A framework for statistical and computational reproducibility in large-scale data analysis projects with a focus on automated forensic bullet evidence comparison

by

Kiegan Erin Rice

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Statistics

Program of Study Committee:
Heike Hofmann, Co-major Professor
Ulrike Genschel, Co-major Professor
Alicia Carriquiry
Dan Nordman
Jennifer Newman

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2020

DEDICATION

I dedicate this work to my family: Mom, Dad, Carrick, and all the Rices, Dempsays, and Binders. You have always supported me completely, encouraged me to pursue my dreams, and loved me for exactly who I am. You have also made sure I never took myself too seriously, and that is something I will forever be grateful for.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

The endeavor of graduate school would not have been possible without the outstanding educators I have had throughout my life. They have encouraged me to pursue my interests and fueled my passion for mathematics and data. I would like to specifically acknowledge Lydia Soldwedel, Haresh Harpalani, Dr. Paul Roback, Dr. Katie Ziegler-Graham, Dr. Mary Walczak, and Dr. Maggie Broner.

To the team at the Center for Statistics and Applications in Forensic Evidence (CSAFE), thank you for your support over the years, including the support of these research projects. I am especially grateful to Dr. Alicia Carriquiry for her guidance and generosity, as well as Harlie Jud for her practical support in organizing the bullet scanning study and emotional support as a friend.

I would also like to thank the Roy J. Carver High Resolution Microscopy Facility for providing the high-resolution data used in most of this dissertation; specifically, the undergraduate bullet lab scanners and the lab director, Curtis Mosher. Their participation in the user variability study made this work possible.

None of this work would have been possible without the guidance and support of my major professors, Dr. Heike Hofmann and Dr. Ulrike Genschel. They both provided me incredible support in both life and work, and have helped me develop and grow immensely throughout the completion of this work. They are both wonderful role models for women in academia. I am incredibly lucky to have worked with both of them these past few years.

I would like to also thank my friends at Iowa State. While the process was difficult, having you all along for the ride made it much more enjoyable. In particular, I would like to thank Amy Crawford for her friendship, support, and commiseration throughout all five years of graduate school. Going through this together was so much better than going through it alone.

Thank you to Burgie's Coffee and Tea Company for keeping me caffeinated and well-fed throughout the writing of this dissertation.

Finally, I would like to thank Nicholas Rekuski for being my biggest cheerleader and best friend. On the days I felt like giving up, he kept me going.

ABSTRACT

The analysis of data can be conceptualized as a process of sequential steps or actions applied to data in order to achieve a quantitative result. An important aspect of the process is how to ensure that it is reproducible. Reproducibility as it applies to Statistics research involves both statistical reproducibility and computational reproducibility. Achieving reproducibility is not trivial, particularly if the problem is complex or involves data from non-standard sources. Automated bullet evidence comparison as proposed by Hare et al. (2017) involves both a complex data analysis as well as a non-standard form of data. Here, it serves as a large-scale motivating example, to help us study the impact of decision-making on the statistical and computational reproducibility of a quantitative result. We first present a method for data pre-processing and assess its impact on bullet land engraved area (LEA) matching accuracy. This is followed by a large user variability study of the high-resolution bullet LEA scanning process and development of an extended Gauge Repeatability and Reproducibility framework. Finally, we propose a framework for adaptive computational reproducibility in a changing landscape of R packages and present software tools to facilitate the study and management of computational reproducibility in R.

CHAPTER 1. INTRODUCTION

1.1 Data Analysis as a Process

Working with data requires making decisions. Decisions about which statistical method to employ, how to treat outliers and missing data, about data transformations, modeling the relationships between variables, or how to quantify results are at the heart of data analysis. Any data-focused project involves various levels of decision-making, beginning with data collection or raw data and ending with a quantitative result. Conceptually, data analyses can be framed as processes or pipelines (Buja et al., 1988), often with a linear and sequential structure.

Any type of data analysis needs to be reproducible. Two of the major components that contribute to reproducibility of a data analysis process are **computational reproducibility** and **statistical reproducibility**. Computational reproducibility is concerned with underlying code in functions and processes applied to data; a computer process applied to the same data input should produce the identical output. Statistical reproducibility is concerned with quantitative and qualitative results and their sensitivity to changes within a data analysis process.

The call for (computational) reproducibility has its origin in computer science: Donald Knuth, the father of TeX, proposed the idea of *literate programming* in the mid 1980s (Knuth, 1984). While this spawned some developments in the statistical computing community (Buckheit and Donoho, 1995; Rossini, 2001; Leisch, 2003) computational reproducibility as it applies to a statistical analysis was often taken for granted. Only in 2004 did 'forensic' work by Baggerly et al. (2004) highlight the importance of a carefully constructed data managing process. Since then, follow-up work in statistical computing (Gentleman and Lang, 2007) and appli-

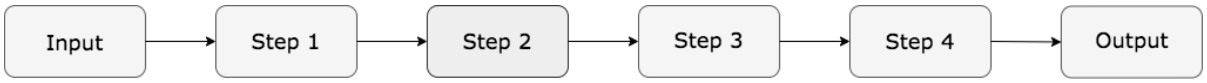

Figure 1.1 A general example of what a data analysis process might look like.

cation areas, such as bioinformatics (Baggerly and Coombes, 2009) and archeology (Marwick, 2016), has led to a renewed focus on computational reproducibility in the data analysis process.

In particular, developments in the statistical programming language R, such as knitr (Xie, 2020) and RMarkdown (Xie et al., 2018; Allaire et al., 2020), have been instrumental in ensuring the feasibility of elements of computational reproducibility as suggested by Sandve et al. (2013), who highlights the importance of the entire framework for reproducibility, and lays out ten rules for reproducible computational research, including accounting for software versioning and providing public access to data and analysis scripts.

Even these measures do not ensure that a complex data analysis can be reproduced on a different system, by a different analyst at a different point in time. Reproducing a complex data analysis generally requires the recreation of an entire protocol or process, and is often costly or time-consuming to implement. This is one reason reproducibility is difficult to assess and the concept often goes unchecked. Creating and documenting a reproducible scientific process is also difficult because there are several facets of reproducibility which intertwine with one another and are difficult to separate.

Rather than assuming complete and perfect computational reproducibility, the approach that we take here is to split a complex data analysis process into separate steps that each allow for an easier assessment of their reproducibility (or lack thereof). This approach is similar to a statistical approach of identifying sources of error in a data analysis process.

Consider the data analysis process depicted in Figure 1.1 as a set of linear, sequential actions applied to an input to produce an output. The number and complexity of steps is unique

to each project, but in general statisticians can define a "pipeline" of identifiable, modular stages such as in Figure 1.1.

In many data analyses, the steps will look something like the process in Figure 1.2(a), with one or more data processing or cleaning steps, followed by defining a set of variables or features, and finally a statistical modeling procedure which leads to an estimate or prediction, considered a quantitative output.

Data processing or cleaning steps, like those depicted in Figure 1.2(a), often rely on computer code. Decisions are made by statisticians about which methods to employ and which software package(s) to use in order to implement the chosen method(s).

To assess whether a process is computationally reproducible, we can consider how vulnerable processing steps are to modifications in underlying code or processing methods; we do this by considering the output obtained when employing multiple methods or different versions of computer code in the processing stage. This is depicted in Figure 1.2(b).

However, there also exist aspects of the data analysis process which rely on human decision-making and the statistician's choices, such as variable selection, modeling methodology, and data format. Sensitivity of quantitative output to human decisions is then a question of statistical, rather than computational, reproducibility. Consider Figure 1.2(c), which depicts a human-dependent data collection process resulting in several unique "inputs" of collected data. To assess reproducibility, we then consider the degree to which quantitative output varies given varying inputs.

Computational and statistical reproducibility can often be observed separately at different points of a process, as in Figure 1.2. However, the entire process as a whole, from input to output, depends on assumptions of reproducibility in both computational and statistical decisions. There is a natural dependence that exists between the two.

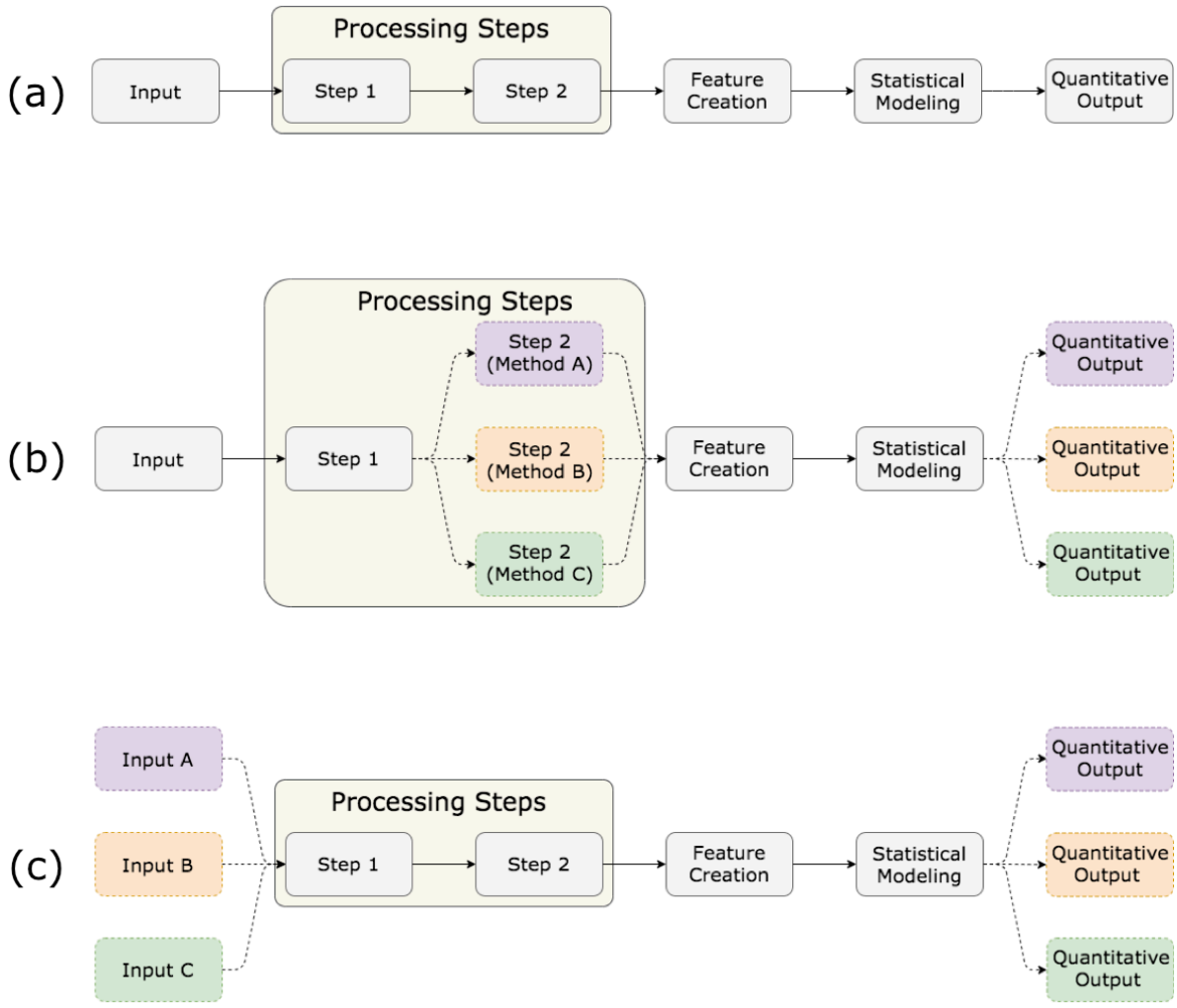

Figure 1.2 How computational and statistical variation impact and interact within a data analysis process. (a) An example of what a data analysis process might look like with two processing steps. (b) The data analysis process with varying methods employed at Step 2. Multiple methods applied to data could result in a range of outputs. (c) The data analysis process with varying inputs of raw data. Multiple inputs which differ could result in a range of outputs.

The following work is a study of reproducibility in data analysis in the context of research from multiple angles. We first study a large-scale data analysis process developed by statisticians and analyze its sensitivity to human decision making. The process, which completes automated analyses of bullet evidence in forensic science, is explored from two angles. First, we assess and improve the data processing steps that are used. Secondly, we complete and analyze a user variability study of the data collection process to test the sensitivity of quantitative output to varying input. To investigate computational reproducibility, we develop and present software tools for assessing and maintaining computational reproducibility when utilizing software packages that are subject to change.

1.2 Reproducibility in Computational Tools

Development of software tools to implement data analysis processes is a major component of statistics research. Tools for manipulating new data types and implementing analyses that involve multiple data processing steps and unique feature engineering steps are critical for utility of a method. Application requires implementation.

One of the most widely-used open-source statistical computing languages is R (R Core Team, 2020). R has a language model which fundamentally relies on including user-developed packages to enhance the language's data analysis capabilities. The framework of package development means that many packages develop and change over time. This can adversely affect applied research teams in three major ways:

1. Code to implement data analyses and obtain quantitative results may change as part of package updates. This can lead to differing quantitative results and lack of numerical reproducibility.

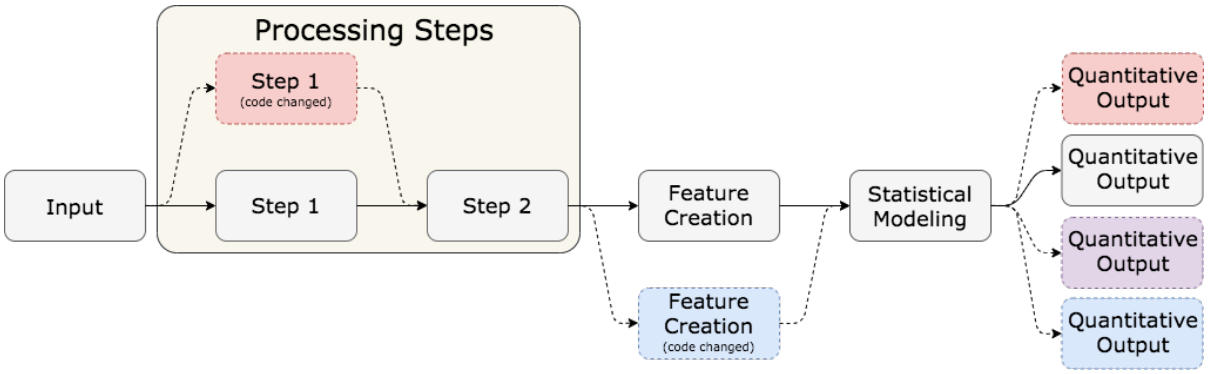

Figure 1.3 An example of how changing computer code within a data analysis process can impact quantitative output. Output may be affected by the Step 1 change (red result), the Feature Creation change (blue result), neither (grey result), or both (purple result).

2. Researchers' own packages in development are vulnerable to changes in packages upon which they depend. The mechanics of a researcher-written function which depends on outside functions that have changed can be affected.

3. Group collaboration efforts on a developing package can lead to miscommunications, including changes in code that can unknowingly affect others' results or written data analysis scripts.

Consider Figure 1.3. Suppose the code change depicted in "Processing Step 1" is researcher-driven, and the code change depicted in the "Feature Creation" step is an underlying code change in a software package in R that researchers have utilized. One or both of these changes can alter the obtained result.

There are several existing approaches to assist with group collaboration and reproducibility of numerical results. Use of version control software such as Git/GitHub[1] with R helps users keep track of changes in shared code and functions. The `knitr` package provides a significant step towards reproducibility by allowing for creation of documents which interweave written

[1] https://github.com

text and R code and output (Xie, 2020). The use of `knitr` to generate reports, slides, and other forms of presentation allows for direct reproduction of figures and results created during data analysis across multiple users and machines. However, the literate programming approach of `knitr` does not account for differences in package versions across those users and machines.

The `packrat` package allows users to catalogue specific versions of packages by defining a project-specific "package library", which can be shared with collaborators. `packrat` allows users to continue to use the same version of a package used during initial analyses, regardless of whether there are more recent updates to the package (Ushey et al., 2018). The `checkpoint` package has a similar goal; by cataloging a version of the Comprehensive R Archive Network (CRAN) each day, it allows users to set a checkpoint date and run all code with the CRAN package versions from that day (Ooi et al., 2020). Group collaboration efforts can be assisted by a combination of these tools; teams can use the exact same versions of packages when running code to ensure numerical reproducibility of scripts across members and computers.

However, neither `packrat` nor `checkpoint` provide adaptive computational reproducibility tools. When packages are updated, they often include added functionality, changing syntax, or improved code implementation. New and novel tools incorporated into a data project can drastically speed up a data wrangling or model fitting process, or even improve accuracy of results.

When the field of statistical computation is moved forward by progress in software packages, it moves the field of statistical application forward as well. However, the speed at which tools are created and updated makes it unrealistic for researchers to identify and track all changes and continually update code as packages change without some form of automated assistance. This problem compounds when packages depend on other packages which are also subject to change.

`pkgnet` provides some functionality to assist with managing package dependency (Burns et al., 2019). Described as a way to "Get Network Representation of an R Package", `pkgnet` allows users to extract and visualize an R package's dependency network on other packages as well as the network of function dependencies within the package. However, it does not provide *adaptive* tools; simply static representations of a package at a certain version or point in time.

To address this gap in adaptability of computational reproducibility tools, we have developed the `manager` package, a set of software tools for the identification and tracking of package and function changes over time.

We anticipate our software tools being widely applicable to a variety of data analysis processes that employ different software packages in R. One large-scale data process – the automated comparison of bullet evidence – inspired the need for adaptive reproducibility tools while studying the process in detail. A deep understanding of that process is not needed to understand our software tools; however, it is needed for other sections of the dissertation and is thus detailed here.

1.3 Evaluation of Evidence in Forensic Science

In the field of forensic science, decision making is a high stakes process. The determination a forensic scientist makes regarding a piece of evidence can change the course of a criminal case.

Forensic science parallels the field of statistics: forensic scientists take in information, use a scientific process to evaluate that information, and make a decision. This decision making process is most often completed on paired pieces of evidence. Pairs of evidence are often compared with the goal of determining whether a questioned piece of evidence matches a reference piece of evidence gathered from a suspect or a suspect's possessions. Where the decision making process diverges between most statistical decision theory and forensic science is the incurred loss associated with a false positive or false negative decision. Statistical decision making is

often associated with a quantitative loss function for misidentification. The loss associated with a misidentification in forensic science is potentially more than quantitative. Convicting an innocent person based on a misidentification of forensic evidence carries an incalculable loss. A false conclusion that a piece of evidence is associated with a suspect when it is not can lead to a wrongful conviction – the worst possible scenario in any criminal trial.

The decision process also differs in the sense that one uses data in the form of quantifiable measurements while the other uses less objective information in the form of visual comparisons. Two major reports in recent years have criticized the lack of objectivity in some forensic science subdisciplines, such as bullet and fingerprint evidence (National Research Council, 2009; President's Council of Advisors on Science and Technology, 2016). The criticisms put forth in both reports focus on a lack of scientific validity in visual comparison disciplines as well as a lack of clear scientifically- and statistically-based standards for decision making.

Much of the concern about the field of forensic science is focused on pattern evidence disciplines: bitemarks, latent fingerprints, firearm and tool marks, handwriting and footwear. Pattern evidence is evaluated through a process of visual feature comparison, in which forensic scientists examine a **questioned** pattern gathered as part of an investigation and compare it to a **known** pattern gathered from a suspect or database. The degree of similarity between the patterns is the primary basis on which forensic scientists make their decision about a possible match.

In the field of forensic firearms analysis, there are two main types of pattern evidence analyzed by forensic scientists: firing pin impressions on cartridge cases, and striation marks on fired bullets. Methods for both types of analysis are based on the assumptions of **consistency** and **uniqueness**: a gun will leave nearly identical marks on each bullet (or cartridge case), and no two guns will produce the same striation marks (or firing pin impressions). These assumptions are at the core of the Association of Firearm and Tool Mark Examiners (AFTE) Theory of Identification:

> "Agreement is significant when it exceeds the best agreement demonstrated between toolmarks known to have been produced by different tools and is consistent with agreement demonstrated by toolmarks known to have been produced by the same tool. The statement that 'sufficient agreement' exists between two toolmarks means that the likelihood another tool could have made the mark is so remote as to be considered a practical impossibility." (AFTE Glossary, 1998)

The scientific basis of these assumptions (consistency and uniqueness) is weak; peer-reviewed studies that address consistency of striation marks over time and uniqueness of striation patterns are few and far between. One significant effort to demonstrate uniqueness of striation marks is the Hamby et al. (2009) study. This study was designed by gathering ten consecutively manufactured Ruger P-85 gun barrels and firing at least three test bullets from each of the ten barrels. Sets of 35 bullets (two known from each barrel and 15 unknown) were sent to firearms examiners and examiners were asked to reach a conclusion on which of the ten barrels each of 15 unknown bullets originated from. The ability of examiners to correctly match striation patterns, separating out patterns from ten consecutively rifled barrels, is proposed as evidence that barrels generate unique engravings. The study reports an error rate of 0.0% with 507 examiners participating. However, the scope of the study is limited to ten barrels of a single type and also fails to address the consistency assumption.

One report which does attempt to address the consistency of striation marks over time is by Bachrach (2006). The author conducted a "barrel wear study" consisting of a sampling of the first 220 shots from nine barrels, each from a different model of firearm. The author concludes that "[T]here appears to be strong evidence to the fact that the number of fires between bullets does have an effect on their similarity". The data presented in the Bachrach (2006) report are not publicly available, so we are unable to reanalyze data using recently developed methods to further address this assumption.

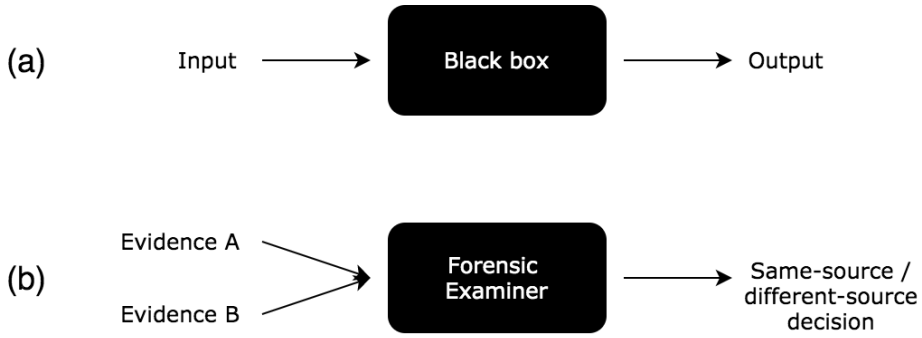

Figure 1.4 A conceptual map of a 'black box'. (a) In general, a black box takes an input and records an output, but the underlying mechanism between cannot be seen. (b) In firearms examination, the black box is the underlying decision-making mechanism the examiner goes through when comparing two pieces of evidence. The mechanism cannot be quantified, but the output can be recorded to assess the mechanism's accuracy.

In addition, the guidelines for performing examinations and reaching and communicating results vary. While firearms examiners must adhere to certain training and proficiency requirements, the standards for reporting decisions vary by state and jurisdiction due to differing policies.

One scientific method in place to guide firearm examiners' decision-making process for bullet striations is based on the concept of Quantitative Consecutively Matching Striae (QCMS), which dates back to a 1959 paper by Biasotti (1959). For traditionally rifled gun barrels, examiners focus on **land engraved areas** (LEAs), areas of the bullet that come into closest contact with the rifling of a gun barrel. A questioned LEA and known LEA are placed under a comparison microscope, and after manually finding the best alignment, examiners count the maximum number of consecutively matching striae they can find.

However, use of the QCMS method differs by examiner, and is not always employed in the decision-making process. The final decision is left to the judgment of the forensic examiner and whether they believe there to be "sufficient agreement".

Although decisions can ultimately be founded on a subjective decision by examiners, there are ways to study the accuracy of a subjective process. This can be done primarily through **black box studies**. Black box studies are research studies which require participants to complete a task and are meant to evaluate an overall decision making process. As their name suggests, black box studies focus solely on the accuracy of a process output, not the underlying mechanism used to reach that output. This concept is depicted in Figure 1.4. In firearms examination, a black box study consists of groups of known bullets originating from labeled barrels, as well as multiple questioned bullets which may or may not originate from one of the labeled barrels. Examiners are asked to report their decisions regarding the origin of each questioned bullet.

When completed on a large scale with many firearms examiners, overall accuracy and false positive rates can be estimated for the decision-making process as a whole. Black box studies are particularly well-suited for visual forensic science disciplines because the underlying decision-making mechanism is difficult to isolate and involves human judgment.

