Visual Sample Plan and Prior Information

What do we need to know to find UXO?

Kenneth A. Flagg

Department of Mathematical Sciences
Montana State University

April 28, 2016

A writing project submitted in partial fulfillment
of the requirements for the degree

Master of Science in Statistics

APPROVAL

of a writing project submitted by

Kenneth A. Flagg

This writing project has been read by the writing project advisor and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the Statistics Faculty.

Date	Megan D. Higgs Writing Project Advisor
Date	Mark C. Greenwood Writing Projects Coordinator

Abstract

Military training and weapons testing activities leave behind munitions debris, including both inert fragments and explosives that failed to detonate. The latter are known as unexploded ordnance (UXO). It is important to find and dispose of UXO items that are located where people could come into contact with them and cause them to detonate.

Typically there exists uncertainty about the locations of UXO items and the sizes of UXO-containing regions at a site, so statistical analyses are used to support decisions made while planning a site remediation project. The Visual Sample Plan software (VSP), published by the Pacific Northwest National Laboratory, is widely used by United States military contractors to guide sampling plan design and to identify regions that are likely to contain UXO.

VSP has many features used for a variety of situations in UXO cleanup and other types of projects. This study focuses on the sampling plan and geostatistical mapping features used to find target areas where UXO may be present. The software produces transect sampling plans based on prior information entered by the user. After the sample data are collected, VSP estimates spatial point density using circular search windows and then uses Kriging to produce a continuous map of point density across the site. I reviewed the software's documentation and examined its output files to provide insight about how VSP does its computations, allowing the software's analyses to be closely reproduced and therefore better understood by users.

I perform a simulation study to investigate the performance of VSP for identifying target areas at terrestrial munitions testing sites. I simulate three hypothetical sites, differing in the size and number of munitions use areas, and in the complexity of the background noise. Many realizations of each site are analyzed using methods similar to those employed by VSP to delineate regions of concentrated munitions use.

I use the simulations to conduct two experiments, the first of which explores the sensitivity of the results to different search window sizes. I analyze two hundred realizations of the simplest site using the same sampling plan and five different window sizes. Based on the results, I select 90% of the minor axis of the target area of interest as the window diameter for the second experiment.

The second experiment studies the effects of the prior information about the target area size and spatial point density of munitions items. For each site, I use four prior estimates of target area size and three estimates of point density to produce twelve sampling plans. One hundred realizations of each site are analyzed with each of the twelve sampling plans. I evaluate the analysis in terms of the detection rates of munitions items and target areas, the distances between undetected munitions items and identified areas, the total area identified, and other practical measures of the accuracy and efficiency of the cleanup effort. I conclude that the most accurate identification of target areas occurs when the sampling plan is based on the true size of the smallest target area present. The prior knowledge of the spatial point density has relatively little impact on the outcome.

Contents

1 Introduction

Warfare, by nature, involves handling and using dangerous munitions items. Even outside of wartime, new weapons are continually developed and tested, and troops are trained in their use. Munitions-related debris are inevitably left behind wherever military activities occur, at battlefields, training sites, and test sites both on land and at sea. These debris are typically bullets and inert metal fragments, but can also include toxic chemicals and live mines, grenades, or explosive shells that failed to detonate. Often, these items remain at a site for years. Each type of munitions device has a unique set of associated hazards and various methods have been developed to address them. This paper focuses specifically on explosives used in terrestrial warfare.

Explosive munitions items which fail to detonate and are left behind are known as *unexploded ordnance* (UXO) and present a potential danger to anyone who encounters them. The United States military conducts ground-based training activities in wilderness and rural areas where there is little immediate danger to bystanders. However, many of these sites are leased from civilian owners and returned to the owners when they are no longer needed by the military. Other sites are made available as public land after they are decommissioned. These areas are often used for agriculture or outdoor recreation, and may be transferred to private ownership for development. The Department of Defense funds projects to clean these sites and reduce the risk that UXO poses to civilians.

For example, the Motlow Range, near Tullahoma, Tennessee, was established in 1941 and used by the US Army for artillery testing. The site is now used largely for agriculture, but is also occupied by several houses and a community college (Pulpisher et al. 2014). In Montana's Helena Valley, part of the Chevallier Ranch was used in the 1940s and 1950s by the Montana Army National Guard for tank and artillery training. Urban growth around the city of Helena has resulted in residential areas encroaching upon the former firing ranges (Neptune and Company, Inc. 2008). There is a possibility that live explosives remain at these locations. In the interest of safety, these sites have undergone efforts to find and remove munitions items before they are unintentionally encountered by the public.

In many cases there exist historical records of where military activities took place, sometimes with detailed information about the locations from which the munitions were fired and the locations of the intended targets. In other cases, such detailed information is not available and so the process of finding and removing ordnance involves a high degree of uncertainty. Sophisticated statistical techniques and software have been developed to help manage this uncertainty.

One software package that has become commonly used in recent years is Visual Sample Plan (VSP Development Team 2016), published by the Pacific Northwest National Laboratory (PNNL). It includes tools to help design statistical sampling plans, analyze sample data to identify possible munitions use areas, and implement quality control measures. VSP is recommended by several government agencies and is designed to be used by individuals who are not experts in statistics.

To produce a sampling plan, VSP relies heavily on prior information contained in the conceptual model of the site. Specifically, the software requires information about the sizes and shapes of the munitions use areas and the spatial density of munitions-related items in these areas. There may be a lot of uncertainty in these quantities, especially if a detailed history of the site is not available. Additionally, mapping the level of contamination across the site after sampling requires

selecting the size of a search window in which the software computes a moving average of spatial point density. This window size parameter has no direct connection to the conceptual site model, so it is also of interest to assess the importance of the window size choice.

Previous case studies and simulation studies performed by the development team at PNNL have verified the accuracy of probabilities and confidence levels reported by VSP, and concluded that sampling plans created with VSP are adequate for verifying that a cleanup is successful or that the quantity of munitions items in a sampling area is acceptably low (Pulsipher et al. 2011, Pulpisher et al. 2014). There is less information available about the quality of VSP's results when the objective is identification of unknown munitions use areas.

The goals of this project are (1) to exposit the methods and assumptions used by VSP to identify munitions use areas, (2) to assess the sensitivity of these methods to different prior information about the site, and (3) to provide a starting point for further studies of target area delineation methodologies.

I accomplish the first goal through a thorough review of guidance documents, the Visual Sample Plan documentation, and the help files included with the software[1] (Sections 2 and 3). A simulation study addresses the second goal (Sections 4–6). Code for the simulations is provided in Appendix A and online.[2] Finally, discussions of future investigations and comparisons with other analysis methods contribute to the third goal (Section 7).

For the simulation study, I define conceptual models of three tank and artillery training sites with different levels of complexity. These models define target areas where munitions items are concentrated, and I simulate many realizations from these models using spatial Poisson processes to generate the locations of munitions items and background noise items. I then implement the entire process of designing a sampling plan and identifying possible munitions-containing regions on the realized sites using the techniques employed by VSP.

Success of the VSP analysis is measured for each realization by the *detection rate* of munitions items (where an item is considered detected if it is within a region identified as a possible target area), the distance between each undetected munitions item and the nearest identified possible target area (*error distance*), and the proportion of the true target areas included in the regions identified as potentially containing munitions. Efficiency of the sampling and cleanup effort is measured through the distance traversed while sampling, the total area identified for cleanup, the proportion of the identified regions not actually belonging to the true target areas (*false positive proportion*), and the number of distinct possible munitions-containing regions that are identified.

I have prepared this document in the hope that it will not only convey the simulation results, but that it will also be useful both to the statistician wishing to learn more about UXO data and VSP, and to the VSP user who desires to gain a deeper understanding of the underlying statistical methods and assumptions.

[1] The most up-to-date help files are also available at `http://vsp.pnnl.gov/help/`.

[2] The files for this project are publicly available on Github at `https://www.github.com/kflagg/vspuxo`.

2 Remediation of Unexploded Ordnance Sites

The general process for clearing a site of UXO is described in detail in the U.S. Army Corps of Engineers Interim Guidance Document, "Technical Guidance for Military Munitions Response Actions" (USACE IGD 14-01). An introduction to the methodology and definitions of important terms are presented below in Section 2.1. I summarize the procedures themselves in Section 2.2. The procedures are the same regardless of the software used for planning and analysis, and IGD 14-01 describes several software options. Software packages with features specifically meant for UXO projects include Visual Sample Plan and UXO Estimator. The guidance document also mentions some general-purpose geostatistical and GIS software packages like ArcGIS, but VSP receives the most thorough discussion because it has the most complete set of UXO-specific features.

2.1 Detecting Unexploded Ordnance

At a military site, most munitions-related items are located in regions of concentrated munitions use surrounding targets, known as *target areas* (TAs). Extant UXO is expected to be contained in a TA, so the standard methodology for locating UXO focuses on finding the TAs. The information in this section comes from Chapters 6 and 8 of IGD 14-01, which contain a great deal of technical details that I omit here.

Munitions are readily detected by magnetometers and *digital geophysical mapping* (DGM) equipment. These devices range in size from handheld sensors to larger arrays that are towed by vehicles. They record a continuous signal in response to magnetic fields as they traverse the surface of a site. Peaks in the signal represent nearby metallic items. These items are called *anomalies*, and the spatial locations of anomalies are recorded when they are encountered during data collection. An anomaly may or may not be a munitions item, but the spatial distribution of anomaly locations is the key to identifying target areas.

Anomalies which are ordnance or related debris are known as *targets of interest* (TOIs). It is important to specifically define TOI for a particular site. This allows DGM equipment to be calibrated to detect the type of munitions that are likely to be present, reducing the number of non-TOI anomalies that are mistaken as possible TOIs. Additionally, ballistic properties have been well documented for all munitions commonly used by the U.S. military. The distance fragments typically travel from the impact point can provide an initial estimate of the size of the TAs, and the maximum penetration depth of the munitions can also be estimated and used to decide how deep to dig during the cleanup.

Anomalies which are not TOIs may be ferrous rocks or other naturally-occurring geological items, or they can be things like bottle caps, nails, and coins that arrived at the site through human activities entirely unrelated to munitions use. These anomalies are considered background noise, so part of the goal of analyzing the DGM data is to separate the noise from the "signal" of true TOI items and find where the TOI items are located.

Background anomalies are typically treated as being homogeneously distributed. The spatial density of the background anomalies can often be known with reasonable certainty based on geologic information or survey data from areas thought to be free of munitions. If a site is believed to contain areas of munitions use, it is generally assumed that elevated anomaly density is due to

the presence of TOI. Therefore, the approach used to find possible target areas is to predict the anomaly density at each location of the site and identify regions of high density. There are many decisions and assumptions that must be made in the process.

2.2 The Cleanup Procedure

Clearing a former military site of dangerous material is typically an expensive and time-consuming project, so it is important that all procedures are carefully defined, and that appropriate quality control measures are used to assess whether the goals of the cleanup project are met. The basic procedure described in IGD 14-01 consists of three phases. The first phase is a preliminary investigation to decide where to focus cleanup efforts and how to allocate resources. The second phase involves the physical retrieval of all potential TOIs from areas believed to be dangerous, a process called *remediation*. The third phase is a quality control measure in which additional data are collected in order to judge whether or not the remediation was successful.

The focus of this study is on the accuracy of statistical tools used in the first phase, but all three phases are described below to provide context.

Phase 1: Remedial Investigation

The remedial investigation is a thorough study of the site in which all available information is compiled into a conceptual site model. This model will be used to plan and justify the remediation actions, so it must describe what types of hazards may be present and where they are likely to be located. It describes all aspects of the site, including munitions use history, other current and historical uses, layout, and geology. Such information is used to provide initial estimates of the number, sizes, and locations of target areas at the site. Any uncertain aspects of the conceptual site model should be augmented by collecting additional data and performing statistical analyses. This can be an iterative process, where information gained through sampling, such as the density of background anomalies or or additional details about target area locations, is used to update the conceptual model. Additional sampling may be needed to answer new questions as the site model is updated.

Sections of the site where munitions use is suspected are sampled with DGM equipment and the spatial anomaly density is mapped. Systematic transect sampling and random grid sampling are both frequently used, but transect sampling is more cost-effective and has been demonstrated via simulation to be more robust to complex background noise (Pulpisher et al. 2014). VSP may be used to develop a transect sampling plan (Section 3.2.1) and then to identify regions with high anomaly density based on the sample data (Sections 3.2.5 and 3.2.6).

Any regions where the spatial density of possible TOI items is much higher than the density due to background noise are considered potential TAs unless the elevated density can be conclusively explained by other aspects of the conceptual site model. Some regions may also be considered to have a high-risk of TOI presence based on prior information. All high-density or high-risk regions are considered for remediation. Ultimately the remediation cannot proceed until the project managers believe they can successfully clear the most hazardous sections of the site while staying under budget, so a risk assessment may be needed to prioritize different possible TAs. The end product of

the remedial investigation is a remediation plan defining the regions to be cleared of possible TOI, the area and depth that must be dug around each possible TOI item, and the maximum number of TOI items that could remain in the remediated regions without posing a risk to future users of the site.

Phase 2: Remediation

Once the remedial investigation concludes and potential TAs have been identified, the remediation occurs. The region selected for remediation undergoes a census in which the full area is swept with metal detectors or DGM devices. All anomalies that are possibly TOI have their locations recorded and are dug up. Several randomly selected anomalies that are not suspected to be TOI should also be dug to check that the DGM equipment is functioning properly.

Phase 3: Post-Remediation Verification

The final phase provides verification that the remediation process successfully cleared the area of TOI according to criteria defined during the remedial investigation. The remediated areas must be revisited and either a sample survey or another census is performed to ensure that no TOI items remain. In the case of a survey, the sampling units can be either transects or the locations of individual anomalies. Transect sampling is typically more cost-effective than anomaly sampling or a complete census (Pulsipher et al. 2011). VSP can compute the number of anomaly locations or transects which must be sampled to make a conclusion at a specified confidence level (Section 3.2.9).

As another check on equipment function, the remediation team seeds several TOI items at the site prior to data collection. If transect sampling is used, the seed items are placed in randomly selected locations within the transects selected for sampling. Otherwise, the seed items are placed at random locations throughout the site, and their locations are added to the list of sampled locations before the list is given to the DGM personnel.

If the anomaly detection equipment detects all the seed items and finds no other TOI items, the remediation is declared successful and the project leaders make a statement such as "we are 95% confident that at least 99% of the area is free of TOI." Otherwise, the remediation is unsuccessful and the project must return to the remedial investigation phase.

3 Visual Sample Plan

The Visual Sample Plan user's guide describes VSP as "a software tool for selecting the right number and location of environmental samples or transects so that the results of statistical tests performed on the data collected via the sampling plan have the required confidence for decision making"(Matzke et al. 2014). It also includes tools for analyzing data collected from the sampling plans that it creates. VSP is meant to provide autonomy for project managers who do not have statistical expertise but need to develop and justify sampling plans.

VSP was originally developed to help project managers implement the Environmental Protection Agency's Data Quality Objectives Process, a sequence of seven steps meant to ensure that envi-

ronmental studies are performed efficiently and effectively (EPA 600-R-96-055). The first six steps involve defining the inputs, outputs, goals, and scope of the project. The seventh step is to choose an optimized sampling design based on the needs stated in the previous steps. VSP's sampling design tools are organized so that users who follow steps 1–6 can enter information about their project and then easily compare different sampling plans to choose one that meets their needs.

Visual Sample Plan Version 1.0 was released in 2001. Its features are meant for geostatistical studies where a continuous response, such as the concentration of a chemical contaminant, can be measured at any point of the site; it has no UXO-specific features and the word "ordnance" does not appear in the user's guide (Davidson et al. 2001). UXO features have been gradually added to subsequent versions, and this expansion of the feature set is at least partly motivated by inappropriate application of VSP's other features to UXO sites (Pulpisher et al. 2014). Visual Sample Plan Version 7.5, released in early 2016, satisfies most or all of the data analysis needs of a typical UXO cleanup project.

However, Visual Sample Plan has by no means become a solely UXO-oriented software package. It includes features applicable to many different types of projects. To illustrate this diversity, Section 3.1 provides a brief overview of some of the software's capabilities. Section 3.2 gives more details about the features that apply specifically to UXO sites. In Section 3.3, I discuss the theory of spatial prediction and the technical details of the implementation in VSP, and Sections 3.3.2 and 3.3.4 describe how VSP interacts with GSLIB (Deutsch and Journel 1998), a library of geostatistical programs that provide VSP with geostatistical mapping abilities.

3.1 General and Non-UXO Features

The user interface of VSP is built around a graphical display of the site. The user can load a background image, or VSP can download maps or satellite images from MapQuest or Open Street Map (Figure 1). Detailed maps can be constructed by pointing and clicking to create polygons and simple shapes. Maps can also be imported and exported as shapefiles. Figure 2 (left) shows a map of a site used in the simulation study in Section 4.1.2). For sites within buildings, rooms may be drawn and viewed in 3D, including a selection of 3D models of furniture that can be placed in rooms (Figure 2, right).

Sections of the map can be defined as *sample areas* (seen as yellow regions in the figures). Within one site or building, multiple sample areas can be created. Each sample area can have its own sampling plan and be analyzed individually.

The sampling design and data analysis tools appear in the "Sampling Goals" menu and are grouped by purpose. There are general tools for estimating means and proportions, and for testing if the mean or proportion exceeds a threshold. Each of these features can create a sampling plan, place samples on the site map, and analyze collected data. Sampling methods include simple random sampling, stratified simple random sampling, systematic grid sampling, adaptive cluster sampling, and others. An example of the "Compare Average to Fixed Threshold" feature appears in Figure 3. These tools use classical Normal- and t-distribution techniques for proportions and means of Normally distributed responses. The user can also choose robust methods for non-Normal quantitative data.

Other features can locate and delineate hot spots of heavy contamination, use the Mann-Kendall test to detect trends, or use Kriging to map a variable across the sample area. There are tools meant specifically for use on radiological sites. The user can manually augment any sampling plan by specifying the locations of judgment samples; the help file includes a warning that the feature is provided "as a convenience and [the developers] cannot be held responsible for its misuse."

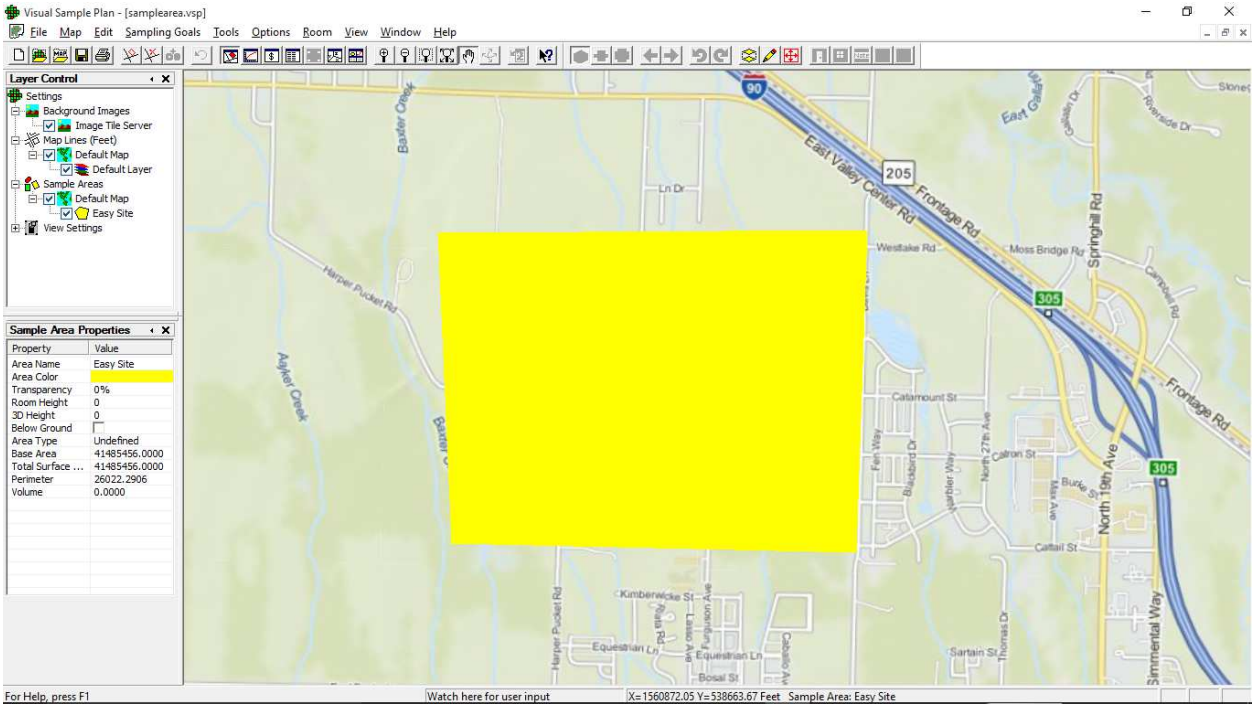


Figure 1: The VSP program window shows the site area used for the simulation study in Section 4. This is a 952.4 acre (roughly 1.4 by 1.1 mile) region situated northwest of Bozeman, Montana. A real geographic location is used only to demonstrate VSP's map feature; the simulated sites are entirely fictitious.

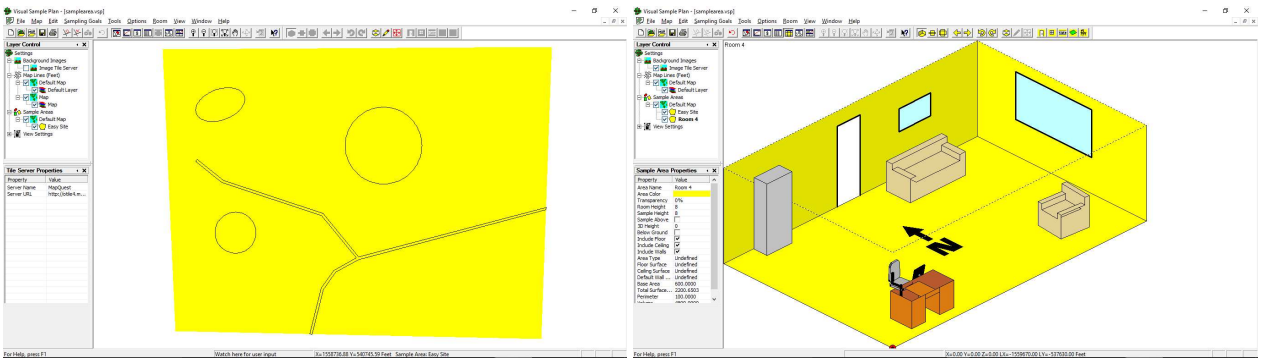


Figure 2: VSP allows drawing detailed maps and buildings with a simple point-and-click interface, and can import maps created in other software. The map (left) was imported via a shapefile. The room (right) was created entirely within VSP.

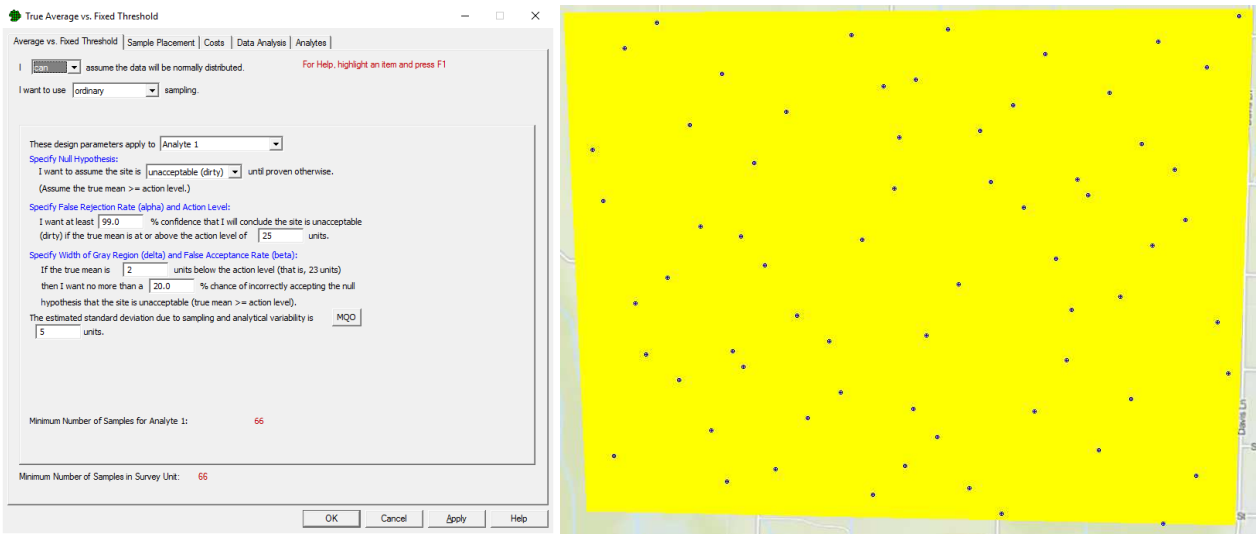


Figure 3: The user provides information about the project goals in the "Compare Average to Fixed Threshold" dialog box, and then VSP a generates a simple random sample of locations to be measured.

3.2 UXO-Specific features

This section describes the options available for analyzing UXO data, presented in the order in which they are employed if an analyst uses VSP for an entire project from start to finish. Sections 3.2.1–3.2.7 discuss remedial investigation features used for mapping and identifying target areas. Mapping may not be applicable to all parts of a site, so Section 3.2.8 describes features used for other aspects of the remedial investigation. Section 3.2.9 documents VSP's verification sampling plan tools.

Sections 3.2.5 and 3.2.6 introduce VSP's anomaly density mapping features, with an emphasis on the interaction between the user and the software. I replicate the methods used by these features in my simulation study, so I devote Section 3.3 to the theoretical background and technical details.

3.2.1 Transect Sampling Plans for Remedial Investigation

At any time during a remedial investigation, if the locations of target areas cannot be precisely estimated based on the information available, additional sample surveys should be conducted to collect more information. VSP constructs systematic plans of evenly-spaced transects for this purpose. The transects can run north-south, east-west, or in both directions to form a grid.

If the sample area has not been previously surveyed, the new transects will cover the entire region. If data have been loaded from a previous survey, VSP can compute the probability that the survey traversed a target area of a specified size, and an option is available to augment the previous survey by placing new transects in sections of the site that were not sampled.

VSP assists the user in selecting the between-transect spacing to achieve a certain probability of detecting a target area of specified size and density. In VSP, *transect spacing* is defined as the distance from the right edge of one transect to the left edge of the next transect to the right. The

space between the centerlines of adjacent transects is the spacing plus the transect width. The width is the footprint of the DGM equipment, and is input by the user. Figure 4 shows an example of a parallel transect plan on a sample area that was not previously surveyed. My simulations use only parallel transect sampling plans and areas not previously surveyed.

3.2.2 Target Area Detection Probability

Visual Sample Plan assists the user in creating a transect sampling plan by displaying a detection probability curve based on user-input information about a target area and the distribution of anomalies at the site. The target area is considered detected if at least one rectangular search window placed on a transect is found to have a higher anomaly density than the assumed background density. VSP computes the detection probabilities using Monte Carlo simulation, which relies on many probabilistic assumptions. The simulation and search window density computations are described in detail in Section 3.2.2.1.

The user inputs a detailed description of the prior information. This includes the transect width, the background anomaly density, the estimated size and anomaly density (above the background level) of the target area, and a false negative rate if it known that the equipment fails to detect a certain proportion of anomalies. The TA is assumed to be circular or elliptical with a random orientation. The user also specifies an interval of transect spacings over which VSP evaluates the detection probability. VSP then plots the detection probability curve as a function of transect spacing and the user clicks a point on the curve to set the spacing for the sampling plan. (Figure 5).

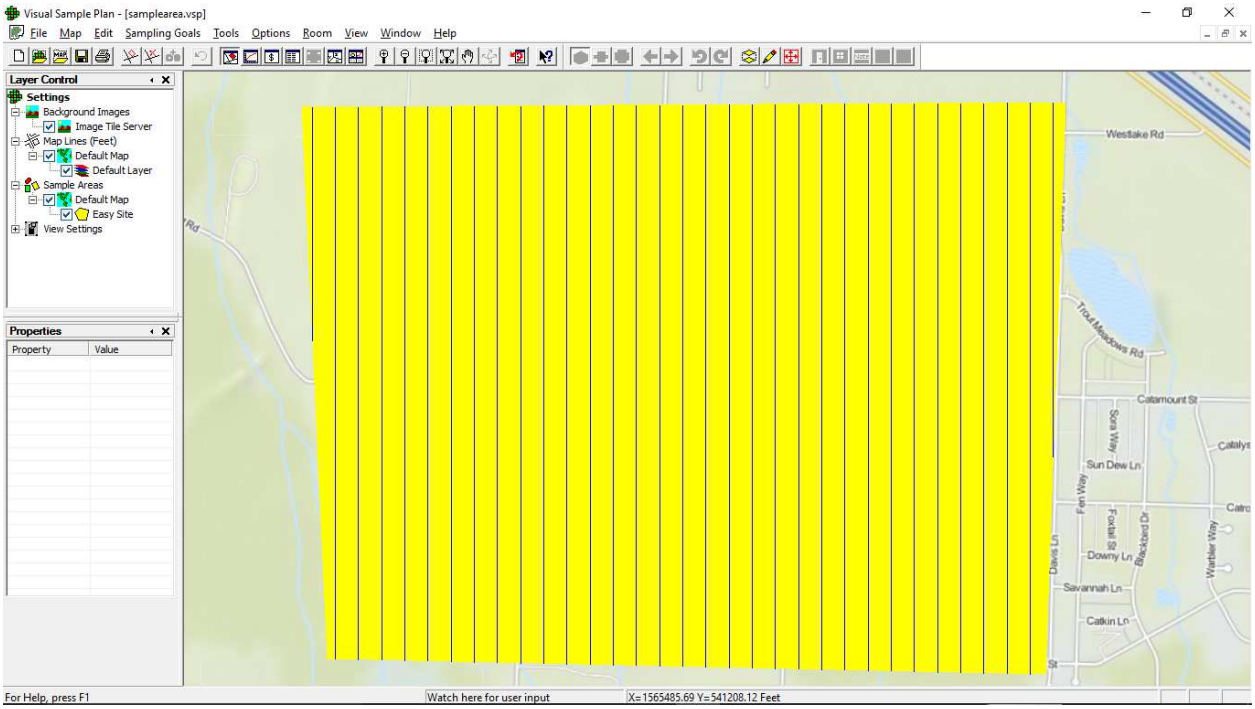


Figure 4: The VSP window shows a parallel transect sampling plan with 225 foot spacing between 6 foot wide transects.

Note that a TA is "detected" if the sample data provide evidence of its presence; the user must be aware that having evidence a TA exists is very different from producing an accurate map of it. If the sampling plan will be used for geostatistical mapping, a transect spacing selected to achieve a specified detection probability is, at best, a rough guideline.

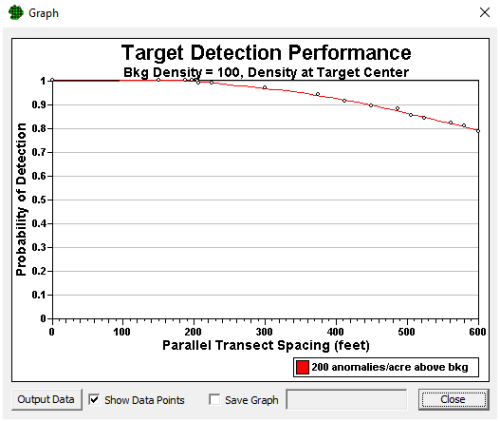


Figure 5: The VSP program window shows a detection probability curve assuming a 1,200 foot by 800 foot elliptical target area. The points on the curve were created by Monte Carlo simulation based on user-input assumptions about the size of the TA, the background anomaly density (100 anomalies per acre), and the TA anomaly density (200 anomalies per acre above the background density). The circles show the point estimates for each transect spacing that VSP simulated.

3.2.2.1 Details of the Detection Probability Simulation

Visual Sample Plan computes the detection probability curve using Monte Carlo simulation, where many realizations of the target area of interest are generated from a spatial point process, and then it applies a transect sampling plan to each realization. The software computes the proportion of realizations where the target area is detected in at least one rectangular search window, and uses that proportion as an estimate of the detection probability. It simulates additional realizations until the variability in the estimated probability falls below a threshold (Section 3.2.2.2). It repeats the simulation process to estimate the detection probability for multiple different transect spacings in a user-specified interval, and then fits a smooth curve to the estimates and plots the curve onscreen.

For each realization, VSP simulates a site with one elliptical target area. The target area has the size and shape specified by the user, and is rotated to a random orientation. The simulated site is just large enough to contain the target area. To generate background anomalies, VSP uses the user-specified background density and equipment false positive rate to compute the expected number of background anomalies that would be detected at the site. The software draws the number of realized background anomalies from a Poisson distribution where the mean is the expected number of detectable background anomalies, and then draws the spatial locations of these anomalies from a bivariate Uniform distribution. VSP then simulates TOI anomalies in a similar manner, using the input TA size and density to compute the expected number of detectable TOI anomalies, drawing the number of anomalies from the corresponding Poisson distribution, and generating the anomaly locations from the user's choice of target area distribution (Uniform or a bivariate Normal). If the

user selects a Uniform distribution, all TOI anomalies are placed inside the ellipse. In the Normal case, the distribution is scaled so that 99% of the TOI anomalies will be inside the ellipse.

