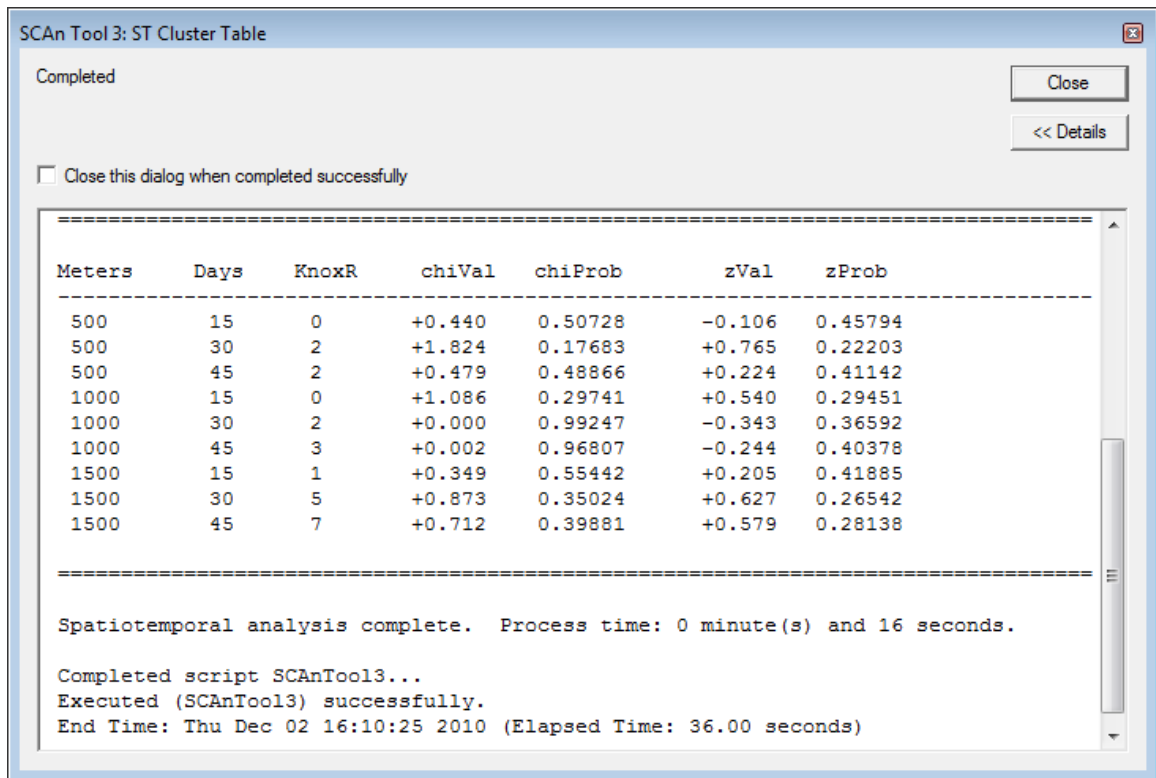
Tool 3: ST Cluster Table

ST Cluster Table evaluates the Knox statistic for a range of user-defined critical parameters and prints out a table with the results according to the spatial and temporal intervals specified (see Figure 3-5). It requires the following:

- 1) The network dataset upon which analysis will occur.
- 2) The impedance attribute of the network. This is a user-defined attribute when creating a network dataset and can be found in the network dataset’s properties dialogue.
- 3) The feature class containing events to be analyzed.
- 4) The temporal granularity desired for the analysis (day or hour).

- 5) A selection of the feature class fields containing the date and hour information.
- 6) Maximum, minimum, and incrementing interval for the critical spatial parameters in integer format according to the units of the network dataset.
- 7) Maximum, minimum, and incrementing interval for the critical temporal parameters as an integer in either days or hours.

The results are printed to the tool's output dialogue which may be copied to a text file or spreadsheet program for further analysis (see Figure 3-6).



SCAn Tool 3: ST Cluster Table

Completed

☐ Close this dialog when completed successfully

Meters	Days	KnoxR	chiVal	chiProb	zVal	zProb
500	15	0	+0.440	0.50728	-0.106	0.45794
500	30	2	+1.824	0.17683	+0.765	0.22203
500	45	2	+0.479	0.48866	+0.224	0.41142
1000	15	0	+1.086	0.29741	+0.540	0.29451
1000	30	2	+0.000	0.99247	-0.343	0.36592
1000	45	3	+0.002	0.96807	-0.244	0.40378
1500	15	1	+0.349	0.55442	+0.205	0.41885
1500	30	5	+0.873	0.35024	+0.627	0.26542
1500	45	7	+0.712	0.39881	+0.579	0.28138

Spatiotemporal analysis complete. Process time: 0 minute(s) and 16 seconds.

Completed script SCAnTool3...

Executed (SCAnTool3) successfully.

End Time: Thu Dec 02 16:10:25 2010 (Elapsed Time: 36.00 seconds)

Figure 3- 6. Output dialogue for SCAn's ST Cluster Table.

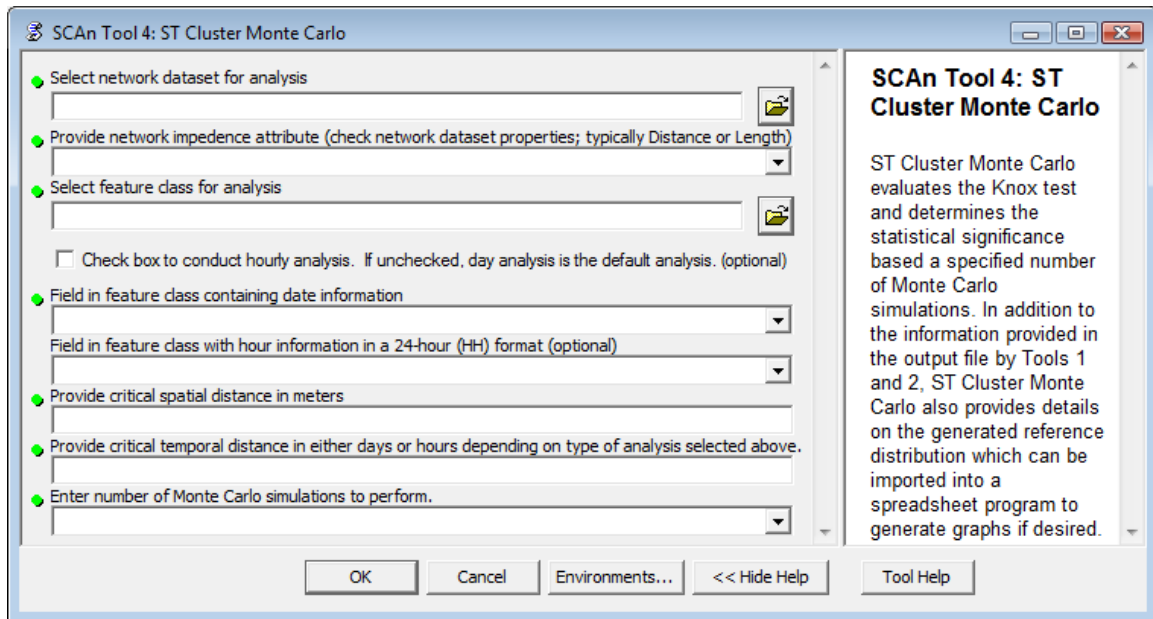


Figure 3- 7. Input screen for SCAn’s ST Cluster Monte Carlo.

Tool 4: ST Cluster Monte Carlo

ST Cluster Monte Carlo evaluates the Knox test and determines the statistical significance based a specified number of Monte Carlo simulations (see Figure 3-7). It requires the following:

- 1) The network dataset upon which analysis will occur.
- 2) The impedance attribute of the network. This is a user-defined attribute when creating a network dataset and can be found in the network dataset’s properties dialogue.
- 3) The feature class containing events to be analyzed.
- 4) The temporal granularity desired for the analysis (day or hour).

- 5) A selection of the feature class fields containing the date and hour information.
- 6) Critical spatial parameter as an integer in the units of the network dataset.
- 7) Critical temporal parameter as an integer in either days or hours.
- 8) Desired number of Monte Carlo simulations.

In addition to the information provided in the output screen by Tools 1 and 2, ST Cluster Monte Carlo's output (see Figure 3-8) also provides details on the generated reference distribution which can be imported into a spreadsheet program to generate graphs if desired (Figures 2-3 and 2-4 were generated in this manner).

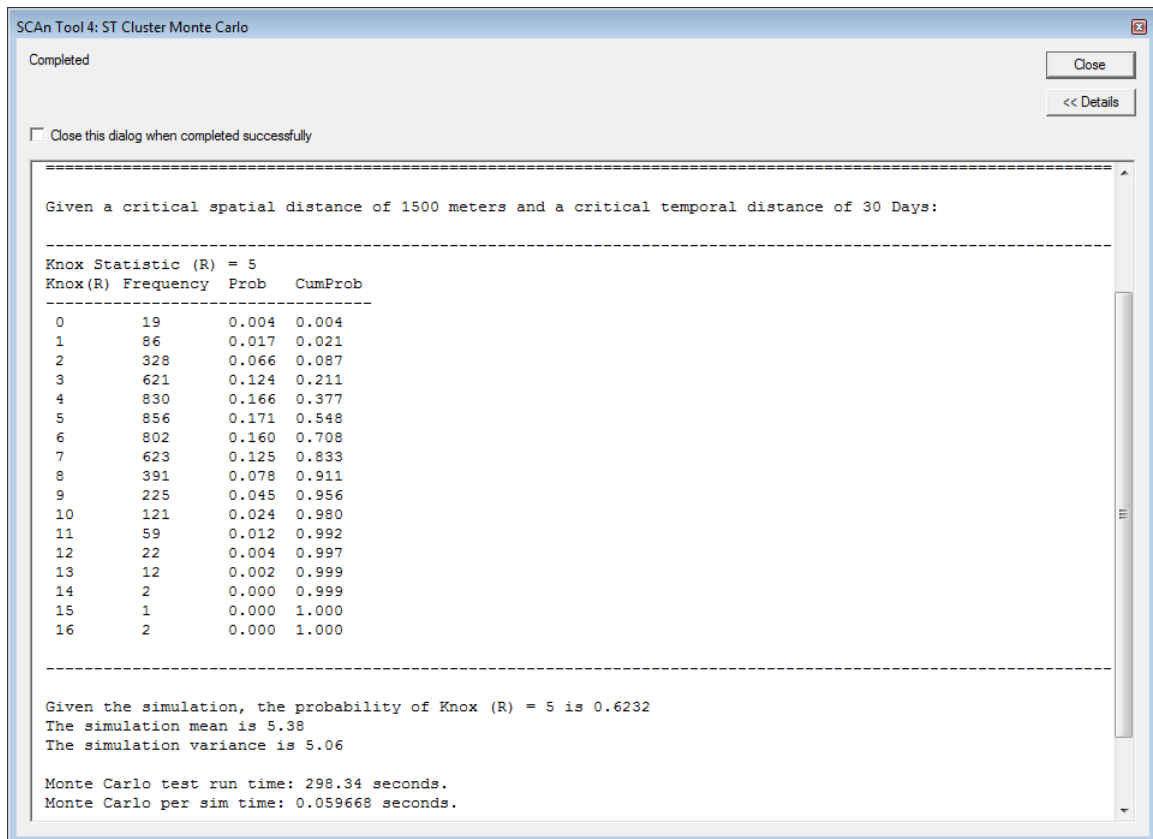


Figure 3- 8. Output dialogue for SCAn's ST Cluster Monte Carlo.

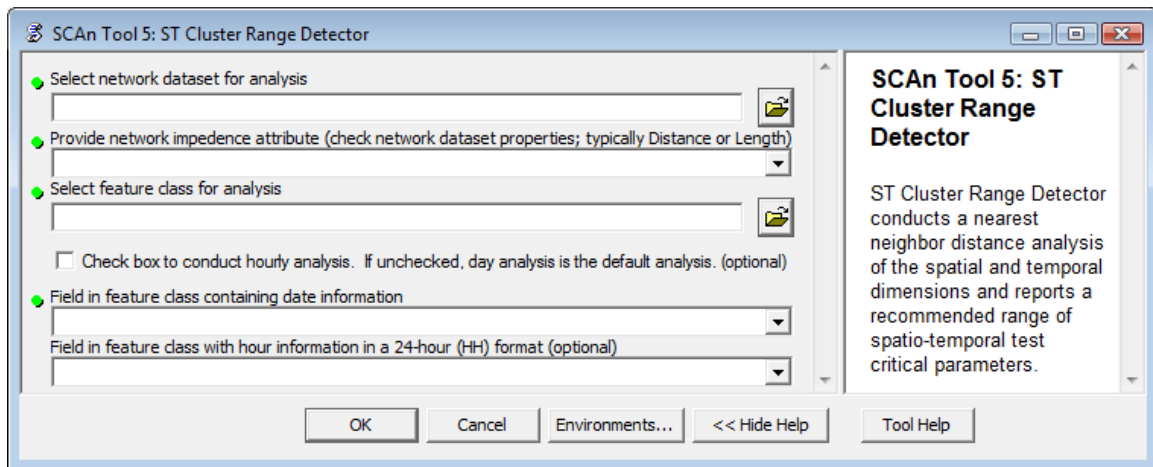


Figure 3- 9. User input screen for SCAn’s ST Cluster Range Detector

Tool 5: ST Cluster Range Detector

ST Cluster Range Detector conducts a nearest neighbor distance analysis of the spatial and temporal dimensions and reports a recommended range of spatiotemporal test critical parameters (see Figures 3-9 and 3-10). The user must provide:

- 1) The network dataset upon which analysis will occur.
- 2) The impedance attribute of the network. This is a user-defined attribute when creating a network dataset and can be found in the network dataset’s properties dialogue.
- 3) The feature class containing events to be analyzed.
- 4) The temporal granularity desired for the analysis (day or hour).
- 5) A selection of the feature class fields containing the date and hour information.

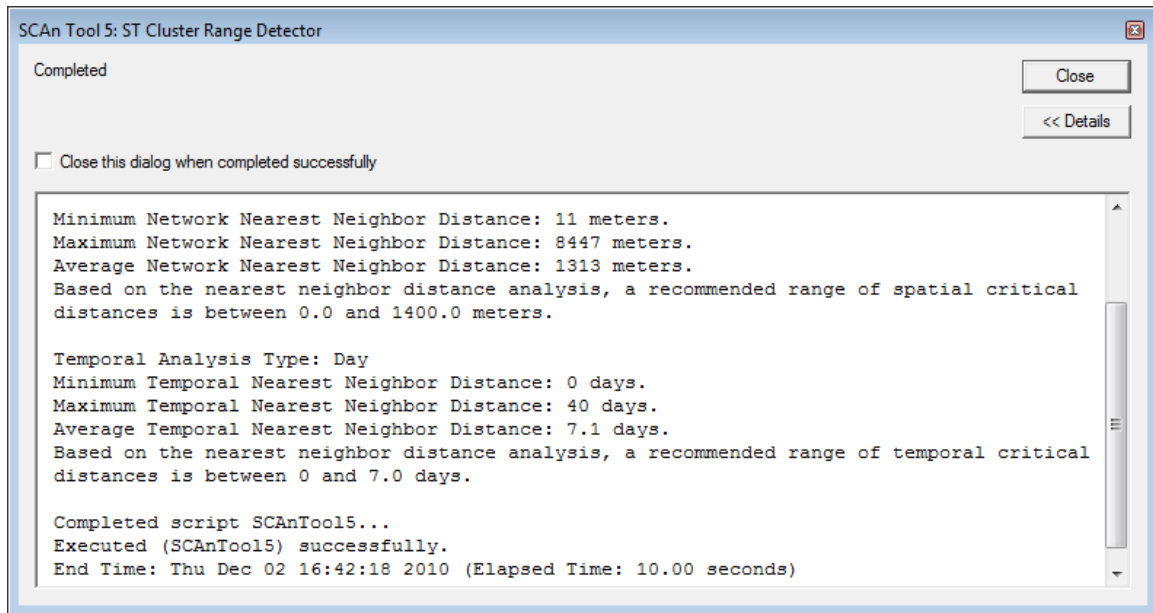


Figure 3- 10. Output dialogue for SCAn's ST Cluster Range Detector.