False positive rate is the most important quantity to estimate in the forensic decision-making process, as it represents the level of risk imposed on an innocent person by employing the methodology. However, there are few peer-reviewed black box studies which estimate the false positive rate of firearms examinations. Those that do exist also must be read carefully. Some studies are closed set studies, in which every questioned bullet matches at least one labeled bullet from within the study. Closed set studies can become a process of "best fit" to find the labeled bullet that *most closely resembles* the bullet in question. Open set studies, studies where unknown bullets originate from both within and outside of the set of known barrels, are preferable to estimate error rates, as they more closely mirror the reality of forensic casework. The few published open set studies reported false positive rates between one and two percent, with one estimating a rate of 1 false positive per 49 cases (Fadul et al. (2013), Baldwin et al. (2014)).

Lack of evidence for scientific assumptions, the ambiguity of the AFTE standard, and a dearth of peer-reviewed black box studies led to the recommendation by the President's Council of Advisors on Science and Technology (PCAST) that future research focus on "[d]eveloping and testing image-analysis algorithms for comparing the similarity of tool marks on bullets" (President's Council of Advisors on Science and Technology, 2016). Following the 2016 report, one major direction of the image-analysis algorithm development process has relied on high resolution 3D microscopy to gather imaging of bullet lands and cartridge cases.

1.4 Imaging Applied to Bullet Evidence

High resolution 3D microscopy captures highly detailed images of 3-dimensional objects. Confocal light microscopes measure the relative height z of an object at an (x,y) location; with magnification, objects can be digitized at resolutions higher than 1 μm per pixel ($1\mu m = 1/1000mm = 1$ micron).

For several decades, this technology has been applied to collect topological images of ballistics evidence, such as bullet land engraved areas (LEAs) and breech faces on cartridge cases. Some of the earliest applications were De Kinder et al. (1998), De Kinder and Bonifanti (1999), Bachrach (2002) and Bachrach (2006).

Early results from algorithms based on 3D topographical images show promise. Several of these methods focus specifically on 3D imaging of toolmarks on cartridge cases (e.g. Petraco and Chan (2012); Riva and Champod (2014); Tai and Eddy (2018)) and report error rates between 0.09% and 2.5%.

Hare et al. (2017) of the Center for Statistics and Applications in Forensic Evidence (CSAFE) propose a fully automated bullet matching algorithm for bullet LEAs. This algorithm, which utilizes random forest methodology, followed other initial studies using 3D topography measurements for bullet signature comparison, based on individual features such as QCMS or

cross-correlation function (e.g. Ma et al. (2004), Chu et al. (2010), Chu et al. (2013)). Using a 3D data set of LEAs scanned on bullets from the Hamby et al. (2009) study described in section 1.3, the Hare et al. (2017) algorithm was able to correctly determine whether each pair of LEAs in the test set originated from the same source or a different source. The result of zero errors on the Hamby set suggests the algorithm's ability to correctly perform the same comparisons examiners complete in the large-scale Hamby study.

The application of 3D imaging to bullet evidence is still a relatively new area of research. Thus, there is a significant amount of work to be done to assess concerns of reproducibility in proposed data processes such as the one in Hare et al. (2017). We can conceptualize the Hare et al. data process as a linear pipeline of decisions within the framework of a general data process described in section 1.1. The Hare et al. (2017) data analysis process, depicted in Figure 1.5, involves two main processing steps, followed by a step of feature extraction and a statistical modeling step which utilizes a random forest method. We will focus on two aspects of this process. The first aspect of interest is the processing step labeled "Groove ID" in Figure 1.5.

The currently accepted best practice for 3D bullet data collection dictates that LEA scans begin and end in their neighboring groove engraved areas (GEAs), the sections of bullets that separate LEAs from one another (McClarin, 2015). A reliable option for automated GEA data removal as part of processing raw scan data has not yet been proposed. Both the LOESS fit used to extract signatures in Hare et al. (2017) and the Gaussian filter used in Chu et al. (2010) are prone to boundary effects, and as such require that GEA data be accurately removed as to not impact the LEA pattern subsequently used for comparison.

To address this gap in the automated process, we have developed an automated method for GEA data identification and removal based on an adapted robust LOESS procedure for global structure removal and subsequent two-class classification techniques applied to the remaining data structure. We validate this method by examining the accuracy of output, as depicted in Figure 1.5(b), when several different methods are used for GEA data identification and removal,

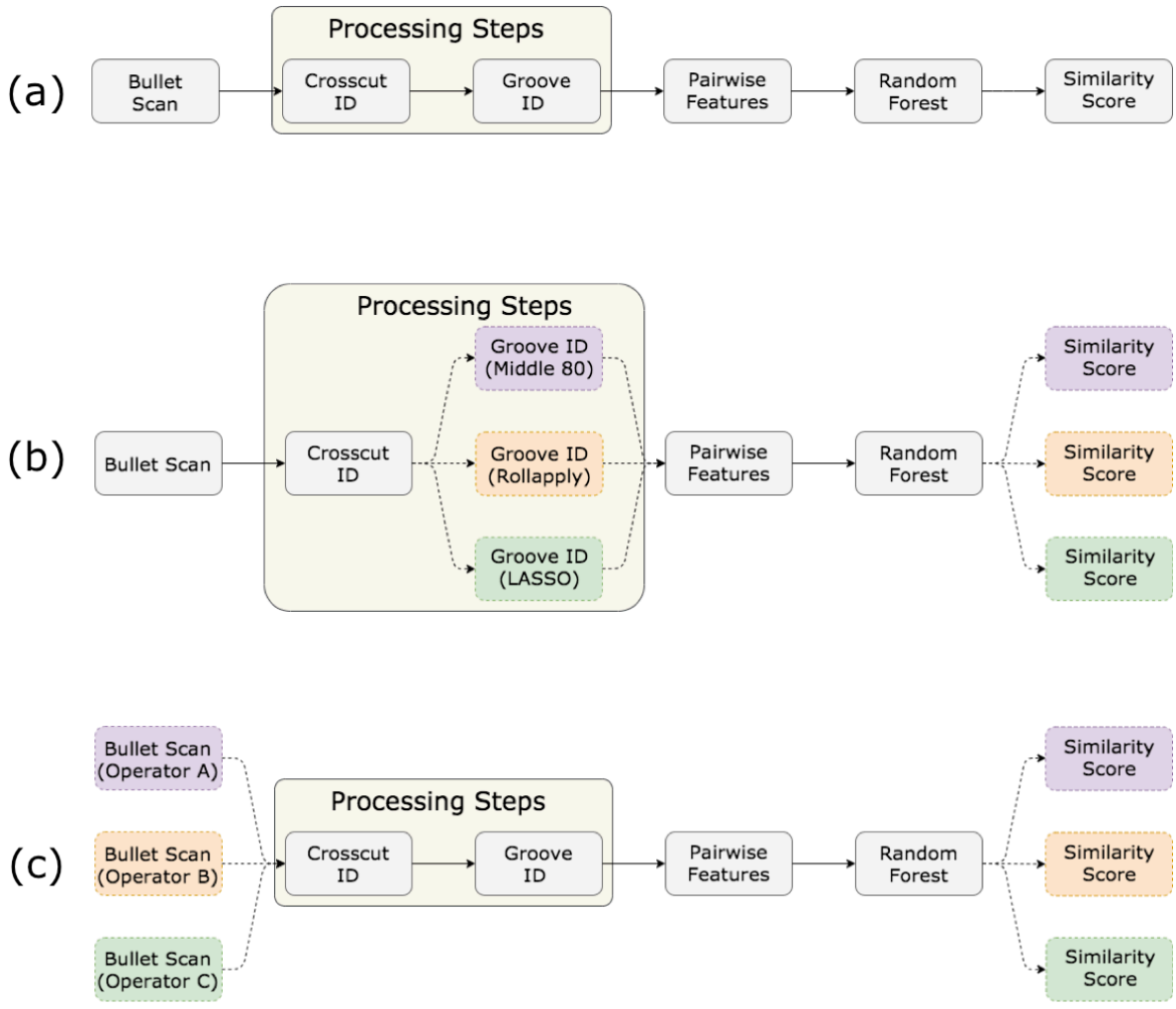

Figure 1.5 A depiction of the bullet comparison process described in Hare et al. (2017) in the framework of a data analysis pipeline. After processing, data are paired together and all subsequent analyses are 'pairwise'. The quantitative output is a similarity score between two paired objects. (a) An overall depiction of the process. (b) Depiction of varying methods used for Groove ID. Varying Groove ID methods result in multiple potential similarity scores. (c) Depiction of varying operators involved in the data scanning process. Varying inputs result in multiple potential similarity scores.

and demonstrate that our method shows significant improvement over the smoothing method proposed in Hare et al. (2017).

The second aspect of the Hare et al. bullet analysis pipeline we address is the process' sensitivity to the data collection process, depicted in Figure 1.5(c). Little formal work has been published on assessing the repeatability and reproducibility of 3D LEA scans when different operators or machines are used to capture a surface profile. Ongoing work by Martin Baiker-Soerensen at the Netherlands Forensic Institute is based on different acquisition methods for 3D imaging, including several brands of microscope. However, results from this study have not yet been published.

Assessment of repeatability and reproducibility of a measurement system is completed in engineering through Gauge R&R studies. These studies are traditionally used to estimate variability of one-dimensional response data. Some R&R concepts have been applied to medical 3D medical imaging data, but data are typically summarized into a one-dimensional object (e.g. Madelin et al. (2012)). Other published applications of the repeatability and reproducibility framework to 3D imaging data are scarce.

Proposed standards by the American Academy of Forensic Sciences Standards Board (ASB) suggest "[u]sing calibrated geometric standards (e.g., sine wave, pitch, step heights)" to measure instrument reproducibility[2]. These proposed methods help ensure a microscope is taking similar measurements over time; however, the geometric standards proposed fail to address the impact on bullet scan objects and subsequent analyses. Given that raw data are subjected to a set of processing and manipulation steps as part of the pipeline depicted in Figure 1.5, studying the variability of output given changes in operator, bullet, and machine is critical to understanding the reproducibility of the process as a whole.

[2] http://www.asbstandardsboard.org/wp-content/uploads/2019/07/061_Std_Ballot01.pdf

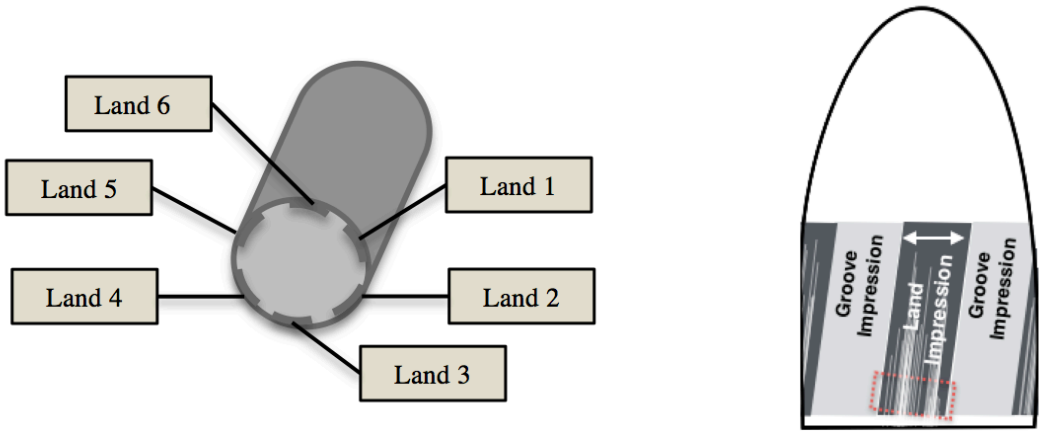

Figure 1.6 (Left) A sketch depicting lands inside a traditionally rifled barrel with six lands. (Right) A sketch of a land engraved area and striation marks engraved on the bullet. Groove engraved areas are found between land engraved areas. The rectangular box denotes the area of a bullet which would be captured as part of a LEA scan.

To address this, we have developed a repeatability and reproducibility framework adapted to the 3D bullet matching process. We estimate variability components in processed bullet signature data, as well as conduct a study of the variability of pairwise similarity scores resulting from pairwise LEA-to-LEA comparisons.

For reference throughout the remainder of this work, we conclude this chapter with a set of relevant vocabulary definitions in forensic firearms analysis as well as a detailed description of the 3D LEA scan data format and processing steps as developed by Hare et al. (2017).

1.5 Relevant Terminology

We present below a list of relevant terminology for the study of forensic firearms analysis, including physical aspects of a bullet.

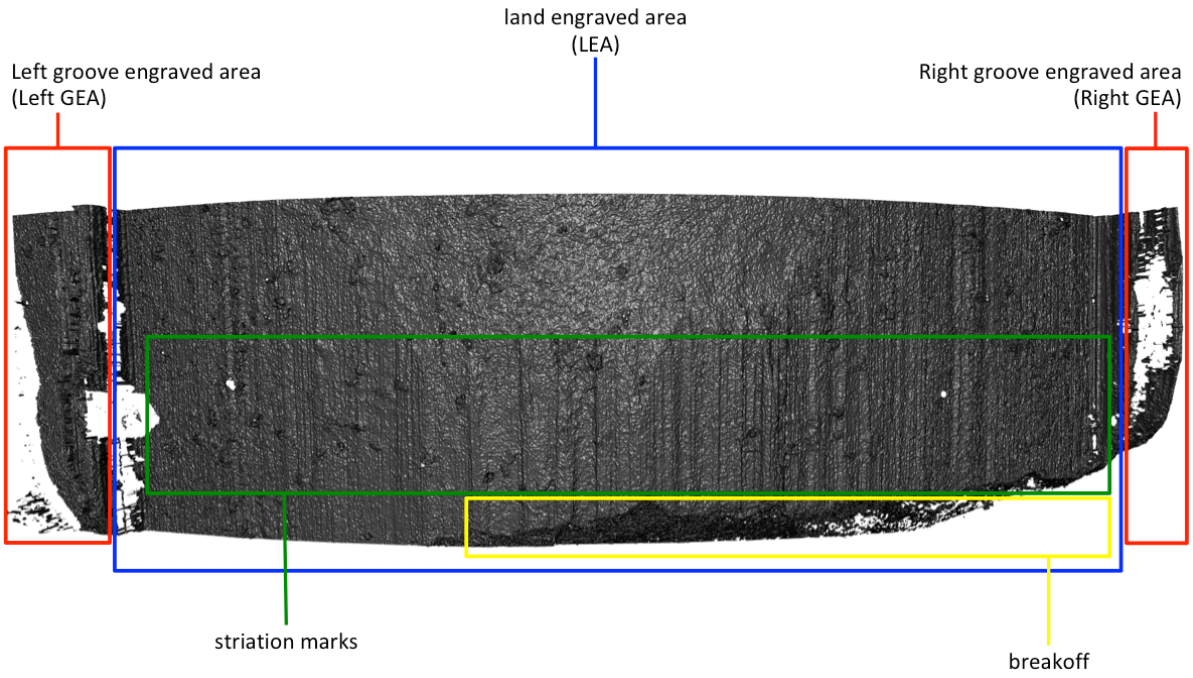

Figure 1.7 Example rendering of a 3D LEA scan with relevant portions of the bullet marked.

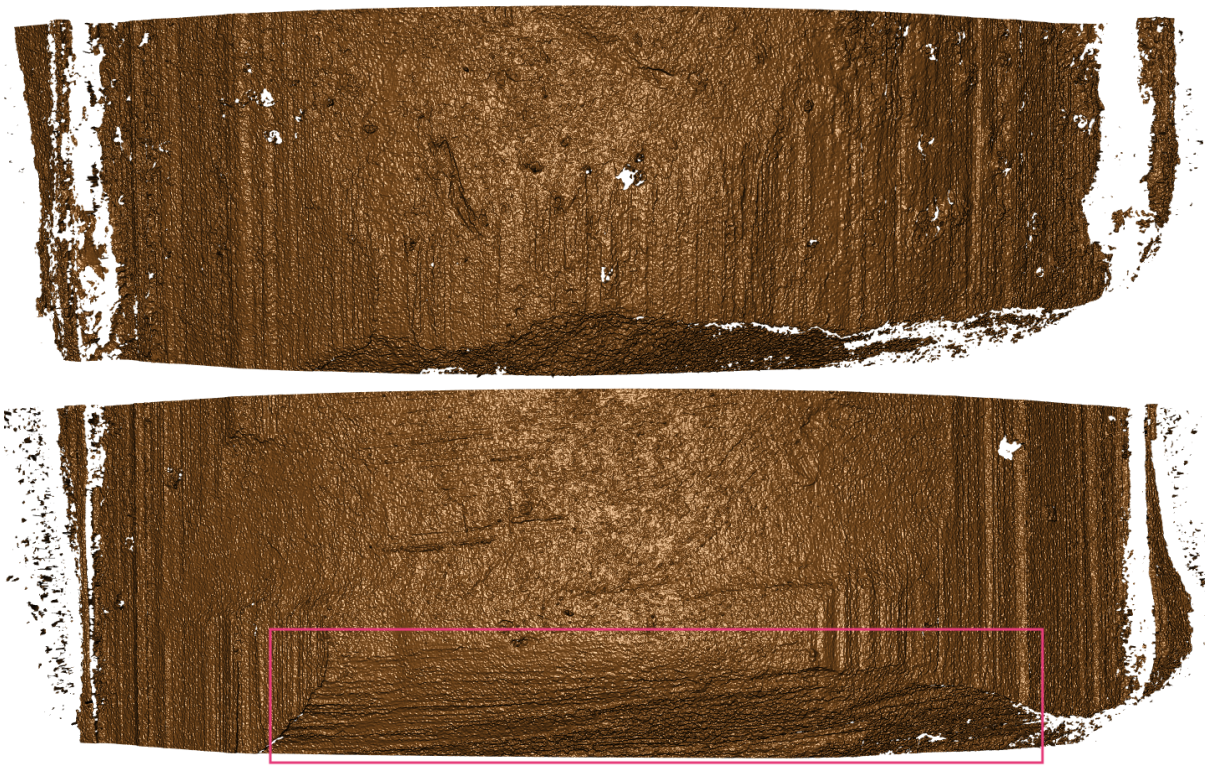

Figure 1.8 (Top) Example rendering of a 3D scan of a LEA with traditional striation marks. (Bottom) Example rendering of a 3D scan of a LEA with pronounced tank rash.

barrel rifling: The internal structure of a firearm barrel which physically causes bullets to twist as they are propelled forward during the firing process. Traditionally rifled barrels are comprised of alternating sections of "lands" and "grooves", depicted in Figure 1.6(left).

land engraved area (LEA): Area near the heel (bottom) of a bullet that is engraved with striation marks during the firing process by contact with a land inside a gun barrel. LEAs alternate with GEAs on bullets fired through traditionally rifled gun barrels. See Figure 1.6(right) and Figure 1.7.

groove engraved area (GEA): Area near the heel (bottom) of a bullet engraved during the firing process by contact with a groove space inside a gun barrel. GEAs alternate with LEAs on bullets fired through traditionally rifled gun barrels. See Figure 1.6(right) and Figure 1.7.

striation marks: Marks physically engraved on the surface of a bullet by a firearm barrel during the firing process due to close contact with imperfections on lands in the rifling. Striation marks are visually vertical on the LEA, shown in Figure 1.7.

breakoff: Area at the heel (bottom) of a LEA where large structural disruptions often occur during the firing process. Breakoff patterns vary by bullet. One example is shown in Figure 1.7.

tank rash: Structural disruptions to a bullet caused when a bullet is fired into a water recovery tank by firearms examiners and it strikes the bottom or side of the tank. Tank rash can destroy striation patterns. Bullets with tank rash are typically deemed unsuitable for forensic comparison. An example of a LEA scan with tank rash is shown in Figure 1.8.

ground truth: A classification of whether two pieces of evidence were created by the same pattern. Ground truth can never be known in forensic casework. It can be determined by researchers when developing test sets by careful data collection and labeling.

known match: The "positive" result in ground truth. If two pieces of evidence are known to have originated from the same source pattern (e.g., two bullets were fired from the same barrel), those two pieces of evidence are a "known match".

known non-match: The "negative" result in ground truth. If two pieces of evidence are known to have originated from different source patterns (e.g., two bullets were fired from two different barrels), those two pieces of evidence are a "known non-match".

similarity score: A quantitative measure of similarity between two objects. A well-defined similarity score for automated bullet comparison will have high values of similarity for known-match pairs of evidence and lower values of similarity for known non-match pairs. The Hare et al. (2017) method uses a pairwise random forest to calculate a similarity score and reports values on $[0, 1]$, with values closer to 1 indicating a higher probability of match.

bullet test set: A set of bullets fired from a limited set of gun barrels designed to test accuracy of firearm striation pattern comparisons. Test sets are divided up into two pieces:

(1) multiple "known" test fires, typically 2-4 bullets each for a small set of barrels, labelled by barrel of origin, and

(2) a set of "unknown" test fires with unknown origin.

Test takers are tasked with determining which, if any, of the "known" bullets match each of the "unknown" bullets. There are two main types of test set:

(1) **closed set**, in which each unknown bullet originates from one of the given known barrels, and

(2) **open set**, in which unknown bullets originate from a mix of the given known barrels and other, out-of-set barrels.

1.6 Data Processing Details

We detail below data structure details and processing steps proposed in Hare et al. (2017). These processing steps are applied to all bullet LEA data in this dissertation. They use primarily the `x3ptools`, `bulletxtrctr`, and `grooveFinder` packages in R, all developed by researchers at the Center for Statistics and Applications in Forensic Evidence (CSAFE).

There are two main stages of processing: first, the translation from raw data to extracted 2D LEA signature, depicted in Figure 1.10. Second, the pairwise alignment and modeling of paired signatures to calculate a pairwise similarity scores.

1.6.1 Physical Bullet to 2D signature

Capture x3p file. Trained microscope operators stage bullets for LEA capture and capture a 3D data representation of a LEA surface[3]. All scans used in the following work were captured on one of two Sensofar confocal light microscopes housed in the Roy J. Carver Microscopy Facility at Iowa State University. Scans conform to the ISO5436-2 standard for x3p (XML 3D Surface Profile) files (ISO 5436-2:2012(en), 2012). x3p is the industry standard format for capture and storage of 3D microscopic topography of bullets. Objects are stored digitally as a 2-dimensional matrix with (x, y) locations corresponding to locations on the physical object; a relative height value z is measured and recorded for each (x, y) location. A screenshot of a small portion of the surface matrix for one x3p file is shown in Figure 1.9(top). All scans were collected at a resolution of 0.645 μm per pixel ($1\ \mu m = 1/1000 mm = 1$ micron).

[3]Standard Operating Procedure document available by request.

x3p format

y locations →

x locations ↓	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	-77.46966	-77.46516	-77.41464	-77.34033	-77.27065	-77.22118
[2,]	-77.80967	-77.74937	-77.66349	-77.55566	-77.44299	-77.37663
[3,]	-78.04388	-77.95883	-77.86028	-77.74922	-77.63812	-77.57217
[4,]	-78.20613	-78.10600	-77.99013	-77.87276	-77.77537	-77.72060
[5,]	-78.28598	-78.18037	-78.04333	-77.91744	-77.84068	-77.80812
[6,]	-78.23099	-78.16338	-78.03343	-77.93237	-77.90619	-77.91022
[7,]	-78.03449	-78.09157	-78.02997	-77.99237	-78.01381	-78.03615
[8,]	-77.80099	-78.04613	-78.08249	-78.09132	-78.11773	-78.12775
[9,]	-77.63317	-78.05255	-78.16391	-78.18111	-78.18398	-78.16466
[10,]	-77.53552	-78.06992	-78.22521	-78.22818	-78.19614	-78.14056

data.frame format

x	y	value
63.855	594.69	-77.46516
64.500	594.69	-77.74937
65.145	594.69	-77.95883
65.790	594.69	-78.10600
66.435	594.69	-78.18037
67.080	594.69	-78.16338
67.725	594.69	-78.09157
68.370	594.69	-78.04613
69.015	594.69	-78.05255
69.660	594.69	-78.06992

Figure 1.9 (Top) Example of data structure for a small section of an x3p surface matrix. (Bottom) Example of data.frame structure after converting x3p to a data.frame.

Convert units. Scan units are recorded in meters. Each height measurement z must therefore be scaled by a factor of 10^7 to ensure data are in terms of microns. Values shown in Figure 1.9 have already been scaled.

Convert data to `data.frame` format. Many data processing methods in R utilize the `data.frame` format, which represents variables as named columns and recorded values of those variables per unit as rows. The `x3p_to_df` function in `x3ptools` reads the x3p surface matrix shown in Figure 1.9(top) and converts each (x,y) location with a recorded z height in the matrix into a single row with variables x, y, and *value*. x and y are location keys while *value* is the measured height value z corresponding to those location keys. An example of the resulting `data.frame` structure is shown in Figure 1.9(bottom).