The target area is considered detected if at least one rectangular search window is found to have a higher point density than the background density. After simulating the anomaly locations, the software lays down transects across the site, with the first transect placed at a random starting point. It places rectangular windows on the transects; the dimensions of the windows are the transect width by a window length parameter (Section 3.2.2.3). Multiple overlapping windows are placed on each transect, with the window centers separated by one-sixth of the window length.

For each window, the software uses a hypothesis test to decide if the anomaly density in the window is higher than the background level. Under the null hypothesis, the number of detected anomalies in the window is assumed to follow a Poisson distribution with a mean equal to the number of background anomalies expected to be found in the window if the prior information entered by the user is correct. If there is enough evidence at a certain significance level to reject the null hypothesis for any window, VSP records that the TA was detected for that realization. The significance level cannot be adjusted by the user and is not stated in the documentation.

One transect sampling plan is applied to each realization, but several different sampling plans are applied to different realizations. The software dynamically creates a set of transect spacings to use for different sampling plans. Initially, only the smallest and largest spacings in the user-specified interval are used; additional spacings are added as needed to increase the smoothness of the estimated detection probability function. For each transect spacing considered, the sample proportion of realizations where the target area is detected is the estimate of the detection probability. VSP uses an undocumented method to fit a smooth curve to the estimated probabilities and plots the curve onscreen.

3.2.2.2 Stopping Criteria for the Detection Probability Simulation

Visual Sample Plan uses parameters called the *maximum error* and the *minimum precision* to decide when to end the simulation. These have default values set, but the user can override the defaults. Their default values are 0.03 and 0.01, respectively; decreasing the values causes the simulation to run longer and results in a smoother detection probability curve.

VSP uses the maximum error to decide when to stop simulating each transect spacing. It treats the set of realizations analyzed with one transect spacing as a random sample from the population of all realizations that could be analyzed using that transect spacing. For each realization, the outcome of "TA detected" or "TA not detected" is recorded. If VSP applies transect spacing s to n_s realizations, the sample proportion of realizations where the TA is detected using transect spacing s is $\widehat{p}_s$. VSP computes

$$\text{MOE} = 1.96\sqrt{\frac{\widehat{p}_s(1-\widehat{p}_s)}{n_s}},$$

the margin of error of a 95% confidence interval for the true proportion of realizations where the TA is detected with a spacing of s. It increases n_s until the MOE is less than the maximum error.

The software uses the minimum precision to assess whether it should simulate additional transect spacings. Once each transect spacing satisfies the maximum error criterion, VSP compares the

estimated detection probabilities for each pair of consecutive transect spacings. If the absolute difference in the estimated probabilities for a pair of transect spacings is larger than the minimum precision, it adds an intermediate transect spacing to the simulation. The simulation continues until all transect spacings satisfy the maximum error criterion and all pairs of adjacent of transect spacings have estimates that differ by less than the minimum precision.

3.2.2.3 Window Length for the Detection Probability Simulation

The window length could have a substantial impact on the simulated detection probabilities, but this effect would be invisible to the user without comparing several window lengths. I investigated the sensitivity of the detection probability curve to the window length and conclude that the default length is a reasonable value to use for a 1,200 foot by 800 foot target area (Figure 6). I had VSP generate ten curves for each of seven window lengths: 10%, 20%, 50%, 80%, 90%, and 125% of the minor axis, as well as 125% of the major axis. I kept the default values for minimum precision and maximum error, and set the false negative rate to 0%. I set all other inputs to the same values as in the Kriging window size experiment (Section 5). Some curves for the two largest windows look very rough; this is an artifact of fitting smooth curves to the point estimates. It could be remedied by increasing the minimum precision, but the other curves appear reasonably smooth.

For transect spacings under 200 feet, the detection probabilities are very close to one for all window sizes. For any window size larger than 200 feet, there is a trend where the probability of detecting the TA decreases as the window size increases. The window lengths of 50% to 90% of the minor axis of the TA yield very similar curves, suggesting that choosing a window length in this interval has little effect on the outcome of VSP's simulation. Therefore, a window length of 90% of the minor axis should result in a detection probability curve that is relatively uninfluenced by the window length choice. For the rest of this project, I use this default window length.

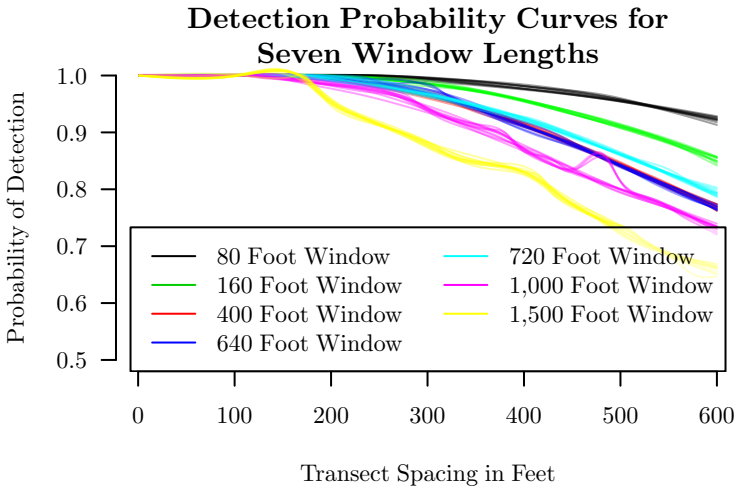


Figure 6: These detection probability curves for a 1,200 ft by 800 ft elliptical were created using Monte Carlo simulation within Visual Sample Plan. The target area is assumed to have a Gaussian distribution and 200 anomalies per acre at the center, and the background density is assumed to be 100 anomalies per acre. For each window length, ten curves illustrate the Monte Carlo error.

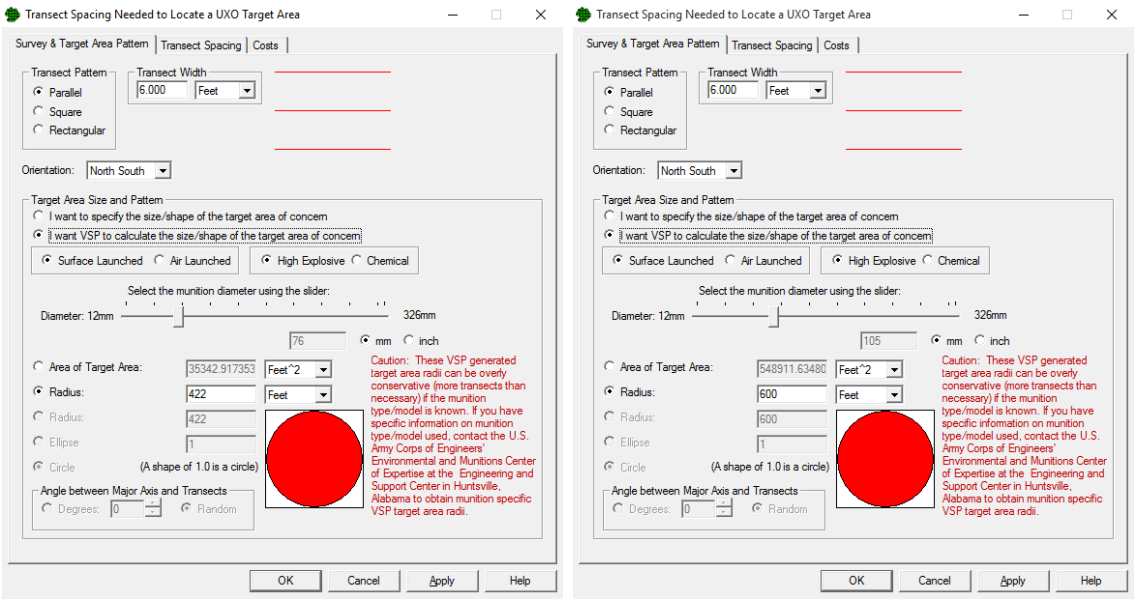


Figure 7: VSP calculates target area sizes for 76 mm and 105 mm surface-launched high-explosive shells.

3.2.3 Target Area Size Calculation

To construct a remedial investigation sampling plan, the user must have some prior knowledge of the target area size. Visual Sample Plan provides size estimates based on the fragmentation distance of a specified munition type fired at a single target. The interface is simple, with a few types of weapons to choose from and a slider to select the size of the munitions. Figure 7 shows the output used to inform the TA sizes for the simulated sites described in Section 4.1.

3.2.4 File Formats for Sample Data

Visual Sample Plan uses two types of files for UXO data: course over ground (COG) files and anomaly files. COG files contain records of the actual path traversed during sampling, represented as a sequence of waypoints. COG files have columns for the longitudinal and latitudinal coordinates, and optionally a column of timestamps. The width of the transect is not contained in the file, but is manually entered into VSP. Anomaly files contain the coordinates of all anomalies detected, and may include a third column for the value of the detector response at the location. This response value is currently ignored by VSP. Both types of files are plain text files with one entry per line and columns separated by commas or tabs.

Additionally, VSP uses the GeoEAS file format to export observed anomaly densities for use by GSLIB. A GeoEAS file is a plain text file with a simple header specifying the number and names of the variables. The data are output with one observation per line and columns are separated by tabs. VSP creates four columns for the longitudinal and latitudinal coordinates of the window center, the elevation, and the spatial anomaly density. VSP works only in two dimensions, so all elevation values are set to zero.

3.2.5 Flag High-Density Locations

The more basic of Visual Sample Plan's two methods of identifying possible target areas is to mark high-density locations along the transects. It does this using circular search windows centered on the transects to compute a moving average of the anomaly density across the sampled portion of the site. The user specifies a diameter for the windows and a critical density value; regions of the site where the observed density exceeds the critical value are flagged as possible target areas. The user chooses either to use the critical value as a hard threshold, or to treat it as the known background level and have VSP find areas where the density is "significantly greater than background density." Locations with high density are turned into polygonal regions.

Windows are centered along the centerline of each transect. VSP places the first windows at the south or west edge of the site. It places subsequent windows farther along the transects with the window centers separated by one-sixth of the window diameter (Figure 8). The density in a window is computed as

$$\text{window density} = \frac{\text{number of anomalies inside the window}}{\text{sampled area inside the window}}.$$

VSP uses large, overlapping windows to smooth out small-scale variation in the observed densities.

If the user chooses the "significantly greater than background density" option, VSP performs a hypothesis test for each window. The number of detected anomalies in the window is assumed to follow a Poisson distribution; the null hypothesis is that the Poisson mean is equal to the user-specified background density multiplied by the sampled area in the window. The significance level defaults to 0.05 but can be changed by the user. VSP marks windows where the null hypothesis is rejected (or where the density is above the hard threshold) by placing "flags" on the map at the window centers (salmon-colored squares in Figure 9).

Once windows are flagged, VSP delineates the high density regions by centering square blocks at the centers of the flagged windows. The user specifies the block size, which should be large enough to overlap blocks from adjacent transects so that a cluster of adjacent blocks appears as a contiguous region. VSP encloses overlapping blocks in polygons and ignores any polygons smaller than a user-specified minimum area.

3.2.5.1 Comments on Statistical Assumptions

Visual Sample Plan's "significantly greater than background density" option is statistically problematic. VSP performs the hypothesis tests as if the windows are independent, but this is not the case because the windows overlap. This dependence can be a source of excess variability in the observed anomaly counts, known as *overdispersion*, which the Poisson distribution does not account for. This leads to the null hypothesis being rejected too easily.

Furthermore, the default significance level is too high for this situation. The significance level is the *Type I error rate*, or the proportion of tests where the null hypothesis is rejected when it is actually true. With a significance level of 0.05, this means that on average 5% of windows containing only background anomalies will be flagged as possible target areas. One analysis can involve hundreds of windows (Figure 8, left, shows 807 windows), so a large number of false positives must be expected. The significance level should be lowered to adjust for the large number of tests.

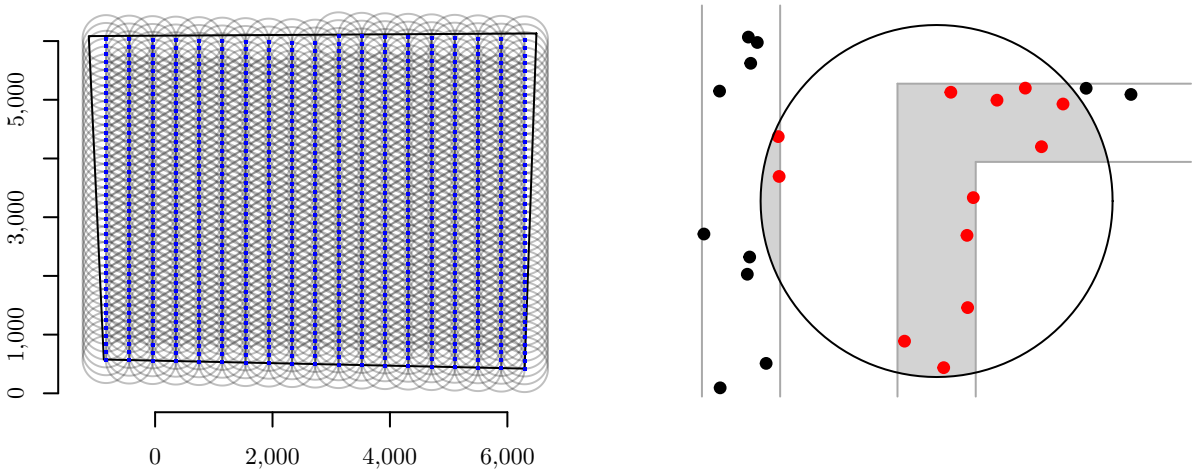


Figure 8: These diagrams illustrate the circular search windows used for calculating the local anomaly density. Left: All of these windows are used to analyze data from one sampling plan with 6 foot wide parallel transects and 390 feet of between-transect spacing. The window diameter is 800 feet. The blue points are the actual window centers generated by VSP. Right: This example shows how the window density is computed. All detected anomalies within the circle (red dots) are counted; the detected anomalies outside the circle (black dots) are ignored. The area used in the computation is the area of the intersection between the transects and the circle (shaded in grey).

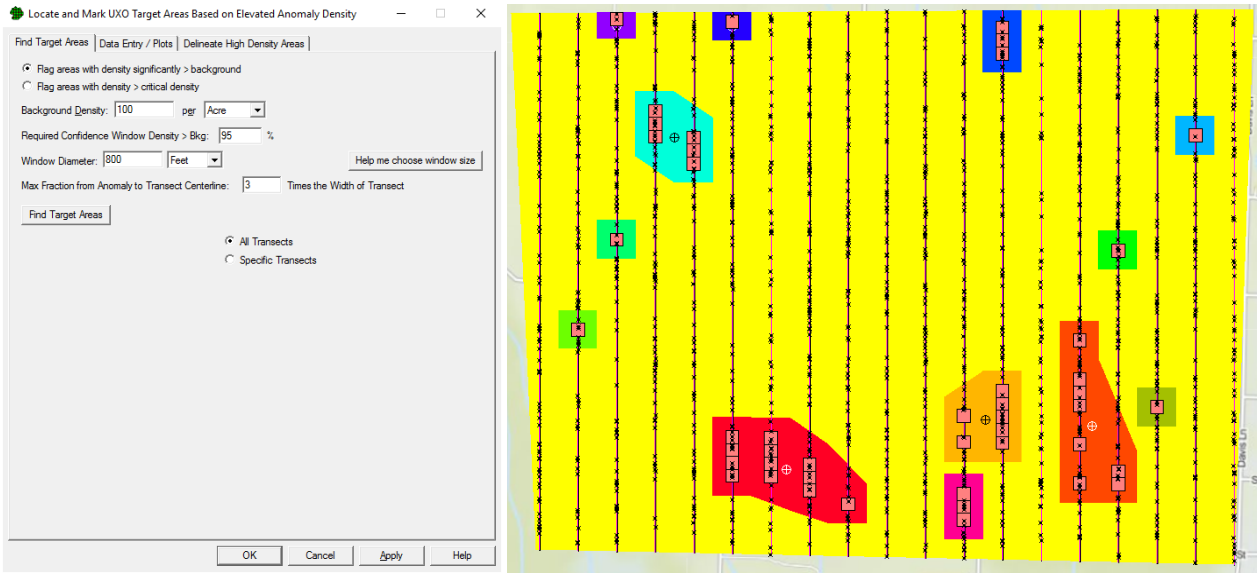


Figure 9: Left: This VSP dialog box is where the user specifies the window size and density threshold. Right: This example shows possible target areas based on 800 foot windows. The small, salmon-colored flags represent windows where the anomaly density is above 100 anomalies per acre at a 0.05 significance level. The program created the larger, colored regions are created by placing a square block at the center of each window. The transect spacing is 390 feet, so a block size of 400 ensures that blocks placed on adjacent transects can overlap. VSP creates the final result by drawing polygons around intersecting blocks.

3.2.5.2 Window Size for Flagging High-Density Locations

The only guidance the user's guide offers for choosing a window size is that the diameter should be at least as large as the between-transect spacing, but smaller than the assumed diameter of the target area of interest. The lower bound ensures that the density estimate undergoes some minimum amount of smoothing, although it is not apparent why the transect spacing in particular is a suitable bound. The upper bound is reasonable because windows that are larger than the target area will always include lower-density sections outside of the target area, causing the density estimate to be too low, and possibly resulting in the failure to detect a target area that truly exists.

These guidelines are only minimally helpful when there is a large difference between the transect spacing and the assumed target area size. VSP includes a sensitivity analysis tool to help make the window size decision (Figure 10). This feature compares density estimates among several window diameters. VSP tabulates the total high-density area delineated by each analysis and presents color-coded maps showing which parts of the site are marked as possible target areas. This information can be used to choose a window size and critical value that give an adequate amount of smoothing and result in identifying a portion of the site that can be remediated or investigated further while staying within the project's budget.

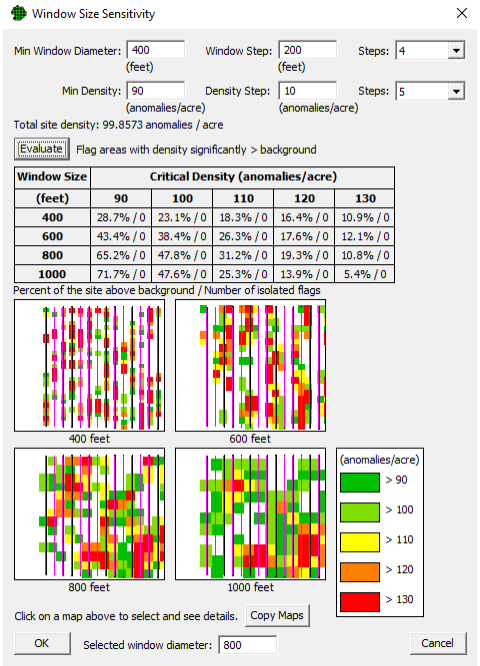


Figure 10: VSP's window size sensitivity analysis tool summarizes the detected high-density regions graphically and in tabular form. The two smallest window diameters (400 and 600 feet) give the appearance of the site being covered in small, isolated high-density regions. In reality these are due to naturally-occurring random variation. The larger window sizes (800 and 1,000 feet) tell more reasonable stories where similar (same-colored) densities are observed in adjacent windows. The user will choose the analysis where the area of the site labeled as high-density is the most consistent with the prior information about the extent of the munitions use areas.

3.2.6 Kriging an Anomaly Density Surface

The more sophisticated of Visual Sample Plan's two methods of identifying target areas is based on geostatistical mapping of the anomaly density over the site. This method approximates a continuous anomaly density surface by Kriging to produce a grid of predicted densities using a parametric covariance model to incorporate spatial dependency (see Section 3.3 for details about covariance functions and Kriging). Then, it finds contiguous regions of the site with high predicted density and delineates them as possible target areas. Kriging is a complicated process where many choices must be made, but the software automates it. The user can click one button to produce a density map using default settings. This Kriging methodology is the primary focus of my project.

VSP's Kriging tools analyze the same moving average anomaly densities as the flagging feature (see Section 3.2.5 for details). The most important difference is that only windows centered on transects can be flagged, but Kriging predicts the anomaly density in grid cells placed both on and between transects.

VSP can either predict numerical densities or predict the probability that the density in a cell exceeds a threshold. From the user's perspective, the software interface is the same whether densities or probabilities are predicted. I use anomaly density prediction to illustrate the Kriging features.

To get started, the user must load COG and anomaly files, and specify the window diameter. Once these tasks are done, the Kriging process occurs in three steps. The user can click a "Launch" button to have VSP automatically do the first two steps in "Basic Mode," which uses default settings. In the first step, VSP estimates a covariance model from the observed data and plots a semivariogram (see Section 3.3.1 for additional details). In the second step, ordinary Kriging is done to construct the anomaly density surface (details in Section 3.3.3). The outcome of these steps is a map of the site, colored by predicted density (Figure 11).

In the third step, VSP thresholds the density surface and places polygons on the map to delineate high-density regions of the site. The software offers the user two options for doing this. Either (1) a fixed threshold for the density is used, or (2) VSP finds regions using an "upper confidence bound" of the predicted density at each grid cell. The latter feature is new in VSP version 7.5 and is not yet documented; I will not discuss it further. Typically, the user will examine the anomaly density map and subjectively decide on a threshold value, as demonstrated in Pulpisher et al. (2014). The software identifies the grid cells where the estimate exceeds the threshold, and draws polygons around contiguous clusters of cells to mark possible target areas. The user must specify a minimum area single delineated region, and any polygons below this area threshold are discarded and their grid cells are not included in the identified high-density area.

3.2.6.1 Window Size for Kriging

The window diameter acts as a smoothing parameter for the density surface (Figure 12). A small window results in a surface with many small hotspots that make it difficult to discriminate between a large target area and background noise. On the other hand, a large window could smooth the surface too much, making target areas appear larger and lower in density than they actually are. It is important to choose a window size that can accurately describe the target areas that are present. VSP does not include any tools to help the user choose a window size for use in Kriging.

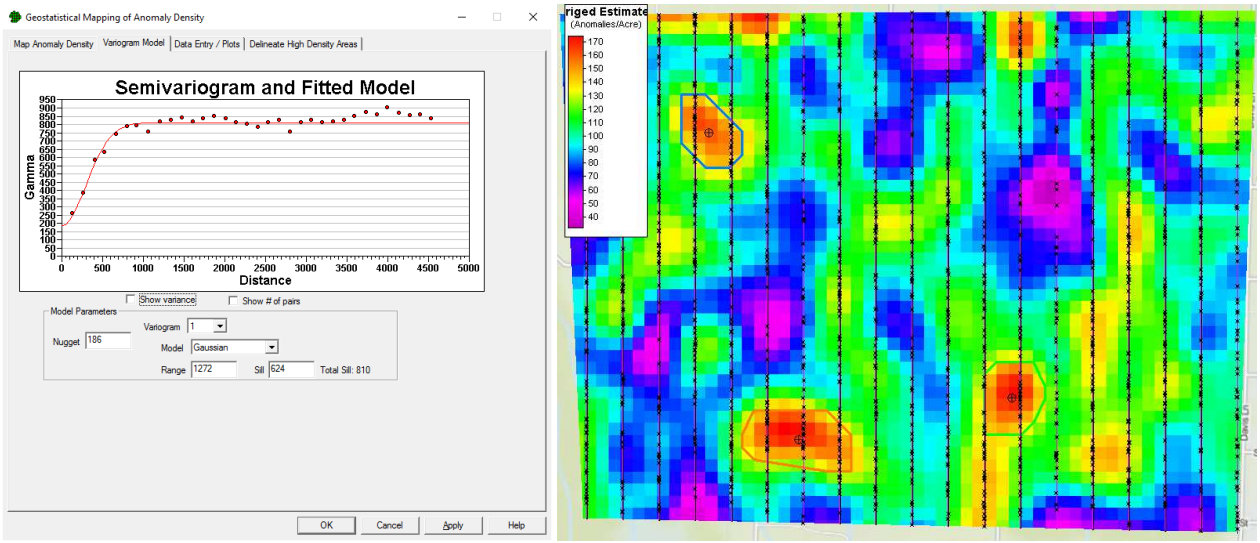
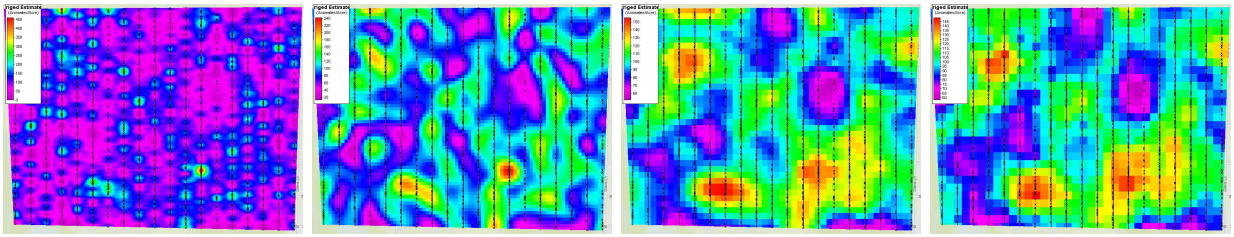


Figure 11: These example Kriging results use 800 foot windows. The transects are 6 feet wide and spaced 390 feet apart. The site contains a true 1,200 foot by 800 foot target area in the upper left and a true 2,000 foot by 900 foot target area in the lower right. Left: VSP's semivariogram selection window shows an empirical semivariogram as points and a parametric semivariogram as a curve. Right: This anomaly density map results from Kriging with the covariance model associated with the semivariogram curve on the left. The orange, green, and blue polygons enclose sets of contiguous grid cells with estimated densities above 140 anomalies per acre and areas above 5 acres. I chose these values because they are able to separate the highest-density (red) elliptical regions from the relatively disordered-looking background (purple, blue and green). The blue polygon correctly delineates the first TA. The green polygon is at the center of the second TA, but much of the TA area is outside the polygon. The orange polygon only contains background noise.



(a) 200 foot windows (b) 500 foot windows (c) 900 foot windows (d) 1,200 foot windows

Figure 12: VSP's automated Kriging procedure makes it easy to re-analyze data with different window sizes. These predicted density surfaces result from using four different window diameters and the same sample data as used in Figure 11. Larger search windows result in smoother surfaces.

3.2.7 Plots of Anomaly Density by Region

Once potential target areas have been delineated within VSP, the user can view histograms and boxplots of the window densities by region. The plots are color-coded by region and are helpful for describing the distributions of the anomaly density, checking that the results fit the prior information about background and target area density, and assessing if the observed anomaly densities can be approximated by a Normal distribution. Figure 13 shows plots corresponding to the regions seen in Figure 11.

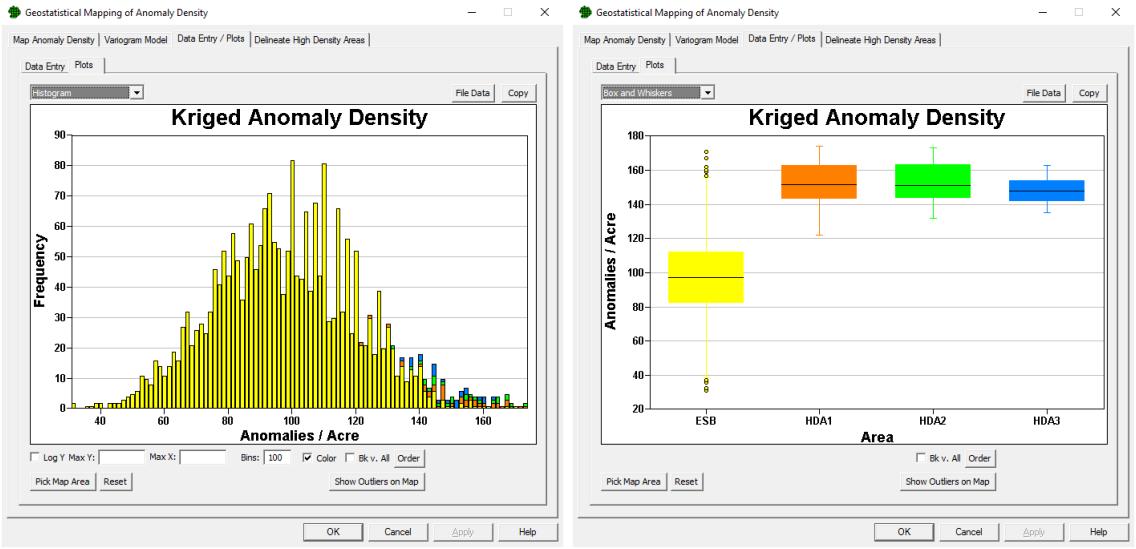


Figure 13: Visual Sample Plan plots the anomaly density in delineated regions as a histogram or boxplots. Yellow corresponds to the portion of the site not marked as a possible target area, and shows the predicted distribution of the background noise given the modeling and threshold choices made by the user.

3.2.8 TOI Rate Estimation and Confirmation that a Site is Clean

The features described in Sections 3.2.1–3.2.7 are used for remedial investigation of regions where prior information suggests concentrated munitions use occurred and many target of interest items are expected to be present. For sections of the site where that is not the case – regions that were not used as target areas so TOI would arrive only by accident or after ricocheting a long distance, or regions that were previously remediated – no high anomaly density areas are expected to be present. Analysis methods for this situation typically use a simple random sample or systematic sample of short transects. The goal for regions expected to have few TOI items is to show that the spatial density of TOIs is satisfactorily low by the criteria defined for the project.

Visual Sample Plan's "Target of Interest (TOI) rate estimation" tool helps the user decide how many transects to sample to meet this goal. The user inputs the size of the transects, the desired confidence level, and the maximum number of TOI items that would be acceptable in the sample area. VSP uses a Binomial model for the number of TOI items found during sampling. The proportion of the total area covered by the transects is used as the probability p of detecting each

item. The total number of TOI items N is unknown, so VSP elicits prior information from the user and uses a Bayesian analysis to find the posterior distribution of N given the number of TOIs detected during sampling. The "confidence level" is defined as the posterior probability that N does not exceed the user-specified maximum. The models used by VSP lead to closed-form posterior distributions, so VSP assumes that zero TOI items will be observed and solves for p. It then recommends the number of transects of the user-specified size that cover p of the site area.

After sampling, the user can input the number of TOIs that were found. The software uses this information to find the posterior quantile corresponding to the confidence level, and then suggests a "confidence" statement that can be made about the density TOI items present, treating the posterior quantile as an upper bound.

The user can choose to use an informative or uninformative prior distribution for N, or to use a maximum likelihood analysis instead of a Bayesian analysis. If the user selects an informative prior, N follows a Poisson distribution with mean λ, and λ follows a Gamma distribution. The user inputs some additional information about λ, and VSP selects one of several pre-programmed shapes for the Gamma distribution and plots the resulting marginal prior distribution of N. If the user desires an uninformative prior, VSP assumes that all nonnegative integers are equally likely values of N, and no additional options are available. The maximum likelihood option uses the same model as the uninformative prior option, but includes additional output regarding the probability of concluding the TOI rate is acceptable when in fact it is higher (analogous to a Type I error rate if "the TOI rate is acceptable" is considered as a null hypothesis).

Figure 14 shows an example of the TOI rate estimation feature. The "Presumptively Clean Validation" feature offers similar functionality, but models the number of small grid cells containing TOIs rather than modeling the number of TOI items. Analyzing grid cells lets the user state conclusions in terms of how much area is free of TOIs instead of the number or density of TOI items.

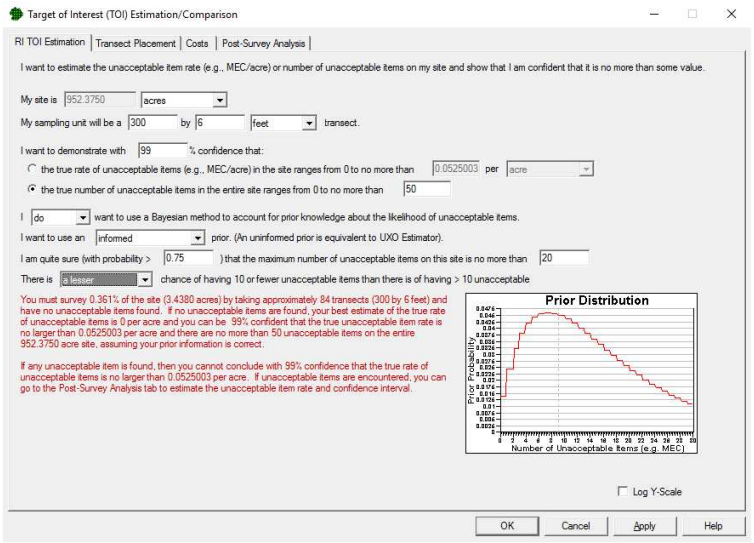


Figure 14: The user provides information to help VSP construct a prior distribution for the number of TOI items in the area. VSP then recommends a number of transects to be sampled.