Program Limitations

While the program requirements described above are not complicated, they do demand some familiarity with ArcGIS and the Network Analyst extension. This may be seen as a limitation for some, but it can easily be overcome using the ArcGIS help menu.

Perhaps the greatest limitation of the *SCAn* toolset is the lengthy amount of time required to execute some tools when large feature classes are analyzed. A detailed analysis defining the time required for the execution of each tool based on the number of events in the analyzed feature class has not been conducted. However, a general observation is that for the Fairfax County dataset, consisting of 125 features, no tool took more than 10 minutes to complete. In fact, for the 125-feature dataset, every *SCAn* tool with the exception of Tool 4 executed in well under 5 minutes. The Franklin County

dataset with almost 600 features required much more time to execute. While Tool 1 and Tool 5 were executed in under 5 minutes, Tool 2 took 15 minutes and Tools 3 and 4 could take much longer depending on the parameters set.

Finally, the need to generate an OD Matrix Layer is in and of itself a limitation when large datasets need to be analyzed. This function is performed internally for each tool, and the *SCAN* tools have not been tested with feature datasets greater than 750 events. From past experience with ArcGIS version 9.3, system memory ran out when attempting to generate an OD Matrix Layer with more than 2000 events.

Future Program Developments

In addition to the capabilities described above, work continues to improve the tool outputs. Ideally, text files, feature classes or other GUI output will be optional outputs for each tool. Progress is also being made on continuous space versions of the tools mentioned here.

SCAN may be obtained from the author free of charge.

Recommendations for Future Research

Spatiotemporal analysis and network-based statistics continue to be areas of growing research. While there are numerous topics within this context that could benefit from focused attention, a few of particular interest to the network-based case of spatiotemporal cluster analysis are addressed here.

One of the improvements observed in continuous-based spatiotemporal cluster analysis is the result of work on the problem created by population shift bias (Mantel 1967; Klauber and Mustacchi 1970; Kulldorff and Hjalmar 1999). In the case of disease studies, when a population shift bias occurs, observed spatiotemporal clustering is likely the result of a change in the underlying geographic population distribution and not the result of some etiological process. A continuous-based solution provided by Kulldorff and Hjalmar (1999) involves linearly interpolating the annual observed population count for smallest sub-regions possible of the given study area and then randomly assigning cases proportionally to each sub-region. Monte Carlo methods are then used to generate an unbiased reference distribution of Knox statistic values for the randomized study area. This unbiased reference distribution is used to determine the significance of the observed Knox statistic.

It may be apparent that the network-based equivalent of population shift bias is a traffic flow bias. Traffic flow across the network has not been considered for this study,

but it seems like it could have a real effect on the significance of observed spatiotemporal clustering, especially in congested urban areas where traffic flow fluctuates at regular temporal intervals. With accurate traffic flow data or models based on real data, it should not be too complicated to extend continuous-based methods that account for population shift bias to the network flow case. This could improve the reliability of network-based spatiotemporal cluster significance testing, and seems worthy of further research.

As described in the introduction, this network extension of the Knox test is a global test, meaning it describes the distribution of points throughout the entire study area, without addressing the significance of local clusters. There are various global continuous space statistics, like Moran's I (1950), that have local counterparts (e.g., Local Indicators of Spatial Autocorrelation (LISA) (Anselin 1995; Ord and Getis 1995)). Local network spatial statistics have also been developed by Yamada and Thill (2007) and Shiode and Shiode (2009). Extension of a local statistic to network-based spatiotemporal cluster analysis appears to be a research area yet to be addressed, but could provide valuable insight for the traffic collision phenomenon when spatiotemporal clustering on congested portions of the network are of interest.

In various continuous space studies, spatiotemporal scan statistics have become popular, which implement a search surface in continuous space, while at the same time extending a vertical search through temporal space, in effect moving a three dimensional space-time search window throughout the study area to identify spatiotemporal clustering at varying scales (Rogerson 2001; Kuldorff 2001, 2006; Block 2007; Chang, Zeng, and Chen 2008; Assuncao and Correa 2009; Mirghani et al. 2010; Nakaya and Yano 2010;

Pei et al. 2010). Significance testing is conducted by implementing Monte Carlo methods. Not only can these techniques identify clusters regardless of predefined critical parameters, they have also been shown to identify clusters spatially as they emerge temporally (Jacquez 2008; Assuncao and Correa 2009). While there may not be intuitive applications for a network-based spatiotemporal scan statistic in traffic collision analysis, such an analysis might be especially beneficial in the study of such phenomena as IED incidence. In the combat zones of Iraq and Afghanistan, insurgents who emplace IEDs are known to relocate their operational areas or adjust their attack windows based on the effectiveness of coalition countermeasures, and a spatiotemporal scan statistic might be able to identify the spatial and temporal position of an insurgent relocation. Again, a network-based spatiotemporal scan statistic appears to be an area wide open to future research.

Finally, while a methodology was presented for determining an acceptable range for spatiotemporal cluster test critical parameters in space and time using nearest neighbor distance values derived from each dimension, a theoretical basis for this technique was not established. Exploring whether or not there is an objective relationship between the distances to nearest neighbors in space and time, and the observation of spatiotemporal clusters might be another informative area of research for spatiotemporal analysis.

APPENDIX

SCAn Python Scripts

Tool 1: ST Cluster Basic

```
#-----
#Name: SCAn Tool 1 - ST Cluster Basic
#Created by: David Eckley
#Date created: 20101201
#Purpose: ST Cluster Basic computes the Knox statistic, determines its
#probability, and lists all of the contributing spatiotemporal clusters
#as coordinate pairs based on the feature class Object ID (OID). A
#feature class is generated with the points contributing to the
#identified cluster which is also stored within a user defined
#workspace.
#-----

#=====
#IMPORT MODULES
#=====
import os, sys, string, arcgisscripting, math, numpy, scipy
from sets import Set
from time import*
from numpy import*
from scipy import*
from scipy import stats
from knoxStats import chiKnox, normalKnox

#=====
#INITIATE GEOPROCESSOR
#=====
gp = arcgisscripting.create(9.3)

#Overwrite any identical outputs
gp.overwriteoutput = 1

#=====
#DEFINE VARIABLES PROVIDED BY USER
#=====

#Network dataset
netDataset = gp.GetParameter(0)

#Impedence/cost attribute for the network dataset
impedence = gp.GetParameterAsText(1)

#Feature class with event details
tempFC = gp.GetParameter(2)
```

```

#Type of temporal analysis (boolean variable).If True, hour
#analysis. If False, day analysis.
timeType = gp.GetParameter(3)

#Label used for temporal column in results table
if timeType == 0:
    tempLabel = "Days"
else:
    tempLabel = "Hours"

#Feature class field containing dates of events
dateFld = gp.GetParameterastext(4)

#Feature class field containing hour event occurred in 24-hour HH
#format (e.g. 7AM is 7; 7PM is 19)
hourFld = gp.GetParameterastext(5)

#Ensure that hour data field was provided if hourly analysis was
#selected above. Otherwise, exit program.
if timeType > 0 and hourFld == "":
    gp.addmessage("")
    gp.addmessage("In order to conduct hourly analysis, a field\
containing hour data in a 24-hour HH format must be provided.")
    gp.addmessage("")
    sys.exit()
else:
    pass

#Critical spatial distance in meters. Convert to integer.
critDist = int(gp.GetParameter(6))

#Critical temporal distance in format described by boolean variable
#above.
critTime = int(gp.GetParameter(7))

#Workspace for the geodatabase in which to store the output feature
#class
outputGDB = gp.GetParameterAsText(8)

#Name given to output feature class
outputF = "PointsInCluster_"+str(critDist)+"_"+str(critTime)
#=====
#BUILD OD COST MATRIX LINES LAYER
#=====

gp.addmessage("")
gp.addmessage("Building Origin-Destination Cost Matrix Layer...")

th1 = clock()

#Using variables above and ArcToolBox tools
odLayer = "ODCostMatrix"

```

```

lineLyr = "ODCostMatrix\\Lines"
odLineLyr = "ODRoute_Layer"
gp.MakeODCostMatrixLayer_na(netDataset, odLayer, impedance, "", "", \
    "", "ALLOW_UTURNS", "", "NO_HIERARCHY", \
    "", "STRAIGHT_LINES")
gp.AddLocations_na(odLayer, "Origins", tempFC, "", "5000 Meters", "", \
    "", "MATCH_TO_CLOSEST", "APPEND", "SNAP", "5\
Meters")
gp.AddLocations_na(odLayer, "Destinations", tempFC, "", "5000 Meters", \
    "", "", "MATCH_TO_CLOSEST", "APPEND", "SNAP", \
    "5 Meters")
gp.Solve_na(odLayer, "SKIP")
gp.SelectData_management(odLayer, "Lines")
gp.MakeFeatureLayer_management(lineLyr, odLineLyr, "", "", "")

#Create variable from the OD Cost Matrix Lines Layer
spaceFC = odLineLyr

#Create variable from the Origin ID Field within the OD Cost Matrix
#Lines Layer
desc2 = gp.Describe(odLineLyr)
fldinfo2 = desc2.FieldInfo
spaceOrigIDfield = fldinfo2.GetFieldName(1)

#Create variable from the Destination ID Field within the OD Cost
#Matrix Lines Layer
spaceDestIDfield = fldinfo2.GetFieldName(2)

#Create variable from the Total Distance/Length Field within the OD
#Cost Matrix Lines Layer
spaceDistfield = fldinfo2.GetFieldName(4)

th2 = clock()

pt1 = th2-th1
ptlint = int(pt1)
ptlmin = int(pt1/60)
ptlsec = int(ptlint-(ptlmin*60))

gp.addmessage("Origin-Destination Cost Matrix Layer complete. Process\
time: " + str(ptlmin) + " minute(s) and " + str(ptlsec) + " seconds.")
gp.addmessage("")

#=====
#BUILD TEMPORAL MATRIX
#=====

gp.addmessage("Creating temporal matrix...")

th3 = clock()

#Define variable that will become the field name for timeline field
calcFld = "CALC_TIME"

```



```

#VB expression that will be used in the calculate field operation in
#hour analysis is conducted
expnHour = "["+str(dateFld)+"]*24+["+str(hourFld)+"]"

#VB expression that will be used in the calculate field operation in
#day analysis is conducted
expnDay = "["+str(dateFld)+"]*1"

#Local variables that will be used in the "make feature layer"
#operation; a prerequisite to calculating the temporal fields.
Output_Layer = "tempFC_L"
Output_Layer2 = "tempFC_L2"

#Assign boolean variable from user inputs as an integer
bool = int(timeType)

#Conduct calculate field operation if the user selects an hourly
#analysis
x = 1
if bool == x:

    # Make feature layer
    gp.MakeFeatureLayer_management(tempFC, Output_Layer)

    # Add timeline field and calculate timeline field value
    gp.AddField_management(Output_Layer, calcFld, "DOUBLE", "", "", \
        "", "", "NULLABLE", "NON_REQUIRED", "")
    gp.CalculateField_management(Output_Layer, calcFld, expnHour, \
"VB", "")

    #Assign variable to feature layer that will be used for temporal
    #analysis and matrix generation
    timeNN = Output_Layer

#Conduct calculate field operation if the user selects a day analysis
else:

    # Make feature layer
    gp.MakeFeatureLayer_management(tempFC, Output_Layer2)

    # Add timeline field and calculate timeline field value
    gp.AddField_management(Output_Layer2, calcFld, "DOUBLE", "", "", \
        "", "", "NULLABLE", "NON_REQUIRED", "")
    gp.CalculateField_management(Output_Layer2, calcFld, expnDay, \
"VB", "")

    #Assign variable to feature layer that will be used for temporal
    #analysis and matrix generation
    timeNN = Output_Layer2

#Rename variable assigned to feature layer
inputTimeFC = timeNN

```

```

#Assign variable to Object ID Field within feature layer
desc = gp.Describe(timeNN)
timeIDfield = desc.OIDFieldName

#Rename variable assigned to timeline field name
timeLinefield = calcFld

#Initiate the python dictionary which stores the featureclass "ID"
#field as a key and the corresponding timeline value
dctTime = {}

#Set a list variable to hold the first "ID" value in the feature class.
#This value will be used to standardize the sequential ordering of
#events to start at a value of "1"
adjustTimeID = []

#Initiate the geoprocessor cursor to extract the first value from the
#user-defined "ID" field and place it in the list variable initiated
#above
cur = gp.SearchCursor(inputTimeFC)
row = cur.Next()
firstTimeID = int(row.GetValue(timeIDfield))
adjustTimeID = [firstTimeID]

#Re-initiate geoprocessor cursor to search all rows in the feature
#layer containing temporal data
cur = gp.SearchCursor(inputTimeFC)
row = cur.Next()

#Store the user-defined temporal ID field and timeline field values
#into the initiated python dictionary
while row:
    #The dictionary key is set to the integer value in the "ID" field
    #of the row; this value is adjusted
    #to start at "1"
    key = (int(row.GetValue(timeIDfield)) - adjustTimeID[0]) + 1
    #Variable is given to the integer value in the field containing the
    #timeline information
    timelineDay = row.GetValue(timeLinefield)
    #The key is set to the respective timeline value for each row
    dctTime[key] = timelineDay
    #Cursor moves to next row in feature layer
    row = cur.Next()

#Initiate the python dictionary which stores the calculated time
#distances between every possible pair of events within the temporal
#feature layer. The key is an "ID" field coordinate pair, i.e. (12,55)
#and the corresponding calculated difference between the timeline
#distance of the coordinate pair is registered in the dictionary as
#the associated value

#STATISTICAL NOTE:
#In order to test the resulting Knox statistic for significance, the
#count of all possible combinations of pairings is necessary. While