Identify optimal crosscut. The `x3p_crosscut_optimize` function in the `bulletxtrctr` package identifies an optimal y location, y_{opt}, at which to take a horizontal slice of a LEA scan in order to capture the striation pattern on the bullet surface. Further details on this procedure can be found in Hare et al. (2017). For y locations, y_j, $j = 1,\ldots,1023$ and locations x_i, $i = 1,\ldots,n_{prof}$ in a given x3p file, `x3p_crosscut_optimize` performs the following:

(1) Compute the percentage of NA values present in the horizontal slice of height values $z_{i,j_{(1)}}$ for $j_{(1)} = 1$.

(2) Compute the cross-correlation value between $(z_{i,j_{(1)}})$ and $(z_{i,j_{(1)+25}})$, the horizontal slice 25 y indices higher than $y_{(1)}$.

(3) If the percentage of NA values is greater than 50% or the cross-correlation value is lower than 0.9, update $j_{(2)} = j_{(1)} + 25$.

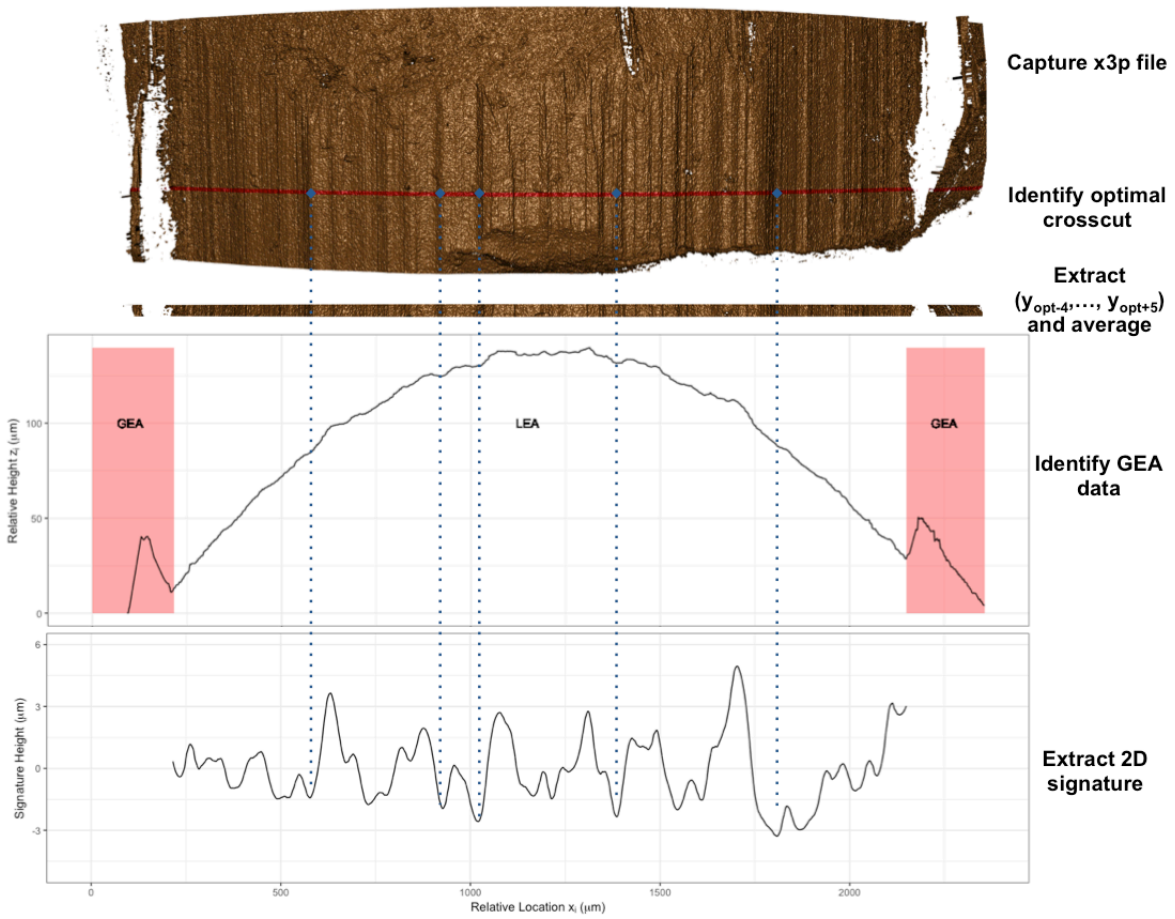

Figure 1.10 Processing steps from raw x3p scan to extracted 2D signature.

(4) Repeat steps (1) - (3) until percentage of NA values is lower than 50% and the cross-correlation is above 0.95, at index $j_{(k)}$. The final index $j_{(k)}$ is then used as the y_{opt} index: $y_{opt} = y_{j_{(k)}}$.

Extract averaged profile. Extract a set of ten horizontal crosscuts, with y indices (y_{opt-4}, y_{opt+5}). Each crosscut consists of vectors of relative locations and relative heights (x_i, z_i) for $i = (1, \ldots, n_{LEA})$, where n_{LEA} is the number of x locations captures for that particular LEA scan. Profiles are averaged in a band around y_{opt} in order to reduce the incidence of missing values and in order to boost the signal of the underlying peaks

and valleys. By including data from ten consecutive crosscuts in an averaged profile, some individual crosscut-level noise is reduced and the striation pattern is more clearly expressed.

Identify GEA data / shoulder locations. The `cc_locate_grooves` function in `bulletxtrctr`, which calls methods defined in the `grooveFinder` package, identifies optimal shoulder locations $(x_L, x_R) \in \mathbf{x}$ where LEA data ends and GEA data begins on an averaged profile. This step is also referred to as "Groove ID". Hare et al. (2017) use a "rollapply" method which first smooths data to reduce noise and subsequently uses a rolling window to search for minima. The search starts from $x_1 = x_{min}$ and moves "inward" $(x_1, x_2, \dots)$ to identify x_L. The search starts from $x_n = x_{max}$ and moves "inward": $(x_n, x_{n-1}, \dots)$ to identify x_R. The focus of Chapter 2 is on developing improved methodologies for completing this processing task.

Extract 2D signature. The `cc_get_signature` function applies 3 steps:

(1) remove data from GEA by filtering x_i values: $x_i \in (x_L, , x_{L+1}, \dots, x_{R-1}, x_R)$ as identified in the previous stage,

(2) apply global smooth (LOESS with span $= 0.75$) to remove bullet curvature, and

(3) apply local smooth (LOESS with span $= 0.03$) to remaining data to reduce noise and capture smooth pattern of "peaks" and "valleys".

1.6.2 2D Signatures to Pairwise Scores

Align two 2D signatures. Paired signatures are aligned in the x direction using the function `sig_align` in `bulletxtrctr`, which uses the maximum cross-correlation function value to identify the $x-$direction shift at which the two signatures are most highly correlated, shown in Figure 1.11.

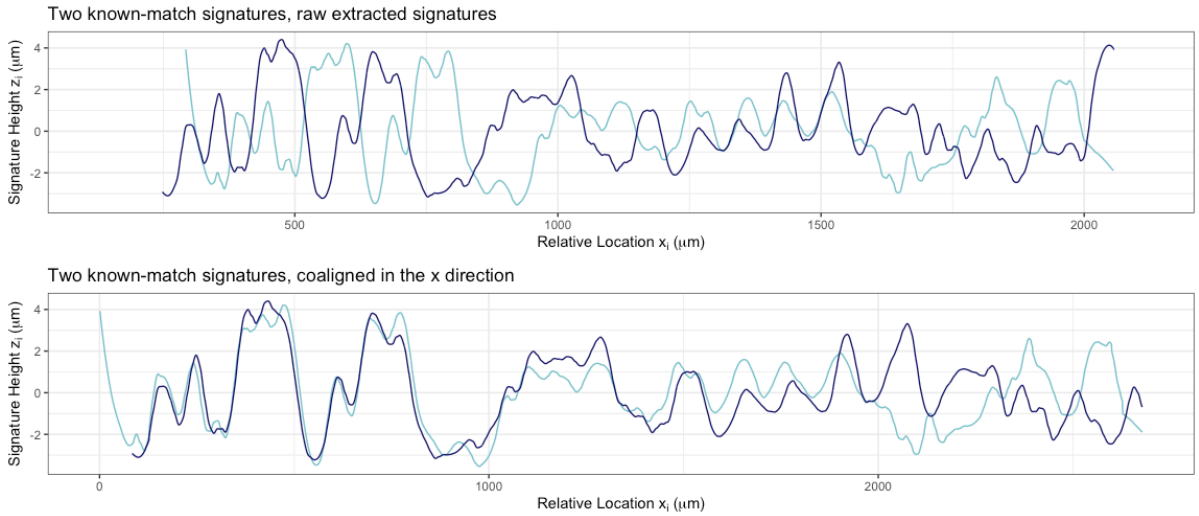

Figure 1.11 Example of two signatures being aligned in the x direction by the `sig_align` function. (Top) Two signatures plotted with raw x locations. (Bottom) Two signatures plotted after coalignment in the x direction.

Extract pairwise features. Once aligned, comparisons of the signatures can be completed by extracting several features. These features include cross-correlation function (`ccf`), maximum consecutively matching striae (`cms`), maximum consecutively non-matching striae (`non_cms`), and the Euclidean distance between the two aligned signatures (`D`). One of the most prominent features is the `ccf`, or max cross-correlation function. Additional features are detailed in Hare et al. (2017).

Apply pre-fit random forest. The random forest algorithm trained in Hare et al. (2017) on the Hamby set 44 data is called using `bulletxtrctr::rtrees`. Predicting a probability of match given the paired features using `rtrees` results in a similarity score (or "random forest score") between 0 and 1. A score closer to 1 indicates a higher degree of similarity between paired signatures, or a higher probability that the two signatures are a match (originated from the same land in the same barrel).

In the following work, we utilize both extracted signature data and pairwise score data resulting from the application of Hare et al. (2017)'s random forest algorithm. Additional details can be found in that publication.

CHAPTER 2. DEVELOPMENT OF AUTOMATED GROOVE IDENTIFICATION TECHNIQUES IN FORENSIC PATTERN ANALYSIS

2.1 Introduction

Forensic pattern analysis aims to address the same-source, different-source problem: whether two impressions were generated by the same object. One of the most significant aspects of the same-source problem in firearms analysis is the determination of whether two bullets were fired through the same gun barrel. The evidence used in visual pattern comparison of bullets are striation marks, which are engraved on a bullet during the firing process by micro imperfections in the barrel.

For barrels with traditional (i.e., not polygonal) rifling, striation marks found on land engraved areas (LEAs) are the primary source of evidence used to compare two bullets. LEAs bear marks engraved by alternating sections of the barrel, and are areas which are assumed to bear unique patterns of striation marks reproduced on any bullet fired through the same barrel (AFTE Glossary, 1998). Figure 1.6 depicts lands inside a barrel, as well as land engraved areas on a fired bullet.

The first step in automated bullet matching algorithms, such as those described in Hare et al. (2017) and Chu et al. (2013), is to process raw three-dimensional data and extract a two-dimensional representation of the striation pattern, called a LEA signature (or 2D signature). The processing steps, detailed in section 1.6, effectively translate a raw 3D object into a 2D signature made up of a series of peaks and valleys. The data processing stages must have

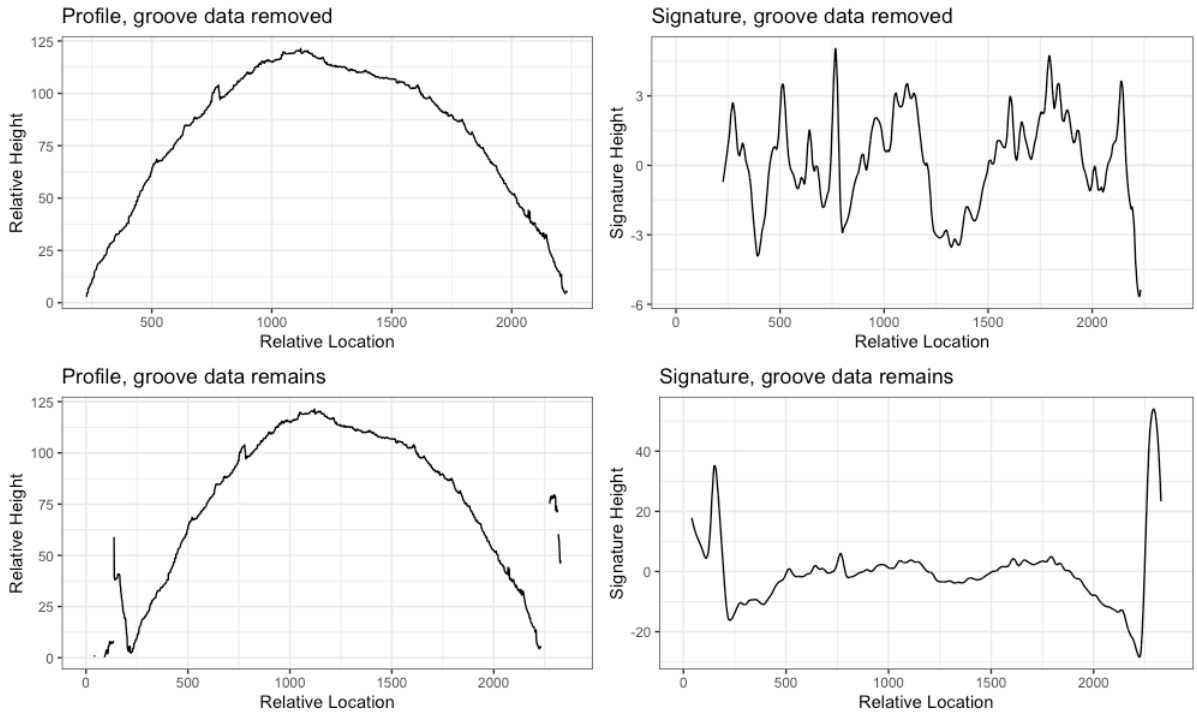

Figure 2.1 An example of the impact failure to remove GEA data can have on an extracted 2D signature. Important data features are obfuscated in the signature by remaining GEA structure.

two properties: (1) processing must be automated, and (2) the automated procedure must be reliable.

The second of these two processing steps is to identify and remove extraneous GEA data. We formulate this as a statistical problem to allow for the process to be automated and to perform at least as well as visual extraction. Failure to correctly remove GEA data, as seen in Figure 2.1, can result in a signature which mischaracterizes the striation patterns present on a LEA.

An approach based on data smoothing and local minima proposed by Hare et al. (2017) which identifies left and right **shoulder locations**, the break points between GEA and LEA

data on either side, is often inaccurate and falls prey to local minima that are not the true shoulder locations.

We propose here an approach to identifying shoulder locations based on robust LOESS and two-class classification techniques, and validate the improvement in automated algorithm accuracy using three different sets of LEA scans.

2.2 Data Source

The bullets used originate from three bullet test sets. Two of the test sets are considered closed sets, while one is an open set. The difference between the two types of test sets is defined in section 1.5.

2.2.1 Test Set Descriptions

The three test sets used are Hamby set 44, the Phoenix PD set, and the Houston-test set. Each test set is made up of a number of bullets from "known" barrels as well as a number of bullets considered to be of "unknown" origin. The barrel types used in all three test sets are 9mm caliber, but come from three distinct barrel models. Details are provided in Table 2.1.

Hamby set 44 is a closed test set consisting of 35 Winchester 9mm copper bullets fired from 10 consecutively rifled Ruger P85 barrels. There are two known bullets for each of the ten barrels, as well as 15 additional questioned bullets. Every LEA was scanned for each of the 35 fired bullets in the set, producing data for 210 individual land engraved areas. Two lands – Barrel 9, Bullet 2, Land 3 and Unknowns, Bullet L, Land 5 – were removed from consideration due to "tank rash". Tank rash results from a bullet striking the bottom of a water recovery tank after exiting the barrel, thereby creating marks on the land that are not due to the contact with the barrel.

Table 2.1 Test set barrel information. Note that for the Houston-test set, while all known bullets originate from 10 barrels and each set of known bullets is grouped as a set of three, some barrels are included twice, resulting in more than three bullets for that barrel. This results in 45 bullets rather than 30. Also note that there are only 208 profiles in the Hamby set 44 as two were removed due to tank rash.

Test Set Name	**Barrel Type**	**Caliber**	**Ammunition Details**	**Test Set Type**
Hamby set 44	Ruger P-85	9mm	Winchester 9mm copper	closed set
Phoenix PD set	Ruger P-95	9mm	American Eagle, 9mm copper	closed set
Houston-test set	Ruger LCP	9mm	American Eagle, 124 grain 9mm copper	open set

Test Set Name	**# known barrels**	**bullets per known barrel**	**# unknown bullets**	**total # bullets**	**total # profiles**
Hamby set 44	10	2 (20 total)	15	35	208
Phoenix	8	3 (24 total)	9	33	198
Houston-test set	10	3-6 (45 total)	24	69	414

The Phoenix PD set is a closed test set consisting of 33 American Eagle 9mm copper bullets fired from 8 Ruger P-95 barrels. There are three known bullets for each of the eight barrels, as well as 9 additional questioned bullets. Every LEA was scanned for each of the 33 fired bullets, producing a total of 198 individual land engraved areas.

The Houston-test set is an open test set consisting of 69 American Eagle 124-grain 9mm copper bullets fired from more than 10 Ruger LCP barrels. There are at least three known bullets for each of the ten barrels, with a total of 45 known bullets. There are 24 questioned bullets that were fired from a combination of the 10 known barrels and additional, out-of-set barrels. Every LEA was scanned for each of the 69 fired bullets, producing a total of 414 individual land engraved areas.

We next describe the format of data resulting from these test sets that is utilized in this chapter.

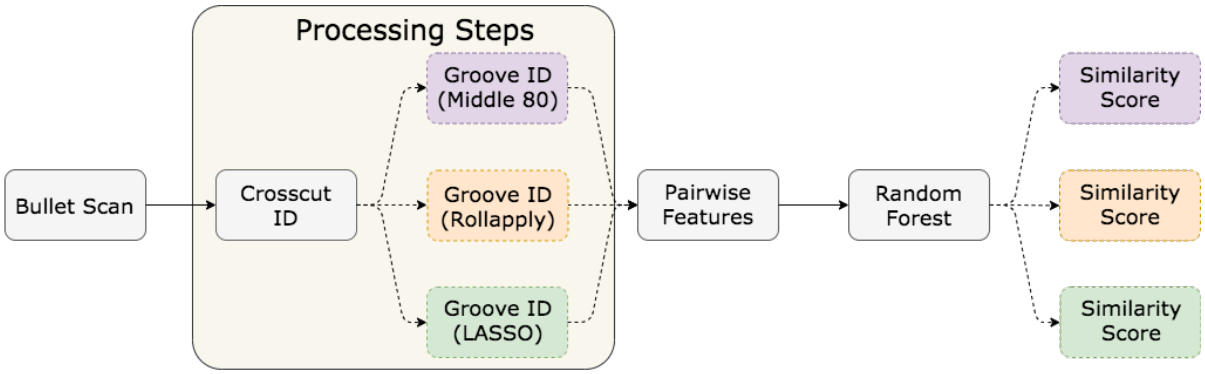

Figure 2.2 A depiction of the data analysis process in Hare et al. (2017) and the potential differences that can occur due to different methods applied at the Groove ID stage. We keep all other stages of the process constant while varying the Groove ID processing step in order to study its impact on quantitative results.

2.2.2 Data Format

We begin with high-resolution 3D scans of bullet LEAs, stored in the x3p file format described in section 1.6. The method in question in this paper is the second of the two main data processing steps, depicted in Figure 2.2. We therefore keep all other aspects of the pipeline constant and focus only on varying the method used for groove identification in order to study its impact on quantitative output.

The data used in this chapter are data which have already gone through the first major processing step – shown in Figure 2.2 – the "Crosscut ID" step. We therefore have data in the form of an averaged set of horizontal crosscuts – what we refer to as an *averaged profile*. Details about the crosscut identification and averaged profile calculation are described in section 1.6 and presented visually in Figure 1.10.

Let us define notation for the data captured in an averaged profile:

$$\mathbf{X} = (\mathbf{x}, \mathbf{z}),$$

where $\mathbf{x} = x_1, \ldots, x_{n_{prof}}$ are equidistant locations x where height measurements $\mathbf{z} = z_1, \ldots, z_{n_{prof}}$ are recorded. $\mathbf{X}$ therefore consists of pairs (x_i, z_i) for locations $i = 1, \ldots, n_{prof}$ where n_{prof} is the number of x locations on a specific averaged profile. Note that while i indices represent equidistant locations in the x direction, there are some missing z_i values recorded as `NA`, so for the following we cannot assume a fully equidistant structure of data points.

Each averaged profile in our data set is defined with the same notation. We thus have many averaged profiles,

$$\mathbf{X_{t,k}}$$

where t denotes the test set, $t = 1, 2, 3$ and k denotes the profile index within the test set t, $k = 1, \ldots, n_t$.

For the description of our methodology, we focus application to a single averaged profile $\mathbf{X} = (\mathbf{x}, \mathbf{z})$ and assume a number of x_i locations n.

2.3 Global Structure Removal

The first stage of the GEA removal process is to remove bullet curvature from the averaged profile. GEAs on the edge of each LEA represent a change in the primary data structure typically characterized by a sharp increase in measured height values, as demonstrated in Figure 1.10 and Figure 2.1. Removal of the primary curvature is expected to leave the secondary structure intact – namely, the sharp increase in height values, z_i. The change in z_i values can subsequently be used to separate LEA data from GEA data.

Bullets are subjected to significant pressure in the process of being fired through a gun barrel. During that process, a bullet may become slightly warped by heat or pressure, leading

to slight changes in the circular structure of the bullet. Thus, we cannot assume completely circular curvature on bullet LEAs. Further, we cannot model bullet LEA curvature as a directly circular or curved structure. We instead use locally estimated scatterplot smoothing (LOESS), a form of locally weighted regression (Cleveland, 1979), to model large-scale structure of the height values z_i. The local approach of LOESS – meaning that structure is fit using data points near one another – provides flexibility on a local level, and can more adeptly capture structural defects that occur for sections of bullet curvature. However, the local nature of LOESS also leaves LOESS susceptible to modeling the secondary structure of the GEA.

Traditional LOESS predicts height values $\widehat{z}_i$ as a function of location x_i by estimating values β_0, β_1 which minimize:

$$\arg\min_{\beta_0,\beta_1} \sum_{k=1}^{n} w_k(x_i)(z_k - (\beta_0 + \beta_1 x_k))^2, \tag{2.1}$$

where $w_k(x_i)$ is a weight assigned to each data point x_k based on its proximity to x_i. Weights w_k decrease as distance to x_i increases, so that data points closest to x_i influence the prediction $\widehat{z}_i$ most. Weights $w_k(x_i)$ are defined using a prespecified decreasing function, traditionally a tricube weight.

An alternative approach which mitigates the impact of GEA data is robust LOESS (see Cleveland (1979)). Robust LOESS iteratively updates weights $w_k(x_i)$ by multiplying by a robustness weight, δ_k, based on the magnitude of each residual $e_k = z_k - \widehat{z}_k$. Larger values of e_k result in a lower weight for that data point, z_k. The process is as follows:

1. Fit a LOESS model to a LEA profile, predicting height z_i using relative location x_i. Assign robustness weights δ_k of 1 to each data point (x_k, y_k).

2. Obtain predicted height values $\widehat{z}_k$, and corresponding residual values $e_k = z_k - \widehat{z}_k$.

3. Calculate updated robustness weights using residual values e_k:

$$\delta_k \times w_k(x_i) = \left(1 - \left(\frac{e_k}{6 * MAD}\right)^2\right)^2 \times w_k(x_i) \qquad \text{if} \quad \left|\frac{e_k}{6 * MAD}\right| < 1, \tag{2.2}$$

where MAD is the median absolute deviation of residuals e_k. This is known as the bisquare function.

4. Repeat steps 1-3 with updated weights at each iteration for m iterations, with 20 iterations as the default.

5. After m iterations of updating the weight vector $\delta_k w_k(x_i)$, fit a LOESS model and obtain residual values e_i for each data point (x_i, z_i).

Re-weighting data as in Step 3 reduces influence of data points with large absolute residual values. When GEA data is present on a profile, largest residual values will occur in areas where GEA data begins as it presents a competing structure with overall LEA curvature.

Figure 2.3 depicts the impact this iterative re-weighting process has on curvature removal for an example LEA from Hamby set 44. Predictions $\widehat{z}_i$ are updated and more closely follow the primary structure of bullet curvature.