3.2.9 Verification Sampling Plans

The final UXO-related feature is used after remediation to assess the success of the cleanup. The project team needs to demonstrate that very few targets of interest remain. VSP uses *accept-on-zero attribute compliance sampling* methodology to guide the user in constructing a sampling plan. The user can choose to use either transects or the locations of remediated anomalies as the sampling units. If the verification sample is collected and no TOI items are found, the team will make a statement of the form "We are $Y\%$ confident that at least $X\%$ of the N possible sampling units do not contain TOIs" where N is the total number of transects or anomaly locations in the remediated region that could have been sampled. At this point, the true number of detectable anomalies is known because remediation involves a census of the whole region.

The data from verification sampling are used for a hypothesis test. Under the null hypothesis, the number of units that contain TOI items in a simple random sample of n units follows a hypergeometric distribution where $N\left(1-\frac{X}{100}\right)$ units contain TOI. $\frac{Y}{100}$ is taken as the probability that no units in the sample contain TOI, so $1-\frac{Y}{100}$ is the Type I error rate. X, Y, N and n have the approximate relationship

$$1-\frac{Y}{100} \approx \left(1-\frac{2n}{2N-N\left(1-\frac{X}{100}\right)+1}\right)^{N\left(1-\frac{X}{100}\right)}$$

presented in Bowen and Bennett (1988, §17.2) in the context of using the hypergeometric distribution with data from accept-on-zero sampling plans where the sampling units are containers of radioactive material. The total number of sampling units N is known. The user specifies two of X, Y, and n, and VSP solves for the third. Figure 15 shows an example where the sampling units are 300 foot by 6 foot transects. The user wants to be 99% confident that at least 99% of the area of the site is free of targets of interest and VSP computes that 454 transects should be sampled.

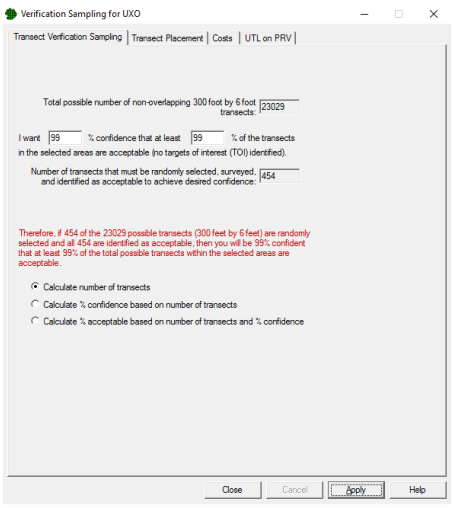


Figure 15: The user specifies a confidence level and the proportion of transects that must be free of targets of interest, and then VSP outputs a sample size.

3.3 Theory of Spatial Prediction

Visual Sample Plan's Kriging feature computes a moving average anomaly density at points along the transects, and uses this moving average density as the response variable in a spatial linear model. It constructs a density surface over the entire site by using the model to predict the moving average density at unsampled locations. To help clarify the issues related to the Kriging analysis in VSP, this section presents an overview of spatial linear models. Sections 3.3.1–3.3.4 discuss the technical details in the context of UXO data and describe the default behavior of VSP's Kriging procedure.

Linear models extend to the realm of spatial data in the following manner (Schabenberger and Gotway 2005, §5.1). Suppose the outcome of a process is observed at spatial locations $\mathbf{s}_1, \ldots, \mathbf{s}_n$. Let $\mathbf{Z} = (Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n))'$ be a vector of responses and let $\mathbf{X}$ be a matrix of explanatory variables where the ith row contains the values at location $\mathbf{s}_i$. in general, $\mathbf{X}$ may contain linear or polynomial functions of the spatial coordinates, as well as other covariates collected along with the response. Visual Sample Plan does not include any covariates or functions of the spatial location in $\mathbf{X}$, but other software can fit more complicated models.

The spatial linear model is

$$\mathbf{Z} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\boldsymbol{\beta}$ is a vector of coefficients and $\boldsymbol{\epsilon}$ is a multivariate Normal random error vector with mean $\mathbf{0}$ and variance-covariance matrix $\boldsymbol{\Sigma}$. The covariance between two observations depends on the distance (and possibly direction) between their locations. These relationships are described by a covariance function and a semivariogram, explained further in Section 3.3.1.

A single real-world site is one realization of some underlying process. In spatial statistics, the interest is typically in mapping the response across the one realization by predicting the response at unobserved locations while using the correlation structure to account for the relationships between nearby points. A family of predictors known as Kriging predictors are the *best linear unbiased predictors* (BLUPs) under squared-error loss in several situations. Important Kriging predictors include the simple Kriging predictor (BLUP when $\mathbf{X}\boldsymbol{\beta}$ is known), the ordinary Kriging predictor (BLUP when $\mathbf{X}\boldsymbol{\beta}$ is unknown but constant), and the universal Kriging predictor (BLUP in the general case where $\mathbf{X}\boldsymbol{\beta}$ is not spatially and $\boldsymbol{\beta}$ is unknown).

The spatial linear model requires a continuous response variable observed at fixed locations. The raw data from a UXO site are the random locations of the anomalies, recorded as a list of points. Visual Sample Plan processes the list of locations into a continuous density variable. It does this by computing the observed spatial anomaly density $Z(\mathbf{s}_i)$ as a moving average in a circular window at locations $\mathbf{s}_i$ on the transects (see Section 3.2.5). VSP does not use any explanatory variables, so the linear model simply describes variation around the overall mean. The model becomes

$$\mathbf{Z} = \mu\mathbf{1} + \boldsymbol{\epsilon}; \quad \boldsymbol{\epsilon} \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma}),$$

where $\mathbf{Z} = (Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n))'$, $\mathbf{1}$ is a vector of ones, and μ is the mean anomaly density across the whole site. An anomaly density map is produced by using the model to predict values of $Z(\mathbf{s})$ at many new values of $\mathbf{s}$.

It should be pointed out that the mean density μ is not the same as the background density because the higher-density target areas are included when computing the mean. If μ is considered unknown,

this superficially looks like an ordinary Kriging problem. However, a fundamental assumption of ordinary Kriging is *second-order stationarity*, meaning that the mean is spatially constant and the covariance between two points depends only on the distance between them, not their absolute locations. The presence of TAs is a violation of stationarity. Nonetheless, Visual Sample Plan uses ordinary Kriging and so I follow suit for my simulation. I revisit these assumptions in Section 7.

3.3.1 Covariance Functions and Semivariograms

In one realization of a spatial process, observations located near each other tend to have similar values. Thus, an observation at one location depends upon the values observed at other locations. Nearby locations are more strongly related than locations that are far apart. For a second-order stationary process, the *covariance function* $C(\mathbf{h}) = \text{Cov}\left(Z(\mathbf{s}), Z(\mathbf{s}+\mathbf{h})\right)$ quantifies the strength of the spatial relationship between two points separated by a displacement vector $\mathbf{h}$. Under the assumption of second-order stationarity, $\mathbf{s}$ is any arbitrary spatial location. The presentation in this paper uses some simplifying assumptions; see Schabenberger and Gotway (2005, Chapter 4) for a more thorough treatment.

The covariance function is a fundamental probabilistic concept, but in geostatistical applications it is more common to describe the spatial dependency by the *semivariogram*, which measures the variability of the difference in response values between a point and its neighbors. Under second-order stationarity, the semivariogram is defined as

$$\gamma(\mathbf{h}) = \frac{1}{2}\text{Var}(Z(\mathbf{s}) - Z(\mathbf{s}+\mathbf{h})).$$

The statistical models used by VSP are both second-order stationary and *isotropic*, meaning that covariance between observations at two points does not depend on the direction between the points. In an isotropic context, the displacement vector can be replaced by the distance between the points, called the *lag*. This assumption makes the semivariogram simple to write out in terms of the covariance function. For any pair of points that are a lag $h = |\mathbf{h}|$ apart, the covariance is denoted $C(h)$. The semivariogram simplifies to

$$\gamma(h) = C(0) - C(h).$$

Semivariograms exhibit a characteristic shape where they start at or near 0, increase, and level off at or near a value of $C(0) = \sigma^2$. The value σ^2 is the variance of the response variable, known in geostatistics as the *sill*.

Other important quantities for summarizing the structure of the spatial dependency are the range and the nugget. The *range* α is the lag beyond which points are uncorrelated, so that $C(h) = 0$ and $\gamma(h) = \sigma^2$ for $h > \alpha$. In situations where the semivariogram approaches the sill asymptotically, α instead represents the *practical range* where the semivariogram becomes approximately equal to the sill. A common definition of the practical range is the lag where the semivariogram reaches 95% of the sill ($\gamma(\alpha) \approx 0.95\sigma^2$).

The *nugget* c_0 is a constant that is added to the semivariogram to represent small-scale variation or measurement error. The canonical example of a nugget arises when mapping mineral concentration across a large region. The statistical model captures general trends in concentration, but nearby observations are not perfectly correlated. One location can contain a nugget of highly concentrated ore, but another location just a few inches away may have a much lower observed concentration.

3.3.1.1 The Empirical Semivariogram

A considerable amount of theory has been developed for both estimating and modeling covariance functions and semivariograms. The techniques used by Visual Sample Plan originate in geostatistics and emphasize the semivariogram over the covariance function, so the rest of this section centers around semivariogram estimation.

The classical semivariogram estimator is known as the *Matheron estimator* or *empirical semivariogram*. It has the form

$$\widehat{\gamma}(h) = \frac{1}{2|L(h)|} \sum_{(i,j)\in L(h)} (Z(\mathbf{s}_i) - Z(\mathbf{s}_j))^2$$

with the *lag class* $L(h)$ defined as the set of distinct pairs of indices such that locations $\mathbf{s}_i$ and $\mathbf{s}_j$ are separated by a lag of h. Usually, lags are binned because there may be only a small number of locations that are a given distance apart. In that case, $L(h)$ is defined to include all pairs with lags in the interval $[h - \epsilon, h + \epsilon)$, and the semivariogram is estimated only for a finite set of lags. Estimated semivariograms can be heavily influenced by the binning, especially when some bins include few pairs. Schabenberger and Gotway suggest choosing ϵ so that each bin includes at least 30 pairs.

3.3.1.2 Parametric Models for the Semivariogram

Spatial statistical models require covariances to be defined for any lag $h \geq 0$, not just for the distances between the observed locations, so a functional form must be specified for the covariance. Many common models are parameterized to include the range, sill, and nugget, and these parameters are estimated from the data. Any covariance function model has an associated semivariogram. Visual Sample Plan allows the user to inspect semivariograms; the covariance functions are

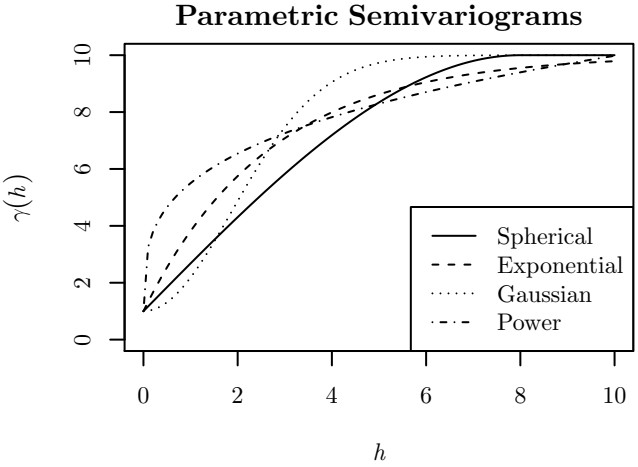


Figure 16: These semivariograms result from different parametric models for the covariance function. Each has a nugget of $c_0 = 1$. The spherical, exponential, and Gaussian models have $\sigma^2 = 9$ and $\alpha = 8$. The power model has $\theta = 4.5$ and $\omega = 0.3$.

hidden behind the scenes. The covariance models supported by Visual Sample plan are the spherical, exponential, and Gaussian models, As implemented in GSLIB (Deutsch and Journel 1998), and including a nugget, they have the following semivariograms. Figure 16 illustrates example semivariograms for the four different models.

The spherical semivariogram is

$$\gamma(h) = \begin{cases} c_0 + \sigma^2 \left(1.5\frac{h}{\alpha} - 0.5 \left(\frac{h}{\alpha}\right)^3 \right), & h < \alpha \\ c_0 + \sigma^2, & h \geq \alpha \end{cases}.$$

In this model, α is the true range.

The exponential semivariogram is

$$\gamma(h) = c_0 + \sigma^2 \left(1 - \exp\left\{ -\frac{3h}{\alpha} \right\} \right).$$

Here, α is the practical range, where $\gamma(\alpha) \approx 0.95\sigma^2$.

The Gaussian semivariogram is

$$\gamma(h) = c_0 + \sigma^2 \left(1 - \exp\left\{ -\left(\frac{3h}{\alpha}\right)^2 \right\} \right).$$

In this case, α is the practical range, with $\gamma(\alpha) \approx 0.99\sigma^2$.

Finally, the power semivariogram has the form

$$\gamma(h) = c_0 + \theta h^{\omega}$$

where $0 < \omega < 2$. The parameters θ and ω are not easily interpretable, but the power model is flexible in that its semivariogram takes on a variety of different shapes for different ω values. It can be useful for describing spatial processes where the range is larger than any of the observed lags.

3.3.1.3 Theory of Fitting Parametric Semivariogram Models

In the past, geostatisticians traditionally fit parametric semivariogram models by trial and error, plotting curves with different parameter values over the empirical semivariogram values until a curve was judged to fit the points adequately. Nonlinear least squares methods for fitting semivariograms gained popularity as computing power increased in the 1980s and 1990s. Now, more sophisticated Bayesian and maximum likelihood methods are available for estimating covariance parameters. If a probability model is fully defined for $(Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n))'$, maximum likelihood (ML) and restricted maximum likelihood (REML) estimators for covariance parameters can be derived. Estimates can be computed numerically using all of the observed data.

However, much of the developed theory regarding the properties of the least squares and maximum likelihood estimators assumes the spatial process is a Gaussian random field, so the observed data vector $(Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n))'$ follows an n-variate Normal distribution. That is a hefty assumption that Schabenberger and Gotway insist is rarely met in reality. In many research fields, it is still

common practice to use least squares to fit a parametric semivariogram function to the empirical semivariogram without stating any particular distribution for the original observations. The covariance function is constructed from the parameter estimates and then treated as if it were the truth, so the resulting Kriging predictions fail to account for the uncertainty associated with the covariance parameter estimates.

The least squares approach with the most theoretical justification is generalized least squares (GLS), which accounts for the correlation between values of the empirical semivariogram at different lags. Schabenberger and Gotway discuss GLS estimation; the expression for the correlation is complicated and GLS is infrequently used in practice.

Weighted least squares (WLS) ignores the correlations but is much easier to implement than GLS. Cressie (1985) proposes the approximation

$$\mathrm{Var}(\widehat{\gamma}(h)) \approx \frac{2\gamma(h,\boldsymbol{\theta})^2}{|L(h)|}$$

which leads to the weighted sum of squares

$$\begin{aligned}\frac{1}{\mathrm{Var}(\widehat{\gamma}(h))}\sum_h \left(\widehat{\gamma}(h)-\gamma(h,\boldsymbol{\theta})\right)^2 &\approx \sum_h \frac{|L(h)|}{2\gamma(h,\boldsymbol{\theta})^2}\left(\widehat{\gamma}(h)-\gamma(h,\boldsymbol{\theta})\right)^2 \\ &= \sum_h \frac{|L(h)|}{2}\left(\frac{\widehat{\gamma}(h)}{\gamma(h,\boldsymbol{\theta})}-1\right)^2.\end{aligned}$$

The vector of parameters $\boldsymbol{\theta}$ is estimated by minimizing the above expression, which is easily done numerically.

A further simplification is to ignore the variance weight and use ordinary least squares (OLS), which is also extremely easy to implement. Schabenberger and Gotway insist that OLS and WLS are generally poor approximations of GLS because the empirical semivariogram values are strongly correlated. Simulation studies (Zimmerman and Zimmerman 1991) suggest, at least for Gaussian random fields, that ML, REML, WLS, and OLS perform similarly well at estimating covariance parameters for Kriging.

3.3.2 Semivariograms in VSP and GSLIB

Visual Sample Plan relies on GSLIB to compute empirical semivariograms. GSLIB is a collection of open source geostatistics computer programs written in FORTRAN by faculty and graduate students at Stanford University. Development of GSLIB began around 1980 (Deutsch and Journel 1998). It was originally written in FORTRAN 77, but has been updated to FORTRAN 90 and is now available in 32-bit and 64-bit versions for Windows, Mac OS X, and Linux (Deutsch and Schnetzler 2003).

GAMV is the GSLIB routine that estimates empirical semivariograms. VSP creates a configuration file which tells GAMV how to set up the binning of the lags. This file specifies a lag separation x_{lag}, a lag tolerance x_{tol} and a number of lags n_{lag}. GAMV reads the file and then outputs the empirical semivariogram values for lags $1x_{\mathrm{lag}}, 2x_{\mathrm{lag}}, 3x_{\mathrm{lag}}, \ldots, n_{\mathrm{lag}}x_{\mathrm{lag}}$. All pairs of observations separated by a distance in the interval $[kx_{\mathrm{lag}}-x_{\mathrm{tol}}, kx_{\mathrm{lag}}+x_{\mathrm{tol}})$ are used to estimate the semivariogram at the k^{th}

lag. Visual Sample Plan uses one-sixth of the window diameter as the lag separation, one-twelfth of the window diameter as the lag tolerance, and requests 36 lags. GAMV offers additional features for dealing with anisotropy and for working with points in three-dimensional space, but VSP does not make use of these.

For large windows, the 36^{th} lag may be larger than the maximum possible distance at the site. In this case, GAMV only estimates the semivariogram for lag classes that contain at least one pair of points. It does not return any values for lags larger than the site. Empirical semivariograms computed using VSP's default settings may include unreliable values for larger lags where few data are available.

Visual Sample Plan estimates covariance parameters itself based on the empirical semivariogram output from GAMV. VSP considers spherical, exponential, Gaussian, and power models; it automatically selects the "best fitting" of these, but the documentation does not explain how it estimates the parameters or how it compares the different models. I have not been able to exactly replicate VSP's estimates, but its estimator behaves similarly to Cressie's WLS estimator. In particular, parametric semivariograms chosen by VSP match the empirical semivariogram closely for small lags and are relatively uninfluenced by extreme values of the empirical semivariogram for large lags.

3.3.3 Ordinary Kriging

Kriging uses a linear function of the observed data to produce estimates and predicted values by interpolating between the observed values. The model assumed by Visual Sample Plan is

$$\mathbf{Z} = \mu\mathbf{1} + \boldsymbol{\epsilon}; \quad \boldsymbol{\epsilon} \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma})$$

where $(Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n))'$ is a vector of observed moving average anomaly densities. Ordinary Kriging assumes the mean μ is unknown but spatially constant and that $\mathrm{Cov}\left(Z(\mathbf{s}_i), Z(\mathbf{s}_j)\right) = C\left(|\mathbf{s}_i - \mathbf{s}_j|\right)$ depends only on the distance between $\mathbf{s}_i$ and $\mathbf{s}_j$. For a point $\mathbf{s}$, the predicted anomaly density is

$$Z(\mathbf{s}) = \widehat{\mu} + \boldsymbol{\sigma}'\boldsymbol{\Sigma}^{-1}\left(\mathbf{Z} - \mathbf{1}\widehat{\mu}\right)$$

where $\boldsymbol{\sigma} = (C(|\mathbf{s} - \mathbf{s}_1|), \ldots, C(|\mathbf{s} - \mathbf{s}_n|))$ and $\widehat{\mu} = \left(\mathbf{1}\boldsymbol{\Sigma}^{-1}\right)^{-1}\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{Z}$. The predictor is derived by using Lagrange multipliers to minimize the squared-error loss function (Schabenberger and Gotway 2005, §5.2). The variance of this predictor is

$$\sigma^2(\mathbf{s}) = C(0) - \boldsymbol{\sigma}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\sigma} + \frac{\left(1 - \mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\sigma}\right)^2}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}},$$

known as the Kriging variance.

Note that the predicted value of $Z(\mathbf{s})$ at an observed point $\mathbf{s}$ will always equal the observed value of $Z(\mathbf{s})$. That is, Kriging preserves the observed anomaly densities. The predicted anomaly densities at unobserved locations are found by interpolating between the moving average densities at observed locations. Kriging produces a continuous density surface, but does not do any smoothing in the sense of removing local variability from the observed anomaly densities. Smoothing comes from computing the observed densities in windows, and the amount of smoothing is controlled by the window size.

3.3.4 Kriging in VSP and GSLIB

Visual Sample Plan sets up a grid over which the anomaly density surface is constructed. The grid cells are squares with side lengths of one-sixth of the window diameter. VSP uses the ordinary Kriging functionality of GSLIB to predict the anomaly density at the center of each cell.

GSLIB's Kriging routine is called KT3D. VSP outputs a configuration file that specifies the form and parameters of the covariance function and instructs KT3D on how to set up the grid of locations where the anomaly density will be predicted. The grid is defined by setting values for the numbers of cells in the longitudinal and latitudinal directions (n_x and n_y), the sizes of the cells in each direction (x_{size} and y_{size}), and the coordinates of the center of southwesternmost cell in the grid ($x_{\min}$ and $y_{\min}$). VSP sets x_{size} and y_{size} to one-sixth of the window diameter. $x_{\min}$ and $y_{\min}$ are set so that the edges of the grid will pass through most southern and western points of the site. Then it chooses n_x and n_y so that the grid will cover the whole site. KT3D then predicts the anomaly density at the center of each grid cell, and VSP reads the KT3D output and plots the predicted values as a heatmap.

KT3D uses only the observations in a local neighborhood to compute predicted values; this is a relic from a time when limited computer memory and processor speed made such compromises necessary. The neighborhood can be defined to include the entire site, but VSP's default settings inexplicably limits the neighborhood to the nearest 50 points. My simulations involve moving average densities computed at hundreds of locations within each realization of a site, so KT3D makes each prediction in ignorance of much of the available information.

KT3D includes many other features that VSP does not use. Some interesting examples include anisotropic covariance models, Kriging in three dimensions, an automated leave-one-out cross validation function, and the ability to compute the prediction error at points in a user-specified validation set for jackknife resampling. The cross-validation and jackknife features could be useful in selecting a search window size (Section 7.1).

4 Simulation Methods

The main component of this project is a simulation study to investigate how decisions made in constructing a sampling plan influence the possible target areas identified after Kriging by Visual Sample Plan's methods. I define three hypothetical sites to provide a realistic context for comparing the performance of the VSP-style analysis across different situations.

The conceptual models of the sites specify the prior information available when creating the sampling plan, and also define the "truth" that the results of the analysis are compared to. Section 4.1 describes the sites. Realizations of the anomaly locations at the sites are generated from spatial Poisson processes. These processes generate events at discrete locations and thus have an entirely different character from the continuous response variable assumed by the models described in Section 3.3; the reader can consult Section 4.2 for a primer on the theory of spatial Poisson processes. I generate 3,000 realizations of each site so I have a large pool available to analyze.

I use sampling plans recommended by VSP based on the prior information in the conceptual site models; Section 4.3 gives the details of how I create the sampling plans and implement them on

the simulated sites. Then I perform spatial prediction and delineate possible target areas using methods meant to mimic those used by VSP. There are a few aspects of VSP's analysis that I either cannot reproduce or consider necessary to modify for simulation purposes, so Section 4.4 explains my prediction and delineation procedure and points out the differences from the VSP procedure. Section 4.4.4 explains how the results of the analysis are summarized. Results from my analysis are presented in Sections 5 and 6.

I use the R software, version 3.2.3 (R Core Team 2015), and the `spatstat` package (Baddeley, Rubak, and Turner 2015; Baddeley and Turner 2005) to automate the simulation, sampling, and analysis. For consistency with VSP, I use GSLIB (Deutsch and Journel 1998) to compute empirical semivariograms and Kriging predictions. The R code is provided in Appendix A. The simulation takes roughly 220 hours to run on a Lenovo Thinkpad SL410.

4.1 Descriptions of Simulated Sites

The sites vary in complexity and are designated as *easy*, *medium*, and *hard* according to the intended difficulty of accurately identifying the true target areas. All sites have the same shape, an approximately 7,400 ft by 5,600 ft quadrilateral (952.4 acres). I use the North American Datum 1983 Montana State Plane Cartesian coordinate system and give the sites a geographic location near Bozeman, Montana. For a site of this size, the Earth's curvature has a negligible effect on distance and area measurements, so I use Euclidean distances measured in U.S. Survey Feet. A real coordinate system is not necessary for the simulation, but adds realism when working in VSP.

All three sites have a background anomaly density of 100 anomalies per acre, and this value is considered known. At the *easy* site, the background anomalies come from a homogeneous process. The *medium* and *hard* sites have more complicated background processes that could make it more difficult to distinguish between the TAs and the background noise.

Each site has two known TAs of different sizes. The *medium* and *hard* sites have a third unknown target area of an intermediate size. The target areas are circular or elliptical, and all have an anomaly density at the center of 200 anomalies per acre above the background level. The density decreases away from the center of the TA. I define the "true" TA region as the ellipse that contains 99% of the target area anomalies lying along any cross-section going through the center of the TA.

4.1.1 Conceptual Model of the Easy Site

The *easy* site is meant to represent the best possible scenario, where the terrain is easy to traverse and the background process is simple. The terrain is open grassland and easily accessible for a vehicle-towed DGM array with a six-foot-wide footprint. The site now part of a wildlife refuge. Recreational activities are prohibited so the area sees little human activity, and thus the background anomalies are mostly due to iron in the soil and rocks. Some preliminary surveys were performed nearby to calibrate the DGM equipment. From these data, it appears reasonable to assume that the background anomalies are homogeneously distributed with a density of 100 anomalies per acre.

In the 1940s, the site was used by the U.S. Army to train tank crews. 76 mm shells were fired from a single firing point in the center into two target areas. A single target in the northeastern corner resulted in a 1,200 ft diameter by 800 ft diameter elliptical target area (TA1). To the south, several

targets in a straight line resulted in a 2,000 ft diameter by 900 ft diameter target area (TA2). At their centers, both TAs have anomaly densities of 200 anomalies per acre above the background.

Retired Army personnel have provided reliable eyewitness accounts of the training activities, so the sizes of the target areas are known but the precise locations are not. Based on the level of site use described in these accounts, the anomaly density in the target areas is expected to be anywhere from 100 to 400 anomalies per acre above the background level.

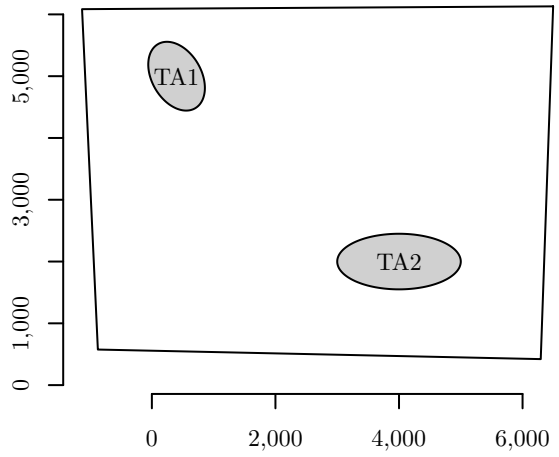


Figure 17: The easy site contains two target areas and homogeneous background anomalies. The target areas are designated TA1 and TA2. The axes on the plot are marked in feet.

4.1.2 Conceptual Model of the Medium Site

The *medium* site is meant to represent a realistic scenario. The terrain is accessible and the background has some debris from human activity. Not all details of the target areas are known. It is in a wilderness area, now part of a National Forest, and is bordered by a mountain range to the north. The environment at the site consists of foothills and thin forest. Any transect would be accessible to a DGM array with a six-foot-wide footprint towed by an ATV.

The site sees moderate human activity. There is a city of 50,000 people to the south, and a lake to the west. The lake is a popular weekend recreation destination. A two-lane highway runs through the site, connecting the city to the lake. A dirt fire road runs from the center of the site to the west, and is used to access trails for hiking, mountain-biking, and cross-country skiing. Background anomalies are from iron in soil and rocks, metallic debris along roads, and occasional metallic debris elsewhere. Equipment tests along the roads have shown that anomalies in the 50 ft wide path along each road have a homogeneous distribution with a density of 200 anomalies per acre. Another UXO cleanup occurred several miles away at a site with similar geology, where the background was found to be homogeneous with 100 anomalies per acre. Several randomly-selected locations at this site were used for equipment calibration; data collected during these activities suggest that both sites have the same distribution of background anomalies.

This site was used during the Second World War for tank and artillery training activities. 76 mm tank shells were fired from two firing points near the west end of the dirt road. Shots from

the northern firing point were fired at targets to the north, against the mountains, into a 1,000 ft diameter by 600 ft diameter elliptical target area (T1). Shots from the southern firing point were fired into an 800 ft diameter circular TA to the southwest (T2). 105 mm artillery shells were fired from a single firing point in the center of the site at several targets in a 1,500 ft diameter circular TA in the northeastern section of the site (A). Visible impact craters make the locations and approximate sizes of T1 and A known, but it is of interest to more precisely map the regions that may contain munitions items. The existence of T2 is suspected due to spent shell casings found at the firing point, but the impact points are now obscured by vegetation so the size and location of the target area are unknown. The anomaly density in the TAs is unknown, but several munitions experts have reckoned the density to be 100, 200, or 400 anomalies per acre above the background level.

4.1.3 Conceptual Model of the Hard Site

The *hard* site provides an example of complicated background noise where the assumption of homogeneity does not hold. It has the same layout as the medium site, with the same military use history and the same prior information available about the munitions use.

The site is a popular hunting and camping area, so background anomalies occur in clusters around camps. The mean background density is 100 anomalies per acre. A ranch occupies the eastern portion of the site. The ranch includes an 800 ft square field used for growing feed (R). The field contains metallic debris from farm equipment, and a corner of the field overlaps the artillery target area. Anomalies from farming activities and vehicles traveling on the roads are distributed homogeneously, with 100 anomalies per acre. These anomalies occur in addition to the those generated by the cluster process, which covers the entirety of the site.

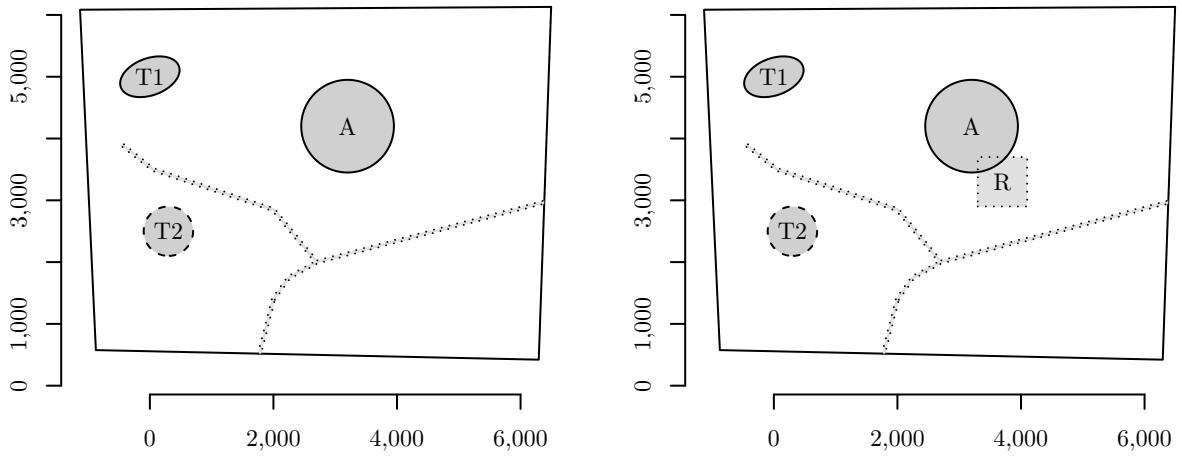


Figure 18: The *medium* (left) and hard (right) sites have two known target areas (T1 and A) and one unknown target area (T2), as well as a road that has elevated anomaly density. Homogeneously distributed background anomalies cover the *medium* site, while background anomalies at the *hard* site occur in clusters. The *hard* site contains an additional high density region due to a ranch (R) which overlaps target area A.

4.2 Spatial Poisson Processes

Anomalies occur at distinct locations throughout the site and can be modeled as the outcome of a *spatial Poisson process*, which is a random set of points in a two-dimensional space (Schabenberger and Gotway 2005, §3.1). A spatial Poisson process is characterized by an *intensity function* $\lambda(\mathbf{s})$ which describes the density (points per unit area) at a location $\mathbf{s} = (x, y)$. The number of points in a region A has a Poisson distribution with mean $\mu(A) = \int_A \lambda(\mathbf{s}) d\mathbf{s}$. Poisson processes fall into two main categories: homogeneous and inhomogeneous.

When Visual Sample Plan calculates the probability that an observed window density is higher than the background level and a target area is detected (Section 3.2.2), it assumes background anomalies come from a *homogeneous* Poisson process where the intensity function is constant, $\lambda(\mathbf{s}) = \lambda$, for all $\mathbf{s}$. The mean number of points in A is simply $\mu(A) = \lambda \times \text{area}(A)$. The locations of the points are uniformly distributed, a property called *complete spatial randomness*. I use a homogeneous process to generate background anomaly locations at the *easy* and *medium* sites. Anomalies from the road at the *medium* and *hard* sites and from the ranch at the *hard* site also come from homogeneous processes.