```

```
#the Knox statistic is the count of all spatiotemporal pairs that fall
#inside the user-defined critical distances, it is also necessary to
#determine the number of spatiotemporal pairs that are inside the
#temporal distance but outside the spatial distance; inside the spatial
#distance but outside the temporal distance; and both inside the
#temporal and spatial distance. In order to determine these
#spatiotemporal pairings, dictionaries are established to capture those
#pairs that fall both inside and outside the temporal distance and both
#inside and outside the spatial distance.
```

```
#Initiate dictionary to capture those temporal pairs which are less
#than (LT) the user-defined critical distance
dctTimeLT = {}
```

```
#Initiate dictionary to capture those temporal pairs that are greater
#than (GT) the user-defined critical distance
dctTimeGT = {}
```

```
#STATISTICAL NOTE:
```

```
#In order to calculate the z statistic, it is necessary to know for a
#given critical temporal distance how many pairs each point has. For
#example, if point 101 is a pair with points 5, 18 and 24 for a given
#critical temporal distance, then point 101 would have a tally of 3.
#The following lists are initiated in order to record this tally for
#every point in the temporal data set.
```

```
timeKeyAlist = []
timeKeyAlistTally = []
```

```
#This variable is defined to register the count of the total number
#of point events being analyzed and to determine the stopping point for
#the subsequent loop operation
numEvents = len(dctTime)
```

```
#Incrementing variables are initiated at "1" because the temporal ID
#sequence was initiated to that value above. The x and y variables will
#be used to number the subsequent "ID" pairs generated by this loop.
x = 1
y = 1
```

```
#The loop will continue until the count of all analyzed points is
#reached. This loop calculates the interpoint temporal distance
#between every possible point combination.
```

```
while x <= numEvents:
    for key in dctTime:
        if key <= numEvents:
            #The temporal distance between the two values is calculated
            timeDif = abs(dctTime[y] - dctTime[x])
            #The temporal pair IDs are set to the current value of x
            #and y
            keyA = x
            keyB = y
            #Those pairs which fall inside the user defined critical
            #temporal distance are stored in the "LT" dictionary.
            #To ensure no duplicate pairings are in the dictionary,
```

```

#i.e. (23,42) and (42,23), the condition of keyA < keyB is
#set
if timeDif <= critTime and keyA < keyB:
    dctTimeLT[keyA, keyB] = timeDif
    #All keyA points are recorded in a list to facilitate
    #a tally of all keyA's
    timeKeyAlist.append(keyA)
#Those pairs which fall outside the user-defined critical
#temporal distance are stored in the "GT" dictionary
else:
    if timeDif > critTime and keyA < keyB:
        dctTimeGT[keyA, keyB] = timeDif
    else:
        pass
    y = y + 1
x = x + 1
y = 1

#The following loop tallies the total number of pairs each point has
#at a given critical temporal distance as well as performing some math
#which will be used by the KnoxStat module to calculate the z
#statistic.
x = 0
while x <= numEvents:
    #For the given x value, the tally of occurrence of that value in
    #the keyA list is assigned to a variable
    occurX = timeKeyAlist.count(x)
    #In accordance with the statistical formula for calculated z for
    #Knox, the following multiplication is calculated
    occurXx2 = occurX * 2
    occurXx2squared = occurXx2 ** 2
    #Each resulting value is stored in another list to be subsequently
    #summed for use in the calculation of z
    timeKeyAlistTally.append(occurXx2squared)
    x = x + 1
#The sum of all values in the list is assigned to variable used in the
#calculation of the z statistic
t2 = sum(timeKeyAlistTally)

#Count of time pairs which fall within the user-defined critical
#distance
numCritTimePairs = len(dctTimeLT)
#Count of time pairs which fall outside the user-defined critical
#distance
numNotCritTimePairs = len(dctTimeGT)

th4 = clock()
pt2 = th4-th3
pt2int = int(pt2)
pt2min = int(pt2/60)
pt2sec = int(pt2int-(pt2min*60))

```

```

gp.addmessage("Temporal matrix complete.  Process time: " + \
str(pt2min) + " minute(s) and " + str(pt2sec) + " seconds.")
gp.addmessage("")

#=====
#BUILD SPATIAL MATRIX
#=====

gp.addmessage("Creating spatial matrix...")

th5 = clock()

#Initiate the python dictionaries which stores the OriginID and
#DestinationID as a key and the corresponding network distance as the
#associated value from the OD Matrix Lines layer

#Stores all spatial pairs which fall inside user-defined spatial
#distance
dctSpaceLT = {}
#Stores all spatial pairs which fall outside user-defined spatial
#distance
dctSpaceGT = {}

#Set list variables to hold the first value in the OriginID and
#DestinationID fields of the OD Matrix "Lines" layer.
#These values will be used to standardize the sequential ordering
#of events to start at a value of "1"
adjustOrigID = []
adjustDestID = []

#Initiate the geoprocessor cursor to extract the first value in the
#OriginID and DestinationID fields of the OD Matrix "Lines" layer
#and place it in the list variables initiated above
cur = gp.SearchCursor(spaceFC)
row = cur.Next()
firstOrigID = int(row.GetValue(spaceOrigIDfield))
firstDestID = int(row.GetValue(spaceDestIDfield))
adjustOrigID = [firstOrigID]
adjustDestID = [firstDestID]

#STATISTICAL NOTE:
#In order to calculate the z statistic, it is necessary to know for a
#given critical spatial distance how many pairs each point has.  For
#example, if point 101 is a pair with points 5, 18 and 24 for a given
#spatial distance, then point 101 would have a tally of 3.  The
#following lists are initiated in order to record this tally for every
#point in the spatial data set.
spaceKeyAlist = []
spaceKeyAlistTally = []

#Initiate geoprocessor cursor to search the Lines Layer containing the
#spatial data of points being analyzed
cur = gp.SearchCursor(spaceFC)
row = cur.Next()

```

```

while row:
    #The dictionary key is set to the integer value in the respective
    #"ID" field of the row; these value are adjusted to start at "1"
    keyA = (int(row.GetValue(spaceOrigIDfield)) - adjustOrigID[0]) + 1
    keyB = (int(row.GetValue(spaceDestIDfield)) - adjustDestID[0]) + 1
    #Extract the associated value for the spatial distance between the
    #two points spaceDif = row.GetValue(spaceDistfield)Those spatial
    #pairs which meet the user-defined critical distance are entered
    #into the "LT" dictionary to ensure no duplicate pairings are in
    #the dictionary, i.e. (23,42) and (42,23), the condition of
    #keyA < keyB is set
    if spaceDif >= 0 and spaceDif <= critDist and keyA < keyB:
        dctSpaceLT[keyA, keyB] = spaceDif
        #All keyA points are recorded in a list to facilitate a tally of
        #all keyA's
        spaceKeyAlist.append(keyA)
    #Those spatial pairs that do not meet the user-defined critical
    #distance are entered into the "GT" dictionary
    else:
        if spaceDif > critDist and keyA < keyB:
            dctSpaceGT[keyA, keyB] = spaceDif
        else:
            pass
    row = cur.Next()

#The following loop tallies the total number of pairs each point has at
#a given critical temporal distance as well as performing some math
#which will be used by the KnoxStat module to calculate the z
#statistic.
x = 0
while x <= numEvents:
    #For the given x value, the tally of occurrence of that value in
    the
    #keyA list is assigned to a variable
    occurX = spaceKeyAlist.count(x)
    #In accordance with the statistical formula for calculated z for
    #Knox, the following multiplication is calculated
    occurXx2 = occurX * 2
    occurXx2squared = occurXx2 ** 2
    #Each resulting value is stored in another list to be subsequently
    #summed for use in the calculation of z
    spaceKeyAlistTally.append(occurXx2squared)
    x = x + 1
#The sum of all values in the list is assigned to variable used in the
#calculation of the z statistic
s2 = sum(spaceKeyAlistTally)

#Count of spatial pairs which fall within the user-defined critical
#distance
numCritSpacePairs = len(dctSpaceLT)

#Count of spatial pairs which fall outside the user-defined critical
#distance

```

```

numNotCritSpacePairs = len(dctSpaceGT)

th6 = clock()

pt3 = th6-th5
pt3int = int(pt3)
pt3min = int(pt3/60)
pt3sec = int(pt3int-(pt3min*60))

gp.AddMessage("Spatial matrix complete. Process time: " + str(pt3min)\
              + " minute(s) and " + str(pt3sec) + " seconds.")
gp.AddMessage("")
gp.AddMessage("")

#=====
#CONDUCT SPATIOTEMPORAL ANALYSIS
#=====

#Compare TimeLT and SpaceLT Dictionaries. If a pair is in both
#dictionaries, count the pair as a spatiotemporal pair. This is the
#Knox Statistic
x = 0
for key in dctSpaceLT:
    if dctTimeLT.has_key(key) == True:
        x = x + 1
#Count of spatiotemporal pairs which fall within the user-defined
#critical distances
knoxAPairs = x

#Compare TimeGT and SpaceLT. If a pair is in both dictionaries,
#count the pair as a spatiotemporal pair
x = 0
for key in dctTimeGT:
    if dctSpaceLT.has_key(key) == True:
        x = x + 1
#Count of spatiotemporal pairs which fall within critical spatial
#distance but outside critical temporal distance
knoxBPairs = x

#Compare TimeLT and SpaceGT. If a pair is in both dictionaries,
#count the pair as a spatiotemporal pair
x = 0
for key in dctSpaceGT:
    if dctTimeLT.has_key(key) == True:
        x = x + 1
#Count of spatiotemporal pairs which fall within critical temporal
#distance but outside critical spatial distance
knoxCPairs = x

#Compare TimeGT and SpaceGT Dictionaries. If a pair is in both
#dictionaries, count the pair as a spatiotemporal pair
x = 0
for key in dctSpaceGT:
    if dctTimeGT.has_key(key) == True:

```

```

        x = x + 1
#Count of spatiotemporal pairs which fall outside the user-defined
#critical distances
knoxDPairs = x

#STATISTICAL OUTPUTS:
#Set the variable to carry the chi squared value for the Knox
Statistic.
#Chi square is calculated according to "chiKnox" which is defined in
the
#knoxStats module.
knoxChiSig = chiKnox(knoxAPairs, knoxBPairs, knoxCPairs, knoxDPairs,\
                    numEvents)
#Probability of knoxChiSig
chiProb = round(scipy.stats.chisqprob(knoxChiSig, 1),5)

#Set the variable to carry the z value for the Knox Statistic. The z
#statistic is calculated according to "normalKnox" which is defined in
#the knoxStats module.
knoxZ = normalKnox(knoxAPairs, numCritSpacePairs, numCritTimePairs,\
s2, t2, numEvents)

#Probability of knoxZ
if knoxZ > 0:
    zProb = round((1 - scipy.stats.zprob(knoxZ)),5)
else:
    zProb = round((scipy.stats.zprob(knoxZ)),5)

#=====
#REPORT RESULTS
#=====

gp.AddMessage("=====\
=====")
gp.AddMessage("")
gp.AddMessage("Meters" + "\t" + tempLabel + "\t" + " KnoxR" + " \
chiVal" + " chiProb" + "\t" + " zVal" + "\t" + " zProb")
gp.AddMessage("-----\
-----")
gp.AddMessage(" %-8d %-5d %-5d %--0.3f %-12s %--0.3f %-12s\
% (critDist, critTime, knoxAPairs, knoxChiSig, chiProb,\
knoxZ, zProb))
gp.AddMessage("")
gp.AddMessage("-----\
-----")
gp.AddMessage("")
gp.AddMessage("Referenced by ID number and in no particular order,\
the spatiotemporal pairs contributing")
gp.AddMessage("to the cluster are as follows:")
gp.AddMessage("")

#Print to the dialouge screen all spatiotemporal pairs which fall
#within the user-defined critical distances (the Knox Statistic pairs)

```



```

clusterIDs = []

for key in dctSpaceLT:
    if dctTimeLT.has_key(key) == True:
        gp.AddMessage(" " + str(key) + " ")
        clusterIDs.append(key)

gp.AddMessage(" ")
gp.AddMessage("=====\\
=====")

#=====
#CREATE OUTPUT FEATURE CLASS
#=====

clusterEvents = []

#If there is no spatiotemporal clustering, write message to the screen
if len(clusterIDs) == 0:
    gp.addmessage(" ")
    gp.addmessage("Knox R = 0, therefore there are no points to export\\
to a cluster feature class.")
    gp.addmessage(" ")
else:
    pass

#Write ObjectIDs contributing to cluster to a list
i = 0
while i < len(clusterIDs):
    clusterEvents.append(clusterIDs[i][0])
    clusterEvents.append(clusterIDs[i][1])
    i = i + 1

#Remove duplicate IDs
clusterEventsSetA = set(clusterEvents)
clusterEventsSetB = set(clusterEvents)
uniqueEvents = clusterEventsSetA|clusterEventsSetB
uniqueEventsList = list(uniqueEvents)

#Generate a SQL statement to be used in the "select by attribute"
#function which will create the output feature class
sqlList = []
i = 0
while i < len(uniqueEventsList):
    statement = '' + str(timeIDfield) + ' = ' + str(uniqueEventsList[i])
    sqlList.append(statement)
    i=i+1

glue = " OR "
clusterSql = glue.join(sqlList)

#From the feature layer, select out only those points which are members
#of the spatiotemporal cluster
gp.SelectLayerByAttribute_management(inputTimeFC, "NEW_SELECTION",\

```

```

clusterSql)
#Create a new feature class from the selected points and save to user
#designated workspace
gp.FeatureClassToFeatureClass_conversion(inputTimeFC, outputGDB,\
                                         outputF)

gp.addmessage("")
gp.addmessage("A feature class file named '" + outputF + "' has been\
saved in the workspace at:")
gp.addmessage("'" + outputGDB + "' which contains the points\
contributing to the spatiotemporal cluster above.")
gp.addmessage("")

#Delete variables
del dctSpaceLT, dctSpaceGT, dctTime, dctTimeLT, dctTimeGT,\
    numCritTimePairs, numNotCritTimePairs, critTime, numEvents
del t2, s2, timeKeyAlist, timeKeyAlistTally, spaceKeyAlist,\
    spaceKeyAlistTally, knoxZ, knoxChiSig,
del knoxAPairs, knoxBPairs, knoxCPairs, knoxDPairs, numCritSpacePairs,\
    numNotCritSpacePairs, adjustOrigID, adjustDestID