However, this method fails to mitigate GEA data impact when GEA structures are more pronounced, such as the example from the Houston-test set in **??**. Thus, we make a small adaptation to the procedure by only updating weights for positive residuals, e_k:

1. Fit a LOESS model to a LEA profile, predicting height z_i using relative location x_i. Assign robustness weights δ_k of 1 to each data point (x_k, y_k).

2. Obtain predicted height values $\widehat{z}_k$, and corresponding residual values $e_k = z_k - \widehat{z}_k$.

3. Calculate updated robustness weights using residual values e_k:

$$\delta_k \times w_k(x_i) = \left(1 - \left(\frac{e_k}{6 * MAD}\right)^2\right)^2 \times w_k(x_i) \qquad \text{if} \quad \left|\frac{e_k}{6 * MAD}\right| < 1, \tag{2.3}$$

where MAD is the median absolute deviation of residuals e_k. δ_k is known as the bisquare function.

4. Assign updated weights as in Step 3 if $e_k > 0$. Else, leave weights as $w_k(x_i)$.

5. Repeat steps 1-4 with updated weights at each iteration for m iterations, with 20 iterations as the default.

6. After m iterations of updating the weight vector $w_k(x_i)$, fit a LOESS model and obtain residual values e_i for each data point (x_i, z_i).

This adapted robust LOESS, while a small procedural change, more adeptly fits bullet curvature. Figure 2.4 demonstrates the impact of the adaptation on predicted height values z_i and residual value e_i. Figure 2.5 depicts the differing levels of impact our adjustment has on each bullet test set. The Houston-test set typically has the most pronounced GEAs, and also benefits the most dramatically from the adapted robust LOESS procedure.

Once the adapted robust LOESS procedure has been applied to a LEA profile, the resulting residuals e_i follow a reliable pattern: small residuals closer to zero in locations associated with LEA data, and sharply increasing, larger residuals in locations associated with GEA data. The resulting residual structure lends itself to statistical techniques to separate the two structures more effectively.

The subsequent GEA identification methods are based on residuals e_i calculated from adapted robust LOESS fit to the global structure of the profile. Both methods result in "shoulder location predictions", predictions for the locations (x_L, x_R) at which LEA data ends and GEA data begins on the left and right side of the profile, respectively.

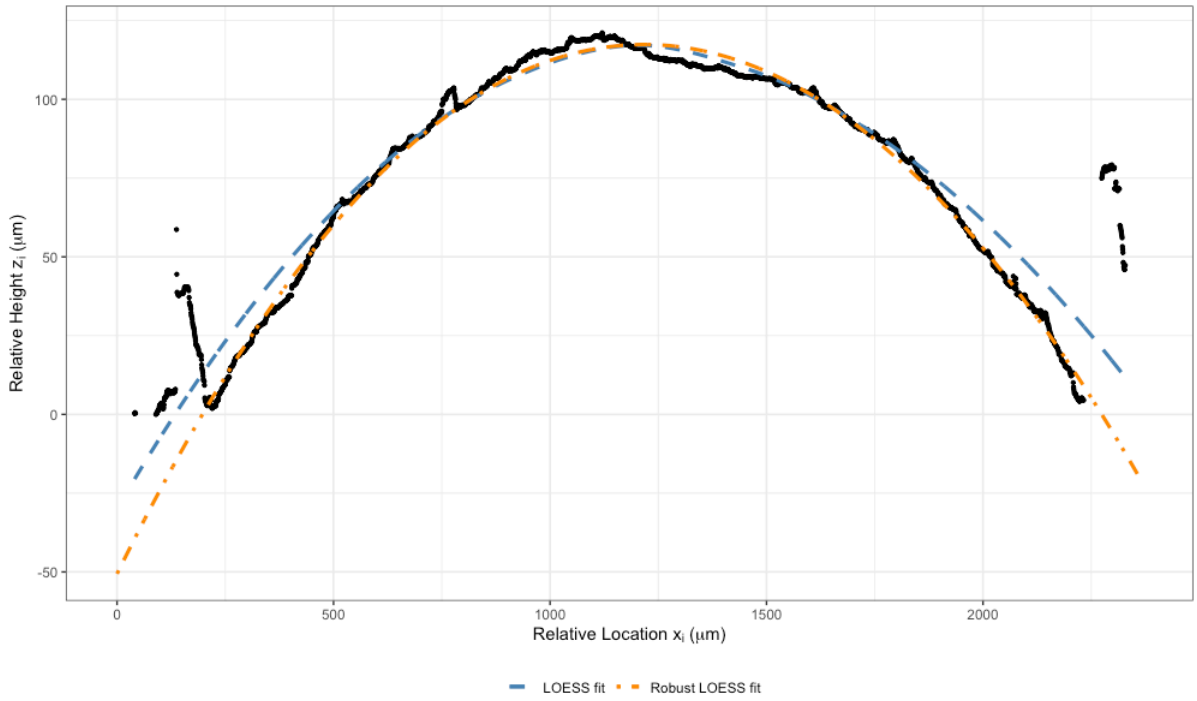

Figure 2.3 An example of the difference between traditional LOESS fit and robust LOESS fit to an LEA profile from Hamby set 44.

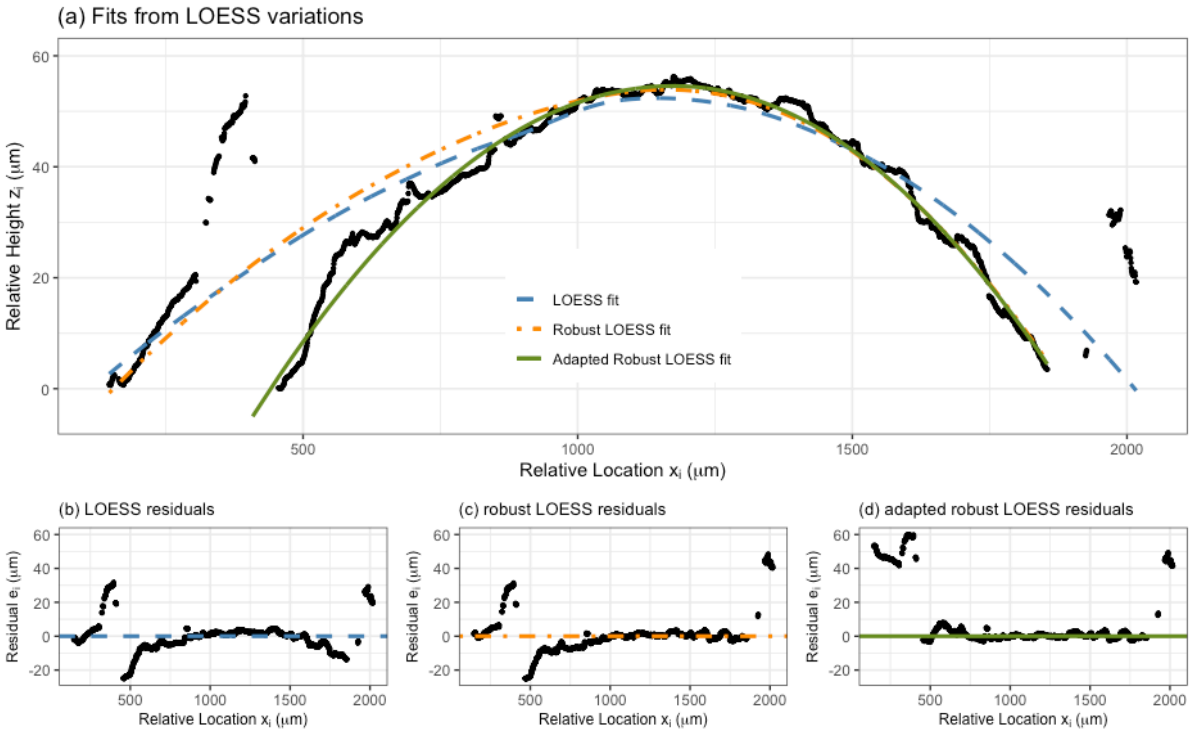

Figure 2.4 An example of the difference between LOESS, robust LOESS, and adapted robust LOESS fits to an LEA profile from the Houston-test set. (a) Predicted curves for all three methods on one LEA. Adapted robust LOESS most closely fits the LEA structure and allows GEA data to remain a separate structure. (b), (c), and (d) depict residuals e_i resulting from each respective prediction method. Adapted robust LOESS in (d) results in the most desirable residual pattern, with LEA data residuals remaining closer to zero, and GEA data residuals being positive and large.

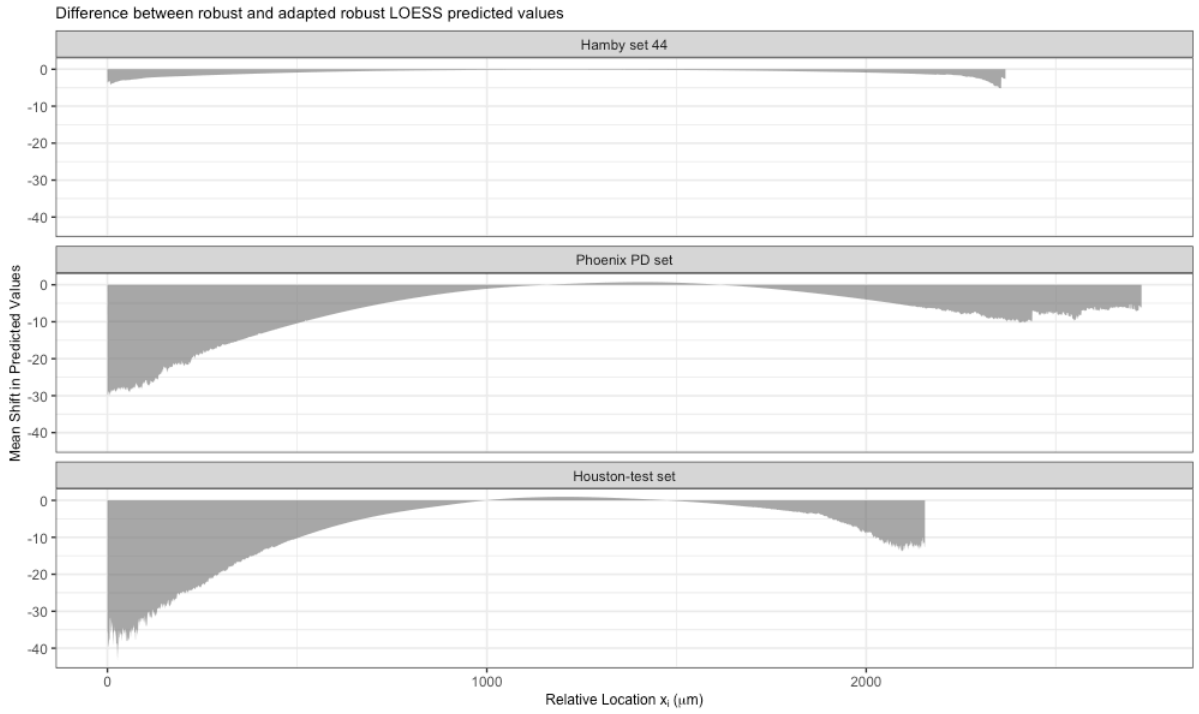

Figure 2.5 Mean shift in predictions when applying the adapted robust LOESS procedure in place of the traditional robust LOESS procedure. For Hamby set 44 predictions are, on average, very similar. The Phoenix PD and Houston-test sets have more significant downwards shifts in predictions near the left and right boundaries.

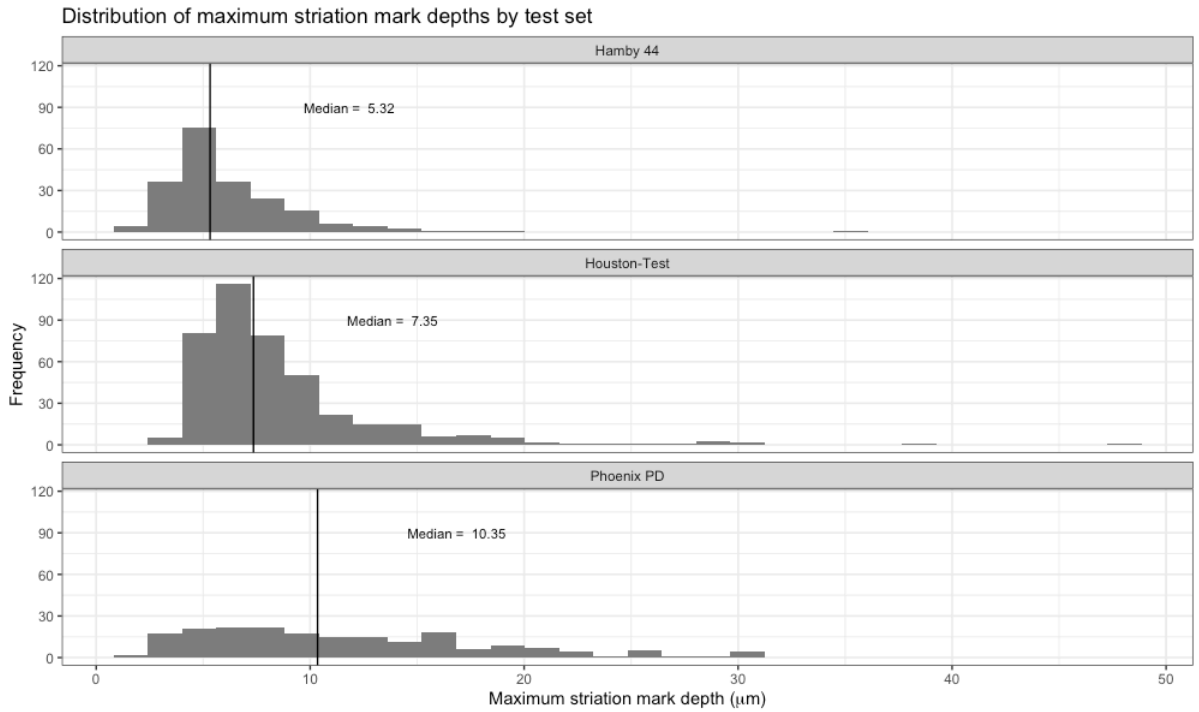

Figure 2.6 Distribution of maximum striation depth for each of the three bullet test sets. Maximum striation depths are calculated as the largest observed absolute signature value in each individual LEA signature. Black vertical lines represent the median depth for each test set. Each test set has a different distribution, which indicates standardization of residual heights is crucial for generalizability of parameter estimates.

2.4 Logistic LASSO

Our approach to predicting shoulder locations first classifies each residual point as one of two classes ("LEA" or "GEA"), and subsequently gathers the range of values classified as "LEA" points.

Classification into "LEA" or "GEA" is first approached by a process of feature engineering based on adapted robust LOESS residuals. While the residuals e_i should demonstrate differing patterns of magnitude, residuals alone are not enough to classify data with high accuracy.

Figure 2.6 demonstrates that each test set has distinct patterns of striation depth. Thus, standardization of features is imperative for transferability of fitted model parameters. The LEA scan context requires some non-traditional standardization practices.

For example, consider the distribution of residual values e_i resulting from adapted robust LOESS. There is reason to believe that the distribution will be quite skewed, which means a standard deviation will not be a good proxy for the spread of the distribution. Thus, rather than the standard deviation, we consider instead the standard deviation of residual values e_i from the middle 50% of x_i locations present in each profile. This alternative acts as a proxy for the depth of striae on each LEA, with higher standard deviations found for lands with deeper striae. Standardizing residual values by this proxy puts all residuals on a comparable scale.

There is also a need for standardization of values in the horizontal x direction, as differing barrel types produce LEAs of differing widths in the x direction. We therefore employ the maximum x location, x_n, as a proxy denominator for features that deal with variation in the x direction. This effectively maps those features to a $(0, 1)$ scale.

The full list of standardized features are as follows:

Standardized residuals e_i (`rlo_resid_std`): Robust LOESS residual value e_i, standardized by dividing by standard deviation of residual values from middle 50% of x_i locations.

Standardized residuals squared, e_i^2 (`(rlo_resid_std)`$^{\mathbf{2}}$): Squared term of `rlo_resid_std`.

Side of scan (`side`): Indicator variable. Denotes whether data point is to left or right of median x_i location.

Standardized depth from scan center (`depth_std`): Distance of data point from median x_i location, standardized by dividing by maximum x_i value (a proxy for the range of x values).

Side of scan, depth interaction (`side:depth_std`): Interaction between `side` and `depth_std` variables.

Left x_i intercept (`xint1_std`): Predicted location x_i at which adapted robust LOESS crosses z axis on left side of profile. That is, the location x_i where $\hat{z}_i < 0$ and $\hat{z}_{i+1} \geq 0$. Standardized by dividing by maximum x_i location (a proxy for the range of x values).

Right x_i intercept (`xint2_std`): Predicted location x_i at which adapted robust LOESS crosses z axis on right side of profile. That is, the location x_i where $\hat{z}_i < 0$ and $\hat{z}_{i-1} \geq 0$. Standardized by dividing by maximum x_i location (a proxy for the range of x values).

Range of local residuals e_i (`range_50_std`): Range of residual values e_i within a 50-point window for indices (x_{i-25}, x_{i+25}) around data point x_i, standardized by dividing by standard deviation of residual values from middle 50% of x_i values.

Number of local missing values (`numNA_50`): Number of missing values within a 50-point window for indices (x_{i-25}, x_{i+25}) around data point x_i.

Magnitude indicator for residual e_i (`ind_2mad`): Indicator of whether `rlo_resid`, e_i at location x_i, is greater than `2*MAD(rlo_resid)`, where *MAD* is the median absolute deviation of residual values e_i for an entire profile, $MAD(\mathbf{e})$.

Number of local positive e_i values (`numpos_50`): Number of positive residual values within a 50-point window for indices (x_{i-25}, x_{i+25}) around data point x_i.

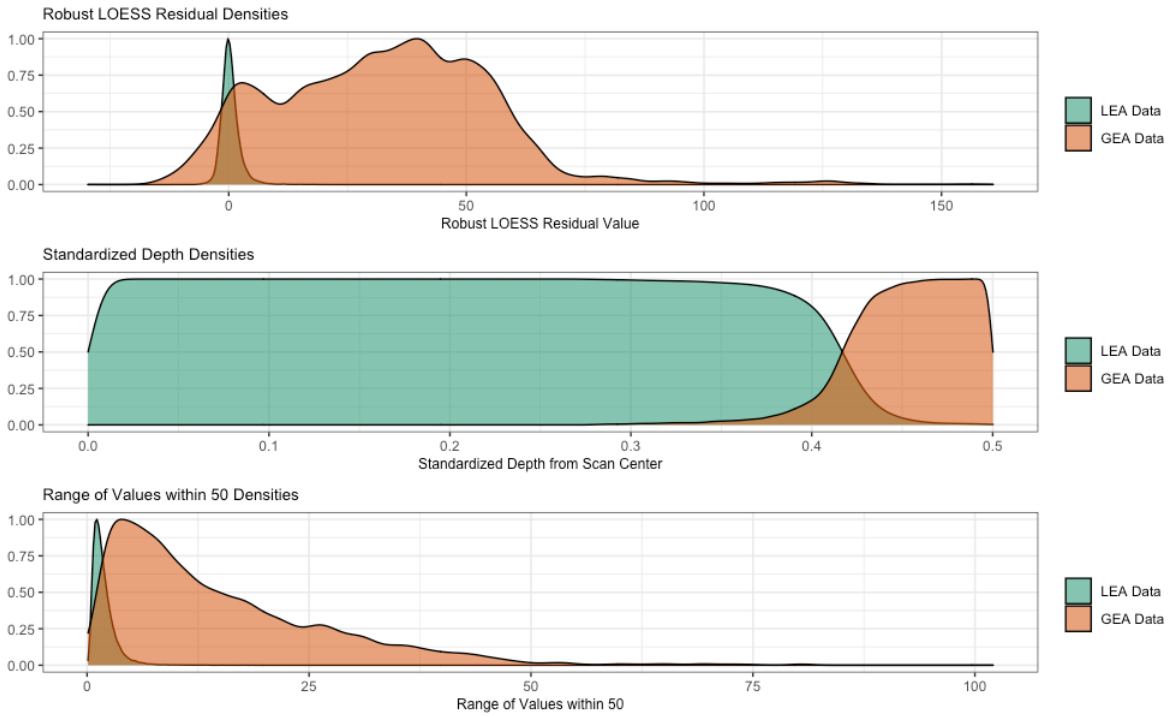

Figure 2.7 Example distributions of features used in two-class classification from Hamby set 44. While depth shows the most clear separation between GEA and LEA data, it alone will not suffice to classify data correctly.

Outside edges indicator (`ind_edges`): Indicator of whether data point is to the left of `xint1` or to the right of `xint2`. x_i locations between `xint1` and `xint2` receive a value of 0, while values on the outside of the two values receive a value of 1.

Examples of the distributions of some of these features can be seen in Figure 2.7.

A logistic LASSO model, a form of penalized regression, was fit using the developed features. LASSO parameter values for p covariates were calculated by identifying:

$$\hat{\beta}_\lambda = \underset{\beta \in \mathbb{R}^p}{\arg\min} \left\{ (Y - X\beta)'(Y - X\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}, \tag{2.4}$$

where X is a matrix with n rows, and a column for each data feature $j = 1, \ldots, p$, described above. β is a vector of estimated parameter values associated with each data feature, and Y is the vector of length n of response class values, either a 1 for GEA class, or a 0 for LEA class. LASSO adds a penalty to the traditional ordinary least squares minimization problem, and uses a tuning parameter λ (Tibshirani, 1996). Predicted probabilities of membership in the GEA class for the data point z_i at location x_i can be calculated as:

$$\widehat{p_i} = \frac{\exp\{X\hat{\beta}_\lambda\}}{1+\exp\{X\hat{\beta}_\lambda\}}. \tag{2.5}$$

A cross-validated logistic LASSO model was fit using the `cv.glmnet` function in the `glmnet` package in R (Friedman et al., 2018). Parameter values from the model with λ_{1se} were used. λ_{1se}, a standard when using LASSO, is the tuning parameter which results in the simplest model that still has cross-validation error within one standard deviation of the best model.

The resulting model uses each of the data features listed above along with pairwise interactions for each of them. The $\hat{\beta}$ vector was estimating using Hamby set 44 data. Parameter values were used to calculate predicted values of GEA membership between 0 and 1; the closer to 1, the higher probability of membership in the "GEA" class.

Two-class classification techniques traditionally employ a cutoff for predicted probabilities and assign predicted class membership using that cutoff; i.e., values above a certain cutoff are classified as part of the "GEA" class, and values below the cutoff are classified as part of the "LEA" class. An equal error rate is typically used for this purpose. Equal error rate is defined as the cutoff at which sensitivity (true positive rate) and specificity (true negative rate) are equal. However, since LEA scans consist of a majority of LEA data by nature, significant class imbalance in our response variable ("GEA" vs. "LEA") dictates a change in procedure for assigning a cutoff for class membership. Employing an equal error rate cutoff to predict class membership would result in a higher number of false positives, here predicting data which is

part of the "LEA" class as "GEA" data. We employ an equal *number of errors* rate rather than an equal error rate to ameliorate this imbalance.

The pipeline of this method for shoulder location identification is as follows:

1. Use adapted robust LOESS procedure to remove bullet curvature and obtain residual values e_i.

2. Extract data features based on x_i locations and residual height values e_i from Step 1.

3. Use fit parameter values from trained logistic LASSO model to calculate probabilities of membership in GEA class.

4. Apply cutoff of .34: classify higher probabilities as GEA data points, and lower probabilities as LEA data points.

5. Identify the minimum x_i location at which $\hat{e}_i$ is predicted as a member of the LEA class, x_L. Identify the maximum x_i location at which $\hat{e}_i$ is predicted as a member of the LEA class, x_R. (x_L, x_R) are the shoulder location predictions.

This method for shoulder location identification can be found in the R package `grooveFinder` as the function `get_grooves_lassofull`.

2.5 Results

To assess the degree of improvement in automated shoulder location identification, we want to quantify the impact our groove prediction method has on the automated bullet matching algorithm's accuracy. Four different shoulder location identification methods will be inserted into the automated bullet matching process:

(1) rollapply, the method proposed by Hare et al. (2017),

(2) middle 80%, an ad-hoc approach that keeps the middle 80% of data by x_i location,

(2) logistic LASSO, and

(3) manual identifications, the "gold standard" for identification.