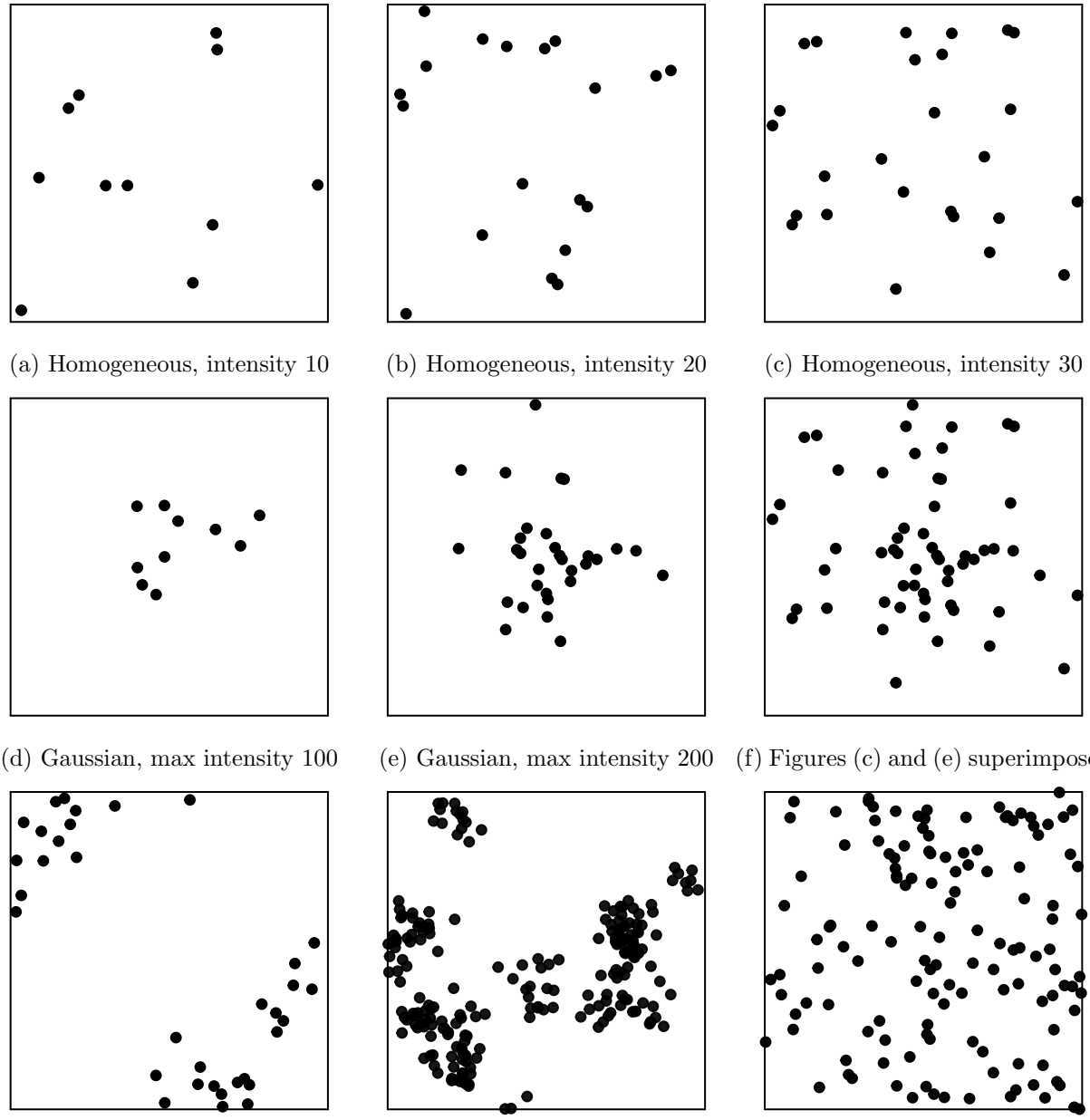
For an *inhomogeneous* Poisson process, $\lambda(\mathbf{s})$ is not constant. Some regions have higher intensity than others. A bell-shaped or Gaussian intensity function,

$$\lambda(\mathbf{s}) \propto \exp\left\{-\frac{1}{2}(\mathbf{s}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{s}-\boldsymbol{\mu})\right\},$$

concentrates points around a center $\boldsymbol{\mu}$. Spatial point density decays when moving away from the center, and the locations of the resulting points follow a Bivariate Normal distribution with mean $\boldsymbol{\mu}$ and variance-covariance matrix $\boldsymbol{\Sigma}$. I use Gaussian intensity functions to simulate the locations of the TOI anomalies.

Clustered points can be produced by a two-stage Poisson process, where cluster centers are the realization of one Poisson process and then child processes generate events around each center. To create a layer of clustered background anomalies at the *hard* site, I generate cluster centers from a homogeneous process with an intensity of two centers per acre, and then generate the anomaly locations from processes that have an average of 50 anomalies with a Gaussian intensity around each center, so the overall background anomaly density is around 100 anomalies per acre.

A complicated UXO site can be modeled as the amalgamation of several processes. Each process generates anomalies for a specific aspect of the site, such as background noise, or a target area, or a road. The simulation could be made more realistic by modeling the physical processes that bring metallic items to their resting places at the site, but I use spatial Poisson processes for simplicity. The R package `spatstat` (Baddeley, Rubak, and Turner 2015; Baddeley and Turner 2005) includes functions to simulate many homogeneous, inhomogeneous, and clustered spatial Poisson processes in regions of any shape or size. Figure 19 shows some example realizations of spatial Poisson processes simulated with `spatstat` to illustrate the building blocks that turn the conceptual site models into realizations of the sites.



(a) Homogeneous, intensity 10 (b) Homogeneous, intensity 20 (c) Homogeneous, intensity 30

(d) Gaussian, max intensity 100 (e) Gaussian, max intensity 200 (f) Figures (c) and (e) superimposed

(g) A few clusters, low intensity (h) More clusters, high intensity (i) Large, overlapping clusters

Figure 19: These example realizations on a one-unit square demonstrate some of the patterns that can arise from spatial Poisson processes. Plot (f) shows how a homogeneous background process and an inhomogeneous foreground process can combine to produce a region with elevated point density, which is how I simulate target areas. I simulate the background noise at the *hard* site using a cluster process similar to the one that generated Plot (i).

4.3 Transect Sampling Plans from the Conceptual Site Models

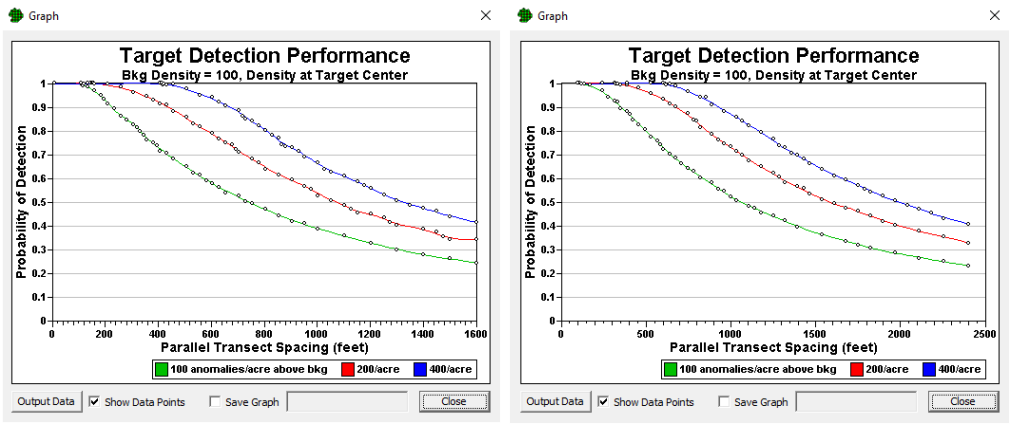
To see how prior information affects the sampling plan and subsequent delineation results, I vary two aspects of the conceptual site model (assumed target area size and assumed target area density) and constructing sampling plans for each site, holding all other sampling plan inputs constant.

The sampling plans are based on four assumed target area sizes. For each site, two of these sizes are the true sizes of the two known target areas. I also use a small size that has half the area of the smallest true TA, and a large size that has twice the area of the largest true TA. The assumed target area anomaly density values I use are the true value (200 anomalies per acre above background), as well as half the true value and twice the true value. Because I create sampling plans based on the true values, as well as sizes that are too small and too large, the results of delineation can be used to asses the importance of accurate prior information.

Visual Sample Plan's detection probability feature (Section 3.2.2) guides the sampling design. I select six-foot-wide parallel transects running north to south, and choose the transect spacing that detects a target area of the assumed size with probability 0.99 as reported by VSP, given the other input values. For the assumed background density, the known value of 100 anomalies per acre is used. A Normal distribution is assumed for the TA anomaly locations, and I leave the maximum error and minimum precision at their default values. I will assume the detection equipment works perfectly, so the false negative rate is set to 0%. The resulting detection probability curves for the *easy* site appear in Figure 20, and the detection probability curves for the *medium* and *hard* sites appear in Figure 21. For a given transect spacing, the reported detection probability increases when either the assumed TA size or the assumed TA density are increased. This is unsurprising because more transects can pass through a larger target area, providing more opportunities for the TA to be detected, and a higher anomaly density inside a target area makes high moving average densities more likely to be observed.

The transect spacings for each site are presented in Tables 1 and 2. The VSP-recommended transect spacing increases with both increasing assumed TA size and increasing assumed TA density. Sampling plans for the *medium* and *hard* sites cover a wider range of transect spacings than the plans for the *easy* site because the assumed TA sizes for the *medium* and *hard* sites are more variable. Note that the largest transect spacing for the *medium* and *hard* sites (1,145 feet) is larger than the size of the smallest true target area (T1, 1,000 feet by 600 feet), so transects from this plan could fail to cross any part of T1.

I implement each sampling plan on each realization of the site by randomly choosing the horizontal position of the first transect, and then placing additional transects separated by the transect spacing plus the width of the detection equipment. A rectangular `spatstat` window object is centered over each transect, and all anomalies within the rectangles are considered detected. The locations of these anomalies are saved as the sample data. I place circular windows along the transects as described in Section 3.2.5, and then save the moving average density values in a GeoEAS file.



(a) TA1

(b) TA2

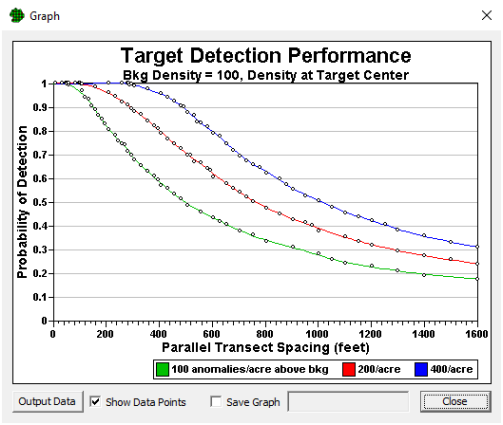
(c) Half the area of TA1

(d) Twice the area of TA2

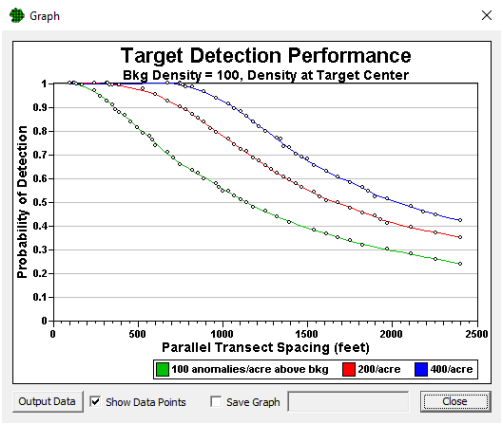
Figure 20: Visual Sample Plan created these detection probability curves by Monte Carlo simulation, based on different assumptions about the target area size and density at the *easy* site. These curves were used to choose the transect spacings for the sampling plans.

	Assumed Anomaly Density Above Background		
Assumed Target Area Size	100 per acre	200 per acre	400 per acre
Small (849 ft by 566 ft, 8.66 acres)	40 ft	100 ft	220 ft
TA1 (1,200 ft by 800 ft, 17.3 acres)	125 ft	225 ft	465 ft
TA2 (2,000 ft by 900 ft, 32.4 acres)	170 ft	390 ft	655 ft
Large (2,828 ft by 1,273 ft, 64.9 acres)	270 ft	565 ft	935 ft

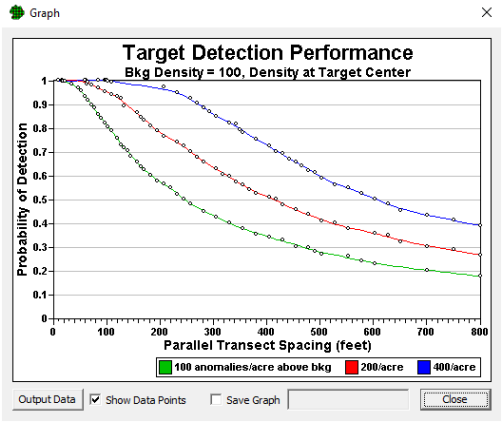
Table 1: These Transect spacings for the *easy* site are recommended by VSP to traverse and detect a target area of each size in 99% of samples, given the specified assumptions about the anomaly density above background at the center of the target area.



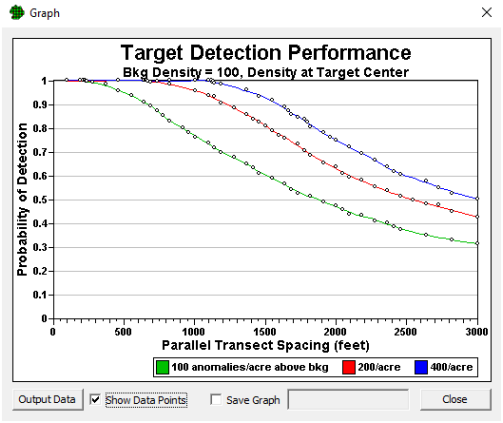
(a) T1



(b) A



(c) Half the area of T1



(d) Twice the area of A

Figure 21: Visual Sample Plan created these detection probability curves by Monte Carlo simulation, based on different assumptions about the target area size and density at the *medium* and *hard* sites. These curves were used to choose the transect spacings for the sampling plans.

	Assumed Anomaly Density Above Background		
Assumed Target Area Size	100 per acre	200 per acre	400 per acre
Small (707 ft by 424 ft, 5.40 acres)	30 ft	65 ft	130 ft
T1 (1,000 ft by 600 ft, 10.8 acres)	70 ft	135 ft	315 ft
A (1,500 ft by 1,500 ft, 40.6 acres)	175 ft	400 ft	785 ft
Large (2,121 ft by 2,121 ft, 81.1 acres)	320 ft	780 ft	1,145 ft

Table 2: These Transect spacings for the *medium* and *hard* sites are recommended by VSP to traverse and detect a target area of each size in 99% of samples, given the specified assumptions about the anomaly density above background at the center of the target area.

4.4 Kriging and Delineation

I recreate Visual Sample Plan's automated Kriging and delineation features (Section 3.2.6) in R (R Core Team 2015) and `spatstat`. Compared to VSP, R makes it much easier to repeat the same analysis on many sets of simulated sample data. Slight differences in the results are expected due to differences in programming. I kept my methods consistent with VSP as much as possible, but I considered it necessary to modify some aspects of the analysis to make automation more practical. These changes are acceptable because the main interest is in the inputs used to create the sampling plan. My analysis does not produce exactly the same delineated regions as the VSP analysis, but since I analyze the sample data from each sampling plan in the same way, changes in the sampling plan should affect my results in the same way they would affect VSP results. For example, my analysis tends to delineate more total area than a VSP analysis would, but if increasing the assumed TA size causes my total delineated area to increase, we can assume the change is due to the change in prior information and that increasing the assumed TA size would also increase the total area delineated by the VSP analysis. Sections 4.4.1–4.4.3 describe my implementation of the Kriging and delineation procedure and point out the differences from the VSP analysis.

4.4.1 Estimating Covariance Parameters and Computing Kriging Predictions

In an attempt to preserve some consistency with VSP, I also use GSLIB to compute empirical semivariograms and Kriging predictions. The VSP installation includes a 32-bit version of GSLIB (which contains GAMV version 2.000 and KT3D version 2.000). However, some of my sampling plans result in sample datasets that are larger than what this version of GSLIB can process. Instead, I use the most recent 64-bit version, which includes GAMV version 2.905 and KT3D version 2.907. I have no problems running the newer versions and they produce the same results for datasets where both versions run. I set up the GAMV configuration file in the same manner as VSP (Section 3.3.2).

The VSP documentation does not explain how VSP estimates covariance parameters, but I get similar estimates using Cressie's weighted least squares method (Cressie 1985). The optim function in R uses the L-BFGS-B algorithm (Byrd et al. 1995) to find the parameter values that minimize the weighted sum of squared errors. I fit spherical, exponential, Gaussian, and power models to the empirical semivariogram and select the model with the lowest weighted sum of squared errors. After selecting a parametric covariance function, I use KT3D to compute Kriging predictions over a grid in the same way that VSP does (Section 3.3.4).

4.4.2 Thresholding the Predicted Densities

There is little guidance available regarding the choice of density threshold value for identifying the high-density regions, so in practice the choice involves some subjectivity. An automated simulation requires a rule that applies to all realizations, but I would not expect a subjective choice to result in consistent performance across different realizations. Some obvious starting points for an objective choice would be the true background density (if known) or the estimate of the mean density. However, sampling variability makes both of these poor choices for the threshold. Some search windows will naturally contain few anomalies and give low observed density values, while other windows that contain only background anomalies will include more anomalies and give observed

densities above the true overall background density. Thus, a large part of the site is erroneously delineated. If most of the site contains only background anomalies, the estimated mean will be close to the true background density and cause the same issue. A better decision rule should account for the natural variability in the observed density values. Future work should look into methods of thresholding the predicted densities based on information contained in the cenceptuak site model and sound statistical assumptions.

My goal for this project is not to find an ideal threshold, but a threshold that always results in most of the site being delineated is unsuitable because it would mask the effects of the sampling plan inputs. I take inspiration from the target area flagging feature in VSP (Section 3.2.5), which provides the option of using a hypothesis test to decide if the density at an observed location is high enough that the location is part of a possible target area. I do not claim that this is a good approach to use, but it accounts for the variability in the moving average densities and allows the simulation to proceed.

For each grid cell, I conduct a hypothesis test. Under the null hypothesis, the predicted density in the grid cell comes from a Normal distribution with a mean equal to the background density (which my site models state as known) and variance equal to the Kriging variance. If

$$\text{predicted density} > \text{background density} + 1.645 \times \sqrt{\text{Kriging variance}}$$

the null hypothesis is rejected at a level of 0.05 and then the grid cell is identified as possibly belonging to a target area.

When working with real data, the normality assumption could be tenuous. The user should always examine VSP's histogram of the observed moving average densities and confirm that it has a bell-shape before using Normal distribution procedures. If the histogram does resemble a Normal distribution, there typically will be enough observed density values that t-distribution procedures are not needed.

This method of identifying high-density grid cells is used to eliminate the need for subjective human input while the simulation runs. It is not entirely sensible when searching for UXO at a real site because a larger Kriging variance makes a cell less likely to be marked as a possible TA than if the prediction had a smaller Kriging variance. In reality, it is would be preferable to construct a rule where greater prediction uncertainty makes a location more likely to be considered a TA. My decision rule also tends to result in delineating more area than I would deem necessary from subjective examination of the density map (Figure 22).

4.4.3 Delineating the High-Density Regions

The final difference between the Visual Sample Plan analysis and the methods I use in the simulation is how the high-density grid cells are combined into regions. VSP draws polygons around clusters of connected cells, giving its delineated regions a smooth appearance. The documentation does not describe the algorithm used to create the polygons, but it appears to connect the corners of cells that extrude from the cluster. It forces some regions to be convex by including lower-density cells, while other regions are allowed to be non-convex. It is also able to merge nearby regions into larger regions.

Any attempt I make to reproduce VSP's polygons would require some arbitrary decisions about how to connect corners and merge regions. Instead, I simply find clusters of connected cells and use their edges as the delineation. This yields minor differences in the area and number of anomalies included in the delineated regions, but both methods give essentially the same regions since they both contain all the cells identified as high-density. My delineated regions appear pixelated compared to the regions produced by VSP, and my results will also include larger numbers of small regions. Figure 22 illustrates the typical differences.

4.4.4 Summarizing Results

The ultimate goal of target area delineation is to find the targets of interest, so I compare the results primarily through the proportion of TOI items that lie within delineated regions, called the *detection rate*. Other informative summary variables are the proportion of the TA area contained in the delineated regions, the area and number of delineated regions, the proportion of the delineated area that is not actually part of a target area (called the *false positive proportion*), and the distance of each undetected TOI item to the nearest delineated region (the *error distance*).

This project is exploratory in nature, and conclusions from my simulation are best used as starting points for future investigations. For that reason, I forego formal statistical inference and instead present the results graphically. I analyze many realizations with each sampling plan, and therefore histograms provide informative displays of the distributions of the summary values. I use these plots throughout the paper to compare results among the sampling plans.

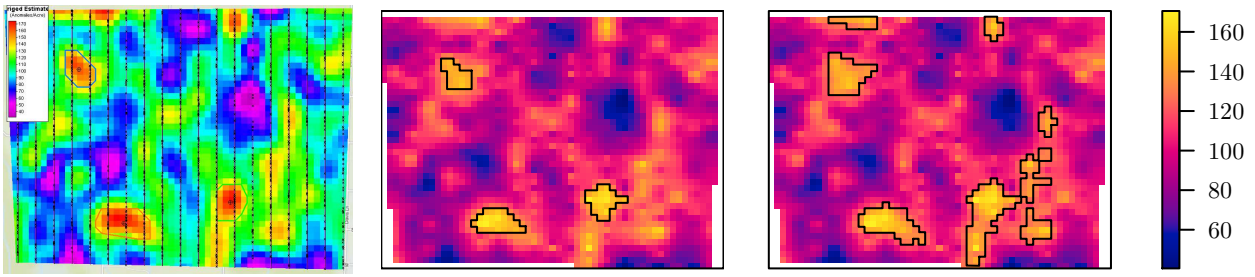


Figure 22: The left and center images show the anomaly density maps produced by the VSP analysis and my analysis from the same sample data. Both maps are thresholded at 140 anomalies per acre, a value which I judged to do well at separating the background noise from the highest-density regions. The delineated regions contain nearly the same grid cells in both cases. The disagreements about which cells are high-density are most likely due to slight differences in how VSP and `spatstat` compute the moving average window densities. The image on the right shows the regions delineated by the decision rule used in the simulation, which marked more low-density regions as possible target areas.

5 Window Size Experiment

The size of the search window that VSP uses to compute moving the average anomaly density can have a major influence on the predicted density surface. Before addressing my main goal of understanding the effects of prior information on delineation results, I look into the choice of window diameter. This experiment involves repeatedly analyzing the same sample data with different window sizes. In a real-world situation it is possible to run multiple analyses on one set of data collected from one realization of a site, but it is difficult to decide which analysis is best if the true target area locations and sizes are unknown. Simulation allows the results of Kriging and delineation to be compared to the known truth. Also, the same analysis can be repeated on many realizations of one site to illustrate the variability that arises simply due to natural variation in the process that generates anomalies. The *easy* site is used so that the window diameter effect can be examined under favorable conditions.

5.1 Design and Analysis

This experiment uses 200 realizations of the *easy* site. Each realization is sampled once with a between-transect spacing of 225 feet. This sampling plan should detect TA1 (800 feet by 1,200 feet) with probability 0.99 (see Table 1 on page 39). Each realization is analyzed using five different window sizes.

Three window diameters (228 feet, 516 feet, 798 feet) range from just above the between-transect spacing to just under the minor axis length of TA1. These sizes are selected in accordance with the Visual Sample Plan user's guide, which recommends choosing a window diameter at least as large as the between-transect spacing but smaller than the diameter of the target area of interest (Matzke et al. 2014). Two additional diameters (150 ft and 1,500 feet) are chosen to be much too small and much too large.

At the end of the analysis, regions of the site with high predicted density are delineated. Visual Sample Plan discards any delineated regions under a user-specified minimum area, but since the window size is a smoothing parameter, very small regions could be indicative of inadequate smoothing. Thus, for the comparisons made within this experiment, I keep all regions regardless of size.

5.2 Results

The success of the analysis is most directly measured by the detection rate, which is the proportion of true TOI items that would be removed from the site if all of the delineated area was remediated. The detection rate increases as the window diameter increases. When using the 1,500 foot window, it is not uncommon for 100% of the TOI items and 100% of the TA area to be contained in the delineated regions (Figure 23, top and center rows).

The amount of area delineated is an important practical consideration because a remediation project has limited resources. It may not be possible to remediate all of the delineated area, so delineating more area than necessary is undesirable. The total area delineated both increases and becomes more variable as window size increases. For some realizations, the 1,500 foot window results in over

half of the 952.4 acre site being identified as high-density (page 47, Figure 24, top right). The 798 foot window gives more manageable results, with the total delineated area being both lower and less variable while still achieving a detection rate over 80% for most realizations.

The area of the smallest and largest individual regions both increase with the window size; for the 150 foot, 228 foot, and 516 foot window sizes, in most realizations the smallest delineated region is a single grid cell. This is a sign that these windows sizes provide inadequate smoothing of the predicted density surface. The 798 foot window does the best job of producing regions that are close to the sizes of the true target area sizes (page 46, Figure 23, bottom row).

At all window sizes, more area is delineated than the area of the true target areas. Large windows have a smoothing effect on the density surface, so small numbers of large regions are delineated. Conversely, small windows are sensitive to local density hotspots and result in large numbers of very small regions being delineated. There is a tradeoff between large windows, which yield a few regions and a large amount of variability in the total area, and small windows, which result in less total area divided up into too many individual regions (page 47, Figure 24, center left).

In all cases, most of the delineated area does not belong to the true target areas (page 47, Figure 24, bottom left). The 516 foot window tends to give a slightly lower false positive proportion than the other window sizes do, but overall there is little variation in false positive proportion among window sizes.

The error distances have extremely right-skewed distributions, with most error distances being much smaller than the sizes of the target areas (page 48, Figure25). The tail of the distribution increases in length as the window size increases because larger windows reduce the detail in the density map, forcing the delineation to be less precise. Most undetected TOIs are very close to the boundary of a delineated possible TA; exceptions are only observed for the 1,500 foot window. For two realizations, the analysis with the largest window completely failed to detect one TA, resulting in a cluster of very large error distances.

There is no window diameter that is obviously the best choice. Larger windows result in more TOIs being found, but the tradeoff is that more area must be remediated to do so. None of the delineations could be considered both accurate and precise. Window diameters larger than a target area or smaller than the space between transects cannot be recommended; these result in, respectively, far too much area delineated or an unrealistically large number of regions identified. When using the 516 foot and 798 foot windows, detection rates above 90% are common. Further studies (perhaps based on real-world sites) could help formalize more general guidelines, but for the *easy* site, it seems advisable to use a window size near the true size of the target area of interest.

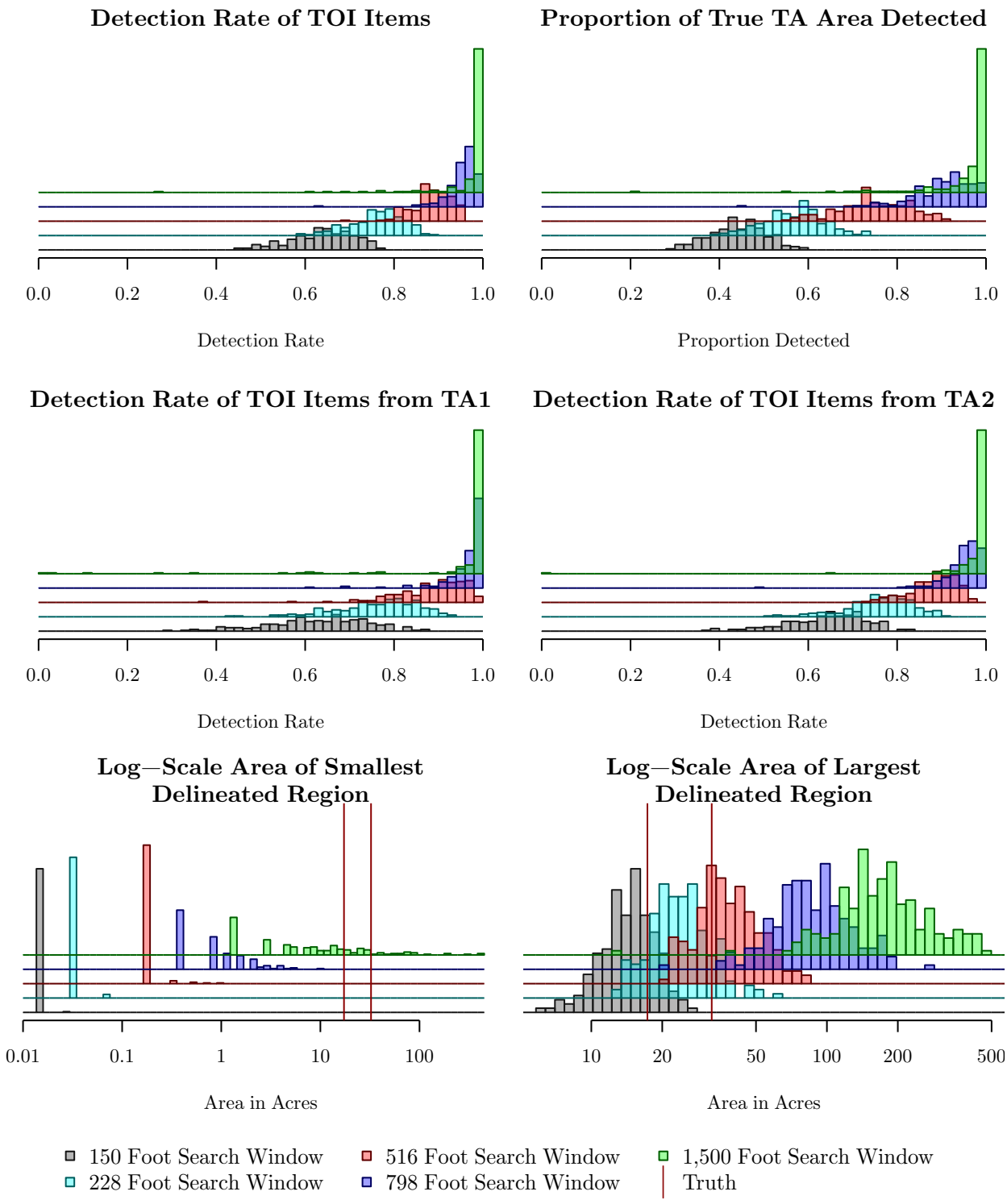


Figure 23: The detection rates and the proportion of the true area delineated measure the ability of the analysis to find the TAs and TOI items. Larger windows result in more of the TOIs being detected. The areas of the smallest and largest regions help in assessing whether the analysis produces regions that tend to be around size of the true TAs. The red lines mark the true areas of TA1 (17.3 acres) and TA2 (32.4 acres).

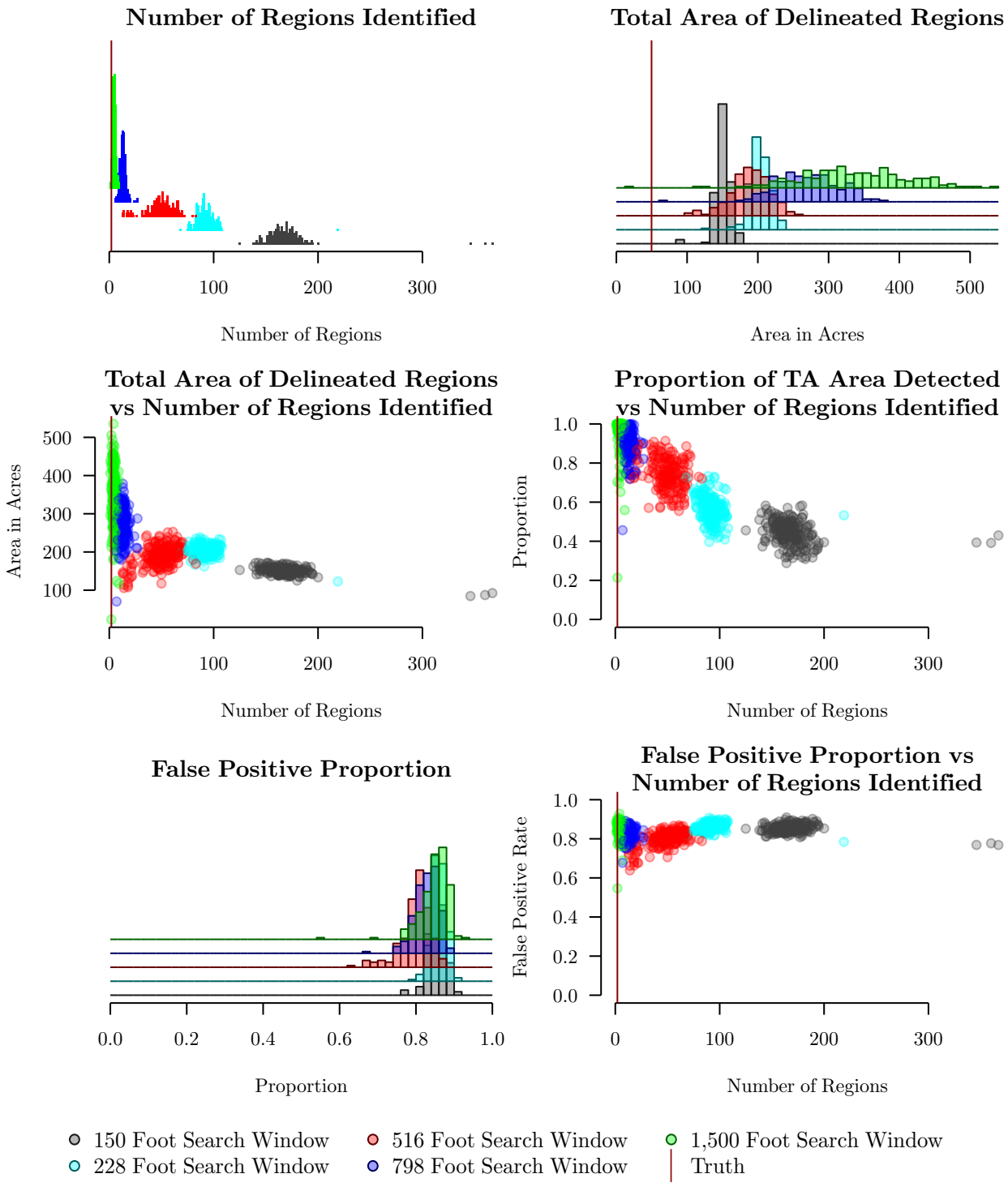


Figure 24: Plots showing the relationships between the number of regions and other variables for the window size experiment. The red vertical lines mark the true total area of the target areas (49.7 acres) and number of target areas (2).