```

Tool 2: ST Cluster Automatic

```
#-----
#Name: SCAN Tool 2 - ST Cluster Automatic
#Created by: David Eckley
#Date created: 20101130
#Purpose: ST Cluster Automatic performs nearest neighbor distance
#analysis on both the spatial and temporal dimension of the dataset to
#determine a range of critical parameters. It then evaluates the
#determined range and reports the Knox statistic and associated
#critical parameters for the result which returned the lowest
#probability of occurrence.
#-----

#=====
#IMPORT MODULES
#=====
import os, sys, string, arcgisscripting, math, numpy, scipy
from time import*
from numpy import*
from scipy import*
from scipy import stats
from kxStats import chiKnox, normalKnox

#=====
#INITIATE GEOPROCESSOR
#=====
gp = arcgisscripting.create(9.3)

gp.addmessage("")
gp.addmessage("Executing spatial and temporal critical range\
analysis...")
gp.addmessage("")

th1 = clock()

#=====
#DEFINE VARIABLES PROVIDED BY USER
#=====

#Network dataset
netDataset = gp.GetParameter(0)

#Impedence/cost attribute for the network dataset
impedence = gp.GetParameterAsText(1)

#Feature class with event details
tempFC = gp.GetParameter(2)

#Type of temporal analysis (boolean variable).If True, hour analysis.
#If False, day analysis.
timeType = gp.GetParameter(3)
```

```

#Label used for temporal column in results table
if timeType == 0:
    tempLabel = "Days"
else:
    tempLabel = "Hours"

#Feature class field containing dates of events
dateFld = gp.GetParameterastext(4)

#Feature class field containing hour event occurred in 24-hour HH
#format (e.g. 7AM is 7; 7PM is 19)
hourFld = gp.GetParameterastext(5)

#Ensure that hour data field was provided if hourly analysis was
#selected above. Otherwise, exit program.
if timeType > 0 and hourFld == "":
    gp.addmessage("")
    gp.addmessage("In order to conduct hourly analysis, a field\
containing hour data in a 24-hour HH format must be provided.")
    gp.addmessage("")
    sys.exit()
else:
    pass

#Variable defining user-defined statistical significance level. Will
#be used to calculate Bonferroni correction.
alpha = gp.GetParameter(6)

#Define list variable to hold the Knox R results for each interval of
#critical parameter pairings
rangeResultsList = []

#=====
#CONDUCT SPATIAL NEAREST NEIGHBOR RANGE ANALYSIS
#=====

# Define local variables used in the Make Closest Facility Layer
#analysis.
CFLayer = "ClosestFacilityLayer"
RouteLayer = "ClosestFacilityLayer\\Routes"
CFRouteLayer = "CFRoutes_Layer"

# Make Closest Facility Layer. This analysis searches for the closest
#and second closest spatial neighbors between two point feature layers;
#in this case both layers are the same.
gp.MakeClosestFacilityLayer_na(netDataset, CFLayer, impedance,\
"TRAVEL_TO", "", "2", "", "ALLOW_UTURNS", "", "NO_HIERARCHY", "",\
"TRUE_LINES_WITH_MEASURES")
gp.AddLocations_na(CFLayer, "Facilities", tempFC, "CurbApproach #\
0;Attr_Length # 0", "5000 Meters", "OBJECTID", "", "MATCH_TO_CLOSEST",\
"APPEND", "SNAP", "5 Meters")
gp.AddLocations_na(CFLayer, "Incidents", tempFC, "CurbApproach #\
0;Attr_Length # 0", "5000 Meters", "OBJECTID", "", "MATCH_TO_CLOSEST",\
"APPEND", "SNAP", "5 Meters")

```

```

gp.Solve_na(CFLayer, "HALT")

# Select only the second nearest neighbors. The first nearest neighbor
#is the point itself since this analysis is looking at two identical
#point layers.
gp.SelectData_management(CFLayer, "Routes")

# Make a feature layer from the selected events above.
gp.MakeFeatureLayer_management(RouteLayer, CFRouteLayer, \
 "\"FacilityRank\" = 2", "", "FacilityID FacilityID VISIBLE\
 NONE;FacilityRank FacilityRank VISIBLE NONE;Name Name VISIBLE\
 NONE;IncidentCurbApproach IncidentCurbApproach VISIBLE\
 NONE;FacilityCurbApproach FacilityCurbApproach VISIBLE NONE;IncidentID\
 IncidentID VISIBLE NONE;Total_Length Total_Distance VISIBLE NONE")

#Rename variable storing feature layer
netNN = CFRouteLayer

#Initiate a list to store the spatial nearest neighbor distance values
nDistList = []

#Initiate the geoprocessor cursor to extract the spatial nearest
#neighbor distance value for each event
cur = gp.SearchCursor(netNN)
row = cur.Next()
while row:
    nDist = int(row.GetValue("Total_Distance"))
    nDistList.append(nDist)
    row = cur.Next()

#Count the values in the list
nNcount = len(nDistList)
#Assign the minimum Spatial Nearest Neighbor Distance to a variable
nNmin = min(nDistList)
#Assign the maximum Spatial Nearest Neighbor Distance to a variable
nNmax = max(nDistList)
#Assign the avg NN spatial dist to a variable
nNavg = sum(nDistList) / nNcount

#Round avg nearest neighbor distance value to the next highest multiple
#of 100, unless the value is exactly a multiple of 100
nNavgRound = round(nNavg / 10)
roundStr = str(nNavgRound)
roundVal = int(roundStr[-3])
if roundVal > 0:
    spaceCritMax = ((10 - float(roundVal)) + nNavgRound)*10
else:
    spaceCritMax = nNavgRound * 10

#Round min nearest neighbor distance value to the next lowest multiple
#of 100, unless the value is exactly a multiple of 100
if nNmin > 0:
    nNminRound = round(nNmin / 10)
    roundStr = str(nNminRound)

```

```

roundVal = int(roundStr[-3])
if roundVal > 0:
    spaceCritMin = (nNminRound - roundVal)*100
else:
    spaceCritMin = nNminRound * 100
else:
    spaceCritMin = 0

gp.addmessage("")
gp.addmessage("Minimum Network Nearest Neighbor Distance: " +\
str(nNmin) + " meters.")
gp.addmessage("Maximum Network Nearest Neighbor Distance: " +\
str(nNmax) + " meters.")
gp.addmessage("Average Network Nearest Neighbor Distance: " +\
str(nNavg) + " meters.")
gp.addmessage("Spatial Range input for ST Cluster Automatic will be\
from " + str(spaceCritMin) + " to " + str(spaceCritMax) + " meters.")

#=====
#CONDUCT SPATIAL NEAREST NEIGHBOR RANGE ANALYSIS
#=====

#Define a variable for the feature layer field that will hold the
#calculated timeline value
calcFld = "CALC_TIME"

#Create SQL statement for the field calculation if hourly analysis is
#selected
expnHour = "["+str(dateFld)+"]*24["+str(hourFld)+"]"
#Create SQL statement for the field calculation if day analysis is
#selected
expnDay = "["+str(dateFld)+"]*1"

#Local variables used in the calculate field process below
Output_Layer = "tempFC_L"
Output_Layer2 = "tempFC_L2"

#Define variable to hold boolean value
bool = int(timeType)

#If hourly analysis is selected, perform this section of code
x = 1
if bool == x:
    gp.addmessage("")
    gp.addmessage("Temporal Analysis Type: Hour")

    # Add and calculate timeline field to feature layer
    gp.MakeFeatureLayer_management(tempFC, Output_Layer)
    gp.AddField_management(Output_Layer, calcFld, "DOUBLE", "", "", "\
", "", "NULLABLE", "NON_REQUIRED", "")
    gp.CalculateField_management(Output_Layer, calcFld, expnHour,\
"VB", "")

```

```

#Assign variable to output layer
timeNN = Output_Layer

#Assign variable to list holding the timeline values
timeList = []

#Search through feature layer and extract out the timeline values
cur = gp.SearchCursor(timeNN)
row = cur.Next()
while row:
    time = row.GetValue(calcFld)
    timeList.append(time)
    row = cur.Next()

#Sort list so that values are ascending
timeList.sort()

#Assign variable to number of items in list
numTimes = len(timeList)

#Search through the list of timeline values and calculate the
#neighbor distance between each event; store in list
nList = []
maxList = max(timeList)
x=1
y=0
while x < numTimes:
    nList.append(timeList[x]-timeList[y])
    x=x+1
    y=y+1

#Search through list of neighbor distances and select the nearest
#neighbor distance for each event
numDist = len(nList)
nDistList = []
x=1
y=0
while x < numDist:
    n=[nList[y],nList[x]]
    nn=min(n)
    nDistList.append(nn)
    x=x+1
    y=y+1

#Count number of values in list
nNcount = len(nDistList)
#Calculate sum of values in list
nNsum = float(sum(nDistList))
#Determine minimum value in list
nNmin = min(nDistList)
#Determine maximum value in list
nNmax = max(nDistList)
#Determine average value in list
nNavg = float(nNsum / nNcount)

```

```

#Round avg nearest neighbor distance value to the next highest
#multiple of 1, unless the value is exactly a multiple of 1
nNavgRound = round(nNavg)
roundStr = str(nNavgRound)
roundVal = int(roundStr[-1])
if roundVal > 0:
    timeCritMax = (10 - float(roundVal)) + nNavgRound
else:
    timeCritMax = nNavgRound

#Round min nearest neighbor distance value to the next lowest
#multiple of 1, unless the value is exactly a multiple of 1
if nNmin > 0:
    nNminRound = round(nNmin)
    roundStr = str(nNminRound)
    roundVal = int(roundStr[-1])
    if roundVal > 0:
        timeCritMin = nNminRound - roundVal
    else:
        timeCritMin = nNminRound
else:
    timeCritMin = 0

gp.addmessage("Minimum Temporal Nearest Neighbor Distance: " + \
str(nNmin) + " hours.")
gp.addmessage("Maximum Temporal Nearest Neighbor Distance: " + \
str(nNmax) + " hours.")
gp.addmessage("Average Temporal Nearest Neighbor Distance: " + \
"%0.1f" % (nNavg) + " hours.")
gp.addmessage("Temporal Range input for ST Cluster Automatic will\
be from " + str(timeCritMin) + " to " + str(timeCritMax) + " hours.")
gp.addmessage("")

#All of the code annotation above is the same for the loop below. The
#following loop performs the same functions for a day analysis instead
#of hourly.
else:
    gp.addmessage("")
    gp.addmessage("Temporal Analysis Type: Day")

gp.MakeFeatureLayer_management(tempFC, Output_Layer2)
gp.AddField_management(Output_Layer2, calcFld, "DOUBLE", "", "", \
"", "", "NULLABLE", "NON_REQUIRED", "")
gp.CalculateField_management(Output_Layer2, calcFld, expnDay, \
"VB", "")

timeNN = Output_Layer2
timeList = []
cur = gp.SearchCursor(timeNN)
row = cur.Next()
while row:
    time = int(row.GetValue(calcFld))

```



```

        timeList.append(time)
        row = cur.Next()

timeList.sort()

numTimes = len(timeList)

nList = []
maxList = max(timeList)
x=1
y=0
while x < numTimes:
    nList.append(timeList[x]-timeList[y])
    x=x+1
    y=y+1

numDist = len(nList)
nDistList = []
x=1
y=0
while x < numDist:
    n=[nList[y],nList[x]]
    nn=min(n)
    nDistList.append(nn)
    x=x+1
    y=y+1

nNcount = len(nDistList)
nNsum = float(sum(nDistList))
nNmin = min(nDistList)
nNmax = max(nDistList)
nNavg = float(nNsum / nNcount)

nNavgRound = round(nNavg)
roundStr = str(nNavgRound)
roundVal = int(roundStr[-1])
if roundVal > 0:
    timeCritMax = (10 - float(roundVal)) + nNavgRound
else:
    timeCritMax = nNavgRound

if nNmin > 0:
    nNminRound = round(nNmin)
    roundStr = str(nNminRound)
    roundVal = int(roundStr[-1])
    if roundVal > 0:
        timeCritMin = nNminRound - roundVal
    else:
        timeCritMin = nNminRound
else:
    timeCritMin = 0

gp.addmessage("Minimum Temporal Nearest Neighbor Distance: " +\
str(nNmin) + " days.")

```

```

gp.addmessage("Maximum Temporal Nearest Neighbor Distance: " + \
str(nNmax) + " days.")
gp.addmessage("Average Temporal Nearest Neighbor Distance: " + \
"%0.1f" % (nNavg) + " days.")
gp.addmessage("Temporal Range input for ST Cluster Automatic will\
be from " + str(timeCritMin) + " to " + str(timeCritMax) + " days.")
gp.addmessage("")

th2 = clock()

pt1 = th2-th1
ptlint = int(pt1)
ptlmin = int(pt1/60)
ptlsec = int(ptlint-(ptlmin*60))

gp.addmessage("")
gp.addmessage("Spatial and temporal critical range analysis complete.\
Process time: " + str(ptlmin) + " minute(s) and " + str(ptlsec) + "\
seconds.")
gp.addmessage("")
gp.addmessage("Creating origin-destination matrix layer...")

th3 = clock()

#=====
#CONDUCT SPATIOTEMPORAL ANALYSIS FOR EACH PAIRING OF
#SPATIAL AND TEMPORAL CRITICAL VALUES DETERMINED ABOVE
#=====

#The annotation for the following portion of code is the same as 'ST
#Cluster Basic'. Refer to that script for explanation.

critTime = int(timeCritMin)

critTimeMax = int(timeCritMax)

critTimeRange = critTimeMax - critTime

if critTimeRange >= 2:
    critTimeMultiple = round(critTimeRange/2)
else:
    if critTimeRange < 2:
        critTimeMultiple = 1

inputTimeFC = timeNN

desc = gp.Describe(timeNN)
timeIDfield = desc.OIDFieldName

timeLinefield = calcFld

odLayer = "ODCostMatrix"
lineLyr = "ODCostMatrix\\Lines"

```

```

odLineLyr = "ODRoute_Layer"
gp.MakeODCostMatrixLayer_na(netDataset, odLayer, impedance, "", "", \
"", "ALLOW_UTURNS", "", "NO_HIERARCHY", "", "STRAIGHT_LINES")
gp.AddLocations_na(odLayer, "Origins", tempFC, "", "5000 Meters", "", \
"", "MATCH_TO_CLOSEST", "APPEND", "SNAP", "5 Meters")
gp.AddLocations_na(odLayer, "Destinations", tempFC, "", "5000 Meters", \
"", "", "MATCH_TO_CLOSEST", "APPEND", "SNAP", "5 Meters")
gp.Solve_na(odLayer, "SKIP")
gp.SelectData_management(odLayer, "Lines")
gp.MakeFeatureLayer_management(lineLyr, odLineLyr, "", "", "")

spaceFC = odLineLyr

desc2 = gp.Describe(odLineLyr)
fldinfo2 = desc2.FieldInfo
spaceOrigIDfield = fldinfo2.GetFieldName(1)

spaceDestIDfield = fldinfo2.GetFieldName(2)

spaceDistfield = fldinfo2.GetFieldName(4)

critDist = int(spaceCritMin)

critDistMax = int(spaceCritMax)

critDistRange = critDistMax - critDist

critDistMultiple = critDistRange/2

th4 = clock()
pt2 = th4-th3
pt2int = int(pt2)
pt2min = int(pt2/60)
pt2sec = int(pt2int-(pt2min*60))

gp.addmessage("Origin-destination matrix layer complete. Process\
time: " + str(pt2min) + " minute(s) and " + str(pt2sec) + " seconds.")
gp.addmessage("")

gp.addmessage("Conducting spatiotemporal analysis...")
gp.addmessage("")

th5 = clock()

gp.AddMessage("=====\
=====")
gp.AddMessage("")
gp.AddMessage("Meters" + "\t" + tempLabel + "\t" + " KnoxR" + "\
chiVal" + " chiProb" + "\t" + " zVal" + "\t" + " zProb")
gp.AddMessage("-----\
-----")

```

```

#Define the number of iterations for which the Knox analysis will be
#conducted based on the spatial and temporal range of values
critDistRuns = (int(critDistRange) / int(critDistMultiple)) + 2
critTimeRuns = (int(critTimeRange) / int(critTimeMultiple)) + 2
critTimeiter = critTime

#Initiate counters
distRun = 0
timeRun = 0
results = 0

#Begin Knox test loop
while distRun < critDistRuns:
    while timeRun < critTimeRuns:

        dctTime = {}

        adjustTimeID = []

        cur = gp.SearchCursor(inputTimeFC)
        row = cur.Next()
        firstTimeID = int(row.GetValue(timeIDfield))
        adjustTimeID = [firstTimeID]

        cur = gp.SearchCursor(inputTimeFC)
        row = cur.Next()

        while row:
            key = (int(row.GetValue(timeIDfield)) - adjustTimeID[0]) + \
1            timelineDay = row.GetValue(timeLinefield)
            dctTime[key] = timelineDay
            row = cur.Next()

        dctTimeLT = {}
        dctTimeGT = {}

        timeKeyAlist = []
        timeKeyAlistTally = []

        numEvents = len(dctTime)

        x = 1
        y = 1
        while x <= numEvents:
            for key in dctTime:
                if key <= numEvents:
                    timeDif = abs(dctTime[y] - dctTime[x])
                    keyA = x
                    keyB = y
                    if timeDif <= critTime and keyA < keyB:
                        dctTimeLT[keyA, keyB] = timeDif
                        timeKeyAlist.append(keyA)