Shoulder location predictions were identified using each of these methods. Predicted values using each method were then used to remove GEA data from each averaged profile. This process was then followed by signature extraction. Finally, extracted signatures were paired using every LEA combination of size two within each test set and pairwise similarity scores were calculated using the Hare et al. (2017) random forest algorithm. Pairwise similarity scores were calculated using the random forest algorithm in the `bulletxtrctr` package and are based on pairwise features described in subsection 1.6.2.

Random forest similarity scores are expected to be high for LEA-to-LEA comparisons between signatures that originate from the same land, and lower for LEA-to-LEA comparisons between signatures from different lands.

We investigate the resulting scores for each individual test set – Hamby set 44, Phoenix PD set, and Houston-test set – after calculating random forest similarity scores for all pairwise LEA-to-LEA comparisons *within* each test set. We look at both visual representations of the random forest score distributions as well as quantitative measures of the random forest method's accuracy in determining whether two lands originate from the same source.

There should be visual separation between "same source" and "different source" score distributions, as is seen for Manual ID predictions for all three test sets in Figure 2.8. Logistic LASSO shows improvement in separation for all three test sets; however, there is still room for improvement when compared to Manual ID distributions in the Phoenix PD and Houston-test sets. Another approach, ROC curves, demonstrates the same relationship in performance (see Figure 2.9).

Figure 2.8 Random forest score distributions for same source and different source LEA-to-LEA comparisons for all test sets. Logistic LASSO demonstrates improvement over Rollapply and Middle 80%, but is still not as well separated as the Manual ID distributions.

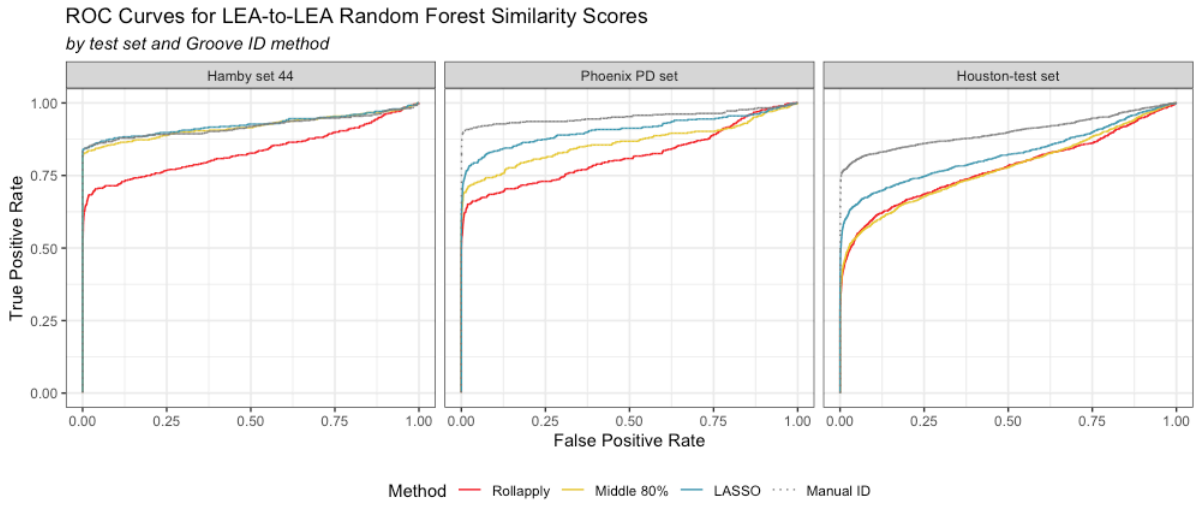

Figure 2.9 ROC curves for predictive accuracy of random forest scores for LEA-to-LEA comparisons for all three test sets, colored by Groove ID method. Manual ID, as the targeted gold standard, is shown in grey. Logistic LASSO demonstrates improvement over Rollapply on all 3 test sets, and clear improvement over the ad-hoc Middle 80 percent method for the Phoenix PD and Houston-test sets.

Quantitatively, there is significant improvement in AUC values for all three test sets, presented in Table 2.2, Table 2.3, and Table 2.4. Of interest is the classification accuracy with a fixed false positive rate. Since false positives - in this case identifying two land engraved areas as same source when they are different source - are the worst possible mistake in the forensic context, we set a cutoff rate for random forest scores based on a fixed false positive rate of .01. A fixed false positive rate of .01 allows us to investigate quantitative success measures when a score cutoff is chosen that results in 1% of same-source conclusions being false positives.

Accuracy is overall high regardless of shoulder location identification method; however, there is a significant reduction in number of false negatives for the logistic LASSO method as compared to the rollapply method for all three test sets. In addition, the decrease in cutoff needed to maintain a false positive rate of .01 also demonstrates the level of improvement in separating the same-source and different-source score distributions.

Table 2.2 LEA-to-LEA comparison results for Hamby set 44 with a class prediction cutoff set based on a controlled false positive rate of .01.

		Controlled FPR = .01					
Method	**AUC**	Cutoff	FN	TP	TN	Accuracy	**Time to Calculate**
Rollapply	0.83	0.86	169	311	21055	0.98	1 min.
Middle 80%	0.92	0.59	82	398	21052	0.99	<1 min.
LASSO	0.93	0.59	75	405	21049	0.99	6 min.
Manual ID	0.92	0.59	75	405	21063	0.99	45 min.

Table 2.3 LEA-to-LEA comparison results for Phoenix PD set with a class prediction cutoff set based on a controlled false positive rate of .01.

		Controlled FPR = .01					
Method	**AUC**	Cutoff	FN	TP	TN	Accuracy	**Time to Calculate**
Rollapply	0.82	0.86	178	292	19048	0.98	1 min.
Middle 80%	0.86	0.79	145	325	19048	0.98	<1 min.
LASSO	0.91	0.79	120	350	19042	0.98	6 min.
Manual ID	0.95	0.62	45	425	19047	0.99	45 min.

The middle 80% method, which simply filters data to keep only the middle 80% of x_i locations on an averaged profile, can be used as an ad-hoc benchmark for whether a method's algorithmic approach brings a more sophisticated or accurate set of predictions. Our pairwise LEA-to-LEA score results demonstrate that rollapply fails to meet this benchmark on two of the three test sets, and performs just about as well as the benchmark on the Houston-test set. The Houston-test bullets, which tend to have larger proportions of GEA data present on LEA scans, may require a more drastic ad-hoc percentage filter than the middle 80% (i.e., the middle 60 or 70%). For all three test sets, logistic LASSO has a higher AUC value and lower number of false negatives than the middle 80% method.

2.6 Conclusions

Adapted robust LOESS methods show a significant improvement over traditional LOESS and traditional robust LOESS in accurately removing bullet curvature from LEA scans. Future

Table 2.4 LEA-to-LEA comparison results for Houston-test with a class prediction cutoff set based on a controlled false positive rate of .01.

		Controlled FPR = .01					
Method	**AUC**	Cutoff	FN	TP	TN	Accuracy	**Time to Calculate**
Rollapply	0.77	0.9	885	656	83540	0.98	2 min.
Middle 80%	0.77	0.89	862	679	83556	0.98	<1 min.
LASSO	0.82	0.75	646	895	83540	0.98	12 min.
Manual ID	0.9	0.59	358	1183	83538	0.99	75 min.

work should include completing additional validation of this method on a larger variety of barrel types.

Our Logistic LASSO method for shoulder location prediction shows significant improvement over existing methods downstream in the Hare et al. (2017) automated bullet matching process on three test sets. This is unsurprising, as one major difference between the two methods is that rollapply takes a "local" approach and identifies local minima, while the logistic LASSO method incorporates a more "global" perspective and uses large-scale structure as well as features extracted across the entire averaged profile. While the middle 80% method can also be considered a global method, as it removes 20% of the global structure without considering local information, it does not incorporate a detailed set of additional information for each data location x_i. The improvement in predictive accuracy using the logistic LASSO method can be attributed to inclusion of more detailed data and more sophisticated predictive methods. Future work to improve performance of the logistic LASSO method should include additional validation on a larger variety of barrel types and test sets. In addition, retraining the LASSO models using a variety of test set data rather than training solely on Hamby set 44 may result in a more robust set of parameter values.

Use of Logistic LASSO for GEA data removal within the bullet analysis would significantly reduce the time required for manual identification; however, manual identification still results in more accurate LEA-to-LEA pairwise classification results for the Phoenix PD and Houston-test

sets. There exists a clear trade-off between accuracy and human intervention in the automated process. Our Logistic LASSO approach is a major improvement to the end goal of process automation, but future improvements are still needed.

As a data analysis process, the predictive strength of the Hare et al. (2017) bullet pipeline is significantly impacted by the method used to identify GEA data. Although the accuracy measure is .98 or above for all methods across all test sets with a controlled false positive rate of .01, the AUC measure and ROC curves demonstrate significant differences in the ability to separate the same-source and different-source distributions. The accuracy measure does not tell the full story, particularly due to significant class imbalance between the number of same-source comparisons and the number of different-source comparisons. The use of automated methods should be considered carefully within this data analysis process, as there are obvious differences in algorithmic performance based on the decision made at this step.

CHAPTER 3. A VARIABILITY STUDY OF 3D BULLET SCANNING AND AUTOMATED MATCHING PROCESS

3.1 Introduction

High resolution 3D scans of bullet land engraved areas (LEAs) are on the cutting edge of automated bullet matching research. Trained microscope operators collect 3D data representations of each LEA – small portions of the bullets' surface that correspond to striated toolmarks engraved during the firing process. Scans can be captured at resolutions in microns (1 micron $= 1\mu m = 1/1000mm$), providing high precision information about the surface of bullet LEAs.

Any data process or pipeline which makes use of precise measurements is dependent on the measurement process itself, relying on the assumption that a measurement will be quantitatively similar regardless of operator and measurement device. In the case of high resolution LEA scans, it is reasonable to assume that regardless of operator or device different scans of the same LEA will reliably produce the same signal for use in automated comparison methods. This assumption has not been established in a scientifically rigorous manner.

The application of high resolution microscopy to bullet LEAs is a measurement system applied to a physical object. Therefore, a natural approach to addressing the assumption of scan repeatability and reproducibility is through the lens of measurement process reproducibility.

3.1.1 Measurement Reproducibility

Gauge Repeatability and Reproducibility (Gauge R&R) studies are used in engineering to assess a measurement system. Gauge R&R studies focus on repeated measurements of multiple similar objects by multiple operators, and aim to calculate variance estimates for two main components:

(1) **repeatability** of measurements under the same environmental conditions (i.e., measurements of the same object by the same operator), and

(2) **reproducibility** of measurements under different environmental conditions (i.e., measurements on the same object by different operators).

Data from Gauge R&R studies are analyzed using random effects models. For $i = 1, \ldots, n$ parts, $j = 1, \ldots, m$ operators, and $k = 1, \ldots, \ell$ repetitions for each part-operator combination, a traditional model is defined as follows. Let y_{ijk} be the recorded measurement value on part i by operator j at repetition k. Then,

$$y_{ijk} = \mu + \alpha_i + \beta_j + \alpha\beta_{ij} + \varepsilon_{ijk}, \tag{3.1}$$

with a fixed but unknown mean of all measurements, μ, and random effects

α_i for part i, following a $N(0, \sigma_\alpha^2)$,

β_j for Operator j, following a $N(0, \sigma_\beta^2)$

$\alpha\beta_{ij}$ for Part $i -$ Operator j, following a $N(0, \sigma_{\alpha\beta}^2)$

ε_{ijk} for measurement error across repetitions, following a $N(0, \sigma^2)$.

The model in Equation 3.1 assumes that all α_i, β_j, $\alpha\beta_{ij}$, and ε_{ijk} are all independent random variables, and σ^2_α, σ^2_β, $\sigma^2_{\alpha\beta}$ and σ^2 are variance components. The summary values

$$\sigma_{repeatability} = \sqrt{\sigma^2}$$
$$\text{and} \quad \sigma_{reproducibility} = \sqrt{\sigma^2_\beta + \sigma^2_{\alpha\beta}}$$

give quantitative estimates of how repeatable and reproducible measurements are using that particular tool or method (see Vardeman and VanValkenburg (1999)). These estimates can be used to determine (1) whether a measurement method or process falls within established standards for measurement variability, and (2) what a reasonably-expected range of measurement values might be for a process which does not yet have established standards. The former can be used to assess a measurement system's repeatability and reproducibility as a whole, while the latter can be used to develop a metric to assess the quality of future measurements of similar objects.

Since the reproducibility of 3D scanning has not yet been addressed in the forensic firearms context, there is a need for both exploring the variability introduced as part of the measurement process and exploring data quality metrics based on the variability we observe. Here we will focus solely on measuring and exploring the variability introduced in the scanning process. Standards proposed in 2019 by the American Academy of Forensic Science Standards Board (ASB) only address measurement reproducibility for microscopes themselves using traditional microscope reproducibility techniques such as a sine wave tool, but not for microscopes as applied to bullet LEAs. The bullet LEA scanning process depends on operators identifying, staging, and capturing an entire surface which is then used for subsequent analysis. Therefore singular measurement repeatability and reproducibility is less relevant than how the measurement system reproduces results of the complex data analysis process. The application of high-resolution

microsocopy to bullet surfaces is a relatively new process, and thus there are not yet widely-accepted standards or standard operating procedures for its use.

The lack of formal assessment of the variability introduced by the scanning process leaves the field vulnerable to questions of reproducibility.

3.1.2 Repeatability and Reproducibility in Automated Bullet Matching

In the forensic firearms context, there are multiple sources of variability introduced in the process of comparing a known bullet to a questioned bullet.

First, when comparing a bullet from a known barrel to a questioned bullet from a crime scene, it is important to address whether quantitative similarity metrics resulting from data analysis will differ based on which bullet is fired from the known barrel. That is, we want to make sure striation patterns engraved by the same barrel are reliably similar within a set of bullets. Examiners may collect several "test fires" from a questioned firearm to use for toolmark comparison to a questioned bullet, and it is of interest whether automated algorithms using scan data will produce similar results regardless of which test fire is used for comparison.

One of the most significant factors with the potential to impact reproducibility across bullets is breakoff patterns on bullets. Two scans of the same LEA from the same bullet, as seen in Figure 3.1, are visually very similar. The same striation patterns (vertical lines engraved on the surface) and breakoff patterns (areas near the bottom of the scan which are disruptions of the overall structure) are observed on both scans. In contrast, LEAs generated by the same barrel but on different bullets, as in Figure 3.2, have very similar striation patterns but have different breakoff patterns. The two objects in Figure 3.2 are less similar than the objects in Figure 3.1.

Figure 3.1 Rendered images of data captured from two scans of the same land engraved area on the same bullet. The scans are visually very similar, but not completely identical. These scans were gathered by the same operator on two different machines.

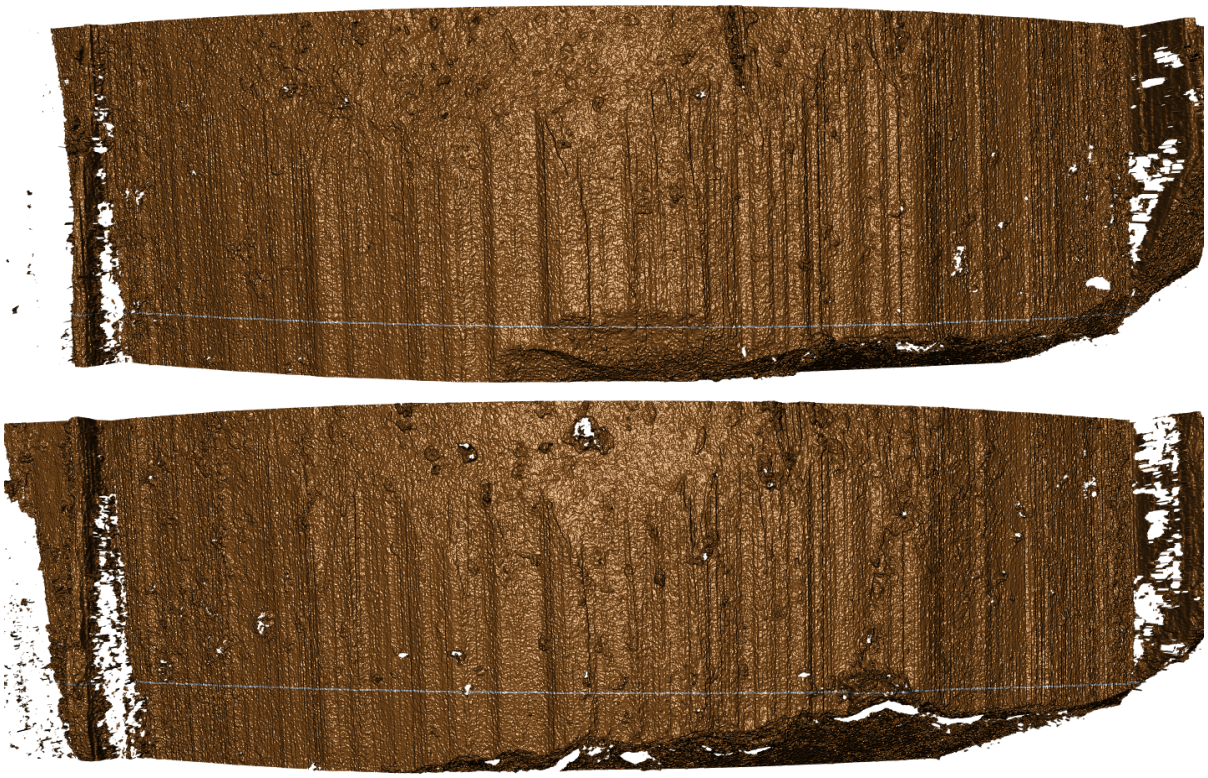

Figure 3.2 Rendered images of data captured from two scans of the same land engraved area on two different bullets. The striation patterns are very similar, but differences in breakoff patterns can be seen for the two bullets. These scans were gathered by the same operator on the same machine.

A second source of variability is introduced during the process of translating physical lands into 3D data objects. As demonstrated by Figure 3.1, measurements of the same physical object are very similar but not identical. The two scans pictured in Figure 3.1 were scanned by the same operator on two different machines. Both operator and machine are sources of variability introduced by translating bullet LEAs into measured data objects.

Thus, we will study the variability observed from three major sources: differing parts (here bullets), differing devices (here microscopes), and differing operators. Automated matching algorithms rely on the data captured by an operator on a microscope. In order to be considered a reproducible data analysis process, an algorithm should reach the same conclusions independent of which test fire (bullet), device, and operator generated the data for a given pair of bullets used to compare striation patterns.

In the case of bullet LEAs, however, the data used are not raw scans; the data are processed. The measurement process, depicted in Figure 3.3, can be assessed at multiple stages.

Traditional Gauge R&R studies model a one-dimensional measurement, typically a single numerical value, acting as the response variable. The LEA scanning process results in a 3D data object, which is then used to extract a 2D signature and ultimately a numerical value – in the form of a random forest score – for a pairwise comparison. To address the variability in the automated matching process, we focus on two stages of the process with two data formats.

The path from 3D object to pairwise similarity score, depicted in Figure 3.4, includes two major types of data that need to be studied. The first is the 2D signature data that is extracted through several processing steps applied to the raw 3D scan objects. There are two main reasons to focus on the variability of signature data: first, because processed signature data is the input data used in the Hare et al. (2017) matching algorithm, and output of any numerical algorithm is dependent on the input. Secondly, due to the relative nature of 3D height measurements, the lack of accurate 3D alignment technology to align multiple data objects, and the complex

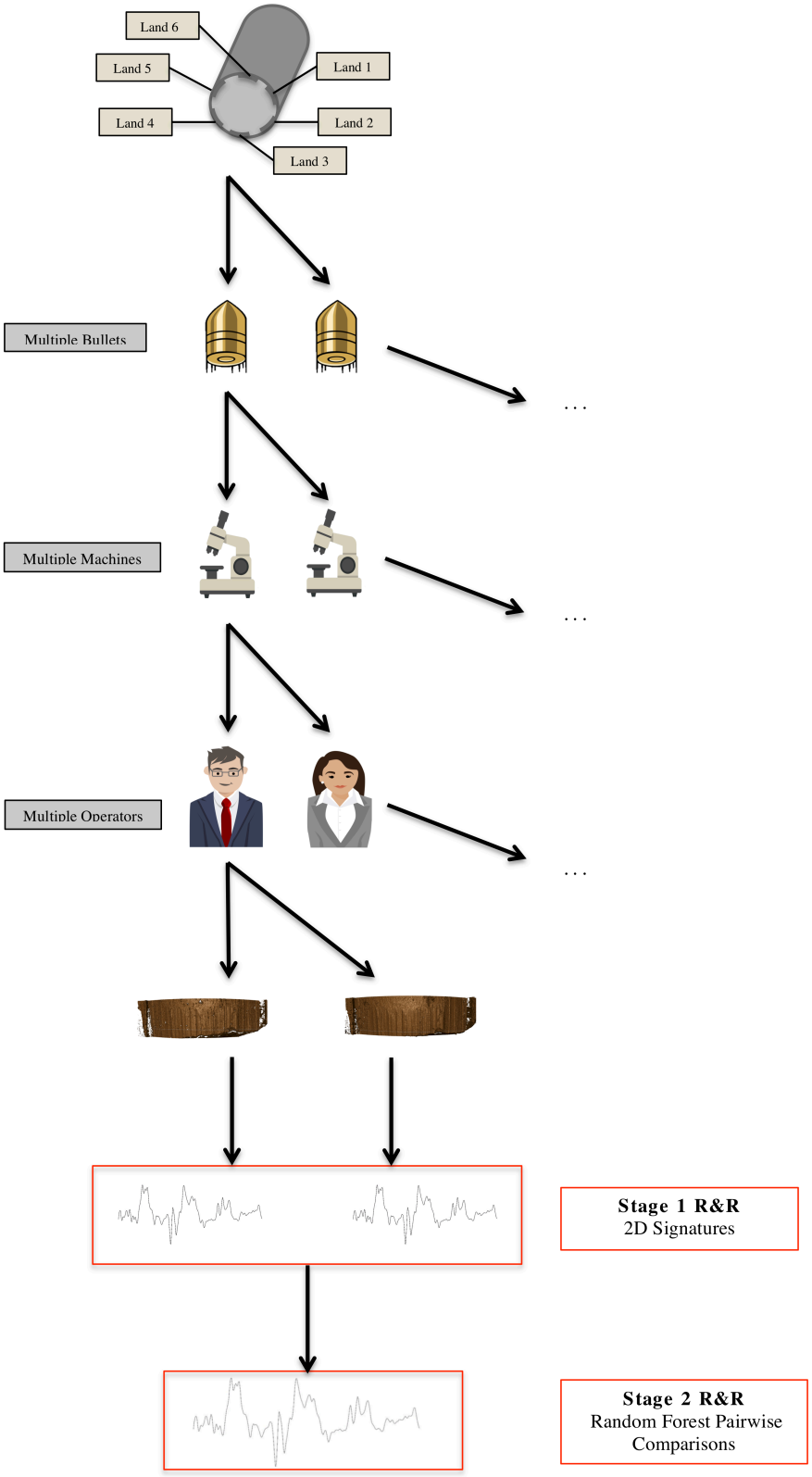

Figure 3.3 A diagram of the bullet LEA scanning process. We assess repeatability and reproducibility at two stages within this process. Reproducibility is assessed for different bullets from the same barrel, different microscopes, and different operators.