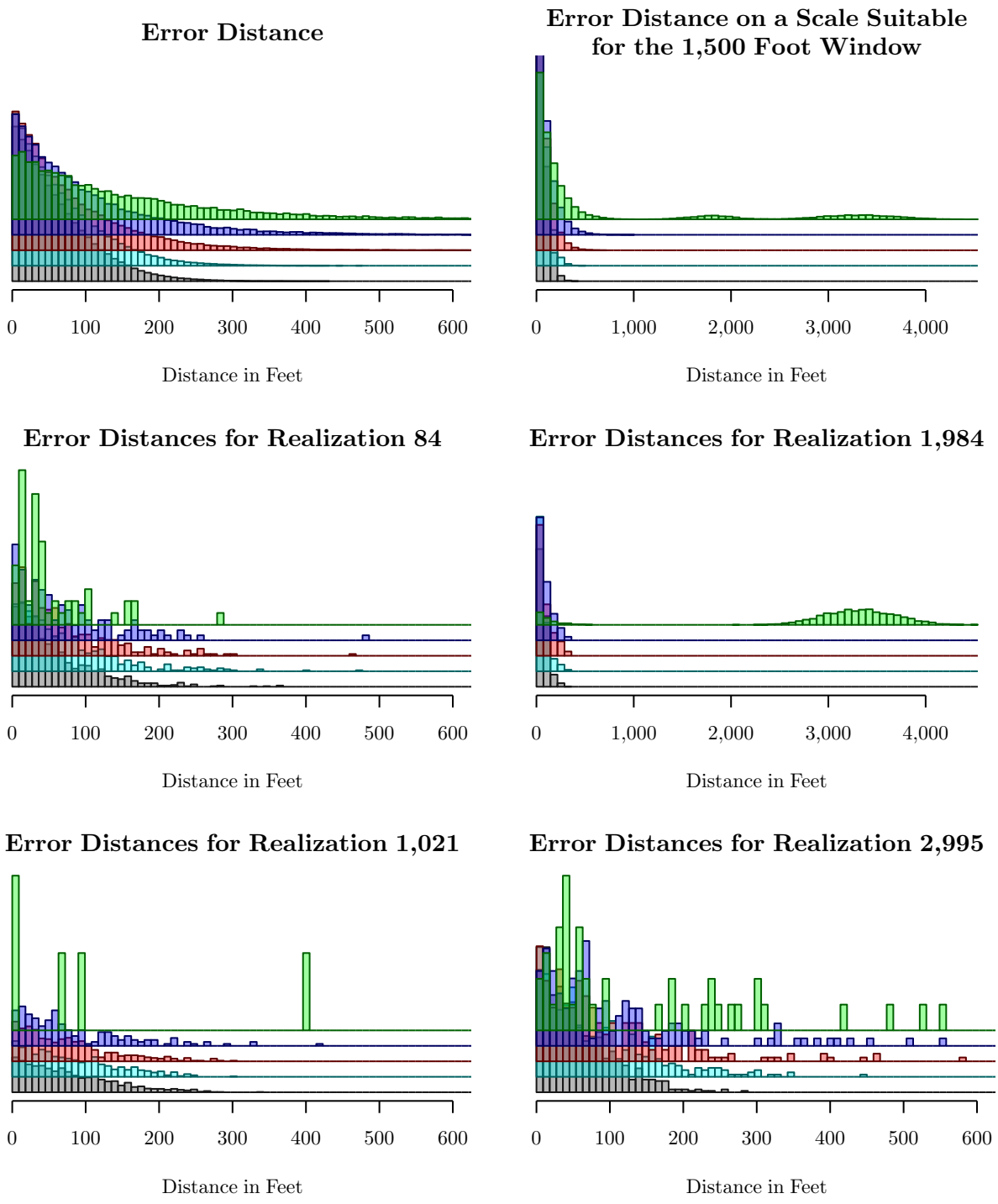


Figure 25: The distribution of error distances is right skewed both for all realizations combined and for individual realizations. Most undetected TOI items are near the boundary of a delineated region. The two clusters seen in the top right plot come from realization 1,984, where the analysis completely failed to detect TA2, and one other realization where TA1 was not detected.

6 Prior Information Experiment

The primary focus of this project is how prior information about the target area size and anomaly density input into Visual Sample Plan affects the delineation of target areas through the sampling plan. The window size experiment (Section 5) involved repeated analysis of one set of sample data from each of multiple realizations of the *easy* site, but the prior information experiment more fully uses the benefits of simulation by using multiple sampling plans on each realization, which would typically be prohibitively expensive at a real site. To illustrate the inherent variability in the anomaly-generation process, this experiment also uses multiple realizations of each site. All three sites (*easy*, *medium*, and *hard*) are used so the effect of their differing complexity can be seen.

6.1 Design and Analysis

The factors of interest are the assumed target area size and the assumed target area density that are input into VSP. The site (*easy*, *medium*, and *hard*) is a third factor. For each site, I consider four levels of assumed TA size: (1) a small size half the area of the smallest TA, (2) the true size of the smallest TA, (3) the true size of the largest TA, and (4) a large size twice the area the largest TA. There are three levels of assumed TA density: (1) 100, (2) 200 (the true value), and (3) 400 anomalies per acre above the background density. In Section 4.3, I used each combination of these factors to create a sampling plan for each site (Tables 1 and 2). For this experiment, I apply all twelve sampling plans to 100 realizations of each site. The true sizes, true densities, and all other sampling plan inputs are held constant.

The window size experiment revealed that a search window size close to the true size of the smallest target area gives relatively good performance for the *easy* site (Section 5.2). In reality, the true TA size would be unknown, but if the project team trusts their prior information they would use whatever they know about the TA size to make the window size decision. To realistically model how the TA size information is used, I let the prior information influence the window size for this experiment. I set the search window diameter to 90% of the minor axis of the assumed TA size.

During a real remedial investigation, the analyst would ignore regions considered too small to be a target area. The analyst enters a minimum area threshold into VSP, and then VSP does not draw any possible TAs under this size on its map (Section 3.2.6). For this experiment, I consider regions under 3 acres too small to be target areas and are I omit them from the results.

As with the window size experiment, I evaluate the results mainly in terms of the TOI detection rate. For the *medium* and *hard* sites, the road and ranch are nuisance regions that have intermediate anomaly density, lower than the density in the target areas, but higher than the background density across the rest of the site. The proportion of these regions that are delineated is also an interesting outcome of the analysis.

6.2 Detection Rate Results

The results from all three sites show similar patterns in how the assumed target area size and density affect the detection rates and in the proportion of the total TA area that is delineated. There is a slight trend of decreasing detection rate as the assumed target area size or the assumed density are

increased, but the most noticeable pattern is that the variability in the detection rate gets larger as either factor is increased. For all sites, sampling plans based on the two largest assumed TA sizes and the largest assumed TA density result in more diverse detection rates compared to the other sampling plans.

For the *easy* site, the detection rates are generally very high, but for some realizations, the sampling plans based on the size of TA2 or the large size result in the analysis doing a poor job of detecting items from the smaller TA1. (page51, Figure 26).

At the *medium* and *hard* sites, when the target area anomaly density is assumed to be 100 anomalies per acre above background, all sampling plans result in detection of nearly all of the TOI and target areas, with the only exception being the sampling plan based on the large TA size (page 52, Figure 27 and page 53, Figure 28). The detection rate gets more variable as the assumed target area size and density increase. For all assumed TA densities, the analysis using the large size tends to either detect all items from T1 or detect none of the items from T1. When the true size of T1 or the small size are used to create the sampling plan, the analysis tends to detect most, but not all, of the TOI items from A; this could be because the small windows do not provide enough smoothing to accurately map the anomaly density in the large artillery range.

The detection rate varies more for the *hard* site than for the *medium* site. A possible explanation is that the clustered background anomalies lead to greater uncertainty in the predicted density, but I leave the details of this issue for a future study.

Within a single site, the observed differences in detection rate are better explained by the assumed TA size than the assumed density.

As a final comment, note that the proportion of the TA area detected tends to be a little lower than the detection rate of TOI items. The "true" area is based on a region drawn on a map, which is essentially an artificial construct meant to help the people understand the site. The analysis finds TOIs; it will not delineate the entire region where munitions use occurred if parts of the region have few TOI items actually present. This result illustrates that there is a subtle difference between finding TAs and finding TOIs, so analysts should not be surprised if the map of TOI locations does not look like the expected map of munitions use areas.

6.3 Identification of the Unknown Target Area

The *medium* and *hard* sites contain a third target area, T2, of an intermediate size between the smaller T1 and the larger A. The conceptual site models state that the size and location of T2 are unknown, so specific information about it is not available for use in creating a sampling plan. I use this simulation to see if sampling plans created from knowledge of the other TAs can identify T2.

At both sites, sampling plans based on the small size or the true size of T1 generally detect nearly all TOI items from T2 (page 54, Figure 29). When the large size or the true size of A are used, the analysis tends to detect either all or none of the T2 items. It is preferable to find some of the TOI items and gain additional information about the TA than to miss the TA entirely, so I recommend using a smaller assumed TA size to design the sampling plan. The assumed density has little effect on the detection rate of TOIs from T2.

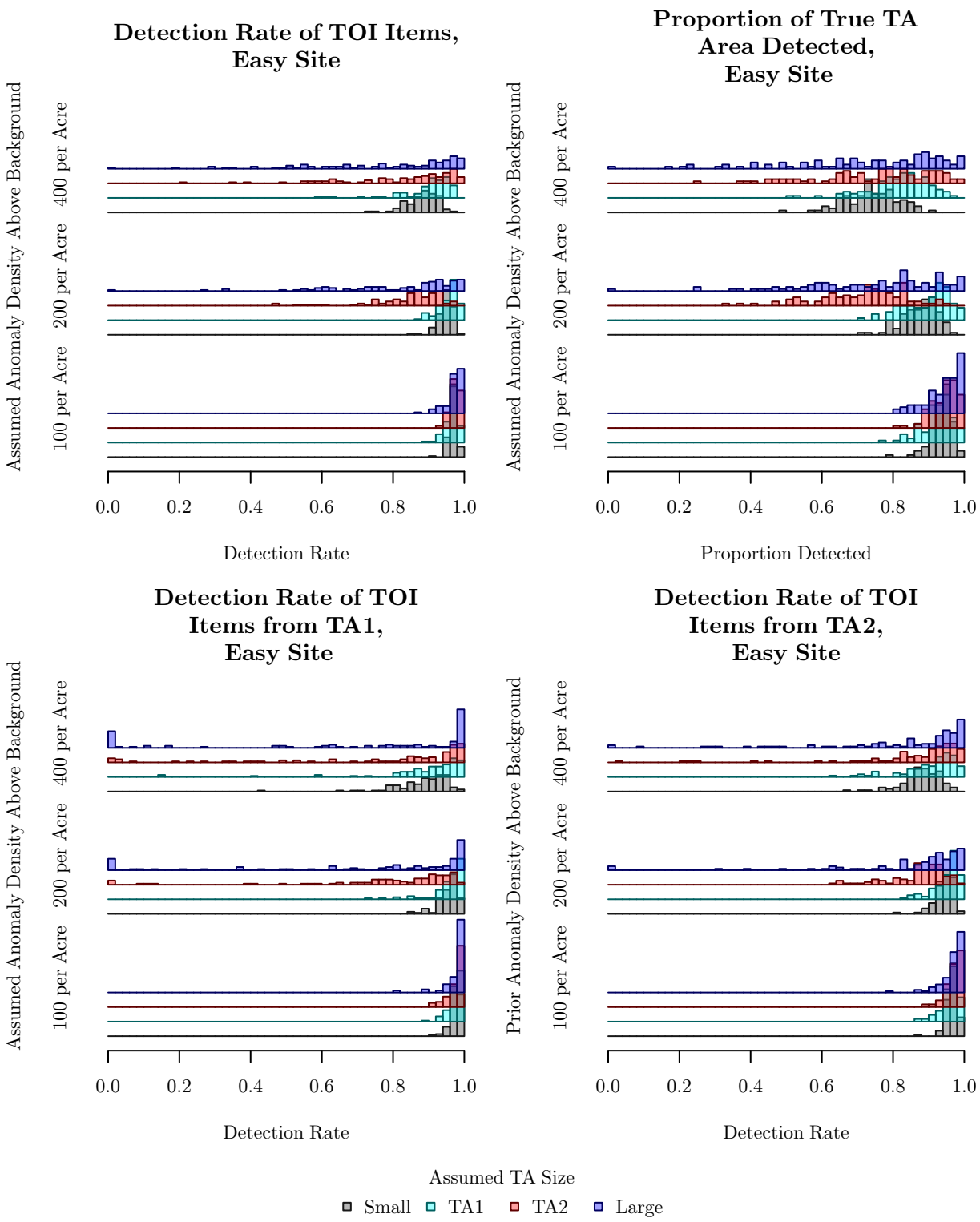


Figure 26: At the *easy* site, all sampling plans tend to detect most or all of the TOI items and TA area. The detection rates are more variable for larger assumed TA size and density.

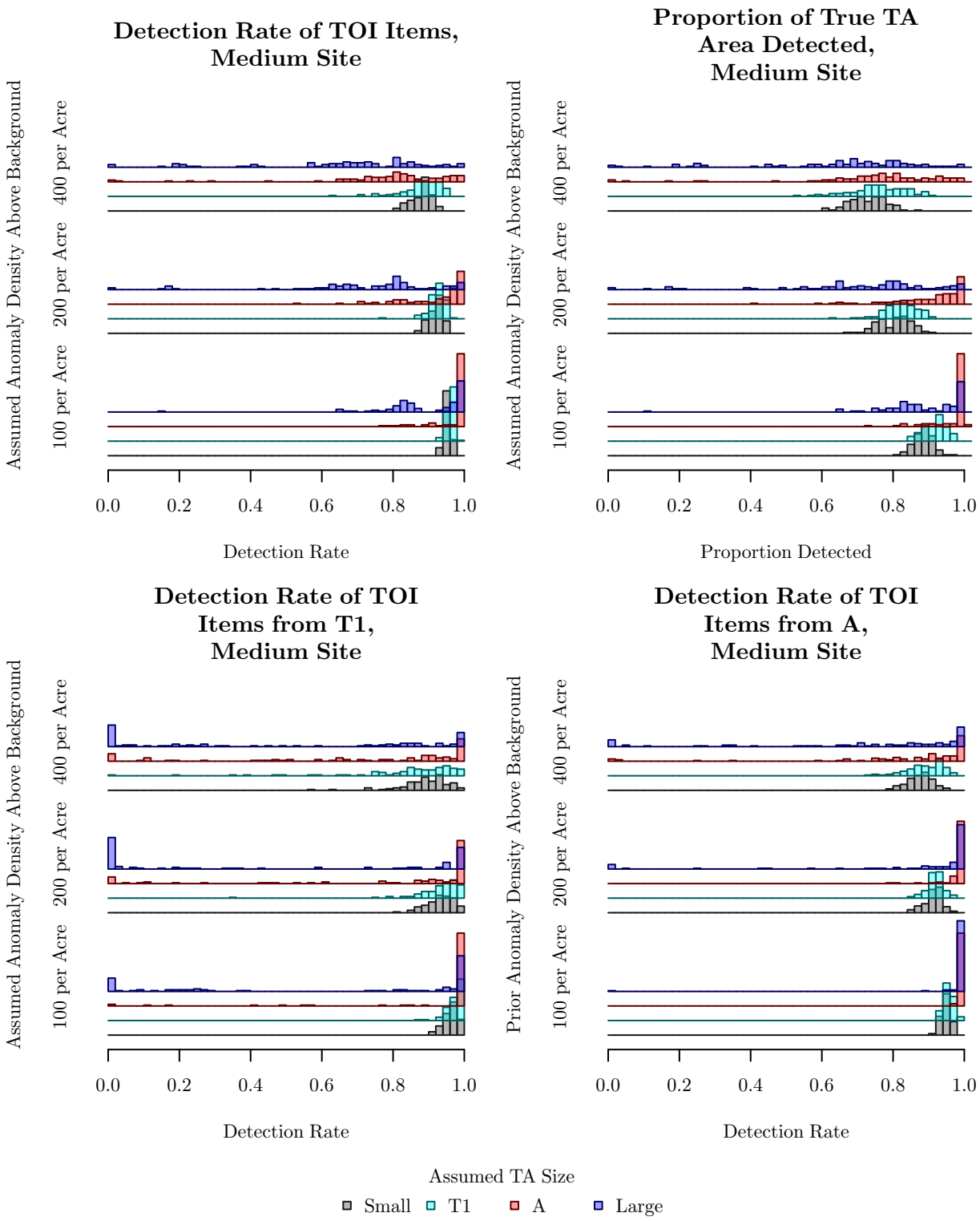


Figure 27: Compared to the *easy* site, detection rates at the *medium* site tend to be lower but show similar relationships with the assumed TA size and density.

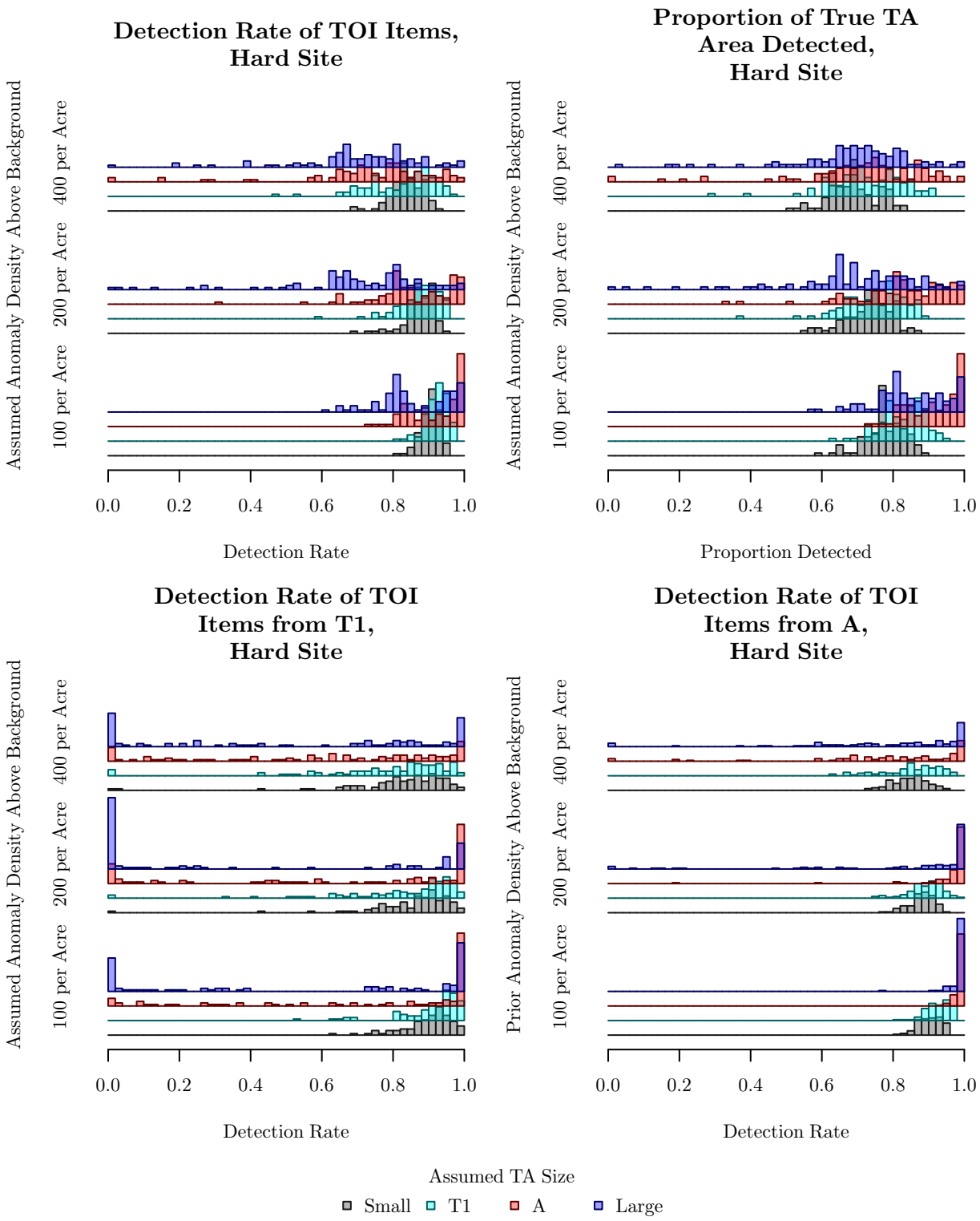


Figure 28: For the *hard* site, detection rates show the same trends as seen at the *medium* site, but the detection rates at the *hard* site are somewhat more variable.

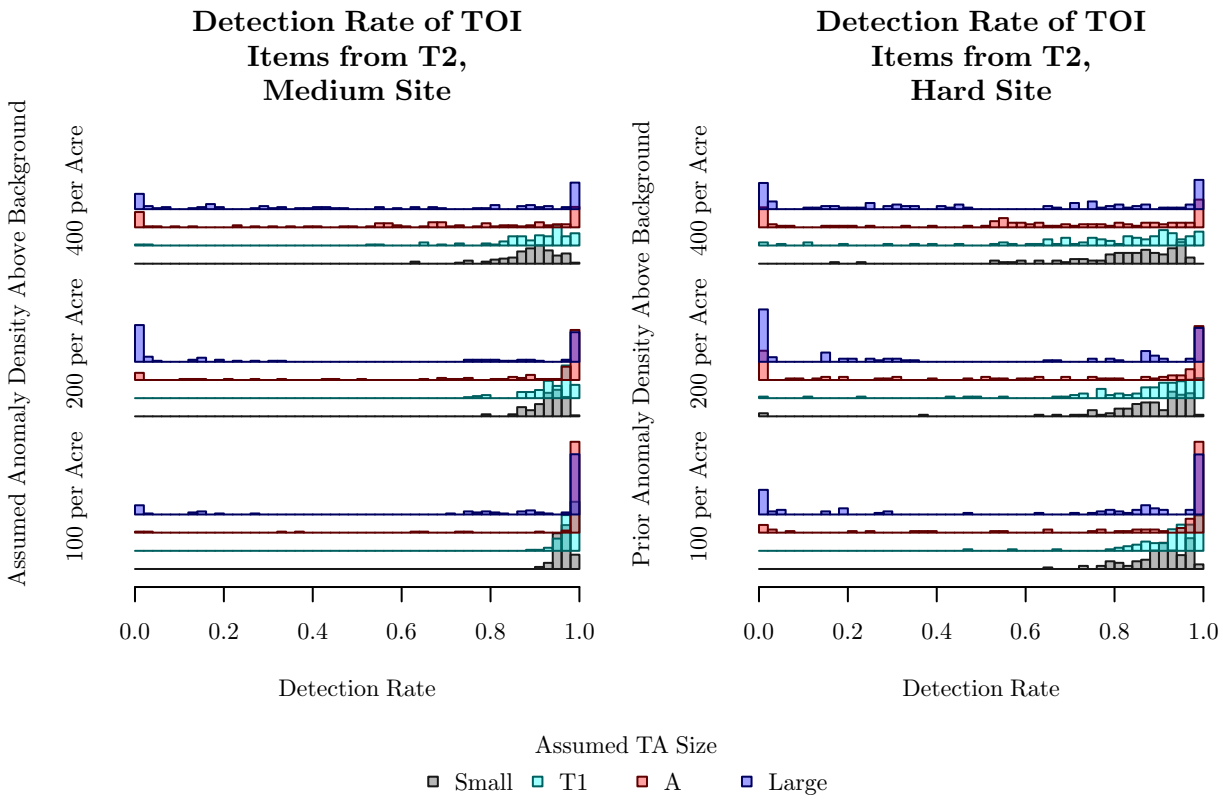


Figure 29: At both the *medium* and the *hard* site, assuming a small target area size is important for finding the third target area, T2.

6.4 Number and Area of Delineated Regions

Ideally, analyzing data from a UXO site should result in the delineation of a few regions that contain TOI items and do not include excess area. Delineating a large number of small regions or a lot of excess area would lead to a costly and unproductive remediation. My analysis delineates more regions than the number of TAs, and these regions contain far too much area. The number and area of the delineated regions are strongly related to the assumed target area size, while the assumed TA density has little effect by itself. The two factors interact, with a decrease in total area being associated with an increase in assumed density only for the small size. Overall, larger assumed TA sizes result in few distinct regions being delineated, but these regions tend to be extremely large.

For the *easy* site, as the assumed TA size increases, the number of regions decreases while the total area delineated has a very slight increasing trend (Figure 30). The total area gets more variable as the assumed density estimate is increased, and this is especially apparent when the assumed TA size is too large. The sampling plan assuming the small size and 400 anomalies per acre has a transect spacing similar to the transect spacing of the sampling plan using the size of TA1 and assuming 200 anomalies per acre (with transect spacings are 220 feet and 225 feet, respectively). These two plans tend to delineate the least area. Additional studies should be done to see if the good performance of these sampling plans can be explained by other sampling plan inputs, or even by the transect spacing itself.

For the *medium* and *hard* sites, the sampling plans separate into two groups corresponding to the two smallest assumed TA sizes and the two largest assumed TA sizes (page 56, Figure 31). Analyses assuming the size of T1 or the small size result in similar numbers of individual regions being identified. The analyses based on the size of A or the large size also yield about the same number of regions, and they find fewer regions than when a smaller TA is assumed. The number of regions found is not affected by the assumed TA density value. On average, the two smallest assumed TA sizes result in the least total area delineated, and the area delineated is much less variable than when the largest TA sizes are used. The total area plots show an interaction similar to the one seen at the *easy* site.

The plots for the *medium* and *hard* sites show similar relationships between the summary values and the factors, but slightly more area is delineated at the *hard* site. The difference is bigger than the 14.7 acres of the ranch, so it is partially due to the clusters of background anomalies.

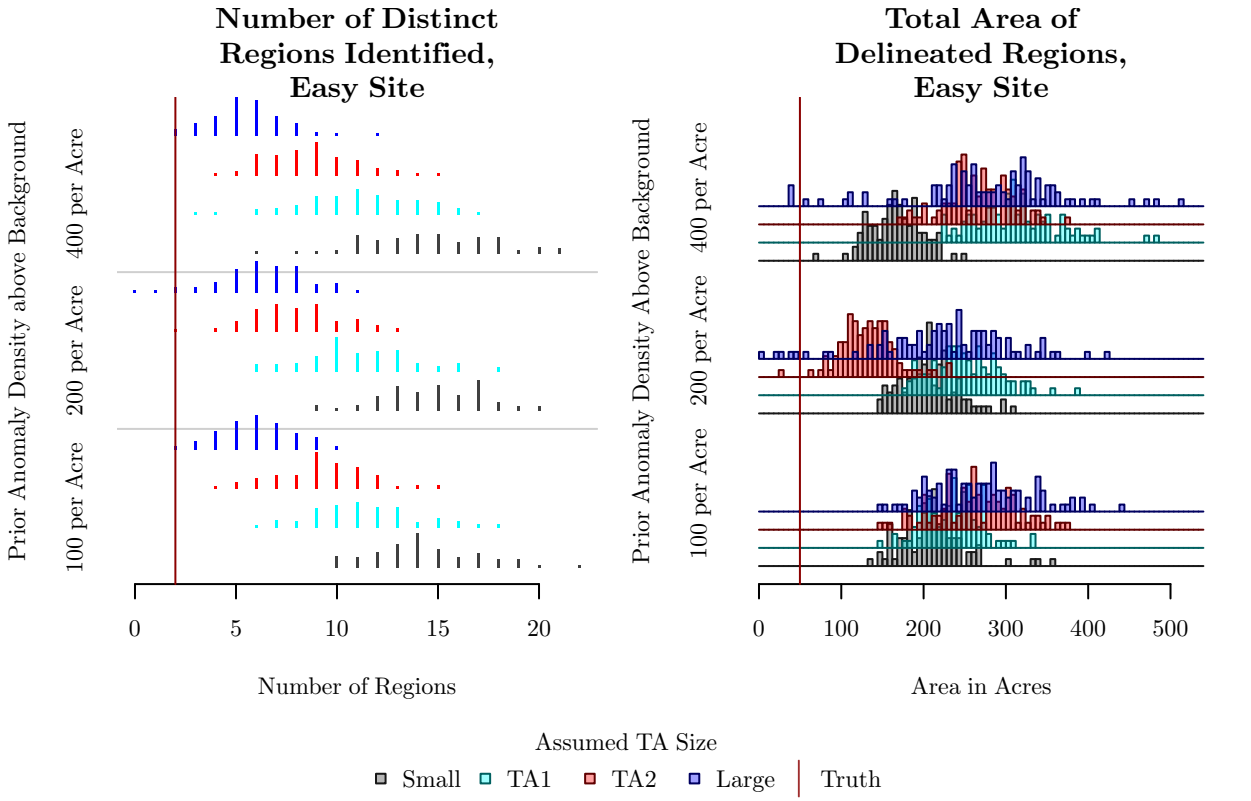


Figure 30: At the *easy* site, the assumed target area density has no effect on the number of regions delineated, but there is an apparent interaction between the assumed TA density and the assumed TA size. A similar pattern is seen at the *medium* and *hard* sites as well. The vertical lines mark the true number (2) and area (49.7 acres) of the target areas.

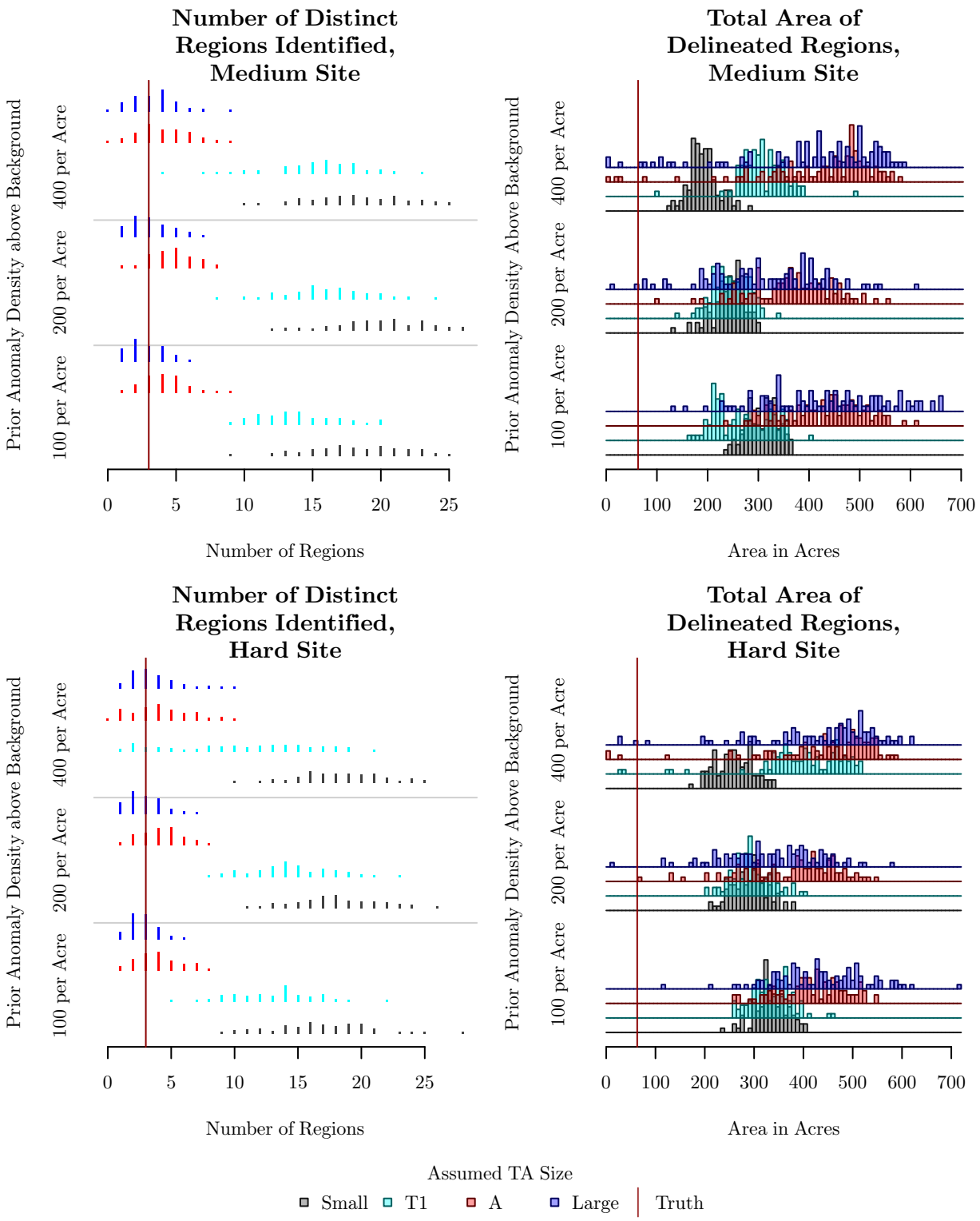


Figure 31: At the *hard* site, slightly more area is delineated than at the *medium* site, indicating that the analysis does not smooth out the clusters in the background anomalies. The vertical lines mark the true number (3) and area (62.9 acres) of the target areas.