```

```

        else:
            if timeDif > critTime and keyA < keyB:
                dctTimeGT[keyA, keyB] = timeDif
            else:
                pass
        y = y + 1
    x = x + 1
    y = 1

x = 0
while x <= numEvents:
    occurX = timeKeyAlist.count(x)
    occurXx2 = occurX * 2
    occurXx2squared = occurXx2 ** 2
    timeKeyAlistTally.append(occurXx2squared)
    x = x + 1

t2 = sum(timeKeyAlistTally)

numCritTimePairs = len(dctTimeLT)
numNotCritTimePairs = len(dctTimeGT)

dctSpaceLT = {}
dctSpaceGT = {}

adjustOrigID = []
adjustDestID = []

cur = gp.SearchCursor(spaceFC)
row = cur.Next()
firstOrigID = int(row.GetValue(spaceOrigIDfield))
firstDestID = int(row.GetValue(spaceDestIDfield))
adjustOrigID = [firstOrigID]
adjustDestID = [firstDestID]

spaceKeyAlist = []
spaceKeyAlistTally = []

cur = gp.SearchCursor(spaceFC)
row = cur.Next()

while row:
    keyA = (int(row.GetValue(spaceOrigIDfield)) -
adjustOrigID[0]) + 1
    keyB = (int(row.GetValue(spaceDestIDfield)) -
adjustDestID[0]) + 1
    spaceDif = row.GetValue(spaceDistfield)
    if spaceDif >= 0 and spaceDif <= critDist and keyA < keyB:
        dctSpaceLT[keyA, keyB] = spaceDif
        spaceKeyAlist.append(keyA)
    else:
        if spaceDif > critDist and keyA < keyB:
            dctSpaceGT[keyA, keyB] = spaceDif
        else:

```

```

        pass
    row = cur.Next()
    x = 0
    while x <= numEvents:
        occurX = spaceKeyAlist.count(x)
        occurXx2 = occurX * 2
        occurXx2squared = occurXx2 ** 2
        spaceKeyAlistTally.append(occurXx2squared)
        x = x + 1
    s2 = sum(spaceKeyAlistTally)
    numCritSpacePairs = len(dctSpaceLT)
    numNotCritSpacePairs = len(dctSpaceGT)

    x = 0
    for key in dctSpaceLT:
        if dctTimeLT.has_key(key) == True:
            x = x + 1

    knoxAPairs = x

    x = 0
    for key in dctTimeGT:
        if dctSpaceLT.has_key(key) == True:
            x = x + 1

    knoxBPairs = x

    x = 0
    for key in dctSpaceGT:
        if dctTimeLT.has_key(key) == True:
            x = x + 1

    knoxCPairs = x

    x = 0
    for key in dctSpaceGT:
        if dctTimeGT.has_key(key) == True:
            x = x + 1

    knoxDPairs = x

    knoxChiSig = chiKnox(knoxAPairs, knoxBPairs, knoxCPairs,\
knoxDPairs, numEvents)
    chiProb = round(scipy.stats.chisqprob(knoxChiSig, 1),5)
    knoxZ = normalKnox(knoxAPairs, numCritSpacePairs,\
numCritTimePairs, s2, t2, numEvents)
    if knoxZ > 0:
        zProb = round((1 - scipy.stats.zprob(knoxZ)),5)
    else:
        zProb = round((scipy.stats.zprob(knoxZ)),5)
    gp.AddMessage(" %-8d  %-5d  %-5d  %--0.3f  %-12s %--0.3f\
%-12s" % (critDist, critTime, knoxAPairs, knoxChiSig, chiProb, knoxZ,\
zProb))

```

```

        #Add resulting statistics from this iteration of the loop into
        #a list. Append the list to a list of all the result lists for
        #all iterations. This list is used to determine the most
        #significant spatiotemporal cluster further down in the script.
        rangeResult = [critDist, critTime, knoxAPairs, knoxChiSig,\
chiProb, knoxZ, zProb]
        rangeResultsList.append(rangeResult)

        critTime = critTime + critTimeMultiple
        timeRun = timeRun + 1
        results = results + 1

        critDist = critDist + critDistMultiple
        critTime = critTimeiter
        distRun = distRun + 1
        timeRun = 0

gp.AddMessage("")
gp.AddMessage("=====\
=====")
gp.AddMessage("")

th6 = clock()

pt3 = th6-th5
pt3int = int(pt3)
pt3min = int(pt3/60)
pt3sec = int(pt3int-(pt3min*60))

gp.AddMessage("Spatiotemporal analysis complete. Process time: " +\
str(pt3min) + " minute(s) and " + str(pt3sec) + " seconds.")

#=====
#DETERMINE THE MOST SIGNIFICANT SPATIOTEMPORAL VALUE
#=====

#Define variables used in the determination of the most significant
#spatiotemporal cluster value
rrl = rangeResultsList
rrlRandZscoresList = []
rrlZscoresList = []
nonzeroRZscoresList = []
rrlbonfc = []
rrlbonfcSig = []

#From the list of results obtained above, select out all the Knox R
#values and associated Z probabilities.
i = 0
x = len(rrl)
while i < x:
    rrlRandZscores = [rrl[i][2], rrl[i][6]]
    rrlRandZscoresList.append(rrlRandZscores)
    rrlZscoresList.append(rrl[i][6])
    if rrlRandZscoresList[i][0] > 0:

```

```

        nonzeroRZscoresList.append(rrlRandZscoresList[i][1])
    else:
        pass
    i = i + 1

#Sort the list with the z probabilities in ascending order
rrlZscoresList.sort()

#Calculate the Bonferroni correction given the user defined alpha and
#add it to a list which contains the associate Knox R and z Prob
#values.
i = 0
y = 0
x = 1
while i < len(rrlZscoresList):
    if rrlZscoresList[i] == rrlRandZscoresList[y][1]:
        bonfcalc = (alpha*x)/(len(rrl))
        bonfcandz = [rrlRandZscoresList[y][0], rrlZscoresList[i],\
bonfcalc]
        rrlbonfc.append(bonfcandz)
        y = 0
        i = i + 1
        x = x + 1
    else:
        y = y + 1

#If the Knox R's associated z Prob value is equal to or less than the
#Bonferroni correction, store it in a new list
i = 0
while i < len(rrlbonfc):
    if rrlbonfc[i][0] > 0 and rrlbonfc[i][1] <= rrlbonfc[i][2]:
        rrlbonfcSig.append(rrlbonfc[i][1])
    else:
        pass
    i = i + 1

#If no Knox R values are returned from the range searched, print
#message to screen
if len(nonzeroRZscoresList) == 0:
    gp.addmessage("KnoxR = 0 for all combinations of spatial and\
temporal critical distances examined.")
    gp.addmessage("")

#Otherwise, continue analysis. If Knox R values were determined from
#the range search, but none were significant given the Bonferroni
#correction then report the Knox R value that had the lowest z Prob not
#within the constraints of the Bonferroni correction
else:
    if len(rrlbonfcSig) == 0:
        minZ = min(nonzeroRZscoresList)
        i = 0
        x = len(rrl)
        while i < x:
            if rrl[i][6] == minZ:

```



```

        bestKnox = rrl[i]
    else:
        pass
    i = i+1
    gp.addmessage("")
    gp.addmessage("There are no Knox R values that meet the\
desired significance level of " + str(alpha) + " considering a\
Bonferroni correction factor")
    gp.addmessage("for multiple tests. Of the range of spatial and\
temporal critical distances searched, below is the most significant\
value:")
    gp.addmessage("")
    gp.AddMessage("Meters" + "\t" + tempLabel + "\t" + "   KnoxR" +\
"   chiVal" + "   chiProb" + "\t" + "   zVal" + "\t" + "   zProb")
    gp.AddMessage("-----\
-----")
    gp.AddMessage(" %-8d  %-5d  %-5d  %--0.3f  %-12s %--0.3f\
%-12s" % (bestKnox[0], bestKnox[1], bestKnox[2], bestKnox[3],\
bestKnox[4], bestKnox[5], bestKnox[6]))
    gp.AddMessage("")
    gp.addmessage("The spatial and temporal critical distances\
displayed here may be entered as parameters for 'ST Cluster Basic' in\
order to generate a feature class containing the points contributing\
to the spatiotemporal cluster.")
    gp.addmessage("")

#If there are Knox R values that are significant given the Bonferroni
#correction, then of those values that were significant, print the one
#which had the lowest z Prob
    else:
        minZ = min(rrlbonfcSig)
        i = 0
        x = len(rrl)
        while i < x:
            if rrl[i][6] == minZ:
                bestKnox = rrl[i]
            else:
                pass
            i = i+1
        gp.addmessage("")
        gp.addmessage("Taking into consideration a Bonferroni\
correction factor for multiple tests, below is the most significant\
Knox Value")
        gp.addmessage("given the range of spatial and temporal\
critical distances searched:")
        gp.addmessage("")
        gp.AddMessage("Meters" + "\t" + tempLabel + "\t" + "   KnoxR" +\
"   chiVal" + "   chiProb" + "\t" + "   zVal" + "\t" + "   zProb")
        gp.AddMessage("-----\
-----")
        gp.AddMessage(" %-8d  %-5d  %-5d  %--0.3f  %-12s %--0.3f\
%-12s" % (bestKnox[0], bestKnox[1], bestKnox[2], bestKnox[3],\
bestKnox[4], bestKnox[5], bestKnox[6]))
        gp.AddMessage("")

```

```

gp.addmessage("The spatial and temporal critical distances\
displayed here may be entered as parameters for 'ST Cluster Basic' in\
order to generate a feature class containing the points contributing\
to the spatiotemporal cluster.")
gp.addmessage("")

#Delete variables
del dctSpaceLT, dctSpaceGT, dctTime, dctTimeLT, dctTimeGT,\
numCritTimePairs, numNotCritTimePairs, critTime, numEvents
del t2, s2, timeKeyAlist, timeKeyAlistTally, spaceKeyAlist,\
spaceKeyAlistTally, knoxZ, knoxChiSig
del knoxAPairs, knoxBPairs, knoxCPairs, knoxDPairs, numCritSpacePairs,\
numNotCritSpacePairs, adjustTimeID, adjustOrigID, adjustDestID

```

Tool 3: ST Cluster Table

```
#-----
#Name: SCAn Tool 3 - ST Cluster Table
#Created by: David Eckley
#Date created: 20101201
#Purpose: ST Cluster Table evaluates the Knox statistic for a range of
#user-defined critical parameters and prints out a table with the
#results according to the spatial and temporal intervals specified.
#-----

#=====
#IMPORT MODULES
#=====
import os, sys, string, arcgisscripting, math, numpy, scipy
from time import*
from numpy import*
from scipy import*
from scipy import stats
from knoxStats import chiKnox, normalKnox

#=====
#INITIATE GEOPROCESSOR
#=====
gp = arcgisscripting.create(9.3)

#=====
#DEFINE VARIABLES PROVIDED BY USER
#=====

#Network dataset
netDataset = gp.GetParameter(0)

#Impedence/cost attribute for the network dataset
impedence = gp.GetParameterAsText(1)

#Feature class with event details
tempFC = gp.GetParameter(2)

#Type of temporal analysis (boolean variable).If True, hour analysis.
#If False, day analysis.
timeType = gp.GetParameter(3)

#Label used for temporal column in results table
if timeType == 0:
    tempLabel = "Days"
else:
    tempLabel = "Hours"

#Feature class field containing dates of events
dateFld = gp.GetParameterAsText(4)
```

```

#Feature class field containing hour event occurred in 24-hour HH
#format (e.g. 7AM is 7; 7PM is 19)
hourFld = gp.GetParameterastext(5)

#Ensure that hour data field was provided if hourly analysis was
#selected above. Otherwise, exit program.
if timeType > 0 and hourFld == "":
    gp.addmessage("")
    gp.addmessage("In order to conduct hourly analysis, a field\
containing hour data in a 24-hour HH format must be provided.")
    gp.addmessage("")
    sys.exit()
else:
    pass

#Minimum critical spatial distance in meters. Convert to integer.
critDist = int(gp.GetParameter(6))

#Maximum critical spatial distance in meters. Convert to integer.
critDistMax = int(gp.GetParameter(7))

#Increment interval for spatial distance range. Convert to integer.
critDistMultiple = int(gp.GetParameter(8))

#Minimum critical temporal distance in units appropriate for desired
#analysis (hours or days). Convert to integer.
critTime = int(gp.GetParameter(9))

#Maximum critical temporal distance in units appropriate for desired
#analysis (hours or days). Convert to integer.
critTimeMax = int(gp.GetParameter(10))

#Increment interval for temporal distance range. Convert to integer.
critTimeMultiple = int(gp.GetParameter(11))

#=====
#CONDUCT SPATIOTEMPORAL ANALYSIS
#=====

#The script annotation for the following code is the same as that
#provided with 'Tool 1: ST Cluster Basic' and 'Tool 2: ST Cluster
#Automatic'. Refer to these tools for explanation.

calcFld = "CALC_TIME"

expnHour = "["+str(dateFld)+"]*24["+str(hourFld)+"]"

expnDay = "["+str(dateFld)+"]*1"

Output_Layer = "tempFC_L"
Output_Layer2 = "tempFC_L2"

bool = int(timeType)

```

```

x = 1
if bool == x:
    gp.addmessage("")
    gp.addmessage("Temporal Analysis Type: Hour")

    gp.MakeFeatureLayer_management(tempFC, Output_Layer)
    gp.AddField_management(Output_Layer, calcFld, "DOUBLE", "", "", "\
", "", "NULLABLE", "NON_REQUIRED", "")
    gp.CalculateField_management(Output_Layer, calcFld, expnHour, \
"VB", "")

    timeNN = Output_Layer

else:
    gp.addmessage("")
    gp.addmessage("Temporal Analysis Type: Day")

    gp.MakeFeatureLayer_management(tempFC, Output_Layer2)
    gp.AddField_management(Output_Layer2, calcFld, "DOUBLE", "", "", "\
", "", "NULLABLE", "NON_REQUIRED", "")
    gp.CalculateField_management(Output_Layer2, calcFld, expnDay, \
"VB", "")

    timeNN = Output_Layer2

gp.addmessage("")
gp.addmessage("Creating origin-destination matrix layer...")
gp.addmessage("")

th3 = clock()

inputTimeFC = timeNN

desc = gp.Describe(timeNN)
timeIDfield = desc.OIDFieldName

timeLinefield = calcFld

odLayer = "ODCostMatrix"
lineLyr = "ODCostMatrix\\Lines"
odLineLyr = "ODRoute_Layer"
gp.MakeODCostMatrixLayer_na(netDataset, odLayer, impedance, "", "", \
", "ALLOW_UTURNS", "", "NO_HIERARCHY", "", "STRAIGHT_LINES")
gp.AddLocations_na(odLayer, "Origins", tempFC, "", "5000 Meters", "", \
", "MATCH_TO_CLOSEST", "APPEND", "SNAP", "5 Meters")
gp.AddLocations_na(odLayer, "Destinations", tempFC, "", "5000 Meters", \
", "", "MATCH_TO_CLOSEST", "APPEND", "SNAP", "5 Meters")
gp.Solve_na(odLayer, "SKIP")
gp.SelectData_management(odLayer, "Lines")
gp.MakeFeatureLayer_management(lineLyr, odLineLyr, "", "", "")

spaceFC = odLineLyr

th4 = clock()