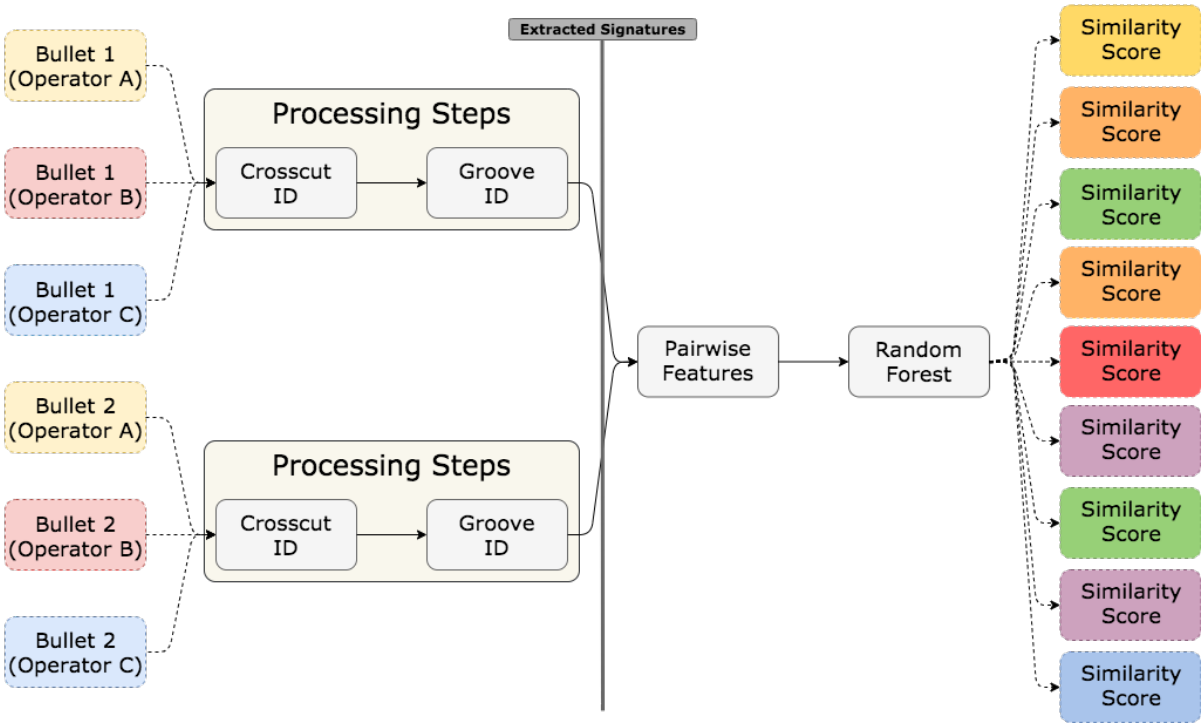

Figure 3.4 Depiction of the bullet comparison process for two bullets when multiple operators scan each bullet. The process can be thought of as a two-stage set of inputs and outputs. Raw data as input goes through processing steps to extract a signature for each data file. Paired signatures are then used as input for a pairwise modeling scheme which results in a similarity score for the paired objects. For two bullets and three operators, we could observe 9 unique similarity scores. Similarity score color denotes the pair of operators present in the paired comparison.

dependence structure of the 3D objects, it is unfeasible and impractical to estimate random effects on the raw 3D data. At the 2D signature level, we want to examine the degree to which signatures vary. Do the signatures differ significantly across bullets, operators, or machines?

In addition to estimating variability of the input data (processed signatures), we can focus on the variability of the output, which for the Hare et al. (2017) algorithm is pairwise similarity scores resulting from extracting pairwise features and applying a random forest model. To what degree do the resulting similarity scores vary depending on bullet of origin, operator, or machine? The stability of pairwise scores across multiple microscopes and operators is perhaps the most important to assess in this process.

To quantify the variability and assess these questions, we designed a repeatability and reproducibility study of the 3D scanning process and collected over 3,500 LEA scans. In the following work, we describe our study design, collected data, model specifications, and adapted Gauge R&R process. We also present the resulting estimates and conclusions on the reproducibility of the automated matching process.

3.2 Study Design

We first introduce vocabulary and components of interest in the study.

The patterns engraved on bullet LEAs are created by contact with a specific **land** within the rifling of a gun barrel. When two LEAs are compared, they are considered to be from the same source if they were created by the *same land in the same barrel*. We will refer to a unique land in a barrel as a **barrel-land**. The barrel-land is the primary object which generates the patterns observed on the secondary objects, the bullets. A visual of the physical barrel-lands can be seen in Figure 3.5.

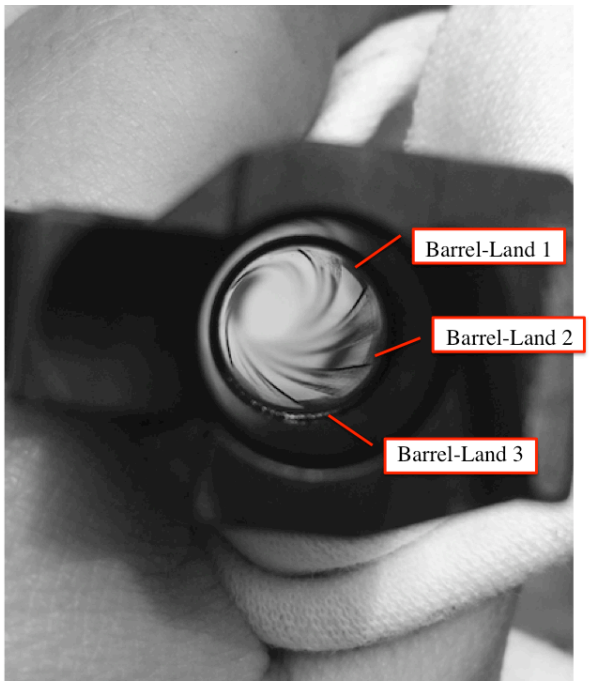

Figure 3.5 A diagram of barrel-lands within the rifling of a gun barrel. Many types of gun barrel, including the barrel pictured here, have six lands.

Since multiple LEAs appear on a single bullet, it would seem intuitive to nest land within bullet, rather than bullet within land. However, we conceptualize the barrel-land as the distinct *pattern* and LEAs found on bullets are *repeated engravings* of that pattern.

Consider the parallel situation in forensic fingerprint analysis. The ridge pattern on a finger is considered the *pattern*, and latent prints left at a crime scene are considered *impressions* of that pattern.

We consider barrel-lands to be the physical units – or **parts** – we are interested in measuring. Within the measurement process, the bullet, operator, and microscope are aspects we vary and model. Each barrel-land produces repeated patterns we can reasonably expect to be similar but not identical. Repetitions of striation patterns from a single barrel-land parallel the $i = 1, \ldots, n$ parts defined in the traditional R&R model in subsection 3.1.1.

Our primary set of research questions aims to assess the repeatability and reproducibility of two-dimensional signatures extracted as part of the automated bullet matching algorithm in Hare et al. (2017). Those questions are as follows:

1. How **repeatable** are signatures when collected under the same experimental conditions?
    - Same operator, same device, same **part (same barrel-land on same bullet)**
2. How **reproducible** are signatures of the same part when collected by different operators, controlling for machine?
3. How **reproducible** are signatures of the same part when collected on different machines, controlling for operator?
4. Does the **reproducibility** of signatures of the same part differ for machines depending on operator?
5. Does the **reproducibility** of signatures change when the part changes (differences in bullet for the same barrel-land)?

Our second set of research questions aims to assess the repeatability and reproducibility of pairwise matching scores calculated using pairwise features and Hare et al. (2017)'s random forest algorithm:

1. How **repeatable** are pairwise similarity scores comparing two signatures scanned under the same experimental conditions?
    - Same operator, device, and part
2. How **reproducible** are pairwise similarity scores comparing two signatures when scans originate from two different bullets?

3. How **reproducible** are pairwise similarity scores comparing two signatures when scanned by two different operators on the same microscope?

4. How **reproducible** are pairwise similarity scores comparing two signatures when scanned on two different devices by the same operator?

5. How **reproducible** are pairwise similarity scores comparing two signatures when scanned by two different operators on two different devices?

3.2.1 Defining Repetition

Prior to data collection and quantitative assessment of measurement repeatability, which focuses on repeated measurements taken under the same environmental conditions, we first must define the nature of repetition in the bullet LEA scanning process. There are two different procedures which can be used to capture scan replicates: **immediate recapture** and **restaging**.

Immediate recapture data are repetitions in which the LEA is staged under the microscope by an operator, and the operator has the machine capture multiple sets of measurements in quick succession without making any adjustments. **Restaging** data are repetitions collected by the same operator on the same machine, but are collected on different days or at different times. Operators must restage the bullet for each repetition, starting from scratch.

In the forensic context, restaging is the more relevant of the two repeated capture methods. Consider a scenario in which a firearms examiner evaluates whether two bullets were fired by the same gun. For the process to be considered repeatable, the examiner should be able to come back to the same comparison a week, a month, or even a year later and still reach the same decision. Therefore, repetitions taken at different times and with new restaging required more closely parallels the question of repeatability as it applies to human visual comparison.

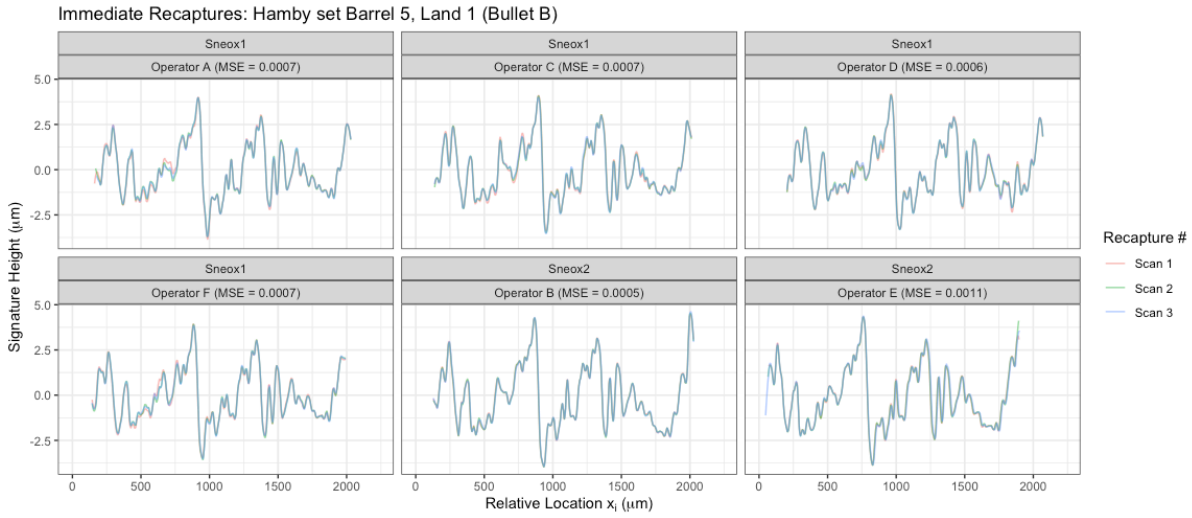

Figure 3.6 Example of immediate recapture data for six operators. Extremely small mean squared error values for each set of recaptures indicate that in many cases, differences due to recapture are negligible.

The decision to define repetitions as restages in the study is also supported by immediate recapture data collected as part of a small pilot study. Extracted signature data from three immediate recapture repetitions from six operators are shown in Figure 3.6. The signatures are nearly identical when graphed together; in addition, each set of repetitions has extremely small mean squared error values ($0.0011\mu m^2$ or less for those pictured in Figure 3.6). Thus, the variability introduced by immediate recapture is miniscule and uninteresting with respect to the larger goals of the study. However, there is one interesting phenomenon to take note of that we observe in the immediate recapture data: recapture drift.

Recapture drift can occur when there is extended time between two recaptures of the same staged LEA. Measured height values are mostly unaffected, by scans are shifted in the x direction, resulting in larger values of the mean squared error (MSE). Figure 3.7 provides examples of recapture drift observed in the study, and demonstrates that coaligning the scans in the x direction removes this effect. While recapture drift will not affect restaging repetitions,

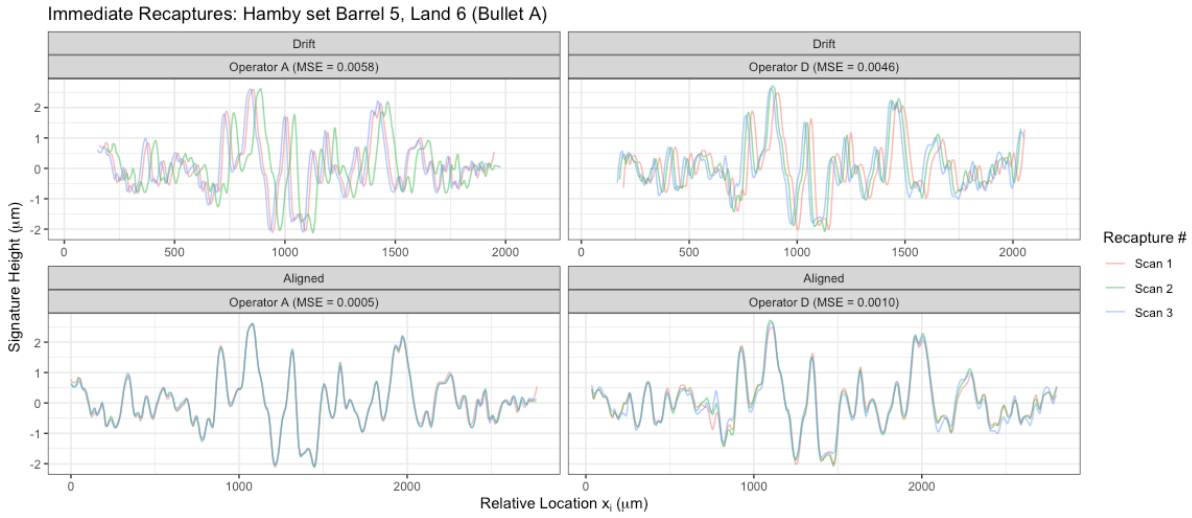

Figure 3.7 (Top) Two examples of recapture drift. Drift can occur when operators leave an LEA staged for a period of time before collecting recapture data rather than gathering captures in immediate succession. (Bottom) Recapture drift is removed by co-aligning scans in the x direction.

Table 3.1 Firearm barrel information. Three barrels were used, with names coded for operator blinding. Each barrel has a 9mm caliber, but each is a different barrel type.

Coded Name	Test Set Name	Barrel Type	Caliber	Ammunition Details
Barrel Orange	Hamby set 224	Ruger P-85	9mm	Winchester 9mm copper
Barrel Pink	Houston set 3	Ruger LCP	9mm	American Eagle 124-grain 9mm copper
Barrel Blue	LAPD	Beretta 92 F/FS	9mm	Winchester 115-grain 9mm copper

note that all signatures in our study are coaligned in the x direction prior to modeling variability to remove unrelated differences in relative x direction.

3.2.2 Scale and Scope of Study

In order to address the proposed research questions on repeatability and reproducibility, we defined the scale of the study in the following way.

A total of nine bullets, three bullets each from three barrels, were selected for repeated scanning. The three barrels are each of a different type, and were given coded barrel names so operators had no prior information about the source of bullets. Details on each barrel used are given in Table 3.1. Different barrel types were used to widen the scope of our study and to investigate whether variability differs for different barrel and bullet types.

Each bullet was labeled within its barrel as Bullet 1, 2 or 3 in order to ensure each LEA scan originating from "Bullet 1" for a particular barrel is a scan of the same part, as it is a factor of interest in the study.

Each microscope operator was tasked with collecting between three and five repeated scans of each bullet on each microscope, spread out over time as a series of "Rounds". A single round consisted of a capture of each LEA on each bullet once on Machine 1 and once on Machine 2. Operators were directed to complete each round in its entirety before moving to the next round.

Data was collected in two phases, with Phase 1 containing Barrels Orange and Pink, and Phase 2 containing Barrels Orange, Pink, and Blue. A total of eight operators completed at least three rounds by the end of Phase 2, with a total of seven operators completing five complete rounds of Barrels Orange and Pink. Due to laboratory staffing changes prior to Phase 2, only five operators scanned Barrel Blue, but all five completed five rounds on Barrel Blue.

An overview of the collected data is displayed in Table 3.2.

A visual overview of the set of repeated signatures extracted from repeated scans is presented in Figure 3.8 for one barrel-land from each of the three barrels in the study.

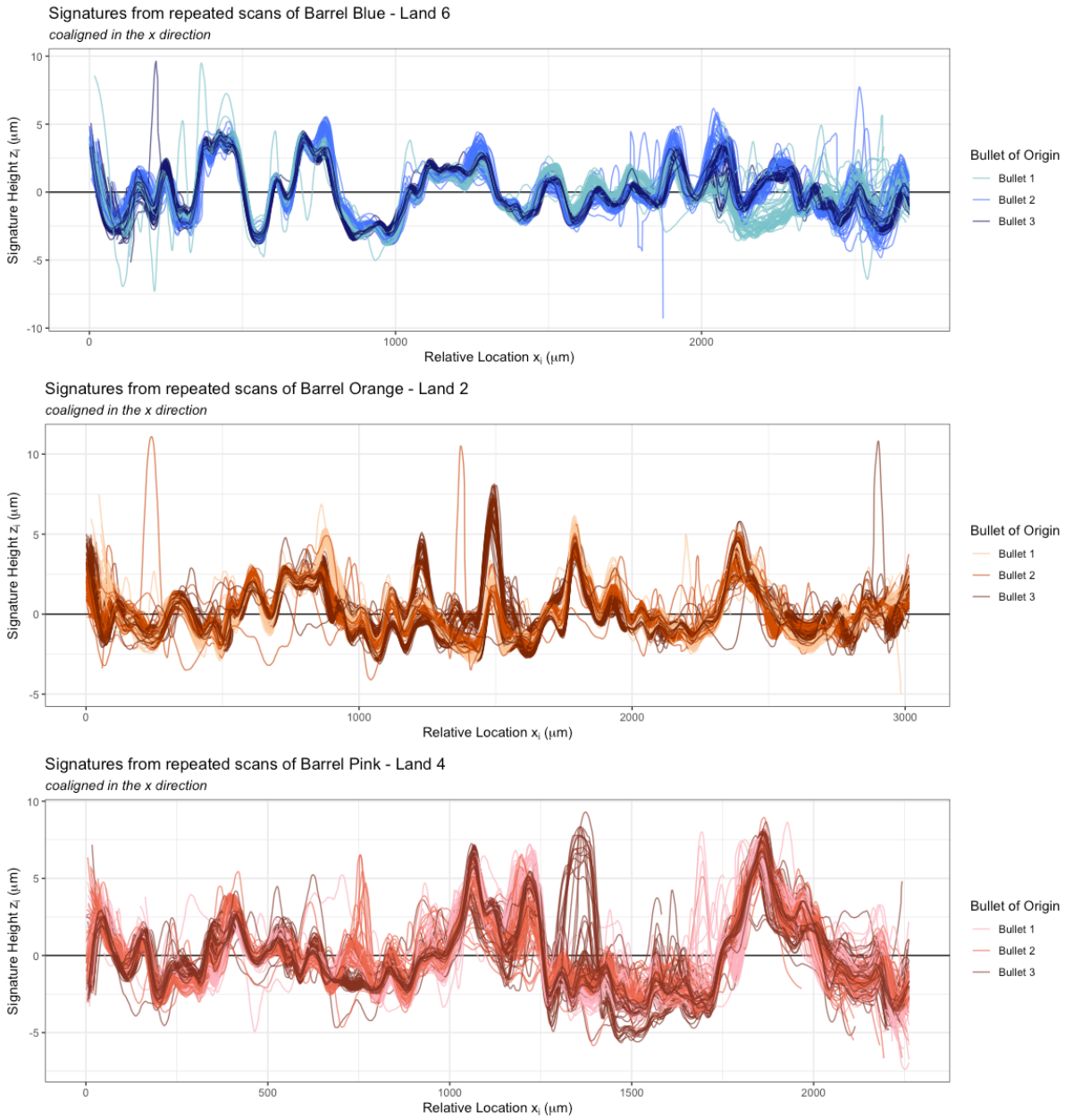

Figure 3.8 A visualization of all signatures collected for three example barrel-lands, one from each barrel. Signatures are colored by bullet of origin, for Bullets 1, 2, and 3 within each barrel.

Table 3.2 Number of repetitions gathered for each bullet, machine, and operator combination. Repetitions shown in purple were scanned during Phase II of the study.

Bullet	Machine	**Op. A**	**Op. B**	**Op. C**	**Op. D**	**Op. E**	**Op. F**	**Op. G**	**Op. H**
O-1	Sneox1	3	5	5	5	5	5	5	5
	Sneox2	3	5	5	5	5	5	5	5
O-2	Sneox1	3	5	5	5	5	5	5	5
	Sneox2	3	5	5	5	5	5	5	5
O-3	Sneox1	3	5	5	5	5	5	5	5
	Sneox2	3	5	5	5	5	5	5	5
P-1	Sneox1	3	5	5	5	5	5	5	5
	Sneox2	3	5	5	5	5	5	5	5
P-2	Sneox1	3	5	5	5	5	5	5	5
	Sneox2	3	5	5	5	5	5	5	5
P-3	Sneox1	3	5	5	5	5	5	5	5
	Sneox2	3	5	5	5	5	5	5	5
B-1	Sneox1		5			5	5	5	5
	Sneox2		5			5	5	5	5
B-2	Sneox1		5			5	5	5	5
	Sneox2		5			5	5	5	5
B-3	Sneox1		5			5	5	5	5
	Sneox2		5			5	5	5	5

3.2.2.1 Imbalance in data collection

It is important to note that while in principle the study design allows for a relatively balanced data structure, data collection errors in a small subset of scans result in some additional imbalance. One type of error observed multiple times in our collected data is accidental double-scanning. Double-scanning can occur when the operator rotates the staged bullet to scan the subsequent LEA, and mistakenly under- or over-rotates, shown in Figure 3.9. Practically, this means the data contains one extra scan of a particular barrel-land and one less scan of a neighboring barrel-land that was skipped.

When modeling variability, imbalance due to double-scanning should have a minimal effect given that there are still 200+ repetitions total for each Orange and Pink barrel-land, and close to 150 repetitions for each Blue barrel-land. However, in the interest of transparency it is important to note that errors of this type do exist in our data.

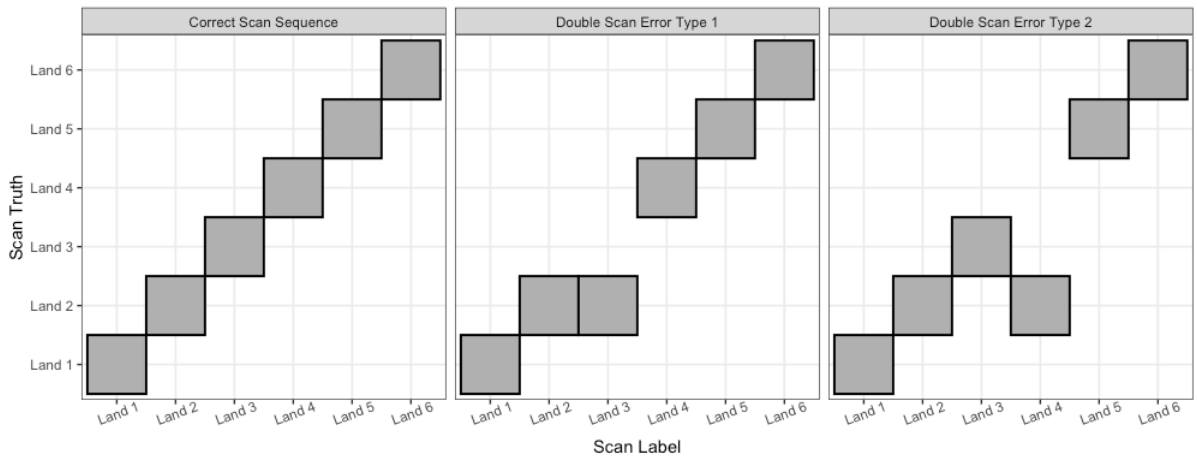

Figure 3.9 Example of scanning sequence errors which result in imbalanced numbers of repetitions for some barrel-lands. (Left) A correct scan sequence, where the scan label matches the true land of origin. (Middle) A scan sequence in which the operator did not rotate far enough and mistakenly scanned the same land twice. (Right) A scan sequence in which the operator rotated backwards during staging and accidentally scanned the same land twice.

A second source of imbalance present in our data is due to the data format. Signatures cannot be assumed to be the same length; one coaligned by maximum cross-correlation function, signatures also begin and end at slightly different x_i locations. Due to this, the number of x_i locations for each signature will be imbalanced when location is considered in modeling. To ameliorate this imbalance, we trim the "tails" of signatures for each barrel-land. To accomplish this, we identify the largest minimum x_i value by barrel-land, as well as the smallest maximum x_i value by barrel-land, and remove all x_i values outside of that range for signature-level modeling. This "tail-trimming" process can be shown for three example barrel-lands in Figure 3.10.

While data is trimmed in the tails to account for location imbalance when modeling signatures, note that no trimming is done prior to calculation of pairwise features and pairwise similarity scores. The Hare et al. (2017) bullet analysis process does not perform any trimming besides GEA data removal and thus it is inappropriate to remove data from the tails when quantifying reproducibility of the scores resulting from the pairwise matching algorithm.