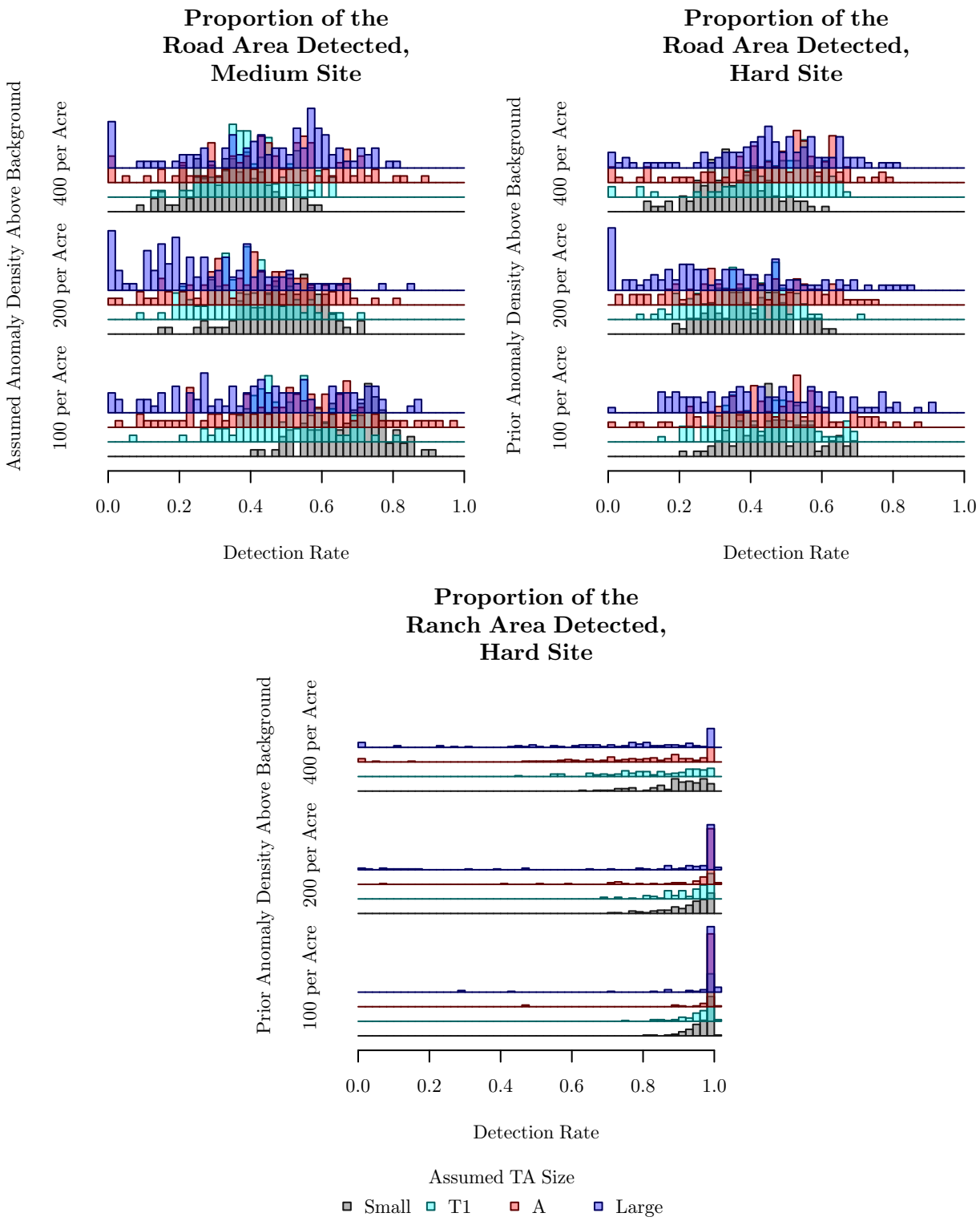


Figure 32: None of the sampling plans are able to smooth the road or the ranch out of the anomaly density map. These regions need to be described in the conceptual site model and analyzed separately.

6.5 Detection of Nuisance Regions

The road and ranch are nuisance regions with anomaly densities of 200 anomalies per acre, which is lower than the true density at the centers of the target areas, but 100 anomalies per acre higher than the background density elsewhere at the site. In reality, TOI presence in these regions would be investigated separately, but in this simulation I want to see how they affect the results of sampling plans developed for other parts of the site.

The *medium* and *hard* sites both include the road. At each site, my analysis does not smooth the road out of the density surface, so on average about half of the road is included in the delineated regions (Figure 32, top). The proportion of the road that is delineated decreases somewhat as the assumed TA size or the assumed TA density increases, but it remains large and highly variable.

Only the *hard* site contains the ranch. It is not unusual for most or all of the ranch to be identified as a possible target area (Figure 32, bottom). Only the sampling plans with the largest transect spacing – those based on the large size or the size of A – occasionally result in very little or none of the ranch area being delineated. The ranch is a true high density region similar in size to a target area, so it should be expected to appear in the predicted density surface. The ranch would need to be sampled and analyzed separately since the anomalies there are known to come from a different process than at other locations within the same site.

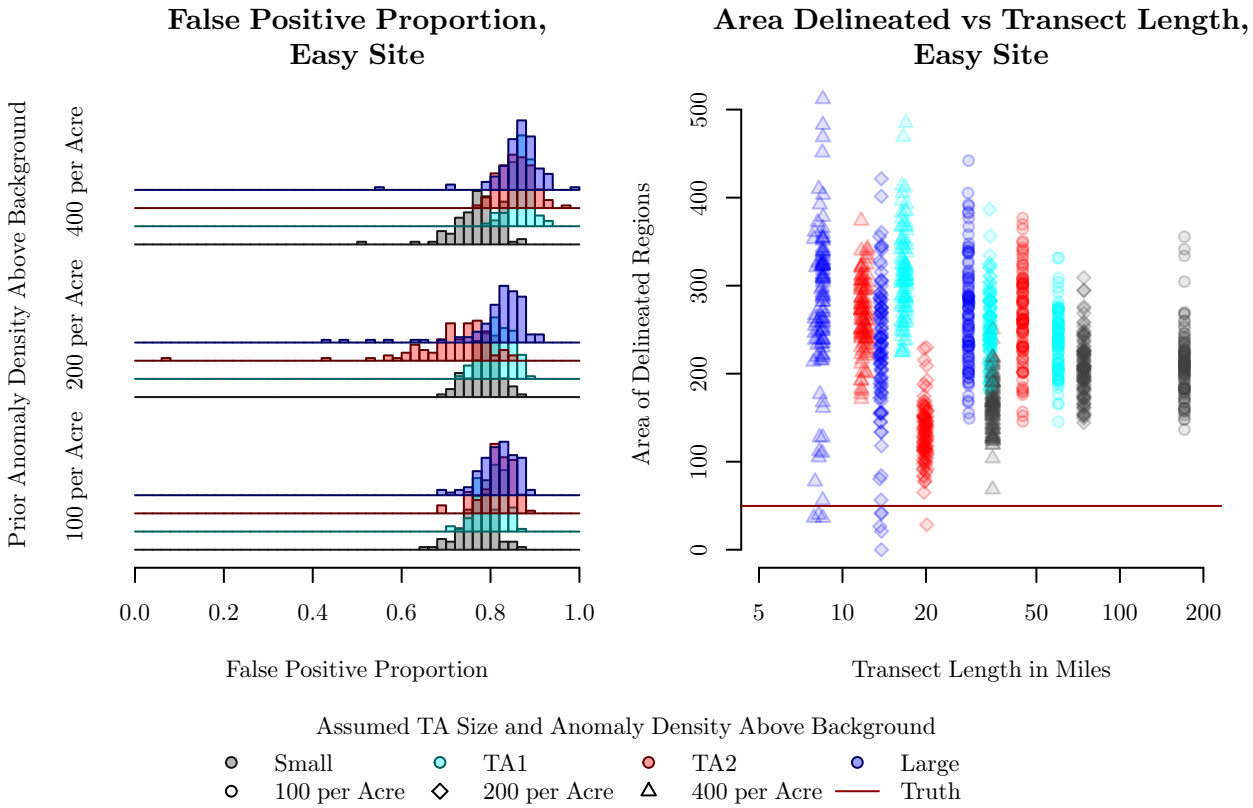


Figure 33: The *easy* site contains 49.7 acres of target areas, but all sampling plans result in far more area than this being delineated.

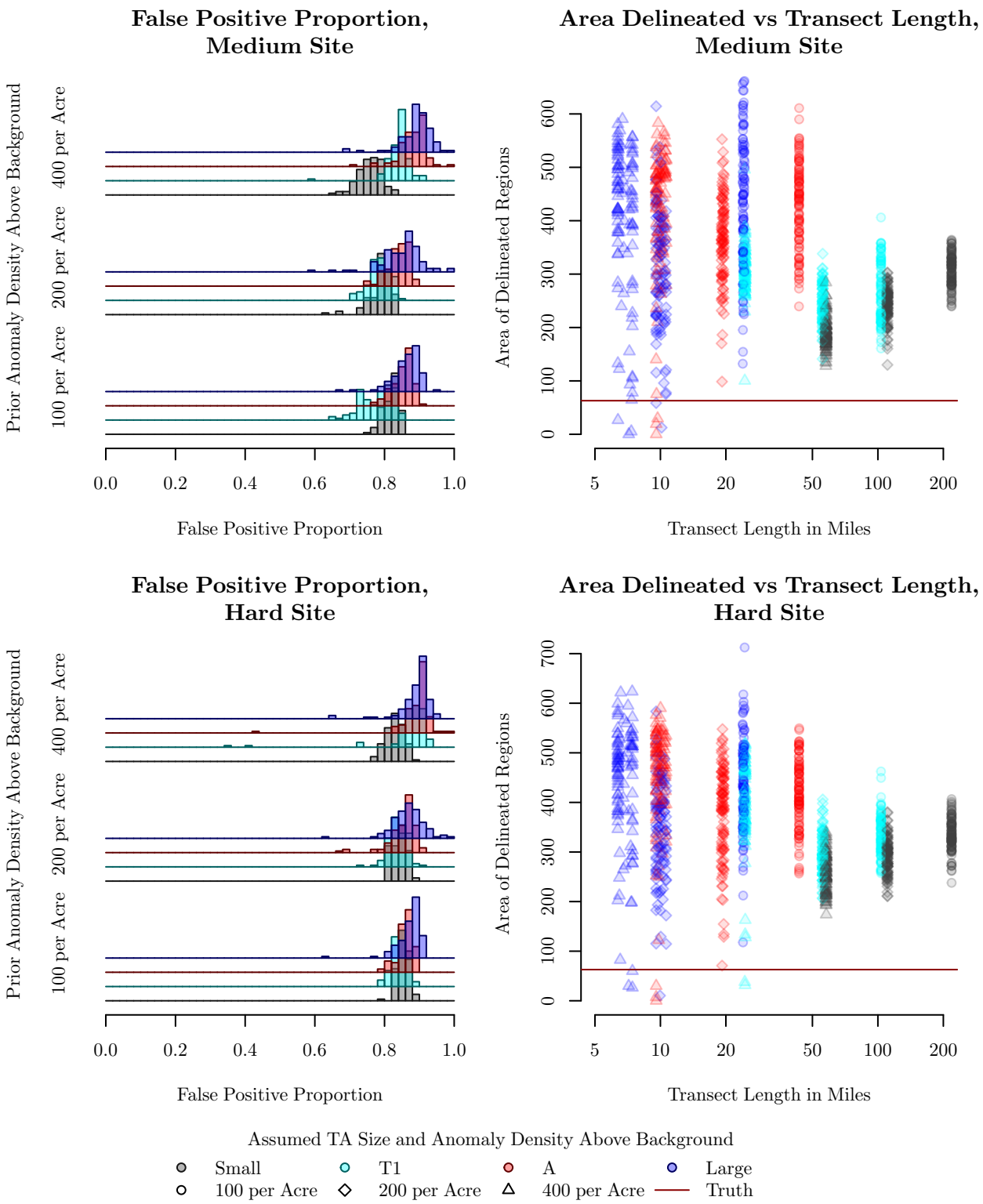


Figure 34: The *medium* and *hard* sites contain 62.9 acres of target areas. As seen at the *easy* site, most of the delineated area does not belong to a target area.

6.6 Sampling Effort

The amount of sampling effort is a very important consideration because a project team needs to balance the cost of sampling with the cost of remediating the possible target areas found. They certainly want to collect more data if it helps identify the target areas more accurately or precisely, but they may wish to avoid spending extra time and money on sampling if it does not save costs during the remediation. I use the false positive proportion, the total area delineated, and the total length of transects traversed to investigate the how the assumed target area size and assumed target area density relate to sampling efficiency.

At all three sites, the false positive proportion is around 0.8 to 0.9 for all sampling plans, meaning 80% to 90% of the delineated area does not belong to the true target areas (Figure 33, left, and Figure 34, left). The plots of total delineated area against distance traversed show that collecting more data from additional transects is associated with lower variability in the amount of area delineated (Figure 33, right, and Figure 34, right). However, the total area delineated does not approach the true area. Collecting more data does not result in a more efficient remediation.

7 Issues and Considerations Related to the VSP Analysis

Visual Sample Plan is an attractive option for analyzing UXO data because it generally does a good job of finding the true target areas and it uses standard geostatistical methods that would be familiar to users with experience working on environmental projects. However, questions remain regarding the choice of Kriging window size for a particular site, the applicability of the linear model and Kriging to anomaly data, and how to use the information gained from mapping and delineating the high-density regions.

7.1 Choosing a Good Window Size

Using an appropriate window size is crucial for obtaining an accurate anomaly density map. Typically, the window size is chosen after the data are collected, but this is problematic because different window sizes yield very different results for a given dataset. If a goal of the project is to map the site with a certain level of detail, it would make more sense to have the window size drive the sampling plan. The window diameter should be selected to achieve a desired resolution for the map, and then the transect spacing set to provide enough data for windows of the chosen size.

If the window size is selected after the sampling plan is created, some effort should be made to choose a window size suitable for the data rather than using a default size. One way to assess optimality is cross-validation, where subsets of observed anomaly densities are omitted while estimating covariance parameters and computing Kriging predictions, and then Kriging is used to predict the densities at the locations of the omitted observations. The mean squared error (MSE) for the omitted observations is used as an optimality criterion. Cross validation is repeated for several window sizes, and the window size with the lowest MSE is selected. Leave-one-out cross validation, where each window is omitted one at a time, is already available in KT3D. K-fold cross validation, where $\frac{1}{K}$ of the windows are randomly selected and omitted at once, generally gives less variability in the computed MSE than leave-one-out cross validation does (James et al. 2013, §5.1)

and could easily be implemented via KT3D's jackknife feature. An algorithm could be developed to adaptively choose additional window sizes to evaluate and create a smooth curve in a manner similar to how VSP's detection probability simulation chooses additional transect spacings.

7.2 Comments on Stationarity

Even though VSP's Kriging methods effectively detect the TOI items in my simulations, the appropriateness of the model and analysis should be considered. Ordinary Kriging assumes that the anomaly densities observed at a site result from a single, stationary process, so that the mean anomaly density is constant across the site. This is a curious assumption to make when it is believed that target areas are present and have higher density than other regions. I argue that this is a reasonable modeling decision, even though the use of ordinary Kriging is inappropriate. The alternative linear model approach is to use universal Kriging with polynomial or spline terms to model the spatial trends, but doing so requires making decisions about how complex the trend surface is allowed to be. Furthermore, a polynomial or spline surface will have a smoothing effect on the density surface, but some smoothing is already done when computing the moving average densities. The additional smoothing needlessly adds complexity to the analysis, so it may be acceptable to use ordinary Kriging if it produces a reasonable anomaly density map for delineation.

7.3 Other Analysis Methods for Spatial Point Data

Perhaps more troublesome than the stationarity assumption is the application of methods for a continuous response variable to the outcome of a point process. Most geostatistical methods are derived and studied under the assumption that the data come from a Gaussian random field, where the joint distribution of the observations is multivariate Normal. In a Gaussian random field, a numerical measurement can be made at any location of the site. This is different from the spatial point process that generates anomlies at random locations around a site.

When Kriging is done in VSP to map a site's spatial anomaly density, the data being used are the window densities, not the anomaly locations. Analysis methods for continuous response variables may incorrectly characterize the outcome of a point process. It could be better to use methods derived specifically for spatial point processes.

Kernel intensity estimation is a method of estimating a Poisson intensity function that can be employed to map anomaly density. A kernel function is centered at the location of each observed anomaly. The sum of the kernels produces a smooth map of the spatial point density. This technique does not require computing window densities, but a smoothing bandwidth must be chosen. Brooks and Marron (1991) discuss cross validation for selecting the bandwidth. Applying kernel estimation to data from transect sampling could be complicated by the fact that points between the transects are not observed; this issue would be worthy of further research.

For situations where a circular target area is being sought, and mapping the anomaly density is not necessary, Kulldorff (1997) proposes a spatial scan statistic that would be applicable. The scan statistic is a likelihood ratio statistic and is used to identify clusters in a point process that are not described by a baseline model. The baseline model has an intensity defined up to a multiplicative constant. When analyzing UXO data, it is widely assumed that background noise has a constant

density. A homogeneous Poisson process would be used for the baseline model, and target areas are identified by finding the cluster centers and diameters which maximize the likelihood.

7.4 A Risk Management Perspective

Once possible target areas are identified, it may not be clear how to use the information, especially if insufficient resources are available to remediate all potential target areas, if less area is delineated than expected, or if high background density makes the analysis difficult. In particularly tough cases, sampling and geostatistical mapping might not even be helpful or necessary to meet the project goals. A risk management study could be useful as an alternative to sampling or to augment information obtained through sampling.

Any historical information can be used to reduce uncertainty about the possible locations of UXO. Neptune and Company, Inc. (2008) demonstrate a Bayesian approach involving ballistic simulations. Historical data about firing locations and ordnance discovered by landowners in Helena Valley are used to develop a simulation of tanks firing rounds into the area where munitions were been found. The simulations lead to revised information about the firing points and the geographical extent of the UXO problem, including identification of a low-density region affected by ricochet. The end result is a spatial map of the risk of UXO encounters, based entirely on historical information and without requiring a new survey. Even for sites with less information available, such simulations could still be used to refine the conceptual site model, potentially saving time and money by reducing the area that must be sampled. This could be especially helpful when there is enough background noise present to make sampling ineffective at identifying target areas.

8 Conclusions

Visual Sample Plan provides tools to address nearly every sampling and data analysis need of an unexploded ordnance cleanup project, and can be used by people with little statistical training. Of particular interest during the remedial investigation phase are the geophysical mapping and target area delineation tools. These features compute the spatial anomaly density in search windows that move along the observed transects, use Kriging to map the anomaly density across the entire site, and threshold the map to delineate high-density regions that correspond to possible target areas and may contain unexploded ordnance. The successful implementation of these tools requires many decisions, assumptions, and pieces of prior information. My simulation study investigates the sensitivity of the delineation to the choice of window size and the prior information about the target area size and anomaly density, but these are just a few of many components.

8.1 Window Size Selection

No single window size is clearly best for all realizations of the sites I constructed. Choosing a window size requires a compromise between detecting a high proportion of the munitions items and wasting effort by remediating a large amount of area that contains little or no munitions debris. A real project should consider several window sizes, but a window close in size to the assumed target area size is a good starting point.

8.2 Prior Information About the Target Area Size and Anomaly Density

Obtaining accurate information about the sizes of possible target areas is very important. Sampling plans based on an assumed size that is too large can fail to detect the full extent of a smaller target area. Assumed TA sizes that are too small lead to excess sampling, unnecessarily increasing costs. If the target area size is used to inform the window size, as in the prior information experiment, an underestimate can result in the Kriging predictions being too sensitive to local variation in anomaly density. In this simulation, the best results occur when the sampling plan is based on the true size of the smallest target area at the site. The prior estimate of the target area anomaly density is less important. Therefore, efforts to gether prior information should focus on obtaining accurate information about the target area sizes.

8.3 Sampling Effort and Remediation Effort

When a reasonable window size is selected and accurate prior information about target area size is available, the methods used in my analysis result in all or nearly all of the munitions items at the simulated sites being contained within the identified high-density regions. Unfortunately, the delineation leads to an inefficient remediation. At all three sites, it is typical that roughly 80% of the delineated area is not part of a true target area. It would be intuitive to expect that collecting more data from more closely-spaced transects would lead to more detailed map of the high-density regions. This is not the case. Additional transect distance decreases the variability in the amount of area delineated, but does not improve the accuracy. Increased sampling effort does not translate into savings in the remediation phase. It would be better to collect as much prior information as possible so the delineation results can be managed and prioritized in the event that more high-density area is identified than can be cleaned up.

8.4 Future Investigations

Much remains to be discovered about how VSP can be used most effectively. Future simulations should use the conceptual models of real sites and could examine prior information about additional quantities such as the background density or distribution of items in the target areas, and true parameters of the site like the actual size and density of the target areas.

Other aspects of the analysis should be studied as well, in particular the optimal window size for one realization. Cross-validation could be an effective way to choose a window size after sampling. Future studies should also consider how to select a window size before sampling, and then create a sampling plan that fits the chosen window size. The window size affects the amount of detail possible in the anomaly density map, so it makes sense to set the window size first. Currently, there is little guidance available on how to create a sampling plan based on a given window size.

The method used to separate high-density regions from the background is another very important topic for further study. The decision rule used in this simulation is a naive attempt to do the delineation objectively, and it is probably the reason my analysis delineates so much excess area. Future work should seek a statistically-justified procedure where any point at the site is considered hazardous unless the anomaly density at that location is shown to be satisfactorily low according to criteria defined for the project.

A R Code Appendix

A.1 Simulations

A.1.1 Easy Site, `easy.r`

```
# Generate an easy site with two elliptical TAs
require(spatstat)
source('rfns/data_functions.r')
source('rfns/spatial_functions.r')

nreps <- 3000

bg.dens <- 100 / 43560 # 100 per acre, converted to square feet
fg.dens <- 200 / 43560 # 200 per acre above bg

# 952.375 acres. At 100 bg per acre, we expect 95237.5 bg anomalies.
sitewindow <- owin(poly = cbind(x = c(1564294, 1564495, 1556870, 1557126),
                                y = c(535421, 541130, 541085, 535576)))

# Generate a vector of random seeds and reseed each iteration
# so the whole set of reps doesn't need to be generated at once.
# Valid seeds are 32 bit signed integers.
set.seed(783614)
seeds <- sample(2^32-1, nreps)-2^31

# Loop to generate many
cat(sprintf('Simulating %d Easy Sites\n', nreps))
pb <- txtProgressBar(max = nreps, style = 3)
timing <- system.time(for(repl in 1:nreps){
  set.seed(seeds[repl])

  # Uniform background
  bg.anomalies <- rpoispp(lambda = bg.dens, win = sitewindow)
  marks(bg.anomalies) <- 0

  # Target Area 1
  fg1 <- rpoispp(lambda = gauss.elliptic, win = sitewindow, mu.x = 1558400, mu.y = 540000,
                 s.a = 800/(2*qnorm(0.995)), s.b = 1200/(2*qnorm(0.995)), r = pi/6, maxrate = fg.dens)
  marks(fg1) <- 1

  # Target Area 2
  fg2 <- rpoispp(lambda = gauss.elliptic, win = sitewindow, mu.x = 1562000, mu.y = 537000,
                 s.a = 2000/(2*qnorm(0.995)), s.b = 900/(2*qnorm(0.995)), r = 0, maxrate = fg.dens)
  marks(fg2) <- 2

  site <- superimpose(bg.anomalies, fg1, fg2)
  save(site, file = sprintf('datasets/easy/full/easy_full_bg%03d_fg%03d_rep%04d.RData',
                            bg.dens*43560, fg.dens*43560, repl))

  setTxtProgressBar(pb, repl)
})
close(pb)
print(timing)
```

A.1.2 Medium Site, `medium.r`

```
# Generate a realistic site with three TAs and some roads
require(spatstat)
source('rfns/data_functions.r')
source('rfns/spatial_functions.r')

nreps <- 3000


bg.dens <- 100 / 43560 # 100 per acre, converted to square feet
road.dens <- 100 / 43560
tank.dens <- 200 / 43560
art.dens <- 200 / 43560

# 952.375 acres. At 100 bg per acre, we expect 95237.5 bg anomalies.
sitewindow <- owin(poly = cbind(x = c(1564294, 1564495, 1556870, 1557126),
                                y = c(535421, 541130, 541085, 535576)))

roadwindow <- intersect.owin(
  dilation(psp(x0 = c(1559750, 1560000, 1560250, 1560700,
                      1560700, 1560000, 1558050),
               y0 = c(535421, 536400, 536750, 537000,
                      537000, 537850, 538500),
               x1 = c(1560000, 1560250, 1560700, 1564495,
                      1560000, 1558050, 1557550),
               y1 = c(536400, 536750, 537000, 538000,
                      537850, 538500, 538900),
               window = boundingbox(sitewindow)),
           25), sitewindow)

# Generate a vector of random seeds and reseed each iteration
# so the whole set of reps doesn't need to be generated at once.
# Valid seeds are 32 bit signed integers.
set.seed(46347)
seeds <- sample(2^32-1, nreps)-2^31

# Loop to generate many
cat(sprintf('Simulating %d Medium Sites\n', nreps))
pb <- txtProgressBar(max = nreps, style = 3)
timing <- system.time(for(repl in 1:nreps){
  set.seed(seeds[repl])

  # Uniform background
  bg.homog <- rpoispp(lambda = bg.dens, win = sitewindow)
  marks(bg.homog) <- 0

  # Uniform background
  bg.road <- rpoispp(lambda = road.dens, win = roadwindow)
  marks(bg.road) <- 1

  # Tank Area 1
  tank1 <- rpoispp(lambda = gauss.elliptic, win = sitewindow, mu.x = 1558000, mu.y = 540000,
                   s.a = 1000/(2*qnorm(0.995)), s.b = 600/(2*qnorm(0.995)), r = -pi/9, maxrate = tank.dens)
  marks(tank1) <- 3
```

```
  # Tank Area 2
  tank2 <- rpoispp(lambda = gauss.elliptic, win = sitewindow, mu.x = 1558300, mu.y = 537500,
                   s.a = 800/(2*qnorm(0.995)), s.b = 800/(2*qnorm(0.995)), r = 0, maxrate = tank.dens)
  marks(tank2) <- 4

  # Artillery Area
  art <- rpoispp(lambda = gauss.elliptic, win = sitewindow, mu.x = 1561200, mu.y = 539200,
                 s.a = 1500/(2*qnorm(0.995)), s.b = 1500/(2*qnorm(0.995)), r = 0, maxrate = art.dens)
  marks(art) <- 5

  site <- superimpose(bg.homog, bg.road, tank1, tank2, art)
  save(site, file = sprintf('datasets/medium/full/medium_full_bg%03d_ro%03d_t%03d_a%03d_rep%04d.RData',
                            bg.dens*43560, road.dens*43560, tank.dens*43560, art.dens*43560, repl))

  setTxtProgressBar(pb, repl)
})
close(pb)
print(timing)
```

A.1.3 Hard Site, `hard.r`

```
# Generate a realistic site with three TAs and some roads
require(spatstat)
source('rfns/data_functions.r')
source('rfns/spatial_functions.r')

nreps <- 3000


# Background has 2 clusters per acre so 50 anomalies per cluster
# gives 100 anomalies per acre
bg.kappa <- 2 / 43560
bg.scale <- 75
bg.mu <- 50

road.dens <- 100 / 43560
ranch.dens <- 100 / 43560
tank.dens <- 200 / 43560
art.dens <- 200 / 43560

# 952.375 acres. At 100 bg per acre, we expect 95237.5 bg anomalies.
sitewindow <- owin(poly = cbind(x = c(1564294, 1564495, 1556870, 1557126),
                                y = c(535421, 541130, 541085, 535576)))

roadwindow <- intersect.owin(
  dilation(psp(x0 = c(1559750, 1560000, 1560250, 1560700,
                      1560700, 1560000, 1558050),
               y0 = c(535421, 536400, 536750, 537000,
                      537000, 537850, 538500),
               x1 = c(1560000, 1560250, 1560700, 1564495,
                      1560000, 1558050, 1557550),
               y1 = c(536400, 536750, 537000, 538000,
```

```
                    537850, 538500, 538900),
                window = boundingbox(sitewindow)),
            25), sitewindow)

ranchwindow <- owin(c(1561300, 1562100), c(537900, 538700))

# Generate a vector of random seeds and reseed each iteration
# so the whole set of reps doesn't need to be generated at once.
# Valid seeds are 32 bit signed integers.
set.seed(23467)
seeds <- sample(2^32-1, nreps)-2^31

cat(sprintf('Simulating %d Hard Sites\n', nreps))
pb <- txtProgressBar(max = nreps, style = 3)
timing <- system.time(for(repl in 1:nreps){
  set.seed(seeds[repl])

  # Clustered background
  bg.clust <- rThomas(kappa = bg.kappa, scale = bg.scale, mu = bg.mu, win = sitewindow)
  marks(bg.clust) <- 0

  # Uniform background
  bg.road <- rpoispp(lambda = road.dens, win = roadwindow)
  marks(bg.road) <- 1

  # Uniform background
  bg.ranch <- rpoispp(lambda = ranch.dens, win = ranchwindow)
  marks(bg.ranch) <- 2

  # Tank Area 1
  tank1 <- rpoispp(lambda = gauss.elliptic, win = sitewindow, mu.x = 1558000, mu.y = 540000,
                   s.a = 1000/(2*qnorm(0.995)), s.b = 600/(2*qnorm(0.995)), r = -pi/9, maxrate = tank.dens)
  marks(tank1) <- 3

  # Tank Area 2
  tank2 <- rpoispp(lambda = gauss.elliptic, win = sitewindow, mu.x = 1558300, mu.y = 537500,
                   s.a = 800/(2*qnorm(0.995)), s.b = 800/(2*qnorm(0.995)), r = 0, maxrate = tank.dens)
  marks(tank2) <- 4

  # Artillery Area
  art <- rpoispp(lambda = gauss.elliptic, win = sitewindow, mu.x = 1561200, mu.y = 539200,
                 s.a = 1500/(2*qnorm(0.995)), s.b = 1500/(2*qnorm(0.995)), r = 0, maxrate = art.dens)
  marks(art) <- 5

  site <- superimpose(bg.clust, bg.road, bg.ranch, tank1, tank2, art)
  save(site, file = sprintf(
    'datasets/hard/full/hard_full_k%02d_s%03d_m%03d_ro%03d_ra%03d_t%03d_a%03d_rep%04d.RData',
    bg.kappa*43560, bg.scale, bg.mu, road.dens*43560, ranch.dens*43560,
    tank.dens*43560, art.dens*43560, repl))

  setTxtProgressBar(pb, repl)
})
close(pb)
print(timing)
```