```

```

pt2 = th4-th3
pt2int = int(pt2)
pt2min = int(pt2/60)
pt2sec = int(pt2int-(pt2min*60))

gp.addmessage("Origin-destination matrix layer complete.  Process\
time: " + str(pt2min) + " minute(s) and " + str(pt2sec) + " seconds.")
gp.addmessage("")

gp.addmessage("Conducting spatiotemporal analysis...")
gp.addmessage("")

th5 = clock()

desc2 = gp.Describe(odLineLyr)
fldinfo2 = desc2.FieldInfo
spaceOrigIDfield = fldinfo2.GetFieldName(1)

spaceDestIDfield = fldinfo2.GetFieldName(2)

spaceDistfield = fldinfo2.GetFieldName(4)

gp.AddMessage("=====\
=====")
gp.AddMessage("")
gp.AddMessage("Meters" + "\t" + tempLabel + "\t" + " KnoxR" + "\
chiVal" + " chiProb" + "\t" + " zVal" + "\t" + " zProb")
gp.AddMessage("-----\
-----")

critDistRange = critDistMax - critDist
critDistRuns = (critDistRange / critDistMultiple) + 1

critTimeRange = critTimeMax - critTime
critTimeRuns = (critTimeRange / critTimeMultiple) + 1

distRun = 0
timeRun = 0

while distRun < critDistRuns:
    while timeRun < critTimeRuns:
        dctTime = {}
        adjustTimeID = []
        cur = gp.SearchCursor(inputTimeFC)
        row = cur.Next()
        firstTimeID = int(row.GetValue(timeIDfield))
        adjustTimeID = [firstTimeID]
        cur = gp.SearchCursor(inputTimeFC)
        row = cur.Next()

        while row:
            key = (int(row.GetValue(timeIDfield)) - adjustTimeID[0])\
+ 1

```

```

        timelineDay = row.GetValue(timeLinefield)
        dctTime[key] = timelineDay
        row = cur.Next()

dctTimeLT = {}
dctTimeGT = {}

timeKeyAlist = []
timeKeyAlistTally = []

numEvents = len(dctTime)

x = 1
y = 1

while x <= numEvents:
    for key in dctTime:
        if key <= numEvents:
            timeDif = abs(dctTime[y] - dctTime[x])
            keyA = x
            keyB = y
            if timeDif <= critTime and keyA < keyB:
                dctTimeLT[keyA, keyB] = timeDif
                timeKeyAlist.append(keyA)
            else:
                if timeDif > critTime and keyA < keyB:
                    dctTimeGT[keyA, keyB] = timeDif
                else:
                    pass
            y = y + 1
        x = x + 1
    y = 1

x = 0
while x <= numEvents:
    occurX = timeKeyAlist.count(x)
    occurXx2 = occurX * 2
    occurXx2squared = occurXx2 ** 2
    timeKeyAlistTally.append(occurXx2squared)
    x = x + 1
t2 = sum(timeKeyAlistTally)

numCritTimePairs = len(dctTimeLT)
numNotCritTimePairs = len(dctTimeGT)

dctSpaceLT = {}
dctSpaceGT = {}

adjustOrigID = []
adjustDestID = []

cur = gp.SearchCursor(spaceFC)
row = cur.Next()
firstOrigID = int(row.GetValue(spaceOrigIDfield))

```

```

firstDestID = int(row.GetValue(spaceDestIDfield))
adjustOrigID = [firstOrigID]
adjustDestID = [firstDestID]

spaceKeyAlist = []
spaceKeyAlistTally = []

cur = gp.SearchCursor(spaceFC)
row = cur.Next()

while row:
    keyA = (int(row.GetValue(spaceOrigIDfield)) -
adjustOrigID[0]) + 1
    keyB = (int(row.GetValue(spaceDestIDfield)) -
adjustDestID[0]) + 1
    spaceDif = row.GetValue(spaceDistfield)
    if spaceDif >= 0 and spaceDif <= critDist and keyA < keyB:
        dctSpaceLT[keyA, keyB] = spaceDif
        spaceKeyAlist.append(keyA)
    else:
        if spaceDif > critDist and keyA < keyB:
            dctSpaceGT[keyA, keyB] = spaceDif
        else:
            pass
    row = cur.Next()
x = 0
while x <= numEvents:
    occurX = spaceKeyAlist.count(x)
    occurXx2 = occurX * 2
    occurXx2squared = occurXx2 ** 2
    spaceKeyAlistTally.append(occurXx2squared)
    x = x + 1
s2 = sum(spaceKeyAlistTally)
numCritSpacePairs = len(dctSpaceLT)
numNotCritSpacePairs = len(dctSpaceGT)

x = 0
for key in dctSpaceLT:
    if dctTimeLT.has_key(key) == True:
        x = x + 1
knoxAPairs = x

x = 0
for key in dctTimeGT:
    if dctSpaceLT.has_key(key) == True:
        x = x + 1
knoxBPairs = x

x = 0
for key in dctSpaceGT:
    if dctTimeLT.has_key(key) == True:
        x = x + 1
knoxCPairs = x

```



```

x = 0
for key in dctSpaceGT:
    if dctTimeGT.has_key(key) == True:
        x = x + 1
knoxDPairs = x

knoxChiSig = chiKnox(knoxAPairs, knoxBPairs, knoxCPairs,\
knoxDPairs, numEvents)
chiProb = round(scipy.stats.chisqprob(knoxChiSig, 1),5)
knoxZ = normalKnox(knoxAPairs, numCritSpacePairs,\
numCritTimePairs, s2, t2, numEvents)

if knoxZ > 0:
    zProb = round((1 - scipy.stats.zprob(knoxZ)),5)
else:
    zProb = round((scipy.stats.zprob(knoxZ)),5)

gp.AddMessage(" %-8d    %-5d    %-5d    %--0.3f    %-12s %--0.3f\
%-12s" % (critDist, critTime, knoxAPairs, knoxChiSig, chiProb, knoxZ,\
zProb))

critTime = critTime + critTimeMultiple
timeRun = timeRun + 1

critDist = critDist + critDistMultiple
critTime = critTime - (critTimeRange + critTimeMultiple)
distRun = distRun + 1
timeRun = 0

gp.AddMessage("")
gp.AddMessage("=====\
=====")
gp.AddMessage("")

th6 = clock()

pt3 = th6-th5
pt3int = int(pt3)
pt3min = int(pt3/60)
pt3sec = int(pt3int-(pt3min*60))

gp.AddMessage("Spatiotemporal analysis complete.  Process time: " +\
str(pt3min) + " minute(s) and " + str(pt3sec) + " seconds.")
gp.AddMessage("")

#Delete variables
del dctSpaceLT, dctSpaceGT, dctTime, dctTimeLT, dctTimeGT,\
numCritTimePairs, numNotCritTimePairs, critTime, numEvents
del t2, s2, timeKeyAlist, timeKeyAlistTally, spaceKeyAlist,\
spaceKeyAlistTally, knoxZ, knoxChiSig,
del knoxAPairs, knoxBPairs, knoxCPairs, knoxDPairs, numCritSpacePairs,\
numNotCritSpacePairs, adjustTimeID, adjustOrigID, adjustDestID

```

Tool 4: ST Cluster Monte Carlo

```
#-----
#Name: SCAn Tool 4: ST Cluster Monte Carlo
#Created by: David Eckley
#Date created: 20101201
#Purpose: ST Cluster Monte Carlo evaluates the Knox test and determines
#the statistical significance based a specified number of Monte Carlo
#simulations. In addition to the information provided in the output
#file by Tools 1 and 2, ST Cluster Monte Carlo also provides details on
#the generated reference distribution which can be imported into a
#spreadsheet program to generate graphs if desired.
#-----

#=====
#IMPORT MODULES
#=====
import os, sys, string, arcgisscripting, math, numpy, scipy
from random import*
from stats import*
from time import*
from numpy import*
from scipy import*
from scipy import stats
from knoxStats import chiKnox, normalKnox

#=====
#INITIATE GEOPROCESSOR
#=====
gp = arcgisscripting.create(9.3)

#=====
#DEFINE VARIABLES PROVIDED BY USER
#=====

#Network dataset
netDataset = gp.GetParameter(0)

#Impedence/cost attribute for the network dataset
impedence = gp.GetParameterAsText(1)

#Feature class with event details
tempFC = gp.GetParameter(2)

#Type of temporal analysis (boolean variable).If True, hour analysis.
#If False, day analysis.
timeType = gp.GetParameter(3)

#Label used for temporal column in results table
if timeType == 0:
    tempLabel = "Days"
else:
    tempLabel = "Hours"
```

```

#Feature class field containing dates of events
dateFld = gp.GetParameterastext(4)

#Feature class field containing hour event occurred in 24-hour HH
#format (e.g. 7AM is 7; 7PM is 19)
hourFld = gp.GetParameterastext(5)

#Ensure that hour data field was provided if hourly analysis was
#selected above. Otherwise, exit program.
if timeType > 0 and hourFld == "":
    gp.addmessage("")
    gp.addmessage("In order to conduct hourly analysis, a field\
containing hour data in a 24-hour HH format must be provided.")
    gp.addmessage("")
    sys.exit()
else:
    pass

#Critical spatial distance in meters. Convert to integer.
critDist = int(gp.GetParameter(6))

#Critical temporal distance in format described by boolean variable
#above.
critTime = int(gp.GetParameter(7))

#Have user define the number of Monte Carlo simulations to run
monteC = int(gp.GetParameter(8))

#=====
#CONDUCT SPATIOTEMPORAL ANALYSIS
#=====

#Where not specifically annotated, refer to Tools 1 through 3 for
#explanation.

gp.addmessage("")
gp.addmessage("Building Origin-Destination Cost Matrix Layer...")

th1 = clock()

#Using variables above and ArcToolBox tools
odLayer = "ODCostMatrix"
lineLyr = "ODCostMatrix\\Lines"
odLineLyr = "ODRoute_Layer"
gp.MakeODCostMatrixLayer_na(netDataset, odLayer, impedance, "", "", \
"", "ALLOW_UTURNS", "", "NO_HIERARCHY", "", "STRAIGHT_LINES")
gp.AddLocations_na(odLayer, "Origins", tempFC, "", "5000 Meters", "", \
"", "MATCH_TO_CLOSEST", "APPEND", "SNAP", "5 Meters")
gp.AddLocations_na(odLayer, "Destinations", tempFC, "", \
"5000 Meters", "", "", "MATCH_TO_CLOSEST", "APPEND", "SNAP", "5\
Meters")
gp.Solve_na(odLayer, "SKIP")

```

```

gp.SelectData_management(odLayer, "Lines")
gp.MakeFeatureLayer_management(lineLyr, odLineLyr, "", "", "")

#Create variable from the OD Cost Matrix Lines Layer
spaceFC = odLineLyr

#Create variable from the Origin ID Field within the OD Cost Matrix
#Lines Layer
desc2 = gp.Describe(odLineLyr)
fldinfo2 = desc2.FieldInfo
spaceOrigIDfield = fldinfo2.GetFieldName(1)

#Create variable from the Destination ID Field within the OD Cost
#Matrix Lines Layer
spaceDestIDfield = fldinfo2.GetFieldName(2)

#Create variable from the Total Distance/Length Field within the OD
#Cost Matrix Lines Layer
spaceDistfield = fldinfo2.GetFieldName(4)

th2 = clock()

pt1 = th2-th1
ptlint = int(pt1)
ptlmin = int(pt1/60)
ptlsec = int(ptlint-(ptlmin*60))

gp.addmessage("Origin-Destination Cost Matrix Layer complete. Process\
time: " + str(ptlmin) + " minute(s) and " + str(ptlsec) + " seconds.")
gp.addmessage("")

#Define variable that will become the field name for timeline field
calcFld = "CALC_TIME"

#VB expression that will be used in the calculate field operation in
#hour analysis is conducted
expnHour = "["+str(dateFld)+"]*24+["+str(hourFld)+"]"

#VB expression that will be used in the calculate field operation in
#day analysis is conducted
expnDay = "["+str(dateFld)+"]*1"

#Local variables that will be used in the "make feature layer"
#operation; a prerequisite to calculating the temporal fields.
Output_Layer = "tempFC_L"
Output_Layer2 = "tempFC_L2"

#Assign boolean variable from user inputs as an integer
bool = int(timeType)

#Conduct calculate field operation if the user selects an hourly
#analysis
x = 1
if bool == x:

```

```

# Make feature layer
gp.MakeFeatureLayer_management(tempFC, Output_Layer)

# Add timeline field and calculate timeline field value
gp.AddField_management(Output_Layer, calcFld, "DOUBLE", "", "", \
"", "", "NULLABLE", "NON_REQUIRED", "")
gp.CalculateField_management(Output_Layer, calcFld, expnHour, \
"VB", "")

#Assign variable to feature layer that will be used for temporal
#analysis and matrix generation
timeNN = Output_Layer

#Conduct calculate field operation if the user selects a day analysis
else:

# Make feature layer
gp.MakeFeatureLayer_management(tempFC, Output_Layer2)

# Add timeline field and calculate timeline field value
gp.AddField_management(Output_Layer2, calcFld, "DOUBLE", "", \
"", "", "", "NULLABLE", "NON_REQUIRED", "")
gp.CalculateField_management(Output_Layer2, calcFld, expnDay, \
"VB", "")

#Assign variable to feature layer that will be used for temporal
#analysis and matrix generation
timeNN = Output_Layer2

#Rename variable assigned to feature layer
inputTimeFC = timeNN

#Assign variable to Object ID Field within feature layer
desc = gp.Describe(timeNN)
timeIDfield = desc.OIDFieldName

#Rename variable assigned to timeline field name
timeLinefield = calcFld

dctTime = {}

adjustTimeID = []

cur = gp.SearchCursor(inputTimeFC)
row = cur.Next()
firstTimeID = int(row.GetValue(timeIDfield))
adjustTimeID = [firstTimeID]

cur = gp.SearchCursor(inputTimeFC)
row = cur.Next()

while row:
    key = (int(row.GetValue(timeIDfield)) - adjustTimeID[0]) + 1