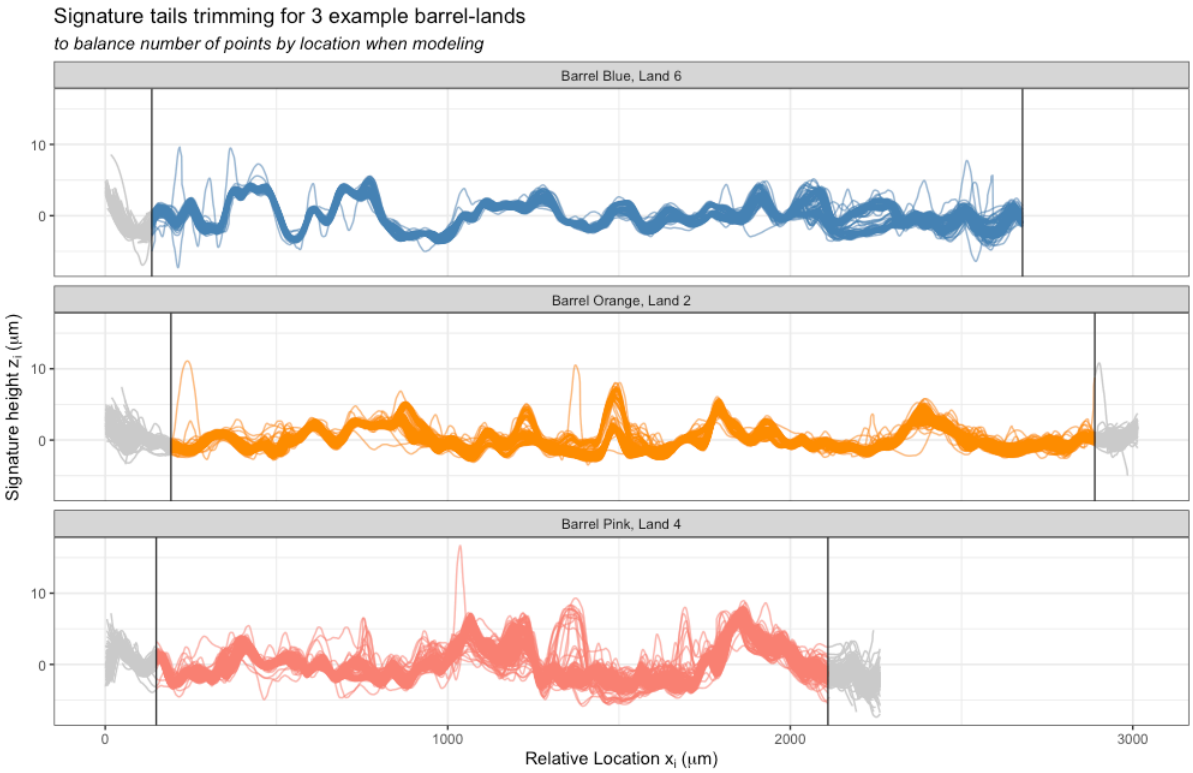

Figure 3.10 Examples of the signature tail trimming process used to balance the number of x locations considered for signature-level modeling. All tails, shown in grey, are removed for signature-level modeling.

We next detail our approaches to modeling repeatability and reproducibility at the signature level and pairwise level.

3.3 Model Specification

We will define two separate models; one for measuring variability at the signature level, and one for measuring variability of pairwise scores. Our study follows a three-factor repeatability and reproducibility framework. For both signatures and pairwise comparisons, we will thus begin with the traditional 3-factor model. Let z_{jkmn} be some measured quantitative response value for bullet (part) j, operator k, device m, and repetition n, and define the model as follows:

$$z_{jkmn} = \mu + b_j + o_k + d_m + bo_{jk} + bd_{jm} + od_{km} + bod_{jkm} + e_{jkmn} \tag{3.2}$$

with a fixed, unknown measurement average μ, and random effects

b_j for Bullet j, following a $N(0, \sigma_b^2)$

o_k for Operator k, following a $N(0, \sigma_o^2)$

d_m for Device m, following a $N(0, \sigma_d^2)$

bo_{jk} for Bullet $j-$Operator k, following a $N(0, \sigma_{bo}^2)$

bd_{jm} for Bullet $j-$Device m, following a $N(0, \sigma_{bd}^2)$

od_{km} for Operator $k-$Device m, following a $N(0, \sigma_{od}^2)$

bod_{jkm} for Bullet $j-$Operator $k-$Device m, following a $N(0, \sigma_{bod}^2)$

e_{jkmn} is error across repetitions, following a $N(0, \sigma^2)$.

We assume each b_j, o_k, d_m, bo_{jk}, bd_{jm}, od_{km}, bod_{jkm} and e_{jkmn} are independent random variables and $\sigma^2_\cdot$ are variance components.

This model is an expanded version of the traditional two-factor model described in subsection 3.1.1. However, in both of our modeling situations – signatures and pairwise scores – this model alone is inappropriate. When modeling signatures, we need to carefully specify the fixed effect structure as well as address issues of inherent dependency in the signature data structure. When modeling pairwise scores, we need to adapt the framework to capture the pairwise nature of the response variable.

3.3.1 Two Dimensional Signatures

To model the measurement variability in two-dimensional signatures, we must consider a more complex specification of the mean structure μ. We first include a mean by location in the model:

$$z_{ijkmn} = \mu_i + b_j + o_k + d_m + bo_{jk} + bd_{jm} + od_{km} + bod_{jkm} + e_{ijkmn} \tag{3.3}$$

with a fixed, unknown mean by location x_i, μ_i, and random effects

b_j for Bullet j, following a $N(0, \sigma^2_b)$

o_k for Operator k, following a $N(0, \sigma^2_o)$

d_m for Device m, following a $N(0, \sigma^2_d)$

bo_{jk} for Bullet $j-$Operator k, following a $N(0, \sigma^2_{bo})$

bd_{jm} is a random effect for Bullet $j-$Device m, following a $N(0, \sigma^2_{bd})$

od_{km} for Operator $k-$Device m, following a $N(0, \sigma^2_{od})$

bod_{jkm} for Bullet $j-$ Operator $k-$ Device m, following a $N(0,\sigma^2_{bod})$

e_{ijkmn} is error across repetitions, following a $N(0,\sigma^2)$

We assume each b_j, o_k, d_m, bo_{jk}, bd_{jm}, od_{km}, bod_{jkm}, and e_{ijkmn} are independent random variables, and each $\sigma^2_{\cdot}$ is a variance component.

The inclusion of a mean by location x_i as a fixed effect is primarily due to our research question. Our focus lies in estimating differences in measured values given differing environmental conditions applied to a pattern – in our case, the striation pattern observed as a signature. Therefore, there is inherent variability due to pattern structure present in the data. This structure is important for subsequent pairwise modeling, but is here most appropriately captured as a fixed mean effect. Estimates of variance components for study factors (bullet, operator, device) are then estimates of variability in measured height *after accounting for signature structure by location*. The difference in mean structure between Equation 3.2 and Equation 3.3 is depicted in Figure 3.11.

3.3.1.1 Signature model grouping structure

In addition to including location as a fixed effect, we adjust our random effect grouping structure following a similar logical argument.

The traditional three-factor Gauge R&R model specification as stated in Equation 3.2 defines a random effect for each factor in the study as well as the interactions between all crossed factors. For example, the component for bullet in Equation 3.2, b_j, is defined as following a $N(0,\sigma^2_b)$ distribution. In order to estimate σ^2_b, data are grouped by bullet regardless of location. In our study, this would result in three groups within the data; one for Bullet 1, one for Bullet

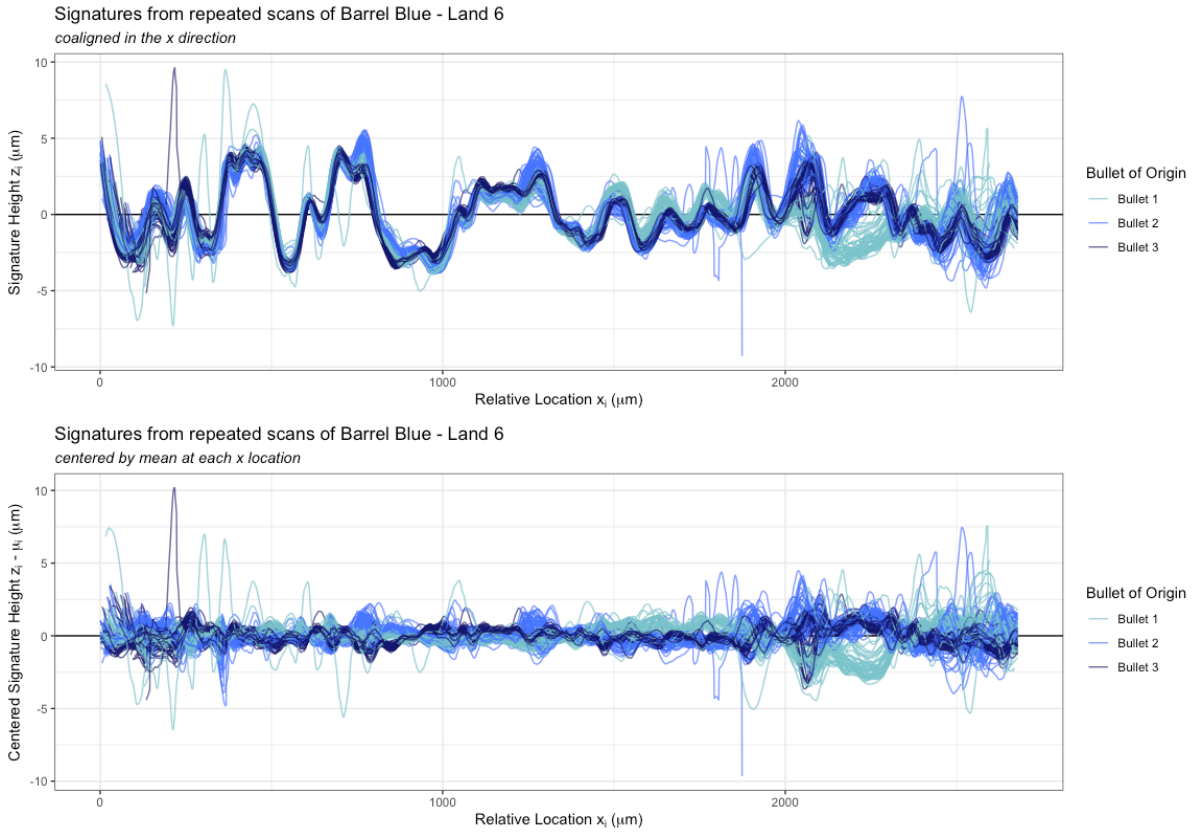

Figure 3.11 Example for Barrel Blue - Land 6 of the difference in data structure for signatures (top) and signatures with mean by location removed (bottom). Including the mean by location as a fixed effect in the model results in the random effects being modeled using data shown in the bottom plot, with much of the structure accounted for.

2, and one for Bullet 3. In order to capture any variability in machine measurements through estimated variance components, the three groups would have to differ in a consistent manner across the signature. In other words, one bullet must have larger measured values than the other regardless of whether the measurement is taken in a peak or a valley on the signature.

The aforementioned grouping structure therefore does not address the research questions of interest. Defining a bullet effect as b_j where $j = 1,2,3$ does not provide a quantification of differences in resultant data; rather, as depicted in Figure 3.12, it demonstrates an exercise in regression to the mean value for a bullet across the entire signature, which we expect to be close to zero. Figure 3.12(a) depicts a small subset (300 x_i locations wide) of repeated signature data for one example barrel-land, and Figure 3.12(b) depicts that same subset centered by location x_i, which depicts the data we will see after accounting for the location fixed effect, μ_i. If we then consider the bullet effect b_j, the effect estimates the difference in distribution of centered values by bullet, depicted in Figure 3.12(c). The distributions shown in (c) have very similar centers, as they aggregate a multitude of centered heights across x_i locations. If we consider the differences in those distributions at a small subset of locations, shown in Figure 3.12(d), we can see major differences in the distributions by location. This phenomenon is not surprising. Particularly in peaks and valleys, one bullet or one machine may have more extreme values; in a peak, their value might be higher, but in a valley, the value might be lower. If we do not consider the location when modeling differences by study factors (bullet shown in Figure 3.12), we incorrectly specify differences in measured height distributions and will therefore underestimate relevant variance components.

Therefore, to more adequately address the research questions of interest, we group each study factor by location. We achieve this in practice by including the x location x_i as an interaction with each previously defined random model component. The resultant model is then:

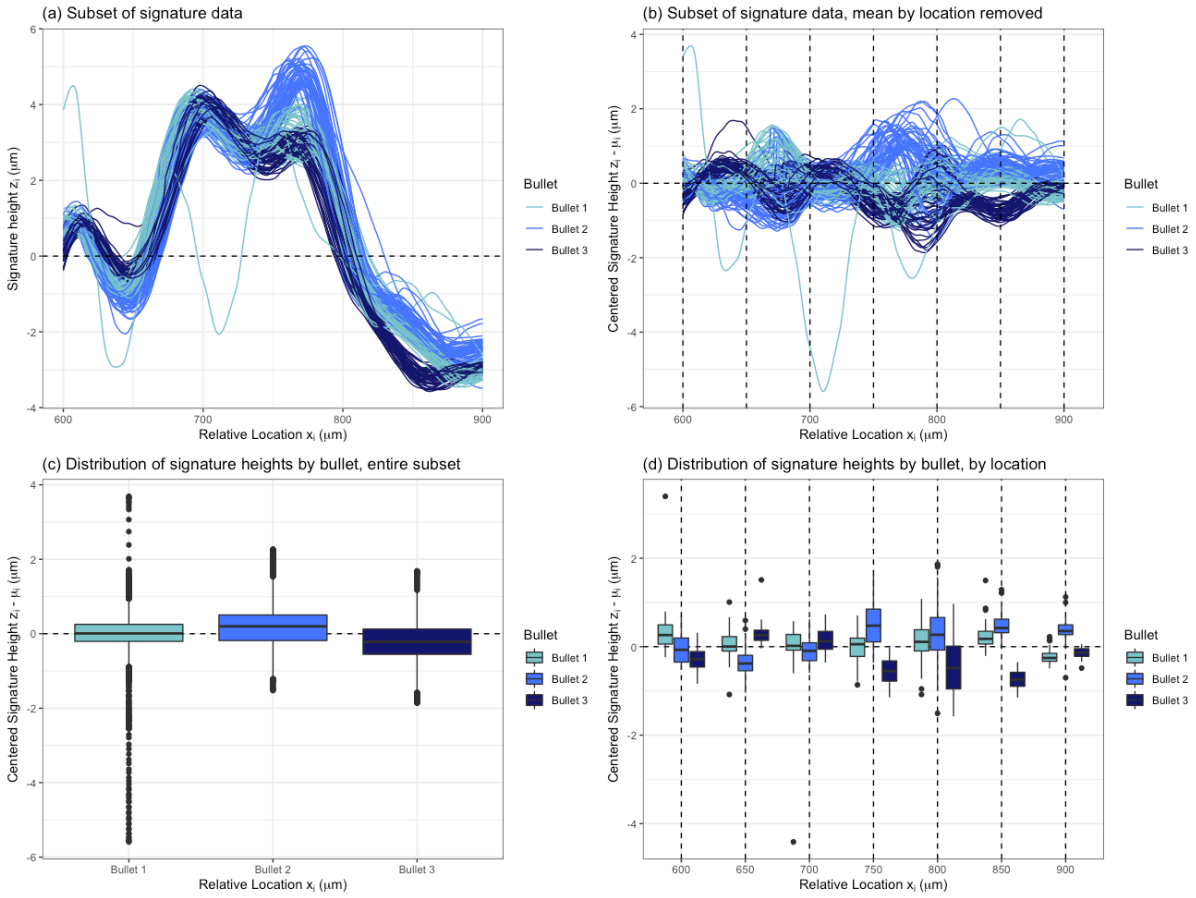

Figure 3.12 Example for Barrel Blue - Land 6 of the difference in data distributions by bullet when considering bullet groupings and bullet by location groupings. (a) A small subset ($x_i \in (600, 900)$) of data on repeated Barrel Blue - Land 6 signatures. (b) The subset in (a) with heights centered by location x_i which here represents the specified mean structure, μ_i. (c) The distribution of centered height values by bullet for all $x_i \in (600, 900)$. When modeling bullet variability, this depicts how similar the average height values will be across bullet. (d) The distribution of centered height values by bullet, considering a small sampling of locations.

$$z_{ijkmn} = \mu_i + b_{ij} + o_{ik} + d_{im} + bo_{ijk} + bd_{ijm} + od_{ikm} + bod_{ijkm} + e_{ijkmn} \tag{3.4}$$

with a fixed, unknown measurement average by location, μ_i, and random effects

b_{ij} for Bullet j by location i, following a $N(0, \sigma_b^2)$

o_{ik} for Operator k by location i, following a $N(0, \sigma_o^2)$

d_{im} for Device m by location i, following a $N(0, \sigma_d^2)$

bo_{ijk} for Bullet j – Operator k by location i, following a $N(0, \sigma_{bo}^2)$

bd_{ijm} for Bullet j – Device m by location i, following a $N(0, \sigma_{bd}^2)$

od_{ikm} for Operator k – Device m by location i, following a $N(0, \sigma_{od}^2)$

bod_{ijkm} for Bullet j – Operator k – Device m by location i, following a $N(0, \sigma_{bod}^2)$

e_{ijkmn} is error across repetitions, following a $N(0, \sigma^2)$.

We assume each b_{ij}, o_{ik}, d_{im}, bo_{ijk}, bd_{ijm}, od_{ikm}, bod_{ijkm}, e_{ijkmn} are independent random variables, and each $\sigma_{\cdot}^2$ is a variance component.

Following the model stated in Equation 3.5, we can estimate random components by location.

The inclusion of location as an interaction also makes estimation of variance components easier, as increasing the number of groups used for estimating each random component will decrease the incidence of model singularities.

However, there is still a major issue to address in our conceptualized model. As alluded to throughout this chapter, the overall structure of the data as a signature comprised of peaks and valleys (the striae we want to capture as a signal) creates inherent dependence in the data by location.

3.3.1.2 Subsampled signature model

To address the dependency by location, we offer a two-pronged argument for a model which uses subsampled data to estimate model components. Our proposed model is:

$$z_{ijkmn} = \mu_i + b_{ij} + o_{ik} + d_{im} + \tag{3.5}$$

$$bo_{ijk} + bd_{ijm} + od_{ikm} + bod_{ijkm} + e_{ijkmn} \tag{3.6}$$

for subsampled location indices

$$i = \ell, \ell + w, \ell + 2w, \ldots, \ell + cw$$

for some starting x index $\ell \in \mathbb{N}$, window size between sampled points $w \in \mathbb{N}$, and positive integer c such that $\ell + cw < I_{BL}$, where I_{BL} is the number of possible x_i locations on signatures for barrel-land BL. We propose a set of indices i created with $w = 100$.

We assume a fixed, unknown measurement average at location i, μ_i, and random effects

b_{ij} for Bullet j by location i, following a $N(0, \sigma_b^2)$

o_{ik} for Operator k by location i, following a $N(0, \sigma_o^2)$

d_{im} for Device m by location i, following a $N(0,\sigma_d^2)$

bo_{ijk} for Bullet $j-$Operator k by location i, following a $N(0,\sigma_{bo}^2)$

bd_{ijm} for Bullet $j-$Device m by location i, following a $N(0,\sigma_{bd}^2)$

od_{ikm} for Operator $k-$Device m by location i, following a $N(0,\sigma_{od}^2)$

bod_{ijkm} for Bullet $j-$Operator $k-$Device m by location i, following a $N(0,\sigma_{bod}^2)$

e_{ijkmn} is error across repetitions, following a $N(0,\sigma^2)$.

Again, we assume each b_{ij}, o_{ik}, d_{im}, bo_{ijk}, bd_{ijm}, od_{ikm}, bod_{ijkm}, e_{ijkmn} are independent random variables and each $\sigma_{\cdot}^2$ are variance components.

First, we argue that a model using data from every 100^{th} x location removes dependence between data points. Second, we argue that a subsample taken at every 100^{th} x location will provide variance component estimates that are consistent with estimates from data sampled on a finer grid which incorporate more data, but contain more underlying dependence.

To argue the first point, we can investigate the autocorrelation functions of the extracted signatures. Autocorrelation function, or ACF, is a measure of dependence typically used to investigate structures in dependent data, such as time series data. The autocorrelation function is defined as;

$$\gamma_Y(k) = Corr(Y_t, Y_{t+k}) \text{ for some lag } k \text{ and } \forall\ t = 1,\ldots,n-k,$$

for a series of dependent data points $Y_1, Y_2, \ldots, Y_n$. ACF is typically calculated for a number of lags, k.

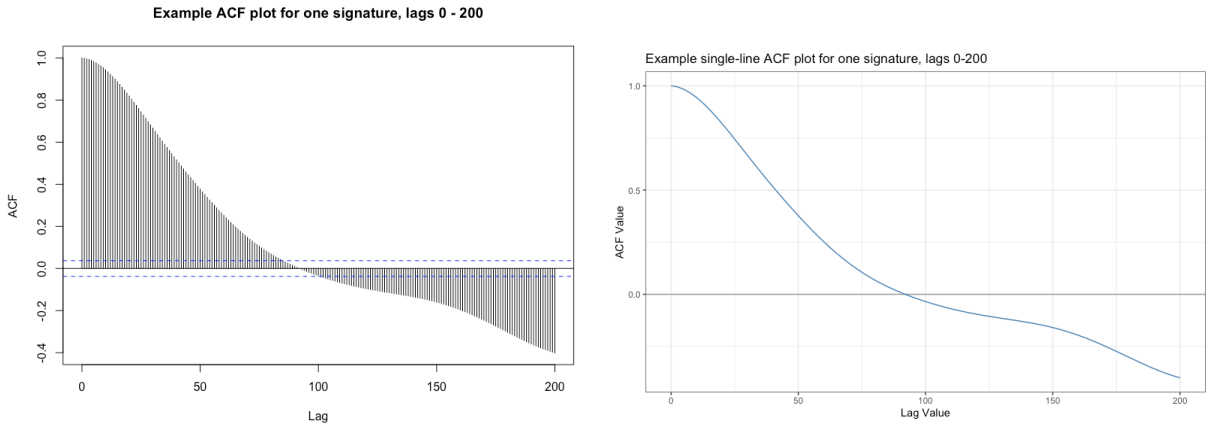

Figure 3.13 Autocorrelation function plots. (Left) Traditional style of ACF plot used in R. (Right) Single-line style of ACF plot, to be used to investigate multiple ACF functions for repeated scans simultaneously.

Each signature can be considered a dependent series of data $Z_1, Z_2, \ldots, Z_n$ and an ACF can be calculated and investigated visually. A traditional ACF plot in `R` is shown in Figure 3.13 along with a single-line representation of the function. We use the single-line version in order to more easily compare many ACFs simultaneously in Figure 3.14. Figure 3.14 demonstrates a steady decrease in autocorrelation function value, decreasing to near or below 0 by lag $k = 100$. This provides support for the argument that subsampling data at every 100^{th} x location, or using a window size $w = 100$, the dependence by location in the remaining data will be minimal to the point where we can assume independence by locations i in our model. This sampling scheme is represented in Figure 3.15.