A.2 Analysis

A.2.1 Window Size Experiment, `experiment1.r`

```
# North-South Transect sampling from the easy site,
# with different window sizes.
require(tcltk)
require(spatstat)
source('rfns/data_functions.r')
source('rfns/sampling_functions.r')
source('rfns/spatial_functions.r')

# Paths to command-line versions of GAMV and KT3D
gamv_exe <- 'gamv'
kt3d_exe <- 'kt3d'

# Paths to input/output
fulldir <- 'datasets/easy/full'
sampdir <- 'datasets/easy/sample'
outdir <- 'datasets/easy/exp1'
# Note: GAM/GAMV only support 40 character file paths.


## EXPERIMENT PARAMETERS

nreps <- 3000
nsamp <- 200 # 200 replicates take about 15 hours on my machine

# Sampling plan parameters, 0.99 prob of detecting the smaller TA
spacing <- 225
width <- 6
bg.dens <- 100
fg.dens <- 200

# Spacing is 225, smaller TA has minor axis diameter 800
window.sizes <- c(150, 228, 516, 798, 1500)
ncells <- length(window.sizes)
nobs <- nsamp * ncells

# True TAs and site
TA1 <- ellipse(800/2, 1200/2, c(1558400, 540000), pi/6)
TA2 <- ellipse(2000/2, 900/2, c(1562000, 537000), 0)
TAs <- union.owin(TA1, TA2)
sitewindow <- owin(poly = cbind(x = c(1564294, 1564495, 1556870, 1557126),
                                y = c(535421, 541130, 541085, 535576)))

# Corners of site and number of discretized rows and columns
corners <- vertices(Frame(sitewindow))
xmin <- min(corners$x) + window.sizes / 12
ymin <- min(corners$y) + window.sizes / 12
nx <- ceiling(6 * (max(corners$x) - min(corners$x)) / window.sizes)
ny <- ceiling(6 * (max(corners$y) - min(corners$y)) / window.sizes)


# Starting values for numerically estimating semivariogram parameters:
```

```
# There should not be a nugget because the simulation has no measurement
# error or microscale variation.
nug.start <- 0

# The number of anomlies in a window follows a Poisson distribution, and
# sites have little area occupied by TAs, so the expected number of
# background anomalies over the area squared (=density over area)
# is a natural starting point for the sill of the local density estimate.
sill.start <- bg.dens / (width*window.sizes/43560)

# The only locations that should be correlated are locations in the same TA,
# so set an initial range on the same order of magnitude as the TA sizes.
range.start <- 1000

# Basic starting point for power model:
slope.start <- 1
power.start <- 0.5


# SELECT THE SAMPLE
set.seed(37478)
seeds <- sample(2^32-1, nreps)-2^31
samp <- sample(nreps, nsamp)


## RESULT STORAGE

# Matrix to store all responses that are not vectors
results <- data.frame(expand.grid('win' = window.sizes,
                                  'Realization' = samp),
                      'length' = numeric(ncells),
                      'detect' = numeric(ncells),
                      'detect1' = numeric(ncells),
                      'detect2' = numeric(ncells),
                      'dens' = numeric(ncells),
                      'detectarea' = numeric(ncells),
                      'detectarea1' = numeric(ncells),
                      'detectarea2' = numeric(ncells),
                      'identarea' = numeric(ncells),
                      'identcount' = numeric(ncells))

# List of lists to store vectors of distances of false negatives to nearest
# delineated regions
ndist <- array(list(), dim = c(nsamp, length(window.sizes)),
               dimnames = list('Realization' = paste0('r', samp),
                               'win' = paste0('w', window.sizes)))

# List of lists to store vectors of areas of disjoint regions
areas <- array(list(), dim = c(nsamp, length(window.sizes)),
               dimnames = list('Realization' = paste0('r', samp),
                               'win' = paste0('w', window.sizes)))


# Loop for each replicate
pb <- tkProgressBar(max = nsamp+1, min = 1, initial = 1,
```

```
                    title = 'Sampling and Kriging',
                    label = 'Sampling and Kriging')
timing <- system.time(for(repl in samp){
  itr <- which(samp==repl)
  r <- (itr - 1) * ncells
  set.seed(seeds[repl])
  setTkProgressBar(pb, itr, label = paste0('Iteration ', which(samp==repl),
                                           ': Sampling rep ', repl))

  ## SAMPLING

  # Read the ground truth file
  load(file = sprintf('%s/easy_full_bg%03d_fg%03d_rep%04d.RData',
                      fulldir, bg.dens, fg.dens, repl))

  # Sample along the transects, starting at a random horizontal coordinate
  sample <- sample.transects.NS(site, width, spacing,
                                offset = runif(1, 0, spacing + width/2))

  # Save the sample
  filepath <- sprintf('%s/easy_sample_sp%04d_bg%03d_fg%03d_rep%04d',
                      sampdir, spacing, bg.dens, fg.dens, repl)
  write.anomaly(sample$anomaly, paste0(filepath, '.anomaly'))
  write.cog(sample$cog, paste0(filepath, '.cog'))

  # Loop for each window size
  for(w in 1:length(window.sizes)){
    setTkProgressBar(pb, itr+w/length(window.sizes),
                     label = paste0('Iteration ', itr, ': Analyzing rep ',
                                    repl, ' with window size ', window.sizes[w]))
    results$length[r+w] <- sample$length

    ## KRIGING

    # Evaluate local density in each window
    datfile <- sprintf('%s/rep%04d_w%04d.dat', outdir, repl, window.sizes[w])
    ldens <- windowed.density.NS(sample, window.sizes[w])
    write.geoeas(ldens, datfile, title = 'Data exported from R')

    # Create GAMV parameter file
    gpar <- sprintf('%s/rep%04d_w%04d_g.par', outdir, repl, window.sizes[w])
    gout <- sprintf('%s/rep%04d_w%04d_g.out', outdir, repl, window.sizes[w])
    cat(gamv_par(datfile, gout, window.sizes[w]), file = gpar)

    # Run GAMV to compute empirical semivariogram
    system2(gamv_exe, input = gpar, wait = TRUE)

    # Read semivariogram and discard lags that were not estimated
    svario <- read.table(gout, row.names = 1,
                         col.names = c('l', 'lag.dist', 'semivariogram',
                                       'n', 'tail.mean', 'head.mean'),
                         header = FALSE, skip = 3)
    svario <- svario[svario$n>0,]

    # Fit parametric semivariograms
```

```
params <- list('sphere' = optim(c(nug.start, sill.start[w], range.start),
                                sv.wss, lags = svario$lag.dist,
                                n = svario$n, ghat = svario$semivariogram,
                                model = sv.sphere, lower = c(0, 0.0001, 0),
                                upper = c(Inf, Inf, Inf),
                                method = 'L-BFGS-B'),
               'expon' = optim(c(nug.start, sill.start[w], range.start),
                               sv.wss, lags = svario$lag.dist,
                               n = svario$n, ghat = svario$semivariogram,
                               model = sv.expon, lower = c(0, 0.0001, 0),
                               upper = c(Inf, Inf, Inf),
                               method = 'L-BFGS-B'),
               'gauss' = optim(c(nug.start, sill.start[w], range.start),
                               sv.wss, lags = svario$lag.dist,
                               n = svario$n, ghat = svario$semivariogram,
                               model = sv.gauss, lower = c(0, 0.0001, 0),
                               upper = c(Inf, Inf, Inf),
                               method = 'L-BFGS-B'),
               'power' = optim(c(nug.start, slope.start, power.start),
                               sv.wss, lags = svario$lag.dist,
                               n = svario$n, ghat = svario$semivariogram,
                               model = sv.power, lower = c(0, 0.0001, 0),
                               upper = c(Inf, Inf, 2),
                               method = 'L-BFGS-B'))

# Find the model with the smallest sum of squares
# The indices match GSLIB's model type numbers
type <- order(sapply(params, function(x){return(x$value)}))[1]

# Create KT3D parameter file
kpar <- sprintf('%s/rep%04d_w%04d_k.par', outdir, repl, window.sizes[w])
kout <- sprintf('%s/rep%04d_w%04d_k.out', outdir, repl, window.sizes[w])
kdbg <- sprintf('%s/rep%04d_w%04d.dbg', outdir, repl, window.sizes[w])
cat(kt3d_par(datfile, kdbg, kout, nx[w], ny[w], xmin[w], ymin[w],
             window.sizes[w], params[[type]]$par[1],
             params[[type]]$par[2], params[[type]]$par[3], type),
    file = kpar)

# Run KT3D to do the kriging
# Note: Value of -999 indicates that the value that was not computed
system2(kt3d_exe, input = kpar, wait = TRUE)


## ANALYSIS

## Read and clean KT3D output
krige.out <- read.geoeas(kout)
krige.out[krige.out$Estimate == -999,] <- rep(NA, 2)
krige.out$EstimationVariance[krige.out$EstimationVariance < 0] <- 0
kest <- im(matrix(krige.out$Estimate, nrow = ny[w], byrow = TRUE),
           seq(xmin[w], length.out = nx[w], by = window.sizes[w]/6),
           seq(ymin[w], length.out = ny[w], by = window.sizes[w]/6),
           unitname = c('foot', 'feet'))
kvar <- im(matrix(krige.out$EstimationVariance, nrow = ny[w], byrow = TRUE),
           seq(xmin[w], length.out = nx[w], by = window.sizes[w]/6),
```

```
            seq(ymin[w], length.out = ny[w], by = window.sizes[w]/6),
            unitname = c('foot', 'feet'))

    # Get the delineated regions, if there are any
    highdens <- kest > bg.dens + qnorm(0.95) * sqrt(kvar)
    if(sum(highdens) > 0){
      identified <- connected(highdens, background = FALSE)
      areas[[itr, w]] <- sapply(levels(identified$v), function(x){
          return(area(Window(connected(identified==x,
                                       background = FALSE))[sitewindow]))
      })

      # Ignore regions with non-positive area
      ignore <- which(areas[[itr, w]] <= 0)
      areas[[itr, w]][ignore] <- NA
      results$identcount[r+w] <- sum(!is.na(areas[[itr, w]]))
      for(i in ignore){
        identified$v[identified$v==i] <- NA
      }
    }
    if(results$identcount[r+w] > 0){
      idboundary <- as.polygonal(Window(identified))[sitewindow]
      idpoints <- site
      Window(idpoints) <- idboundary
      missedpoints <- site
      Window(missedpoints) <- complement.owin(idboundary,
        frame = dilation(Frame(sitewindow), window.sizes[w]))

      results$detect[r+w] <- sum(marks(idpoints)>0) / sum(marks(site)>0)
      results$detect1[r+w] <- sum(marks(idpoints)==1) / sum(marks(site)==1)
      results$detect2[r+w] <- sum(marks(idpoints)==2) / sum(marks(site)==2)
      results$dens[r+w] <- sum(marks(idpoints)>0) / area(idpoints) * 43560
      a <- intersect.owin(Window(idpoints), TAs, fatal = FALSE)
      results$detectarea[r+w] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), TA1, fatal = FALSE)
      results$detectarea1[r+w] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), TA2, fatal = FALSE)
      results$detectarea2[r+w] <- ifelse(is.null(a), 0, area(a))
      results$identarea[r+w] <- area(idpoints)
      ndist[[itr, w]] <- nncross(missedpoints[marks(missedpoints)>0],
                                 edges(idboundary), what = 'dist')
    }
  }
})
setTkProgressBar(pb, nsamp+1, label = 'Done')
invisible(close(pb))
print(timing)

save(results, file = paste0('datasets/easy/results/easy_winresults_sp', spacing,
                            '_fg', fg.dens, '.RData'))
save(ndist, file = paste0('datasets/easy/results/easy_winndist_sp', spacing,
                          '_fg', fg.dens, '.RData'))
save(areas, file = paste0('datasets/easy/results/easy_winareas_sp', spacing,
                          '_fg', fg.dens, '.RData'))
```

A.2.2 Prior Information Experiment (*Easy* Site), `experiment2e.r`

```
# North-South Transect sampling from the easy site,
# with twelve different sampling plans.
require(tcltk)
require(spatstat)
source('rfns/data_functions.r')
source('rfns/sampling_functions.r')
source('rfns/spatial_functions.r')

# Paths to command-line versions of GAMV and KT3D
gamv_exe <- 'gamv'
kt3d_exe <- 'kt3d'

# Paths to input/output
fulldir <- 'datasets/easy/full'
sampdir <- 'datasets/easy/sample'
outdir <- 'datasets/easy/exp2'
# Note: GAM/GAMV only support 40 character file paths.


## EXPERIMENT PARAMETERS

nreps <- 3000
nsamp <- 100 # Takes about 25 hours

# Site parameters
width <- 6
bg.dens <- 100
fg.dens <- 200 # TRUE density for the simulation

# Sampling plan parameters (treatments)
ta.prior <- c('Small', 'TA1', 'TA2', 'Large')
fg.prior <- c(100, 200, 400)
spacings <- matrix(c(40, 125, 170, 270,
                     100, 225, 390, 565,
                     220, 465, 655, 935), ncol = 3)
window.sizes <- 0.9 * c(566, 800, 900, 1273)
ncells <- length(ta.prior) * length(fg.prior)
nobs <- nsamp * ncells

# True TAs and site
TA1 <- ellipse(800/2, 1200/2, c(1558400, 540000), pi/6)
TA2 <- ellipse(2000/2, 900/2, c(1562000, 537000), 0)
TAs <- union.owin(TA1, TA2)
sitewindow <- owin(poly = cbind(x = c(1564294, 1564495, 1556870, 1557126),
                                y = c(535421, 541130, 541085, 535576)))

# Corners of site and number of discretized rows and columns
corners <- vertices(Frame(sitewindow))
xmin <- min(corners$x) + window.sizes / 12
ymin <- min(corners$y) + window.sizes / 12
nx <- ceiling(6 * (max(corners$x) - min(corners$x)) / window.sizes)
ny <- ceiling(6 * (max(corners$y) - min(corners$y)) / window.sizes)
```

```
# Starting values for numerically estimating semivariogram parameters:

# There should not be a nugget because the simulation has no measurement
# error or microscale variation.
nug.start <- 0

# The number of anomlies in a window follows a Poisson distribution, and
# sites have little area occupied by TAs, so the expected number of
# background anomalies over the area squared (=density over area)
# is a natural starting point for the sill of the local density estimate.
sill.start <- bg.dens / (width*window.sizes/43560)

# The only locations that should be correlated are locations in the same TA,
# so set an initial range on the same order of magnitude as the TA sizes.
range.start <- 1000

# Basic starting point for power model:
slope.start <- 1
power.start <- 0.5


# SELECT THE SAMPLE
set.seed(87235)
seeds <- sample(2^32-1, nreps)-2^31
samp <- sample(nreps, nsamp)


## RESULT STORAGE

# Matrix to store all responses that are not vectors
results2e <- data.frame(expand.grid('Target' = ta.prior,
                                    'fg' = fg.prior,
                                    'Realization' = samp),
                        'length' = numeric(ncells),
                        'detect' = numeric(ncells),
                        'detect1' = numeric(ncells),
                        'detect2' = numeric(ncells),
                        'dens' = numeric(ncells),
                        'detectarea' = numeric(ncells),
                        'detectarea1' = numeric(ncells),
                        'detectarea2' = numeric(ncells),
                        'identarea' = numeric(ncells),
                        'identcount' = numeric(ncells))

# List of lists to store vectors of distances of false negatives to nearest
# delineated regions
ndist2e <- array(list(), dim = c(nsamp, length(ta.prior), length(fg.prior)),
                 dimnames = list('Realization' = paste0('r', samp),
                                 'Target' = paste0('Target', ta.prior),
                                 'fg' = paste0('fg', fg.prior)))

# List of lists to store vectors of areas of disjoint regions
areas2e <- array(list(), dim = c(nsamp, length(ta.prior), length(fg.prior)),
                 dimnames = list('Realization' = paste0('r', samp),
                                 'Target' = paste0('Target', ta.prior),
```

```
                                'fg' = paste0('fg', fg.prior)))


# Loop for each realization
pb <- tkProgressBar(max = nsamp+1, min = 1, initial = 1,
                    title = 'Sampling and Kriging',
                    label = 'Sampling and Kriging')
timing <- system.time(for(itr in seq_along(samp)){
  r <- (itr-1) * ncells
  repl <- results2e$Realization[r+1]
  set.seed(seeds[repl])
  setTkProgressBar(pb, r, label = paste0('Iteration ', itr,
                                         ': Loading rep ', repl))

  # Read the ground truth file
  load(file = sprintf('%s/easy_full_bg%03d_fg%03d_rep%04d.RData',
                      fulldir, bg.dens, fg.dens, repl))

  # Loop for each treatment combination
  for(trt in seq_len(ncells)){
    ta <- which(ta.prior==results2e$Target[r+trt])
    fg <- which(fg.prior==results2e$fg[r+trt])
    setTkProgressBar(pb, itr+trt/ncells,
                     label = paste0('Iteration ', itr,
                                    ': Analyzing rep ', repl,
                                    ' with spacing ', spacings[ta, fg]))


    ## SAMPLING

    # Sample along the transects, starting at a random horzontal coordinate
    sample <- sample.transects.NS(site, width, spacings[ta, fg],
                                  offset = runif(1, 0, spacings[ta, fg] + width/2))

    # Save the sample
    filepath <- sprintf('%s/easy_sample_t%s_p%03d_bg%03d_fg%03d_rep%04d',
                        sampdir, ta.prior[ta], fg.prior[fg], bg.dens, fg.dens, repl)
    write.anomaly(sample$anomaly, paste0(filepath, '.anomaly'))
    write.cog(sample$cog, paste0(filepath, '.cog'))
    results2e$length[r+trt] <- sample$length


    ## KRIGING

    # Evaluate local density in each window
    datfile <- sprintf('%s/rep%04d_s%04d.dat', outdir, repl, spacings[ta, fg])
    ldens <- windowed.density.NS(sample, window.sizes[ta])
    write.geoeas(ldens, datfile, title = 'Data exported from R')

    # Create GAMV parameter file
    gpar <- sprintf('%s/rep%04d_s%04d_g.par', outdir, repl, spacings[ta, fg])
    gout <- sprintf('%s/rep%04d_s%04d_g.out', outdir, repl, spacings[ta, fg])
    cat(gamv_par(datfile, gout, window.sizes[ta]), file = gpar)

    # Run GAMV to compute empirical semivariogram
```

```
system2(gamv_exe, input = gpar, wait = TRUE)

# Read semivariogram and discard lags that were not estimated
svario <- read.table(gout, row.names = 1,
                     col.names = c('l', 'lag.dist', 'semivariogram',
                                   'n', 'tail.mean', 'head.mean'),
                     header = FALSE, skip = 3)
svario <- svario[svario$n>0,]

# Fit parametric semivariograms
params <- list('sphere' = optim(c(nug.start, sill.start[ta], range.start),
                                sv.wss, lags = svario$lag.dist,
                                n = svario$n, ghat = svario$semivariogram,
                                model = sv.sphere, lower = c(0, 0.0001, 0),
                                upper = c(Inf, Inf, Inf),
                                method = 'L-BFGS-B'),
               'expon' = optim(c(nug.start, sill.start[ta], range.start),
                               sv.wss, lags = svario$lag.dist,
                               n = svario$n, ghat = svario$semivariogram,
                               model = sv.expon, lower = c(0, 0.0001, 0),
                               upper = c(Inf, Inf, Inf),
                               method = 'L-BFGS-B'),
               'gauss' = optim(c(nug.start, sill.start[ta], range.start),
                               sv.wss, lags = svario$lag.dist,
                               n = svario$n, ghat = svario$semivariogram,
                               model = sv.gauss, lower = c(0, 0.0001, 0),
                               upper = c(Inf, Inf, Inf),
                               method = 'L-BFGS-B'),
               'power' = optim(c(nug.start, slope.start, power.start),
                               sv.wss, lags = svario$lag.dist,
                               n = svario$n, ghat = svario$semivariogram,
                               model = sv.power, lower = c(0, 0.0001, 0),
                               upper = c(Inf, Inf, 2),
                               method = 'L-BFGS-B'))

# Find the model with the smallest sum of squares
# The indices match GSLIB's model type numbers
type <- order(sapply(params, function(x){return(x$value)}))[1]

# Create KT3D parameter file
kpar <- sprintf('%s/rep%04d_s%04d_k.par', outdir, repl, spacings[ta, fg])
kout <- sprintf('%s/rep%04d_s%04d_k.out', outdir, repl, spacings[ta, fg])
kdbg <- sprintf('%s/rep%04d_s%04d.dbg', outdir, repl, spacings[ta, fg])
cat(kt3d_par(datfile, kdbg, kout, nx[ta], ny[ta], xmin[ta], ymin[ta],
             window.sizes[ta], params[[type]]$par[1],
             params[[type]]$par[2], params[[type]]$par[3], type),
    file = kpar)

# Run KT3D to do the kriging
# Note: Value of -999 indicates that the value that was not computed
system2(kt3d_exe, input = kpar, wait = TRUE)


## ANALYSIS
```

```
    ## Read and clean KT3D output
    krige.out <- read.geoeas(kout)
    krige.out[krige.out$Estimate == -999,] <- rep(NA, 2)
    kest <- im(matrix(krige.out$Estimate, nrow = ny[ta], byrow = TRUE),
               seq(xmin[ta], length.out = nx[ta], by = window.sizes[ta]/6),
               seq(ymin[ta], length.out = ny[ta], by = window.sizes[ta]/6),
               unitname = c('foot', 'feet'))
    kvar <- im(matrix(krige.out$EstimationVariance, nrow = ny[ta], byrow = TRUE),
               seq(xmin[ta], length.out = nx[ta], by = window.sizes[ta]/6),
               seq(ymin[ta], length.out = ny[ta], by = window.sizes[ta]/6),
               unitname = c('foot', 'feet'))

    # Get the delineated regions, if there are any
    highdens <- kest > bg.dens + qnorm(0.95) * sqrt(kvar)
    if(sum(highdens) > 0){
      identified <- connected(highdens, background = FALSE)
      areas2e[[itr, ta, fg]] <- sapply(levels(identified$v), function(x){
          return(area(Window(connected(identified==x,
                                       background = FALSE))[sitewindow]))
      })

      # Ignore regions less than 3 acres
      ignore <- which(areas2e[[itr, ta, fg]] < 3*43560)
      areas2e[[itr, ta, fg]][ignore] <- NA
      results2e$identcount[r+trt] <- sum(!is.na(areas2e[[itr, ta, fg]]))
      for(i in ignore){
        identified$v[identified$v==i] <- NA
      }
    }
    if(results2e$identcount[r+trt] > 0){
      idboundary <- as.polygonal(Window(identified))[sitewindow]
      idpoints <- site
      Window(idpoints) <- idboundary
      missedpoints <- site
      Window(missedpoints) <- complement.owin(idboundary,
        frame = dilation(Frame(sitewindow), window.sizes[ta]))

      results2e$detect[r+trt] <- sum(marks(idpoints)>0) / sum(marks(site)>0)
      results2e$detect1[r+trt] <- sum(marks(idpoints)==1) / sum(marks(site)==1)
      results2e$detect2[r+trt] <- sum(marks(idpoints)==2) / sum(marks(site)==2)
      results2e$dens[r+trt] <- sum(marks(idpoints)>0) / area(idpoints) * 43560
      a <- intersect.owin(Window(idpoints), TAs, fatal = FALSE)
      results2e$detectarea[r+trt] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), TA1, fatal = FALSE)
      results2e$detectarea1[r+trt] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), TA2, fatal = FALSE)
      results2e$detectarea2[r+trt] <- ifelse(is.null(a), 0, area(a))
      results2e$identarea[r+trt] <- area(idpoints)
      ndist2e[[itr, ta, fg]] <- nncross(missedpoints[marks(missedpoints)>0],
                                        edges(idboundary), what = 'dist')
    }
  }
})
setTkProgressBar(pb, nsamp+1, label = 'Done')
invisible(close(pb))
```

```
print(timing)

save(results2e, file = 'datasets/easy/results/easy_exp2results.RData')
save(ndist2e, file = 'datasets/easy/results/easy_exp2ndist.RData')
save(areas2e, file = 'datasets/easy/results/easy_exp2areas.RData')
```

A.2.3 Prior Information Experiment (*Medium* Site), `experiment2m.r`

```
# North-South Transect sampling from the medium site,
# with twelve different sampling plans.
require(tcltk)
require(spatstat)
source('rfns/data_functions.r')
source('rfns/sampling_functions.r')
source('rfns/spatial_functions.r')

# Paths to command-line versions of GAMV and KT3D
gamv_exe <- 'gamv'
kt3d_exe <- 'kt3d'

# Paths to input/output
fulldir <- 'datasets/medium/full'
sampdir <- 'datasets/medium/sample'
outdir <- 'datasets/medium/exp2'


## EXPERIMENT PARAMETERS

nreps <- 3000
nsamp <- 100 # Takes about 60 hours

# Site parameters
width <- 6
bg.dens <- 100
r.dens <- 100
t.dens <- 200
a.dens <- 200

# Sampling plan parameters (treatments)
ta.prior <- c('Small', 'T1', 'A', 'Large')
fg.prior <- c(100, 200, 400)
spacings <- matrix(c(30,  70, 175,  320,
                     65, 135, 400,  780,
                    130, 315, 785, 1145), ncol = 3)
window.sizes <- 0.9 * c(424, 600, 1500, 2121)
ncells <- length(ta.prior) * length(fg.prior)
nobs <- nsamp * ncells

# True TAs and site
T1 <- ellipse(1000/2, 600/2, c(1558000, 540000), -pi/9)
T2 <- disc(800/2, c(1558300, 537500))
A  <- disc(1500/2, c(1561200, 539200))
TAs <- union.owin(T1, T2, A)
```

```
sitewindow <- owin(poly = cbind(x = c(1564294, 1564495, 1556870, 1557126),
                                y = c(535421, 541130, 541085, 535576)))
roadwindow <- intersect.owin(
  dilation(psp(x0 = c(1559750, 1560000, 1560250, 1560700,
                      1560700, 1560000, 1558050),
               y0 = c(535421, 536400, 536750, 537000,
                      537000, 537850, 538500),
               x1 = c(1560000, 1560250, 1560700, 1564495,
                      1560000, 1558050, 1557550),
               y1 = c(536400, 536750, 537000, 538000,
                      537850, 538500, 538900),
               window = boundingbox(sitewindow)),
           25), sitewindow)

# Corners of site and number of discretized rows and columns
corners <- vertices(Frame(sitewindow))
xmin <- min(corners$x) + window.sizes / 12
ymin <- min(corners$y) + window.sizes / 12
nx <- ceiling(6 * (max(corners$x) - min(corners$x)) / window.sizes)
ny <- ceiling(6 * (max(corners$y) - min(corners$y)) / window.sizes)


# Starting values for numerically estimating semivariogram parameters:

# There should not be a nugget because the simulation has no measurement
# error or microscale variation.
nug.start <- 0

# The number of anomlies in a window follows a Poisson distribution, and
# sites have little area occupied by TAs, so the expected number of
# background anomalies over the area squared (=density over area)
# is a natural starting point for the sill of the local density estimate.
sill.start <- bg.dens / (width*window.sizes/43560)

# The only locations that should be correlated are locations in the same TA,
# so set an initial range on the same order of magnitude as the TA sizes.
range.start <- 1000

# Basic starting point for power model:
slope.start <- 1
power.start <- 0.5


# SELECT THE SAMPLE
set.seed(73578)
seeds <- sample(2^32-1, nreps)-2^31
samp <- sample(nreps, nsamp)


## RESULT STORAGE

# Matrix to store all responses that are not vectors
results2m <- data.frame(expand.grid('Target' = ta.prior,
                                    'fg' = fg.prior,
                                    'Realization' = samp),
```

```
                        'length' = numeric(ncells),
                        'detect' = numeric(ncells),
                        'detect1' = numeric(ncells),
                        'detect3' = numeric(ncells),
                        'detect4' = numeric(ncells),
                        'detect5' = numeric(ncells),
                        'dens' = numeric(ncells),
                        'detectarea' = numeric(ncells),
                        'detectarea1' = numeric(ncells),
                        'detectarea3' = numeric(ncells),
                        'detectarea4' = numeric(ncells),
                        'detectarea5' = numeric(ncells),
                        'identarea' = numeric(ncells),
                        'identcount' = numeric(ncells))

# List of lists to store vectors of distances of false negatives to nearest
# delineated regions
ndist2m <- array(list(), dim = c(nsamp, length(ta.prior), length(fg.prior)),
                 dimnames = list('Realization' = paste0('r', samp),
                                 'Target' = paste0('Target', ta.prior),
                                 'fg' = paste0('fg', fg.prior)))

# List of lists to store vectors of areas of disjoint regions
areas2m <- array(list(), dim = c(nsamp, length(ta.prior), length(fg.prior)),
                 dimnames = list('Realization' = paste0('r', samp),
                                 'Target' = paste0('Target', ta.prior),
                                 'fg' = paste0('fg', fg.prior)))


# Loop for each realization
pb <- TkProgressBar(max = nsamp+1, min = 1, initial = 1,
                    title = 'Sampling and Kriging',
                    label = 'Sampling and Kriging')
timing <- system.time(for(itr in seq_along(samp)){
  r <- (itr-1) * ncells
  repl <- results2m$Realization[r+1]
  set.seed(seeds[repl])
  setTkProgressBar(pb, r, label = paste0('Iteration ', itr,
                                         ': Loading rep ', repl))

  # Read the ground truth file
  load(file = sprintf('%s/medium_full_bg%03d_ro%03d_t%03d_a%03d_rep%04d.RData',
                      fulldir, bg.dens, r.dens, t.dens, a.dens, repl))

  # Loop for each treatment combination
  for(trt in seq_len(ncells)){
    ta <- which(ta.prior==results2m$Target[r+trt])
    fg <- which(fg.prior==results2m$fg[r+trt])
    setTkProgressBar(pb, itr+trt/ncells,
                     label = paste0('Iteration ', itr,
                                    ': Analyzing rep ', repl,
                                    ' with spacing ', spacings[ta, fg]))


    ## SAMPLING
```

```
# Sample along the transects, starting at a random horzontal coordinate
sample <- sample.transects.NS(site, width, spacings[ta, fg],
                              offset = runif(1, 0, spacings[ta, fg] + width/2))

# Save the sample
filepath <- sprintf('%s/medium_sample_t%s_p%03d_bg%03d_rep%04d',
                    sampdir, ta.prior[ta], fg.prior[fg], bg.dens, repl)
write.anomaly(sample$anomaly, paste0(filepath, '.anomaly'))
write.cog(sample$cog, paste0(filepath, '.cog'))
results2m$length[r+trt] <- sample$length


## KRIGING

# Evaluate local density in each window
datfile <- sprintf('%s/rep%04d_s%04d.dat', outdir, repl, spacings[ta, fg])
ldens <- windowed.density.NS(sample, window.sizes[ta])
write.geoeas(ldens, datfile, title = 'Data exported from R')

# Create GAMV parameter file
gpar <- sprintf('%s/rep%04d_s%04d_g.par', outdir, repl, spacings[ta, fg])
gout <- sprintf('%s/rep%04d_s%04d_g.out', outdir, repl, spacings[ta, fg])
cat(gamv_par(datfile, gout, window.sizes[ta]), file = gpar)

# Run GAMV to compute empirical semivariogram
system2(gamv_exe, input = gpar, wait = TRUE)

# Read semivariogram and discard lags that were not estimated
svario <- read.table(gout, row.names = 1,
                     col.names = c('l', 'lag.dist', 'semivariogram',
                                   'n', 'tail.mean', 'head.mean'),
                     header = FALSE, skip = 3)
svario <- svario[svario$n>0,]

# Fit parametric semivariograms
params <- list('sphere' = optim(c(nug.start, sill.start[ta], range.start),
                                sv.wss, lags = svario$lag.dist,
                                n = svario$n, ghat = svario$semivariogram,
                                model = sv.sphere, lower = c(0, 0.0001, 0),
                                upper = c(Inf, Inf, Inf),
                                method = 'L-BFGS-B'),
               'expon' = optim(c(nug.start, sill.start[ta], range.start),
                                sv.wss, lags = svario$lag.dist,
                                n = svario$n, ghat = svario$semivariogram,
                                model = sv.expon, lower = c(0, 0.0001, 0),
                                upper = c(Inf, Inf, Inf),
                                method = 'L-BFGS-B'),
               'gauss' = optim(c(nug.start, sill.start[ta], range.start),
                                sv.wss, lags = svario$lag.dist,
                                n = svario$n, ghat = svario$semivariogram,
                                model = sv.gauss, lower = c(0, 0.0001, 0),
                                upper = c(Inf, Inf, Inf),
                                method = 'L-BFGS-B'),
               'power' = optim(c(nug.start, slope.start, power.start),
                                sv.wss, lags = svario$lag.dist,
```

```
                                n = svario$n, ghat = svario$semivariogram,
                                model = sv.power, lower = c(0, 0.0001, 0),
                                upper = c(Inf, Inf, 2),
                                method = 'L-BFGS-B'))

# Find the model with the smallest sum of squares
# The indices match GSLIB's model type numbers
type <- order(sapply(params, function(x){return(x$value)}))[1]

# Create KT3D parameter file
kpar <- sprintf('%s/rep%04d_s%04d_k.par', outdir, repl, spacings[ta, fg])
kout <- sprintf('%s/rep%04d_s%04d_k.out', outdir, repl, spacings[ta, fg])
kdbg <- sprintf('%s/rep%04d_s%04d.dbg', outdir, repl, spacings[ta, fg])
cat(kt3d_par(datfile, kdbg, kout, nx[ta], ny[ta], xmin[ta], ymin[ta],
             window.sizes[ta], params[[type]]$par[1],
             params[[type]]$par[2], params[[type]]$par[3], type),
    file = kpar)

# Run KT3D to do the kriging
# Note: Value of -999 indicates that the value that was not computed
system2(kt3d_exe, input = kpar, wait = TRUE)


## ANALYSIS

## Read and clean KT3D output
krige.out <- read.geoeas(kout)
krige.out[krige.out$Estimate == -999,] <- rep(NA, 2)
krige.out$EstimationVariance[krige.out$EstimationVariance < 0] <- 0
kest <- im(matrix(krige.out$Estimate, nrow = ny[ta], byrow = TRUE),
           seq(xmin[ta], length.out = nx[ta], by = window.sizes[ta]/6),
           seq(ymin[ta], length.out = ny[ta], by = window.sizes[ta]/6),
           unitname = c('foot', 'feet'))
kvar <- im(matrix(krige.out$EstimationVariance, nrow = ny[ta], byrow = TRUE),
           seq(xmin[ta], length.out = nx[ta], by = window.sizes[ta]/6),
           seq(ymin[ta], length.out = ny[ta], by = window.sizes[ta]/6),
           unitname = c('foot', 'feet'))

# Get the delineated regions, if there are any
highdens <- kest > bg.dens + qnorm(0.95) * sqrt(kvar)
if(sum(highdens) > 0){
  identified <- connected(highdens, background = FALSE)
  areas2m[[itr, ta, fg]] <- sapply(levels(identified$v), function(x){
      return(area(Window(connected(identified==x,
                                   background = FALSE))[sitewindow]))
  })

  # Ignore regions less than 3 acres
  ignore <- which(areas2m[[itr, ta, fg]] < 3*43560)
  areas2m[[itr, ta, fg]][ignore] <- NA
  results2m$identcount[r+trt] <- sum(!is.na(areas2m[[itr, ta, fg]]))
  for(i in ignore){
    identified$v[identified$v==i] <- NA
  }
}
```

```
    if(results2m$identcount[r+trt] > 0){
      idboundary <- as.polygonal(Window(identified))[sitewindow]
      idpoints <- site
      Window(idpoints) <- idboundary
      missedpoints <- site
      Window(missedpoints) <- complement.owin(idboundary,
        frame = dilation(Frame(sitewindow), window.sizes[ta]))

      results2m$detect[r+trt] <- sum(marks(idpoints)>2) / sum(marks(site)>2)
      results2m$detect1[r+trt] <- sum(marks(idpoints)==1) / sum(marks(site)==1)
      results2m$detect3[r+trt] <- sum(marks(idpoints)==3) / sum(marks(site)==3)
      results2m$detect4[r+trt] <- sum(marks(idpoints)==4) / sum(marks(site)==4)
      results2m$detect5[r+trt] <- sum(marks(idpoints)==5) / sum(marks(site)==5)
      results2m$dens[r+trt] <- sum(marks(idpoints)>0) / area(idpoints) * 43560
      a <- intersect.owin(Window(idpoints), TAs, fatal = FALSE)
      results2m$detectarea[r+trt] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), roadwindow, fatal = FALSE)
      results2m$detectarea1[r+trt] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), T1, fatal = FALSE)
      results2m$detectarea3[r+trt] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), T2, fatal = FALSE)
      results2m$detectarea4[r+trt] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), A, fatal = FALSE)
      results2m$detectarea5[r+trt] <- ifelse(is.null(a), 0, area(a))
      results2m$identarea[r+trt] <- area(idpoints)
      ndist2m[[itr, ta, fg]] <- nncross(missedpoints[marks(missedpoints)>0],
                                        edges(idboundary), what = 'dist')
    }
  }
})
setTkProgressBar(pb, nsamp+1, label = 'Done')
invisible(close(pb))
print(timing)

save(results2m, file = 'datasets/medium/results/medium_exp2results.RData')
save(ndist2m, file = 'datasets/medium/results/medium_exp2ndist.RData')
save(areas2m, file = 'datasets/medium/results/medium_exp2areas.RData')
```