```

```

        timelineDay = row.GetValue(timeLinefield)
        dctTime[key] = timelineDay
        row = cur.Next()

dctTimeLT = {}
numEvents = len(dctTime)
timeShuffleList = []

x = 1
y = 1
while x <= numEvents:
    for key in dctTime:
        if key <= numEvents:
            timeA = dctTime[x]
            timeB = dctTime[y]
            timeDif = abs(dctTime[y] - dctTime[x])
            #Load all the temporal distance between pairs values into a
#list to be used for Monte Carlo simulations below
            timeShuffleList.append(timeDif)
            keyA = x
            keyB = y
            if timeDif <= critTime and keyA < keyB:
                dctTimeLT[keyA, keyB] = timeDif
            else:
                pass
            y = y + 1
        x = x + 1
        y = 1

numCritTimePairs = len(dctTimeLT)

gp.AddMessage("=====\n\n")
gp.AddMessage("")
gp.AddMessage("Given a critical spatial distance of " +\ str(critDist)
+ " meters and a critical temporal distance of " + str(critTime) + " "\
+ tempLabel + ":\n")
gp.AddMessage("")

dctSpaceLT = {}
spaceShuffleList = []

adjustOrigID = []
adjustDestID = []

cur = gp.SearchCursor(spaceFC)
row = cur.Next()
firstOrigID = int(row.GetValue(spaceOrigIDfield))
firstDestID = int(row.GetValue(spaceDestIDfield))
adjustOrigID = [firstOrigID]
adjustDestID = [firstDestID]

cur = gp.SearchCursor(spaceFC)
row = cur.Next()

```

```

while row:
    keyA = (int(row.GetValue(spaceOrigIDfield)) - adjustOrigID[0]) + 1
    keyB = (int(row.GetValue(spaceDestIDfield)) - adjustDestID[0]) + 1
    spaceDif = row.GetValue(spaceDistfield)
    #Load all the spatial distance between pairs values into a list to
#be used for Monte Carlo simulations below
    spaceShuffleList.append(spaceDif)
    if spaceDif >= 0 and spaceDif <= critDist and keyA < keyB:
        dctSpaceLT[keyA, keyB] = spaceDif

    else:
        pass
    row = cur.Next()

del spaceFC

numCritSpacePairs = len(dctSpaceLT)

x = 0
for key in dctSpaceLT:
    if dctTimeLT.has_key(key) == True:
        x = x + 1

knoxAPairs = x

gp.AddMessage("-----\
-----")
gp.AddMessage("Knox Statistic (R) = " + str(knoxAPairs))

spaceShuffleListLength = len(spaceShuffleList)
timeShuffleListLength = len(timeShuffleList)
dctSpaceLength = len(dctSpaceLT)

gp.AddMessage("Knox(R) Frequency  Prob   CumProb")
gp.AddMessage("-----")

start6 = clock()

mcKnoxlist = []
randrangeTimeShuffle = int(timeShuffleListLength)

#Initiate loop that will perform the individual Monte Carlo simulations
#of the Knox Test
m = 1
while m <= monteC:
    #Clear the list holding the spatial distance between pairs values
#that met the critical spatial distance parameter
    dctSpaceLT.clear()
    #Shuffle the list containing all the spatial distances between
#pairs
    shuffle(spaceShuffleList)

    #Conduct the spatiotemporal analysis

```

```

x = 1
y = 1
s = 0
while x <= numEvents:
    for key in dctTime:
        if key <= numEvents:
            spaceDif = spaceShuffleList[s]
            keyA = x
            keyB = y
            if spaceDif >= 0 and spaceDif <= critDist and keyA < \
keyB:
                dctSpaceLT[keyA, keyB] = spaceDif
            else:
                pass
            y = y + 1
            s = s + 1
        x = x + 1
        y = 1

k = 0
for key in dctSpaceLT:
    if dctTimeLT.has_key(key) == True:
        k = k + 1
knoxR = k
#Append resulting Knox R to a list of all Knox R values created by
#each iteration
mcKnoxlist.append(knoxR)
m = m + 1

stop6 = clock()
timePassed6 = round((stop6 - start6),2)
perSim = timePassed6 / monteC

start7 = clock()

#From list of Knox R values generated during the simulation above,
#select the max
maxKnox = max(mcKnoxlist)
#From list of Knox R values generated during the simulation above,
#select the min
minKnox = min(mcKnoxlist)

#Initiate dictionary to store the calculated Knox R probability values
mcKnoxprobDict = {}
smallestProb = 1 / float(monteC)

x = minKnox
cumProb = float(0)

#Calculate the probability for each Knox R value based on its
#occurrence during within the simulations generated
while x <= maxKnox:
    key = x
    countKnox = mcKnoxlist.count(x)

```



```

        countKnoxprob = float(countKnox) / float(monteC)
        mcKnoxprobDict[key] = countKnoxprob
        cumProb = cumProb + countKnoxprob
        gp.AddMessage(" %-6d  %-6d  %0.3f  %0.3f" % (x, countKnox,\
countKnoxprob, cumProb))
        x = x + 1

knoxP = 0
for item in mcKnoxlist:
    if item >= knoxAPairs:
        knoxP = knoxP + 1

if knoxP == 0:
    probKnox = ("< " + str(smallestProb))
else:
    probKnox = float(knoxP)/float(monteC)

#Calculate the mean and variance of the Knox R values occuring in the
#simulation
avgKnox = round(lmean(mcKnoxlist),2)
varKnox = round(lvar(mcKnoxlist),2)

gp.addmessage("")
gp.AddMessage("-----\
-----")
gp.AddMessage("")
gp.AddMessage("Given the simulation, the probability of Knox (R) = " +\
str(knoxAPairs) + " is " + str(probKnox))
gp.AddMessage("The simulation mean is " + str(avgKnox))
gp.AddMessage("The simulation variance is " + str(varKnox))

stop7 = clock()
timePassed7 = round((stop7 - start7),2)

gp.AddMessage("")

gp.AddMessage("Monte Carlo test run time: " + str(timePassed6) + \
" seconds.")
gp.AddMessage("Monte Carlo per sim time: " + str(perSim) + " seconds.")

gp.AddMessage("")

#Delete variables
del dctSpaceLT, dctTime, dctTimeLT, numCritTimePairs, critTime,\
numEvents, smallestProb, spaceShuffleList
del timeShuffleList, avgKnox, varKnox, maxKnox, minKnox, mcKnoxlist,\
probKnox, mcKnoxprobDict, spaceShuffleListLength, timeShuffleListLength
del knoxAPairs, numCritSpacePairs, adjustTimeID, adjustOrigID,\
adjustDestID

```

Tool 5: ST Cluster Range Detector

```
#-----
#Name: SCAn Tool 5 - ST Cluster Range Detector
#Created by: David Eckley
#Date created: 20101128
#Purpose: ST Cluster Range Detector conducts a nearest neighbor
#distance analysis of the spatial and temporal dimensions and reports a
#recommended range of spatio-temporal test critical parameters.
#-----

#=====
#IMPORT MODULES
#=====
import sys, string, os, arcgisscripting, math

#=====
#INITIATE GEOPROCESSOR
#=====
gp = arcgisscripting.create(9.3)

#=====
#DEFINE VARIABLES PROVIDED BY USER
#=====

#Network dataset
netDataset = gp.GetParameter(0)

#Impedence/cost attribute for the network dataset
impedence = gp.GetParameterAsText(1)

#Feature class with event details
tempFC = gp.GetParameter(2)

#Type of temporal analysis (boolean variable).If True, hour analysis.
#If False, day analysis.
timeType = gp.GetParameter(3)

#Feature class field containing dates of events
dateFld = gp.GetParameterAsText(4)

#Feature class field containing hour event occurred in 24-hour HH
#format (e.g. 7AM is 7; 7PM is 19)
hourFld = gp.GetParameterAsText(5)

#Ensure that hour data field was provided if hourly analysis was
#selected above. Otherwise, exit program.
if timeType > 0 and hourFld == "":
    gp.addmessage("")
    gp.addmessage("In order to conduct hourly analysis, a field\
containing hour data in a 24-hour HH format must be provided.")
    gp.addmessage("")
```

```

        sys.exit()
    else:
        pass

#=====
#CONDUCT SPATIAL NEAREST NEIGHBOR RANGE ANALYSIS
#=====

# Define local variables used in the Make Closest Facility Layer
#analysis.
CFLayer = "ClosestFacilityLayer"
RouteLayer = "ClosestFacilityLayer\\Routes"
CFRouteLayer = "CFRoutes_Layer"

# Make Closest Facility Layer. This analysis searches for the closest
#and second closest spatial neighbors between two point feature layers;
#in this case both layers are the same.
gp.MakeClosestFacilityLayer_na(netDataset, CFLayer, impedance, \
"TRAVEL_TO", "", "2", "", "ALLOW_UTURNS", "", "NO_HIERARCHY", "", \
"TRUE_LINES_WITH_MEASURES")
gp.AddLocations_na(CFLayer, "Facilities", tempFC, "CurbApproach #\
0;Attr_Length # 0", "5000 Meters", "OBJECTID", "", "MATCH_TO_CLOSEST", \
"APPEND", "SNAP", "5 Meters")
gp.AddLocations_na(CFLayer, "Incidents", tempFC, "CurbApproach #\
0;Attr_Length # 0", "5000 Meters", "OBJECTID", "", "MATCH_TO_CLOSEST", \
"APPEND", "SNAP", "5 Meters")
gp.Solve_na(CFLayer, "HALT")

# Select only the second nearest neighbors. The first nearest neighbor
#is the point itself since this analysis is looking at two identical
#point layers.
gp.SelectData_management(CFLayer, "Routes")

# Make a feature layer from the selected events above.
gp.MakeFeatureLayer_management(RouteLayer, CFRouteLayer, \
 "\"FacilityRank\" = 2", "", "FacilityID FacilityID VISIBLE\
NONE;FacilityRank FacilityRank VISIBLE NONE;Name Name VISIBLE\
NONE;IncidentCurbApproach IncidentCurbApproach VISIBLE\
NONE;FacilityCurbApproach FacilityCurbApproach VISIBLE NONE;IncidentID\
IncidentID VISIBLE NONE;Total_Length Total_Distance VISIBLE NONE")

#Rename variable storing feature layer
netNN = CFRouteLayer

#Initiate a list to store the spatial nearest neighbor distance values
nDistList = []

#Initiate the geoprocessor cursor to extract the spatial nearest
#neighbor distance value for each event
cur = gp.SearchCursor(netNN)
row = cur.Next()
while row:
    nDist = int(row.GetValue("Total_Distance"))
    nDistList.append(nDist)

```

```

row = cur.Next()

#Count the values in the list
nNcount = len(nDistList)
#Assign the minimum Spatial Nearest Neighbor Distance to a variable
nNmin = min(nDistList)
#Assign the maximum Spatial Nearest Neighbor Distance to a variable
nNmax = max(nDistList)
#Assign the avg NN spatial dist to a variable
nNavg = sum(nDistList) / nNcount

#Round avg nearest neighbor distance value to the next highest multiple
#of 100, unless the value is exactly a multiple of 100
nNavgRound = round(nNavg / 10)
roundStr = str(nNavgRound)
roundVal = int(roundStr[-3])
if roundVal > 0:
    spaceCritMax = ((10 - float(roundVal)) + nNavgRound)*10
else:
    spaceCritMax = nNavgRound * 10

#Round min nearest neighbor distance value to the next lowest multiple
#of 100, unless the value is exactly a multiple of 100
if nNmin > 0:
    nNminRound = round(nNmin / 10)
    roundStr = str(nNminRound)
    roundVal = int(roundStr[-3])
    if roundVal > 0:
        spaceCritMin = (nNminRound - roundVal)*100
    else:
        spaceCritMin = nNminRound * 100
else:
    spaceCritMin = 0

gp.addmessage("")
gp.addmessage("Minimum Network Nearest Neighbor Distance: " + \
str(nNmin) + " meters.")
gp.addmessage("Maximum Network Nearest Neighbor Distance: " + \
str(nNmax) + " meters.")
gp.addmessage("Average Network Nearest Neighbor Distance: " + \
str(nNavg) + " meters.")
gp.addmessage("Based on the nearest neighbor distance analysis, a\
recommended range of spatial critical distances is between " + \
str(spaceCritMin) + " and " + str(spaceCritMax) + " meters.")

#=====
#CONDUCT SPATIAL NEAREST NEIGHBOR RANGE ANALYSIS
#=====

#Define a variable for the feature layer field that will hold the
#calculated timeline value
calcFld = "CALC_TIME"

```

```

#Create SQL statement for the field calculation if hourly analysis is
#selected
expnHour = "["+str(dateFld)+"]*24+["+str(hourFld)+"]"
#Create SQL statement for the field calculation if day analysis is
#selected
expnDay = "["+str(dateFld)+"]*1"

#Local variables used in the calculate field process below
Output_Layer = "tempFC_L"
Output_Layer2 = "tempFC_L2"

#Define variable to hold boolean value
bool = int(timeType)

#If hourly analysis is selected, perform this section of code
x = 1
if bool == x:
    gp.addmessage("")
    gp.addmessage("Temporal Analysis Type: Hour")

    # Add and calculate timeline field to feature layer
    gp.MakeFeatureLayer_management(tempFC, Output_Layer)
    gp.AddField_management(Output_Layer, calcFld, "DOUBLE", "", "", \
"", "", "NULLABLE", "NON_REQUIRED", "")
    gp.CalculateField_management(Output_Layer, calcFld, expnHour, \
"VB", "")

    #Assign variable to output layer
    timeNN = Output_Layer

    #Assign variable to list holding the timeline values
    timeList = []

    #Search through feature layer and extract out the timeline values
    cur = gp.SearchCursor(timeNN)
    row = cur.Next()
    while row:
        time = row.GetValue(calcFld)
        timeList.append(time)
        row = cur.Next()

    #Sort list so that values are ascending
    timeList.sort()

    #Assign variable to number of items in list
    numTimes = len(timeList)

    #Search through the list of timeline values and calculate the
#neighbor distance between each event; store in list
    nList = []
    maxList = max(timeList)
    x=1
    y=0
    while x < numTimes:

```

```

        nList.append(timeList[x]-timeList[y])
        x=x+1
        y=y+1

    #Search through list of neighbor distances and select the nearest
    #neighbor distance for each event
    numDist = len(nList)
    nDistList = []
    x=1
    y=0
    while x < numDist:
        n=[nList[y],nList[x]]
        nn=min(n)
        nDistList.append(nn)
        x=x+1
        y=y+1

    #Count number of values in list
    nNcount = len(nDistList)
    #Calculate sum of values in list
    nNsum = float(sum(nDistList))
    #Determine minimum value in list
    nNmin = min(nDistList)
    #Determine maximum value in list
    nNmax = max(nDistList)
    #Determine average value in list
    nNavg = float(nNsum / nNcount)

    #Round avg nearest neighbor distance value to the next highest
    #multiple of 1, unless the value is exactly a multiple of 1
    nNavgRound = round(nNavg)
    roundStr = str(nNavgRound)
    roundVal = int(roundStr[-1])
    if roundVal > 0:
        timeCritMax = (10 - float(roundVal)) + nNavgRound
    else:
        timeCritMax = nNavgRound

    #Round min nearest neighbor distance value to the next lowest
    #multiple of 1, unless the value is exactly a multiple of 1
    if nNmin > 0:
        nNminRound = round(nNmin)
        roundStr = str(nNminRound)
        roundVal = int(roundStr[-1])
        if roundVal > 0:
            timeCritMin = nNminRound - roundVal
        else:
            timeCritMin = nNminRound
    else:
        timeCritMin = 0

    gp.addressmessage("Minimum Temporal Nearest Neighbor Distance: " +\
str(nNmin) + " hours.")