A.2.4 Prior Information Experiment (*Hard* Site), `experiment2h.r`

```
# North-South Transect sampling from the hard site,
# with twelve different sampling plans.
require(tcltk)
require(spatstat)
source('rfns/data_functions.r')
source('rfns/sampling_functions.r')
source('rfns/spatial_functions.r')

# Paths to command-line versions of GAMV and KT3D
gamv_exe <- 'gamv'
kt3d_exe <- 'kt3d'

# Paths to input/output
```

```
fulldir <- 'datasets/hard/full'
sampdir <- 'datasets/hard/sample'
outdir <- 'datasets/hard/exp2'


## EXPERIMENT PARAMETERS

nreps <- 3000
nsamp <- 100 # Takes about 60 hours

# Site parameters
width <- 6
bg.mu <- 50
bg.kappa <- 2
bg.dens <- bg.mu * bg.kappa
bg.scale <- 75
r.dens <- 100
ra.dens <- 100
t.dens <- 200
a.dens <- 200

# Sampling plan parameters (treatments)
ta.prior <- c('Small', 'T1', 'A', 'Large')
fg.prior <- c(100, 200, 400)
spacings <- matrix(c(30,  70, 175,  320,
                     65, 135, 400,  780,
                    130, 315, 785, 1145), ncol = 3)
window.sizes <- 0.9 * c(424, 600, 1500, 2121)
ncells <- length(ta.prior) * length(fg.prior)
nobs <- nsamp * ncells

# True TAs and site
T1 <- ellipse(1000/2, 600/2, c(1558000, 540000), -pi/9)
T2 <- disc(800/2, c(1558300, 537500))
A  <- disc(1500/2, c(1561200, 539200))
TAs <- union.owin(T1, T2, A)
sitewindow <- owin(poly = cbind(x = c(1564294, 1564495, 1556870, 1557126),
                                y = c(535421, 541130, 541085, 535576)))
roadwindow <- intersect.owin(
  dilation(psp(x0 = c(1559750, 1560000, 1560250, 1560700,
                      1560700, 1560000, 1558050),
               y0 = c(535421, 536400, 536750, 537000,
                      537000, 537850, 538500),
               x1 = c(1560000, 1560250, 1560700, 1564495,
                      1560000, 1558050, 1557550),
               y1 = c(536400, 536750, 537000, 538000,
                      537850, 538500, 538900),
               window = boundingbox(sitewindow)),
           25), sitewindow)
ranchwindow <- owin(c(1561300, 1562100), c(537900, 538700))

# Corners of site and number of discretized rows and columns
corners <- vertices(Frame(sitewindow))
xmin <- min(corners$x) + window.sizes / 12
ymin <- min(corners$y) + window.sizes / 12
```

```
nx <- ceiling(6 * (max(corners$x) - min(corners$x)) / window.sizes)
ny <- ceiling(6 * (max(corners$y) - min(corners$y)) / window.sizes)


# Starting values for numerically estimating semivariogram parameters:

# There should not be a nugget because the simulation has no measurement
# error or microscale variation.
nug.start <- 0

# The number of anomlies in a window follows a Poisson distribution, and
# sites have little area occupied by TAs, so the expected number of
# background anomalies over the area squared (=density over area)
# is a natural starting point for the sill of the local density estimate.
sill.start <- bg.dens / (width*window.sizes/43560)

# The only locations that should be correlated are locations in the same TA,
# so set an initial range on the same order of magnitude as the TA sizes.
range.start <- 1000

# Basic starting point for power model:
slope.start <- 1
power.start <- 0.5


# SELECT THE SAMPLE
set.seed(67623)
seeds <- sample(2^32-1, nreps)-2^31
samp <- sample(nreps, nsamp)


## RESULT STORAGE

# Matrix to store all responses that are not vectors
results2h <- data.frame(expand.grid('Target' = ta.prior,
                                    'fg' = fg.prior,
                                    'Realization' = samp),
                        'length' = numeric(ncells),
                        'detect' = numeric(ncells),
                        'detect1' = numeric(ncells),
                        'detect2' = numeric(ncells),
                        'detect3' = numeric(ncells),
                        'detect4' = numeric(ncells),
                        'detect5' = numeric(ncells),
                        'dens' = numeric(ncells),
                        'detectarea' = numeric(ncells),
                        'detectarea1' = numeric(ncells),
                        'detectarea2' = numeric(ncells),
                        'detectarea3' = numeric(ncells),
                        'detectarea4' = numeric(ncells),
                        'detectarea5' = numeric(ncells),
                        'identarea' = numeric(ncells),
                        'identcount' = numeric(ncells))

# List of lists to store vectors of distances of false negatives to nearest
```

```
# delineated regions
ndist2h <- array(list(), dim = c(nsamp, length(ta.prior), length(fg.prior)),
                 dimnames = list('Realization' = paste0('r', samp),
                                 'Target' = paste0('Target', ta.prior),
                                 'fg' = paste0('fg', fg.prior)))

# List of lists to store vectors of areas of disjoint regions
areas2h <- array(list(), dim = c(nsamp, length(ta.prior), length(fg.prior)),
                 dimnames = list('Realization' = paste0('r', samp),
                                 'Target' = paste0('Target', ta.prior),
                                 'fg' = paste0('fg', fg.prior)))


# Loop for each realization
pb <- TkProgressBar(max = nsamp+1, min = 1, initial = 1,
                    title = 'Sampling and Kriging',
                    label = 'Sampling and Kriging')
timing <- system.time(for(itr in seq_along(samp)){
  r <- (itr-1) * ncells
  repl <- results2h$Realization[r+1]
  set.seed(seeds[repl])
  setTkProgressBar(pb, r, label = paste0('Iteration ', itr,
                                         ': Loading rep ', repl))

  # Read the ground truth file
  load(file = sprintf('%s/hard_full_k%02d_s%03d_m%03d_ro%03d_ra%03d_t%03d_a%03d_rep%04d.RData',
                      fulldir, bg.kappa, bg.scale, bg.mu, r.dens, ra.dens, t.dens, a.dens, repl))

  # Loop for each treatment combination
  for(trt in seq_len(ncells)){
    ta <- which(ta.prior==results2h$Target[r+trt])
    fg <- which(fg.prior==results2h$fg[r+trt])
    setTkProgressBar(pb, itr+trt/ncells,
                     label = paste0('Iteration ', itr,
                                    ': Analyzing rep ', repl,
                                    ' with spacing ', spacings[ta, fg]))


    ## SAMPLING

    # Sample along the transects, starting at a random horzontal coordinate
    sample <- sample.transects.NS(site, width, spacings[ta, fg],
                                  offset = runif(1, 0, spacings[ta, fg] + width/2))

    # Save the sample
    filepath <- sprintf('%s/hard_sample_t%s_p%03d_rep%04d',
                        sampdir, ta.prior[ta], fg.prior[fg], repl)
    write.anomaly(sample$anomaly, paste0(filepath, '.anomaly'))
    write.cog(sample$cog, paste0(filepath, '.cog'))
    results2h$length[r+trt] <- sample$length


    ## KRIGING

    # Evaluate local density in each window
```

```
datfile <- sprintf('%s/rep%04d_s%04d.dat', outdir, repl, spacings[ta, fg])
ldens <- windowed.density.NS(sample, window.sizes[ta])
write.geoeas(ldens, datfile, title = 'Data exported from R')

# Create GAMV parameter file
gpar <- sprintf('%s/rep%04d_s%04d_g.par', outdir, repl, spacings[ta, fg])
gout <- sprintf('%s/rep%04d_s%04d_g.out', outdir, repl, spacings[ta, fg])
cat(gamv_par(datfile, gout, window.sizes[ta]), file = gpar)

# Run GAMV to compute empirical semivariogram
system2(gamv_exe, input = gpar, wait = TRUE)

# Read semivariogram and discard lags that were not estimated
svario <- read.table(gout, row.names = 1,
                     col.names = c('l', 'lag.dist', 'semivariogram',
                                   'n', 'tail.mean', 'head.mean'),
                     header = FALSE, skip = 3)
svario <- svario[svario$n>0,]

# Fit parametric semivariograms
params <- list('sphere' = optim(c(nug.start, sill.start[ta], range.start),
                                sv.wss, lags = svario$lag.dist,
                                n = svario$n, ghat = svario$semivariogram,
                                model = sv.sphere, lower = c(0, 0.0001, 0),
                                upper = c(Inf, Inf, Inf),
                                method = 'L-BFGS-B'),
               'expon' = optim(c(nug.start, sill.start[ta], range.start),
                                sv.wss, lags = svario$lag.dist,
                                n = svario$n, ghat = svario$semivariogram,
                                model = sv.expon, lower = c(0, 0.0001, 0),
                                upper = c(Inf, Inf, Inf),
                                method = 'L-BFGS-B'),
               'gauss' = optim(c(nug.start, sill.start[ta], range.start),
                                sv.wss, lags = svario$lag.dist,
                                n = svario$n, ghat = svario$semivariogram,
                                model = sv.gauss, lower = c(0, 0.0001, 0),
                                upper = c(Inf, Inf, Inf),
                                method = 'L-BFGS-B'),
               'power' = optim(c(nug.start, slope.start, power.start),
                                sv.wss, lags = svario$lag.dist,
                                n = svario$n, ghat = svario$semivariogram,
                                model = sv.power, lower = c(0, 0.0001, 0),
                                upper = c(Inf, Inf, 2),
                                method = 'L-BFGS-B'))

# Find the model with the smallest sum of squares
# The indices match GSLIB's model type numbers
type <- order(sapply(params, function(x){return(x$value)}))[1]

# Create KT3D parameter file
kpar <- sprintf('%s/rep%04d_s%04d_k.par', outdir, repl, spacings[ta, fg])
kout <- sprintf('%s/rep%04d_s%04d_k.out', outdir, repl, spacings[ta, fg])
kdbg <- sprintf('%s/rep%04d_s%04d.dbg', outdir, repl, spacings[ta, fg])
cat(kt3d_par(datfile, kdbg, kout, nx[ta], ny[ta], xmin[ta], ymin[ta],
             window.sizes[ta], params[[type]]$par[1],
```

```
            params[[type]]$par[2], params[[type]]$par[3], type),
    file = kpar)

# Run KT3D to do the kriging
# Note: Value of -999 indicates that the value that was not computed
system2(kt3d_exe, input = kpar, wait = TRUE)


## ANALYSIS

## Read and clean KT3D output
krige.out <- read.geoeas(kout)
krige.out[krige.out$Estimate == -999,] <- rep(NA, 2)
krige.out$EstimationVariance[krige.out$EstimationVariance < 0] <- 0
kest <- im(matrix(krige.out$Estimate, nrow = ny[ta], byrow = TRUE),
           seq(xmin[ta], length.out = nx[ta], by = window.sizes[ta]/6),
           seq(ymin[ta], length.out = ny[ta], by = window.sizes[ta]/6),
           unitname = c('foot', 'feet'))
kvar <- im(matrix(krige.out$EstimationVariance, nrow = ny[ta], byrow = TRUE),
           seq(xmin[ta], length.out = nx[ta], by = window.sizes[ta]/6),
           seq(ymin[ta], length.out = ny[ta], by = window.sizes[ta]/6),
           unitname = c('foot', 'feet'))

# Get the delineated regions, if there are any
highdens <- kest > bg.dens + qnorm(0.95) * sqrt(kvar)
if(sum(highdens) > 0){
  identified <- connected(highdens, background = FALSE)
  areas2h[[itr, ta, fg]] <- sapply(levels(identified$v), function(x){
      return(area(Window(connected(identified==x,
                                   background = FALSE))[sitewindow]))
  })

  # Ignore regions less than 3 acres
  ignore <- which(areas2h[[itr, ta, fg]] < 3*43560)
  areas2h[[itr, ta, fg]][ignore] <- NA
  results2h$identcount[r+trt] <- sum(!is.na(areas2h[[itr, ta, fg]]))
  for(i in ignore){
    identified$v[identified$v==i] <- NA
  }
}
if(results2h$identcount[r+trt] > 0){
  idboundary <- as.polygonal(Window(identified))[sitewindow]
  idpoints <- site
  Window(idpoints) <- idboundary
  missedpoints <- site
  Window(missedpoints) <- complement.owin(idboundary,
    frame = dilation(Frame(sitewindow), window.sizes[ta]))

  results2h$detect[r+trt] <- sum(marks(idpoints)>2) / sum(marks(site)>2)
  results2h$detect1[r+trt] <- sum(marks(idpoints)==1) / sum(marks(site)==1)
  results2h$detect2[r+trt] <- sum(marks(idpoints)==2) / sum(marks(site)==2)
  results2h$detect3[r+trt] <- sum(marks(idpoints)==3) / sum(marks(site)==3)
  results2h$detect4[r+trt] <- sum(marks(idpoints)==4) / sum(marks(site)==4)
  results2h$detect5[r+trt] <- sum(marks(idpoints)==5) / sum(marks(site)==5)
  results2h$dens[r+trt] <- sum(marks(idpoints)>0) / area(idpoints) * 43560
```

```
      a <- intersect.owin(Window(idpoints), TAs, fatal = FALSE)
      results2h$detectarea[r+trt] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), roadwindow, fatal = FALSE)
      results2h$detectarea1[r+trt] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), ranchwindow, fatal = FALSE)
      results2h$detectarea2[r+trt] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), T1, fatal = FALSE)
      results2h$detectarea3[r+trt] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), T2, fatal = FALSE)
      results2h$detectarea4[r+trt] <- ifelse(is.null(a), 0, area(a))
      a <- intersect.owin(Window(idpoints), A, fatal = FALSE)
      results2h$detectarea5[r+trt] <- ifelse(is.null(a), 0, area(a))
      results2h$identarea[r+trt] <- area(idpoints)
      ndist2h[[itr, ta, fg]] <- nncross(missedpoints[marks(missedpoints)>0],
                                        edges(idboundary), what = 'dist')
    }
  }
})
setTkProgressBar(pb, nsamp+1, label = 'Done')
invisible(close(pb))
print(timing)

save(results2h, file = 'datasets/hard/results/hard_exp2results.RData')
save(ndist2h, file = 'datasets/hard/results/hard_exp2ndist.RData')
save(areas2h, file = 'datasets/hard/results/hard_exp2areas.RData')
```

A.3 Miscellanea

A.3.1 Detection Probability Curves, `spacings.r`

```
# Create several detection probability curves for the small TA at the easy site
require(RDCOMClient)

nreps <- 10
window.sizes <- c(80, 160, 400, 640, 720, 1000, 1500)
min.spacing <- 1
max.spacing <- 600
probs <- data.frame('spacing' = min.spacing:max.spacing,
                    matrix(numeric(),
                           nrow = length(min.spacing:max.spacing),
                           ncol = length(window.sizes) * nreps))
colnames(probs)[-1] <- paste0('w', rep(window.sizes,
                                       rep(nreps, length(window.sizes))),
                                       '.', rep(1:nreps, length(window.sizes)))
major.diam <- 1200
minor.diam <- 800
fg.dens <- 200
bg.dens <- 100
width <- 6

# Start VSP
vsp <- COMCreate('VSample.Document')
```

```
pb <- winProgressBar(title = 'Computing Detection Probabilities',
                     label = 'Computing Detection Probabilities',
                     min = 1, max = length(window.sizes)+1, initial = 1)
for(w in window.sizes){
  for(i in 1:nreps){
    setWinProgressBar(pb, which(window.sizes==w)+(i-1)/nreps,
                      label = paste0(w, ' foot window, rep ', i))

    # If this function returns a positive integer, that number is the index of
    # the argument that it didn't like
    vsp$Plan()$UXOPowerCurveS(width, # xsect width
                              0, # map units (feet = 0)
                              0, # xsect pattern (parallel = 0)
                              fg.dens, # fg density
                              3, # density units (per acre = 3?)
                              2, # density at (center = 2?)
                              3, # swath ratio
                              major.diam/2, # target radius
                              minor.diam/major.diam, # target shape
                              TRUE, # random angle
                              0, # angle
                              bg.dens, # bg density
                              1, # 1-false negative rate
                              0.05, # alpha
                              w, # window length
                              0.03, # min precision
                              0.01, # max error
                              min.spacing, # min spacing
                              max.spacing, # max spacing
                              TRUE) # bivariate normal?
    probs[,paste0('w', w, '.', i)] <- sapply(probs$spacing, function(x){
        vsp$Plan()$UXOPowerCurveY(x)
      })
  }
}
setWinProgressBar(pb, length(window.sizes)+1, label = 'Done')
close(pb)

# Close VSP
vsp$Window()$Close('VSampl1')

save(probs, file = 'datasets/plan/detectionprobs.RData')
```

A.3.2 Functions for Reading and Writing Data Files, `data_functions.r`

```
# Write VSP anomaly files
write.anomaly <- function(data, file, col.names = TRUE, row.names = FALSE){
  return(write.table(data, file, row.names = row.names, col.names = col.names,
                     quote = FALSE, sep = ','))
}

# Read VSP anomaly files
```

```
read.anomaly <- function(file, header = TRUE, sep = ',', ...){
  return(read.table(file = file, header = header, sep = sep, ...))
}

# Write VSP cog files
write.cog <- function(data, file, col.names = TRUE, row.names = FALSE){
  return(write.table(data, file, row.names = row.names, col.names = col.names,
                     quote = FALSE, sep = ','))
}

# Read and write GeoEAS files such as used by GAM/GAMV and KT3D
read.geoeas <- function(file){
  header <- readLines(file, n = 2)
  headstr <- strsplit(header[2], ' ')[[1]]
  headstr <- headstr[headstr!='']
  ncol <- as.numeric(headstr[1])
  header2 <- readLines(file, n = 2 + ncol)
  cnames <- make.names(header2[2+(1:ncol)])
  return(read.table(file = file, col.names = cnames, skip = 2 + ncol))
}
write.geoeas <- function(data, file, title = file){
  cat(c(title, length(colnames(data)), colnames(data)), sep = '\n', file = file)
  return(write.table(data, file = file,
                     row.names = FALSE, col.names = FALSE, append = TRUE))
}
```

A.3.3 Functions for Spatial Models, `spatial_functions.r`

```
# Bivariate Normal kernel
# a is horizontal axis, b is vertical axis, r is rotation angle
gauss.elliptic <- function(x, y, mu.x = 0, mu.y = 0, s.a = 1, s.b = 1,
                           r = 0, maxrate = 1){
  rot <- zapsmall(matrix(c(cos(r), sin(r), -sin(r), cos(r)), nrow = 2))
  ab <- diag(c(s.a^2, s.b^2))
  sigma <- rot %*% ab %*% t(rot)
  siginv <- solve(sigma)
  mu <- matrix(c(mu.x, mu.y), nrow = 2)
  mat <- matrix(rbind(x, y), nrow = 2)
  return(maxrate * apply(mat, 2, function(vec){
      exp(-t(vec - mu) %*% siginv %*% (vec - mu) / 2)
    }))
}


# Parametric semivariograms
# h is the lag distance, theta = c(nugget, sill, range)
# ghat is empirical semivariogram, lags is a vector of lag distances

# Spherical
sv.sphere <- function(h, theta){
  return(theta[2] * ifelse(h<theta[3], (1.5*h/theta[3]-0.5*(h/theta[3])^3), 1))
}
# Exponential
```

```
sv.expon <- function(h, theta){
  return(theta[2] * (1 - exp(-3*h/theta[3])))
}
# Gaussian
sv.gauss <- function(h, theta){
  return(theta[2] * (1 - exp(-(3*h/theta[3])^2)))
}

# Power
# theta[2] is slope, theta[3] is exponent
sv.power <- function(h, theta){
  return(theta[2] * h^theta[3])
}

# OLS objective function
sv.ss <- function(theta, lags, ghat, model){
  theta <- ifelse(theta>0, theta, 0)
  return(sum((ghat-theta[1]-sapply(lags, model, theta = theta))^2))
}

# WLS objective function
sv.wss <- function(theta, lags, n, ghat, model){
  theta <- ifelse(theta>0, theta, 0)
  return(sum(n/2*(ghat/(theta[1]+sapply(lags, model, theta = theta))-1)^2))
}
```

A.3.4 Functions Used in Sampling and Analysis, sampling_functions.r

```
# Take a sample along evenly-spaced transects going north-south
sample.transects.NS <- function(data, width, spacing, offset = 0){
  first <- min(vertices(Frame(data))$x) + width/2 + offset
  last <- max(vertices(Frame(data))$x) - width/2
  bottom <- min(vertices(Frame(data))$y)
  top <- max(vertices(Frame(data))$y)
  x <- seq(first, last, spacing+width)
  xsect <- do.call(union.owin, lapply(x, function(i){
      owin(c(i-width/2, i+width/2), c(bottom, top))
    }))
  sample.cog <- data.frame('x' = sort(rep(x, 2)),
                           'y' = rep(c(bottom-width, top+width,
                                       top+width, bottom-width),
                                     ceiling(length(x)/2))[1:(2*length(x))])
  sample.data <- data
  Window(sample.data) <- intersect.owin(xsect, Window(data))
  ybd <- sapply(x, function(i){
      return(range(crossing.psp(
        edges(sample.data),
        psp(x0 = i, x1 = i, y0 = min(sample.cog$y), y1 = max(sample.cog$y),
            window = owin(c(i-1, i+1), range(sample.cog$y))))$y))
    })
  sample.length <- sum(ybd[2,]-ybd[1,])
  return(list('anomaly' = sample.data,
              'cog' = sample.cog,
```

```
              'length' = sample.length))
}

# Compute the density in one window
local.density <- function(center, data, winsize){
  new.win <- intersect.owin(Window(data),
                            disc(radius = winsize/2, centre = center))
  Window(data) <- new.win
  return(npoints(data)/area(new.win))
}

# Compute the density in evenly-spaced windows centered along transects
windowed.density.NS <- function(data, winsize){
  x <- unique(data$cog$x)
  ybd <- sapply(x, function(i){
      return(range(crossing.psp(
        edges(data$anomaly),
        psp(x0 = i, x1 = i, y0 = min(data$cog$y), y1 = max(data$cog$y),
            window = owin(c(i-1, i+1), range(data$cog$y))))$y))
    })
  y <- lapply(seq_len(ncol(ybd)), function(i){
      return(seq(min(ybd[,i]), max(ybd[,i]), winsize/6))
  })
  centers <- cbind('X' = rep(x, sapply(y, length)), 'Y' = unlist(y))
  return(data.frame(centers, 'Z' = 0,
    'density' = 43560 * apply(centers, 1, local.density,
                              data = data$anomaly, winsize = winsize)))
}

# Par file templates
gamv_par <- function(dat, out, win){
  return(paste0('                  Parameters for GAMV
                 ******************

START OF PARAMETERS:
', dat, '    \\file with data
1  2  0    \\  columns for X, Y, Z coordinates
1  4    \\  number of variables,col numbers
-1.0e+21 1.0e+21    \\  trimming limits
', out, '    \\file for variogram output
36    \\number of lags
', win/6, '    \\lag separation distance
', win/12, '    \\lag tolerance
1    \\number of directions
0.00  90.00 7625.00 0.00  90.00  50.00  \\azm,atol,bandh,dip,dtol,bandv
0    \\standardize sills? (0=no, 1=yes)
1    \\number of variograms
1  1  1  \\tail var., head var., variogram type
')
  )
}
kt3d_par <- function(dat, dbg, out, nx, ny, xmin, ymin, win,
                     nug, sill, range, type){
  return(paste0('                 Parameters for KT3D
                ******************
```

```
START OF PARAMETERS:
', dat, '               \\file with data
0  1  2  0  4  0                \\   columns for DH,X,Y,Z,var,sec var
-1.0e21   1.0e21                \\   trimming limits
0                               \\option: 0=grid, 1=cross, 2=jackknife
nodata                          \\file with jackknife data
1   2   0    4    0             \\   columns for X,Y,Z,vr and sec var
1                               \\debugging level: 0,1,2,3
', dbg, '               \\file for debugging output
', out, '              \\file for kriged output
', nx, '  ', xmin, '  ', win/6, '             \\nx,xmn,xsiz
', ny, '  ', ymin, '  ', win/6, '             \\ny,ymn,ysiz
1    0.5    1.0                 \\nz,zmn,zsiz
1    1      1                   \\x,y and z block discretization
2    50                         \\min, max data for kriging
8                               \\max per octant (0-> not used)
', win*2, '  ', win*2, '  0.0               \\maximum search radii
 0.0   0.0   0.0                \\angles for search ellipsoid
1     0                         \\0=SK,1=OK,2=non-st SK,3=exdrift
0 0 0 0 0 0 0 0 0               \\drift: x,y,z,xx,yy,zz,xy,xz,zy
0                               \\0, variable; 1, estimate trend
nodata                          \\gridded file with drift/mean
4                               \\  column number in gridded file
1  ', nug,'                        \\nst, nugget effect
', type, '  ', sill, '  0.0   0.0   0.0     \\it,cc,ang1,ang2,ang3
         ', range, '  ', range, '  0.0     \\a_hmax, a_hmin, a_vert
')
  )
}
```

References

Baddeley, Adrian, Ege Rubak, and Rolf Turner (2015). *Spatial Point Patterns: Methodology and Applications with R*. London: Chapman and Hall/CRC Press.

Baddeley, Adrian and Rolf Turner (2005). "spatstat: An R Package for Analyzing Spatial Point Patterns". In: *Journal of Statistical Software* 12.6, pp. 1–42.

Bivand, Roger, Tim Keitt, and Barry Rowlingson (2015). *rgdal: Bindings for the Geospatial Data Abstraction Library*. R package version 1.1-1. URL: `https://CRAN.R-project.org/package=rgdal`.

Bivand, Roger and Nicholas Lewin-Koh (2015). *maptools: Tools for Reading and Handling Spatial Objects*. R package version 0.8-37. URL: `https://CRAN.R-project.org/package=maptools`.

Bowen, W Michael and Carl Allen Bennett (1988). *Statistical methods for nuclear material management*. Tech. rep. Pacific Northwest Lab., Richland, WA: Nuclear Regulatory Commission, Office of Nuclear Regulatory Research.

Brooks, Maria Mori and J Stephen Marron (1991). "Asymptotic optimality of the least-squares cross-validation bandwidth for kernel estimates of intensity functions". In: *Stochastic Processes and their Applications* 38.1, pp. 157–165.

Byrd, Richard H et al. (1995). "A limited memory algorithm for bound constrained optimization". In: *SIAM Journal on Scientific Computing* 16.5, pp. 1190–1208.

Cressie, Noel (1985). "Fitting variogram models by weighted least squares". In: *Journal of the International Association for Mathematical Geology* 17.5, pp. 563–586.

Davidson, James et al. (2001). *Visual Sample Plan Version 1.0 User's Guide*. Pacific Northwest National Laboratory. Richland, Washington. URL: `http://www.pnl.gov/main/publications/external/technical_reports/PNNL13490.pdf`.

Deutsch, Clayton and André Journel (1998). *GSLIB: Geostatistical Software Library and User's Guide*. New York: Oxford University Press.

Deutsch, Clayton and Emmanuel Schnetzler (2003). *GSLIB*. Statios. URL: `http://www.gslib.com/`.

EPA 600-R-96-055 (1994). *Guidance for the Data Quality Objectives Process*. URL: `http://www3.epa.gov/epawaste/hazard/correctiveaction/resources/guidance/qa/epaqag4.pdf`.

James, Gareth et al. (2013). *An Introduction to Statistical Learning*. New York: Springer.

Kulldorff, Martin (1997). "A spatial scan statistic". In: *Communications in Statistics-Theory and methods* 26.6, pp. 1481–1496.

Lang, Duncan Temple (2012). *RDCOMClient: R-DCOM Client*. R package version 0.93-0.2. URL: `http://www.omegahat.org/RDCOMClient`.

Matzke, Brett et al. (2014). *Visual Sample Plan Version 7.0 User's Guide*. Pacific Northwest National Laboratory. Richland, Washington. URL: `http://vsp.pnnl.gov/docs/PNNL-23211.pdf`.

Neptune and Company, Inc. (2008). *Helena Valley Probabilistic Risk Assessment of MEC for the Montana Army National Guard*.

Pulpisher, Brent et al. (2014). *Demonstration Report for Visual Sample Plan Remedial Investigation (VSP-RI) Sampling Methods at the Motlow Site in Tennessee*. Pacific Northwest National Laboratory. Richland, Washington. URL: `http://www.pnnl.gov/main/publications/external/technical_reports/PNNL-23210.pdf`.

Pulsipher, Brett et al. (2011). *Demonstration Report for Visual Sample Plan (VSP) Verification Sampling Methods at the Navy/DRI Site*. Richland, Washington: Pacific Northwest National Laboratory. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA581965.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Schabenberger, Oliver and Carol Gotway (2005). *Statistical Methods for Spatial Data Analysis*. New York: Chapman and Hall/CRC Press.

USACE IGD 14-01 (2013). *Technical Guidance for Military Munitions Response Actions*. URL: http://www.denix.osd.mil/edqw/upload/Technical-guidance-for-Military-Munitions.pdf.

VSP Development Team (2016). *Visual Sample Plan: A Tool for Design and Analysis of Environmental Sampling*. Version 7.5. Pacific Northwest National Laboratory. Richland, Washington. URL: http://vsp.pnnl.gov/.

Zimmerman, Dale L and M Bridget Zimmerman (1991). "A comparison of spatial semivariogram estimators and corresponding ordinary kriging predictors". In: *Technometrics* 33.1, pp. 77–91.