```

```

        gp.addmessage("Maximum Temporal Nearest Neighbor Distance: " +\
str(nNmax) + " hours.")
        gp.addmessage("Average Temporal Nearest Neighbor Distance: " +\
"%0.1f" % (nNavg) + " hours.")
        gp.addmessage("Based on the nearest neighbor distance analysis, a\
recommended range of temporal critical distances is between " +\
str(timeCritMin) + " and " + str(timeCritMax) + " hours.")
        gp.addmessage("")

        del expnHour, expnDay, nNcount, nNsum, nNmin, nNmax, nNavg,\
numDist, nDistList, nList, numTimes, timeNN

#All of the code annotation above is the same for the loop below. The
#following loop performs the same functions for a day analysis instead
#of hourly.
else:
    gp.addmessage("")
    gp.addmessage("Temporal Analysis Type: Day")

    gp.MakeFeatureLayer_management(tempFC, Output_Layer2)
    gp.AddField_management(Output_Layer2, calcFld, "DOUBLE", "", "",\
"", "", "NULLABLE", "NON_REQUIRED", "")
    gp.CalculateField_management(Output_Layer2, calcFld, expnDay,\
"VB", "")

    timeNN = Output_Layer2
    timeList = []

    cur = gp.SearchCursor(timeNN)
    row = cur.Next()
    while row:
        time = int(row.GetValue(calcFld))
        timeList.append(time)
        row = cur.Next()

    timeList.sort()
    numTimes = len(timeList)

    nList = []
    maxList = max(timeList)
    x=1
    y=0
    while x < numTimes:
        nList.append(timeList[x]-timeList[y])
        x=x+1
        y=y+1

    numDist = len(nList)
    nDistList = []
    x=1
    y=0
    while x < numDist:
        n=[nList[y],nList[x]]
        nn=min(n)

```

```

        nDistList.append(nn)
        x=x+1
        y=y+1

nNcount = len(nDistList)
nNsum = float(sum(nDistList))
nNmin = min(nDistList)
nNmax = max(nDistList)
nNavg = float(nNsum / nNcount)

nNavgRound = round(nNavg)
roundStr = str(nNavgRound)
roundVal = int(roundStr[-1])
if roundVal > 0:
    timeCritMax = (10 - float(roundVal)) + nNavgRound
else:
    timeCritMax = nNavgRound

if nNmin > 0:
    nNminRound = round(nNmin)
    roundStr = str(nNminRound)
    roundVal = int(roundStr[-1])
    if roundVal > 0:
        timeCritMin = nNminRound - roundVal
    else:
        timeCritMin = nNminRound
else:
    timeCritMin = 0

gp.addmessage("Minimum Temporal Nearest Neighbor Distance: " +\
str(nNmin) + " days.")
gp.addmessage("Maximum Temporal Nearest Neighbor Distance: " +\
str(nNmax) + " days.")
gp.addmessage("Average Temporal Nearest Neighbor Distance: " +\
"%0.1f" % (nNavg) + " days.")
gp.addmessage("Based on the nearest neighbor distance analysis, a\
recommended range of temporal critical distances is between " +\
str(timeCritMin) + " and " + str(timeCritMax) + " days.")
gp.addmessage("")

del expnHour, expnDay, nNcount, nNsum, nNmin, nNmax, nNavg,\
numDist, nDistList, nList, numTimes, timeNN

```


References

References

- Anon.2009. Weather Underground. *Weather History for Columbus, OH*. March.
<http://www.wunderground.com/>.
- Anon.2010a. Significance Tests for Counts. *The Statistics Calculator*.
<http://www.statpac.com/statistics-calculator/counts.htm>.
- Anon.2010b. Ohio Highway Shapefile from OpenStreetMap (www.openstreetmap.com).
www.mapcruzin.com. 7. <http://www.mapcruzin.com/free-united-states-shapefiles/free-ohio-arcgis-maps-shapefiles.htm>.
- Anselin, L. 1995. Local indicators of spatial association: LISA. *Geographical Analysis* 27: 93-115.
- Assuncao, R, and T Correa. 2009. Surveillance to detect emerging space-time clusters. *Computational Statistics and Data Analysis* 53, no. 8 (June 15): 2817-2830.
- Assuncao, R, A Tavares, T Correa, and M Kulldorff. 2007. Space-time cluster identification in point processes. *The Canadian Journal of Statistics* 35, no. 1: 9-25.
- Baker, R. 1996. Testing for space-time clusters of unknown size. *Journal of Applied Statistics* 23, no. 5 (10): 543-554.
- Black, W. 1991. Highway accidents: a spatial and temporal analysis. *Transportation Research Record*, no. 1318: 75-82.
- . 1992. Network autocorrelation in transport network and flow systems. *Geographical Analysis* 24, no. 3: 207-222.
- Block, R. 2007. Scanning for clusters in space and time - A tutorial review of SaTScan. *Social Science Computer Review* 25, no. 2: 272-278.
- Chang, W, D Zeng, and H Chen. 2008. A stack-based prospective spatio-temporal data analysis approach. *Decision Support Systems* 45, no. 4 (November): 697-713.

- Clark, P, and F Evans. 1954. Distance to Nearest Neighbor as a Measure of Spatial Relationships in Populations. *Ecology* 35, no. 4 (10): 445.
- Clements, F. 1905. *Research Methods in Ecology*. Lincoln, NE: University Publishing Company.
- Cliff, A, and J Ord. 1981. *Spatial Processes: Models & Applications*. London: Pion.
- David, F, and D Barton. 1966. Two Space-Time Interaction Tests for Epidemicity. *British Journal of Preventive and Social Medicine* 20, no. 1 (January): 44-48.
- Flahaut, B, M Mouchart, E Martin, and I Thomas. 2003. The local spatial autocorrelation and the kernel method for identifying black zones a comparative approach. *Accident Analysis & Prevention* 35: 991-1004.
- Glass, A, and N Mantel. 1969. Lack of Time-Space Clustering of Childhood Leukemia in Los Angeles County, 1960–1964. *Cancer Research* 29, no. 11 (November): 1995-2001.
- Hochberg, Y. 1988. A sharper Bonferroni procedure for multiple test of significance. *Biometrika* 75: 800-802.
- Holland, B, and M Copenhave. 1987. An improved sequentially rejective Bonferroni test procedure. *Biometrics* 43: 417-423.
- Jacquez, G. 1996. A k-nearest neighbour test for space-time interaction. *Statistics in Medicine* 15, no. 18 (9): 1935-1949.
- . 2008. Spatial Cluster Analysis. In *The Handbook of Geographic Information Science*, ed. S Fotheringham and J Wilson, 395-416. Blackwell Publishing.
- Kennedy, E. 2010. ODPS Crash Extracts. *Ohio Traffic Safety Office*. 7. <http://www.dps.state.oh.us/CrashRequests/extract.aspx>.
- Khan, G, K Santiago-Chaparro, X Qin, and D Noyce. 2009. Application and Integration of Lattice Data Analysis, Network K-Functions, and Geographic Information System Software to Study Ice-Related Crashes. *Transportation Research Record*, no. 2136: 67-76.
- Klauber, M. 1971. Two-Sample Randomization Tests for Space-Time Clustering. *Biometrics* 27, no. 1 (March): 129-142.
- Klauber, M, and P Mustacchi. 1970. Space-time clustering of childhood leukemia in San Francisco. *Cancer Research* 30: 1969-1973.

- Knox, E, and M Bartlett. 1964. The Detection of Space-Time Interactions. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 13, no. 1: 25-30.
- Knox, E, and E Gilman. 1992. Leukaemia clusters in Great Britain. 1. Space-time interactions. *Journal of Epidemiology and Community Health* 46, no. 6 (December): 566-572.
- Kuldorff, M. 2001. Prospective time periodic geographical disease surveillance using a scan statistic. *Journal of the Royal Statistical Society A* 164, no. 1: 61-72.
- Kulldorff, M. 2006. *SaTScan user guide*. August. <http://www.satscan.org>.
- Kulldorff, M, and U Hjalmar. 1999. The Knox method and other tests for space-time interaction. *Biometrics* 55, no. 2 (June): 544-552.
- Lesniak, A, and Z Isakow. 2009. Space-time clustering of seismic events and hazard assessment in the Zabrze-Bielszowice coal mine, Poland. *International Journal of Rock Mechanics and Mining Sciences* 46, no. 5 (July): 918-928.
- Lloyd, S, and C Roberts. 1973. A Test for Space Clustering and Its Application to Congenital Limb Defects in Cardiff. *British Journal of Preventive and Social Medicine* 27, no. 3: 188-191.
- Lord, D, and F Mannering. 2010. The statistical analysis of crash-frequency data: A review and assessment of methodological alternatives. *Transportation Research Part A: Policy and Practice* 44, no. 5 (June): 291-305.
- Mantel, N. 1967. The Detection of Disease Clustering and a Generalized Regression Approach. *Cancer Research* 27, no. 2 (February): 209-220.
- Meighan, S, and G Knox. 1965. Leukemia in childhood. Epidemiology in oregon. *Cancer* 18, no. 7 (7): 811-814.
- Mennis, D. 2010. Multidimensional Map Algebra: Design and Implementation of a Spatio-Temporal GIS Processing Language. *Transactions in GIS* 14, no. 1 (2): 1-21.
- Michaelsen, J. 2007. Confirmatory vs Exploratory Data Analysis January 8, University of California Santa Barbara.
http://www.geog.ucsb.edu/~joel/g210_w07/lecture_notes/lect01/oh07_01_2.html.
- Mielke, P. 1978. Clarification and Appropriate Inferences for Mantel and Valand's Nonparametric Multivariate Analysis Technique. *Biometrics* 34, no. 2: 277-282.

- Miller, H. 1999. Measuring space-time accessibility benefits within transportation networks: basic theory and computational procedures. *Geographical Analysis* 31: 187-212.
- Mirghani, S, B Nour, S Bushra, I Elhassan, R Snow, and A Noor. 2010. The spatial-temporal clustering of Plasmodium falciparum infection over eleven years in Gezira State, The Sudan. *Malaria Journal* 9.
- Moran, P. 1950. Notes on continuous stochastic phenomena. *Biometrika* 37: 17-23.
- Mountrakis, G, and K Gunson. 2009. Multi-scale spatiotemporal analyses of moose-vehicle collisions: a case study in northern Vermont. *International Journal of Geographical Information Science* 23, no. 11: 1389-1412.
- Nakaya, T, and K Yano. 2010. Visualising Crime Clusters in a Space-time Cube: An Exploratory Data-analysis Approach Using Space-time Kernel Density Estimation and Scan Statistics. *Transactions in GIS* 14, no. 3 (6): 223-239.
- von Neumann, J. 1941. Distribution of the Ratio of the Mean Square Successive Difference to the Variance. *The Annals of Mathematical Statistics* 12, no. 4 (December): 367-395.
- Okabe, A, B Boots, K Sugihara, and S Chui. 2000. *Spatial Tessellations - Concepts and Applications of Voronoi Diagrams*. John Wiley.
- Okabe, A, K Okunuki, and SANET Team. 2009. *SANET: A Spatial Analysis on Networks*. Tokyo, Japan. <http://sanet.csis.u-tokyo.ac.jp/>.
- Okabe, A, and T Satoh. 2006. Uniform network transformation for points pattern analysis on a non-uniform network. *Journal of Geographical Systems* 8: 25-37.
- Okabe, A, and I Yamada. 2001. The k-function method on a network and its computational implementation. *Geographical Analysis* 33, no. 3: 271-290.
- Okabe, A, H Yomono, and M Kitamura. 1995. Statistical Analysis of the Distribution of Points on a Network. *Geographical Analysis* 27, no. 2: 152-175.
- Okabe, A, T Yoshikawa, A Fujii, and K Oikawa. 1988. The statistical analysis of a distribution of activity points in relation to surface-like elements. *Environment and Planning A* 20, no. 5: 609 – 620.
- Okunuki, K, and A Okabe. 2002. Solving the Huff-based competitive location model on a network with link-based demand. *Annals of Operations Research* 111: 239-252.

- Ord, J, and A Getis. 1995. Local spatial autocorrelation statistics: distributional issues and an application. *Geographical Analysis* 27: 286-306.
- Parzen, E. 1962. On estimation of a probability density function and mode. *Annals of Mathematical Statistics* 33: 1065-1076.
- Pei, T, C Zhou, A Zhu, B Li, and C Qin. 2010. Windowed nearest neighbour method for mining spatio-temporal clusters in the presence of noise. *International Journal of Geographical Information Science* 24, no. 6: 925-948.
- Pinder, D, and M Witherick. 1973. Nearest-neighbour analysis of linear point patterns. *Tijdschrift voor Economische en Sociale Geografie* 64: 160-163.
- Ripley, D. 1981. *Spatial Statistics*. Chichester: Wiley.
- Roberts, C, K Laurence, and S Lloyd. 1975. An Investigation of Space and Space-Time Clustering in a Large Sample of Infants with Neural Tube Defects Born in Cardiff. *British Journal of Preventive and Social Medicine* 29, no. 3: 202-204.
- Rogerson, P. 2001. Monitoring point patterns for the development of space-time clusters. *Journal of the Royal Statistical Society Series A - Statistics in Society* 164: 87-96.
- Shiode, S. 2008. Analysis of a distribution of point events using the network-based quadrat method. *Geographical Analysis* 40: 380-400.
- Shiode, S, and N Shiode. 2009. Detection of multi-scale clusters in network space. *International Journal of Geographical Information Science* 23, no. 1: 75-92.
- Sidak, Z. 1967. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association* 62: 626-633.
- Siemiatycki, J. 1978. Mantel's space-time clustering statistic: computing higher moments and a comparison of various data transforms. *Journal of Statistical Computation and Simulation* 7, no. 1: 13-31.
- Simes, R. 1986. An Improved Bonferroni Procedure for Multiple Tests of Significance. *Biometrika* 73, no. 3 (December): 751-754.
- Smith, P, M Pike, M Till, and R Hardisty. 1976. Epidemiology of childhood leukaemia in greater london: A search for evidence of transmission assuming a possibly long latent period. *British Journal of Cancer* 33, no. 1: 1-8.
- Tukey, J. 1977. *Exploratory Data Analysis*. Philippines: Addison-Wesley.

- Upton, G. 1985. *Spatial Data Analysis by Example*. Chichester: Wiley.
- Voronoi, G. 1907. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik* 133: 97-178.
- Williams, G. 1984. Time-space clustering of disease. In *Statistical Methods for Cancer Studies*, ed. R Cornell, 479. New York: Marcel Dekker, Inc.
- Yamada, I, P Rogerson, and G Lee. 2009. GeoSurveillance: a GIS-based system for the detection and monitoring of spatial clusters. *Journal of Geographical Systems* 11, no. 2: 155-173.
- Yamada, I, and J Thill. 2004. Comparison of planar and network k-functions in traffic accident analysis. *Journal of Transport Geography* 12: 149-158.
- . 2007. Local Indicators of Network-Constrained Clusters in Spatial Point Patterns. *Geographical Analysis* 39, no. 3: 268-292.
- . 2010. Local Indicators of Network-Constrained Clusters in Spatial Patterns Represented by a Link Attribute. *Annals of the Association of American Geographers* 100, no. 2: 269-285.
- Young, D. 1982. The Linear Nearest Neighbour Statistic. *Biometrika* 69, no. 2: 477-480.

Curriculum Vitae

David C. Eckley graduated from West Valley High School, Cottonwood, California, in 1996. He received his Bachelor of Science in Mapping, Charting, and Geodesy from the United States Military Academy, West Point, New York in 2000. He has served as an officer in the United States Army for the past ten years and received his Master of Science in Geographic and Cartographic Science from George Mason University in 2011.