Figure 3.15 also demonstrates the extent of data reduction in the subsampled model with $w = 100$. We therefore also approach the defense of the subsampled model from the perspective of information loss. This is accomplished by fitting the proposed model not only at every 100^{th} x location, but additionally investigating a range of window sizes w and by fitting multiple phases of each model for each window size w. For example, if we consider window size $w = 100$ and investigate 10 phases of the $w = 100$ model, we use the following 10 subsets for location x_i:

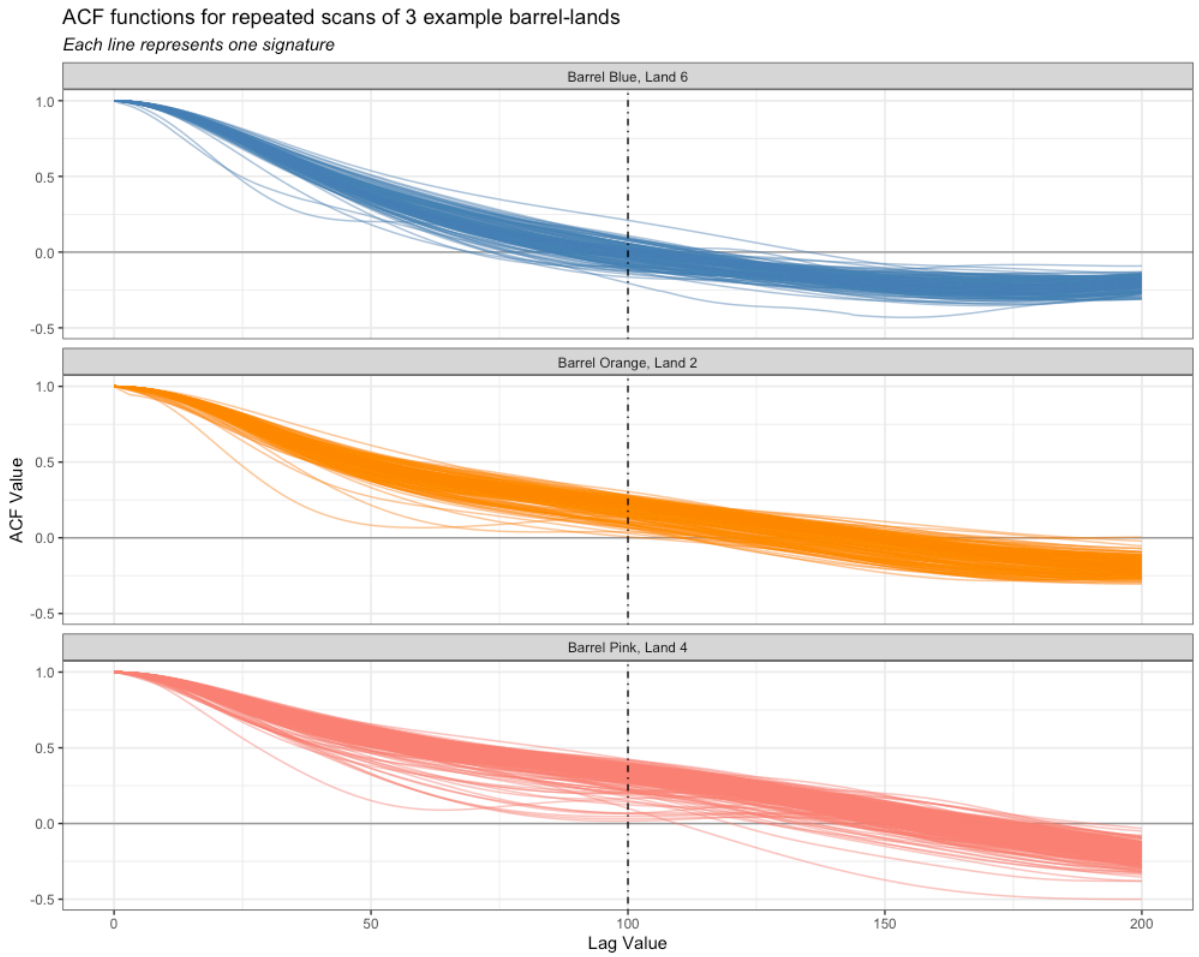

Figure 3.14 Autocorrelation functions for lags 0 to 200 for repeated scans of three example barrel-lands. Each line represents the relationship between lag and autocorrelation value for a single signature. The vertical black line represents lag k = 100.

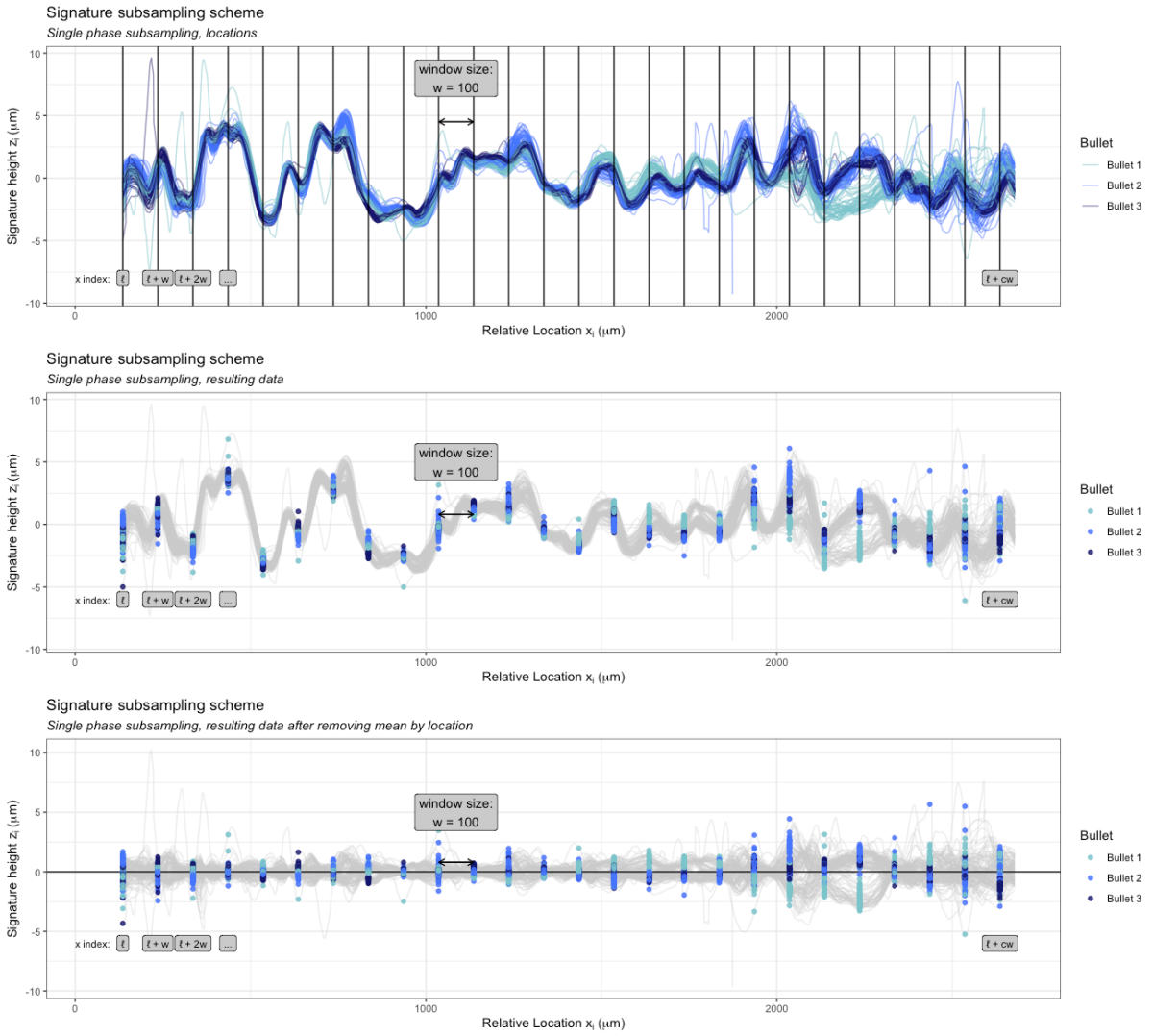

Figure 3.15 A visual depiction of the subsampling scheme proposed in Equation 3.6, sampling data at every 100^{th} x_i location. (Top) Subsampling locations on a signature. (Middle) Resulting data points extracted from subsampling locations. (Bottom) Data points centered by mean by x_i location, our proposed model fixed effect.

$$
\begin{aligned}
i_1 &= \ell, && \ell+100, \ell+2\times 200, \ldots, && \ell+c\times 100\\
i_2 &= \ell+10, && \ell+10+100, \ell+10+2\times 100, \ldots, && \ell+10+c\times 100\\
i_3 &= \ell+20, && \ell+20+100, \ell+20+2\times 100, \ldots, && \ell+20+c\times 100\\
&\vdots = \vdots &&&&\\
i_{10} &= \ell+90, && \ell+90+100, \ell+90+2\times 100, \ldots, && \ell+90+c\times 100
\end{aligned}
$$

This proposed set of indices $(i_1, \ldots, i_{10})$ is represented visually in Figure 3.16. Figure 3.16(bottom) demonstrates that one-tenth of data are used with a ten-phase approach and window size $w = 100$. We investigate the impact of that information loss by fitting a series of models, with window sizes $w = (10, 20, \ldots, 100)$, and a ten-phase approach for each model. We therefore get ten models for each window size w and investigate the resulting variability estimates to determine whether $w = 100$ will adequately capture the components of interest. Note that with window size $w = 10$ and ten phases, every single x location is utilized within one of the ten phased models.

The resulting model estimates, shown for a sampling of one barrel-land in each barrel in Figure 3.17, demonstrate three major properties. First, estimated variance components remain consistent on average as the window size w increases. Secondly, the variability of the variance components increases as window size increases. This is expected, as the model sample size decreases with larger window size w and less data are being captured *per model*, leading to estimates which naturally vary more. Finally, the relationship *between* variance components within each barrel-land is relatively consistent regardless of window size or phase. Respective magnitude of components is estimated consistently, which is of interest for our research questions.

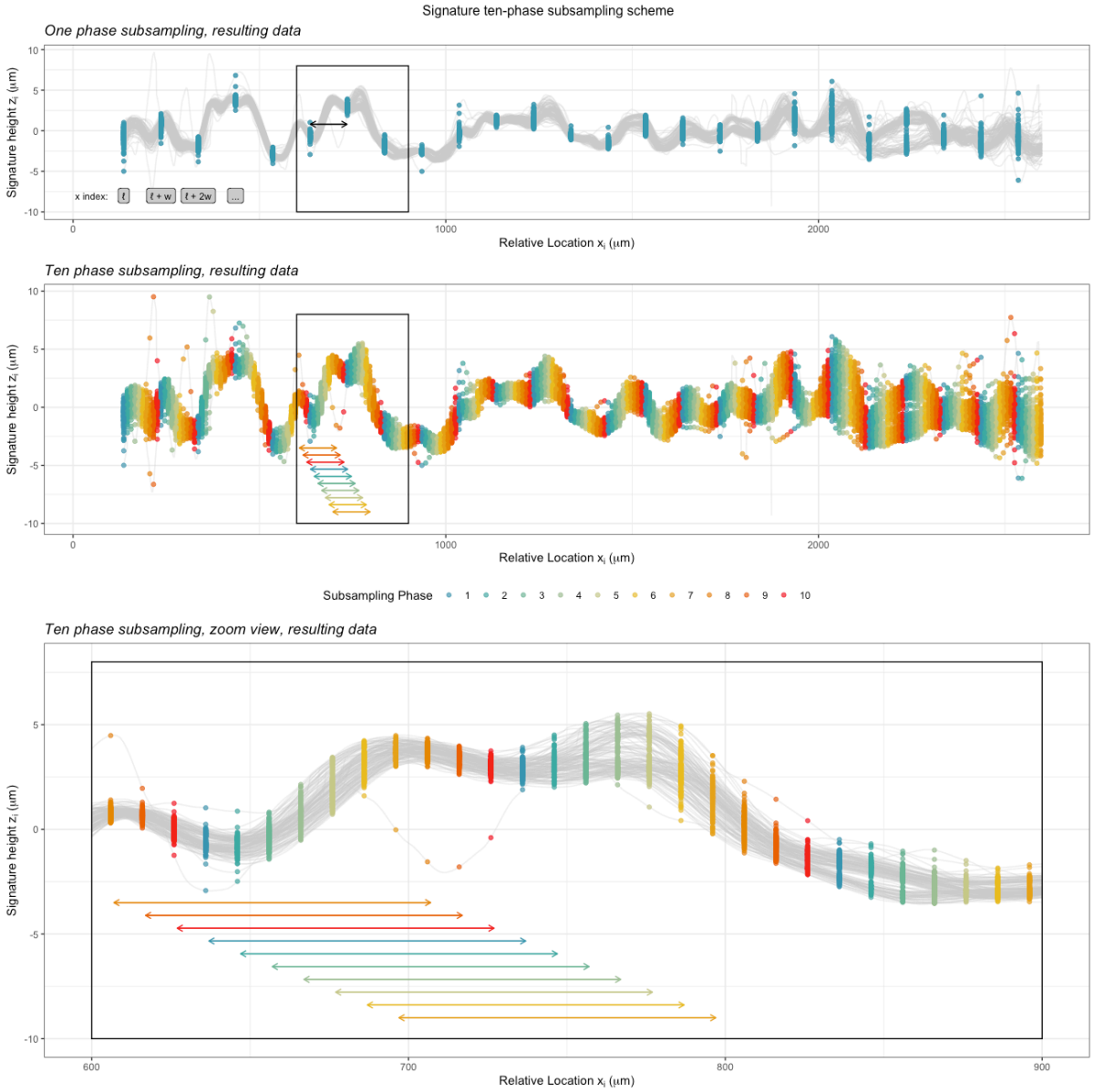

Figure 3.16 Visual representation of the data used in the ten-phase subsampling approach with window size w=100. (Top) Data subsampled for a one-phase approach; one model is fit. (Middle) Data subsampled for a ten-phase approach; one model is fit for each phase, resulting in ten independent models. (Bottom) A zoomed view of the ten-phase subsampling scheme shown.

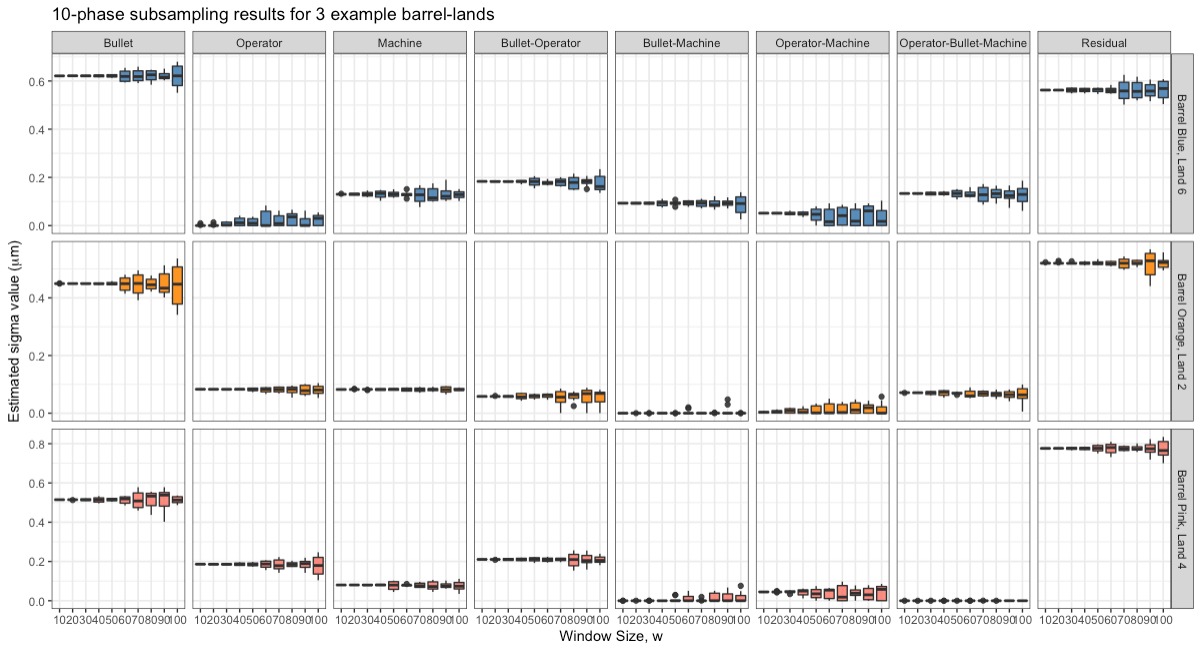

Figure 3.17 Distributions of variance component estimates for a subsampling model with window sizes w = 10, 20, ..., 100. Ten models were fit as equidistant phases spread out along the width of the window size w. Models were fit for Barrel Blue, Land 6 (top), Barrel Orange, Land 2 (middle), and Barrel Pink, Land 4 (bottom).

We therefore fit ten phases of models using a window size $w = 100$ and report the average estimate for each component across ten models as well as the range of estimates calculated by the phased models.

3.3.1.3 Addressing model singularities

There exists one consideration remaining with our model: the potential for model singularities. Singularities occur when the model matrix becomes rank deficient during maximum likelihood optimization, which can occur for several reasons:

1. The model is overspecified, or there is not enough data to support estimating parameters;
2. The variance components are very small (near zero), and the model reports them as zero; or
3. The number of grouping levels for a random effect is too small to support estimating a variance component

We have already addressed number (3) in subsubsection 3.3.1.1 by including location as an interaction with each random component to increase the number of grouping levels. In addition, our model specification matches our study design as a three-factor Gauge R&R model, so we do not want to alter the model per number (1). However, we may have some variance components, likely interaction terms, which are small in magnitude or don't have enough data to support their estimation. Therefore, we will likely see models with singularities. Practically, this means one or more random effect estimates will be reported as 0 in any model which is singular.

To address the behavior of models with singularities, we perform a small simulation study which simulates data and investigates distributions of model estimates from that data.

The simulation study was completed using the following procedure:

1. Calculate values μ_i for a chosen Barrel-Land (Orange-1 was used) and 20 locations x_i spaced at every 100^{th} x_i location on barrel-land BL.

2. Define a number of grouping levels for each study design factor to mirror the study design used: $j = 3$ bullets, $k = 8$ operators, $m = 2$ devices, $n = 3$ repetitions.

3. Create a shell data frame which mirrors the model set-up. This is accomplished by creating a row for every unique combination of location i, bullet j, operator k, machine m, and repetition n.

4. Define a set of random effect parameter values:

$$\Sigma = (\hat{\sigma}_b, \hat{\sigma}_o, \hat{\sigma}_d, \hat{\sigma}_{bo}, \hat{\sigma}_{bd}, \hat{\sigma}_{od}, \hat{\sigma}_{bod}, \hat{\sigma})$$

   as target values to estimate by obtaining a model fit for the chosen Barrel-Land (Orange-1).

5. For each random effect parameter in $\Sigma = (\hat{\sigma}_b, \hat{\sigma}_o, \ldots, \hat{\sigma})$, and a simulation number s, simulate $n_{grouping-levels}$ values from a $N(0, \hat{\sigma}_{..}^2)$. For example, simulate $j \times i = 3 \times 20 = 60$ values $\tilde{b}_{ij}^{(s)}$ from a $N(0, \hat{\sigma}_b^2)$ distribution and assign $\tilde{b}_{1,1}^{(s)}$ to all rows with location $i = 1$ and bullet $j = 1$.

6. Calculate simulated data points $\tilde{z}_{ijkmn}^{(s)} = \tilde{\mu}_i^{(s)} + \tilde{b}_{ij}^{(s)} + \cdots + \tilde{e}_{ijkmn}^{(s)}$.

7. Using the simulated response values $\tilde{\mathbf{z}}^{(s)}$, fit the model defined in Equation 3.6 and save resulting estimates of $\hat{\Sigma}^{(s)} = (\hat{\sigma}_b^{(s)}, \hat{\sigma}_o^{(s)}, \ldots, \hat{\sigma}^{(s)})$ for $s = 1000$ simulation runs.

Results of this simulation using target values calculated from Barrel Orange Land 1 are presented in Figure 3.18, colored by models resulting in singular and non-singular fits. We utilized two sets of target Σ vectors. We observe that for small "target" effect sizes (e.g., Operator-Machine, Operator-Bullet-Machine), regardless of whether the model is singular, our model overestimates the value of the target effect size. However, as the singular models some-

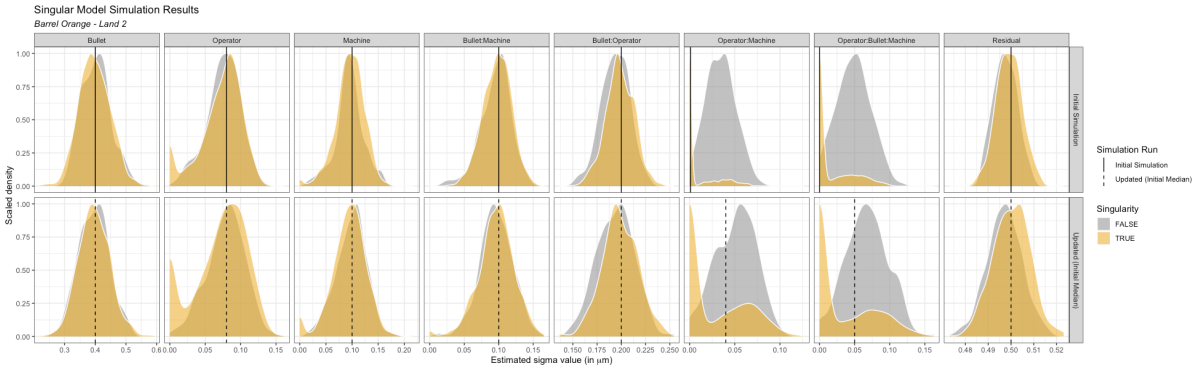

Figure 3.18 Results from the simulation study investigating model singularities, using Barrel Orange - Land 2 fixed effects as a basis for simulated data. (Top) Initial simulation, using estimated variance components from one phase of the Barrel Orange - Land 2 model as the target random effect sizes. (Bottom) Updated simulation, using median of over-estimated parameters from the initial simulation as the target random effect sizes.

times report a value of 0 for small effect sizes, we can see that the distribution of simulated variance components for singular models sometimes captures the small magnitude of the effect. Our second set of target values uses the mode of the distribution of over-estimated values for Operator-Machine and Operator-Bullet-Machine effects in our initial simulation. This was done to assess whether a larger effect size would be captured more effectively in simulation, and while we do note an improvement, the model still tends to overestimate the effect sizes for those interaction terms.

However, for larger effect sizes, the model estimates are well-distributed with the simulated estimates having a mode aligned with the target effect size, regardless of whether the model resulted in a singularity or not. This is most likely due to the larger effects not being the cause of the singularity; their effect sizes are almost never the estimates being set to 0 during optimization. Therefore, we can have a singular model but still observe accurate parameter estimation for the majority of the parameters of interest.

The overestimation of small effect sizes in non-singular models is an important phenomenon to take note of. We will address this in part using our phased modeling approach described in subsubsection 3.3.1.2. By fitting ten distinct subsampling models, we can obtain a better sense of how the model may vary and include a range of estimates across the ten phases. This also provides an "upper bound" for the effect size.

We will also fit a "pooled" model for each barrel which pools data from all 6 barrel-lands. The pooled model incorporates a fixed effect structure by Barrel-Land and Location, so grouping by location does not cross Barrel-Lands. The pooled model provides a larger sample size for maximum likelihood optimization and reduces the instance of singularities.

3.3.2 Pairwise Similarity Scores

The second model framework of interest is modeling the variability in pairwise random forest scores measuring similarity of paired sets of repeated signatures. Quantifying the variability of similarity scores is of interest because it demonstrates the degree of sensitivity the pairwise modeling approach has to varying environmental conditions of LEA scanning. As discussed in section 3.2, we must conceptualize our research questions, and therefore modeling approach, using a *pairwise* approach.

Consider the model defined in Equation 3.2, with some mean structure μ and random effects for each factor in the study design:

$$z_{jkmn} = \mu + b_j + o_k + d_m + bo_{jk} + bd_{jm} + od_{km} + bod_{jkm} + e_{jkmn}.$$

This model does not capture the mechanism we are interested in for pairwise scores, for one primary reason. The initial model defines a response which is a singular measurement for single levels of j, k, m, and n – that is, z_{jkmn} represents a measured response for a bullet j,

an operator k, machine m, and repetition n. However, pairwise random forest scores naturally incorporate one or more levels of the study design because they are paired across levels. For this same reason, random effects which are defined for a study factor – bullets j, for example – will not capture the difference in scores when comparing signatures *across* bullets.

To address this issue, we propose a model which instead models a pairwise response for pairs of study design levels. Let $z_{(L)'(j)'(k)'(m)'(n)'}$ be the pairwise similarity score calculated for barrel-land pair $(L)'$, bullet pair $(j)'$, operator pair $(k)'$, machine pair $(m)'$, and repetition pair $(n)'$. The indices in our paired model represent a list of pairings which exhaust all possible combinations of the levels, $\binom{n_{levels}}{2}$. For example, for the bullet study factor, we have bullet pairs $(j)' = (1-1),(1-2),(1-3),(2-3),(3-3)$, which is an exhaustive set of comparisons between bullets $j = (1,2,3)$.

Given the new indices, the proposed model is defined as follows:

$$z_{(L)'(j)'(k)'(m)'(n)'} = \mu + b_{(j)'} + o_{(k)'} + d_{(m)'} + bo_{(j)'(k)'} + bd_{(j)'(m)'} + od_{(k)'(m)'} + bod_{(j)'(k)'(m)'} + e_{(j)'(k)'(m)'(n)'}. \tag{3.7}$$

However, there is still one additional aspect to consider carefully: the mean structure. While we might expect same-source pairings to have score distributions that fall high, close to 1, we do not expect different-source pairings to differ *in the same way.* For example, a pairing between BL O-1 and BL O-2 might have scores that typically fall near 0.4, but a pairing between BL O-1 and BL O-4 might have scores that fall closer to 0.1. Differences in pairwise score distributions by barrel-land pairing on Barrel Blue can be seen in Figure 3.19(left). For this reason, we want to use the barrel-land pairing index, $(L)'$ as a fixed effect to remove the inherent structure present in our scores due to barrel-land pairings. Score distributions centered by barrel-land pairing mean are shown in Figure 3.19(right).

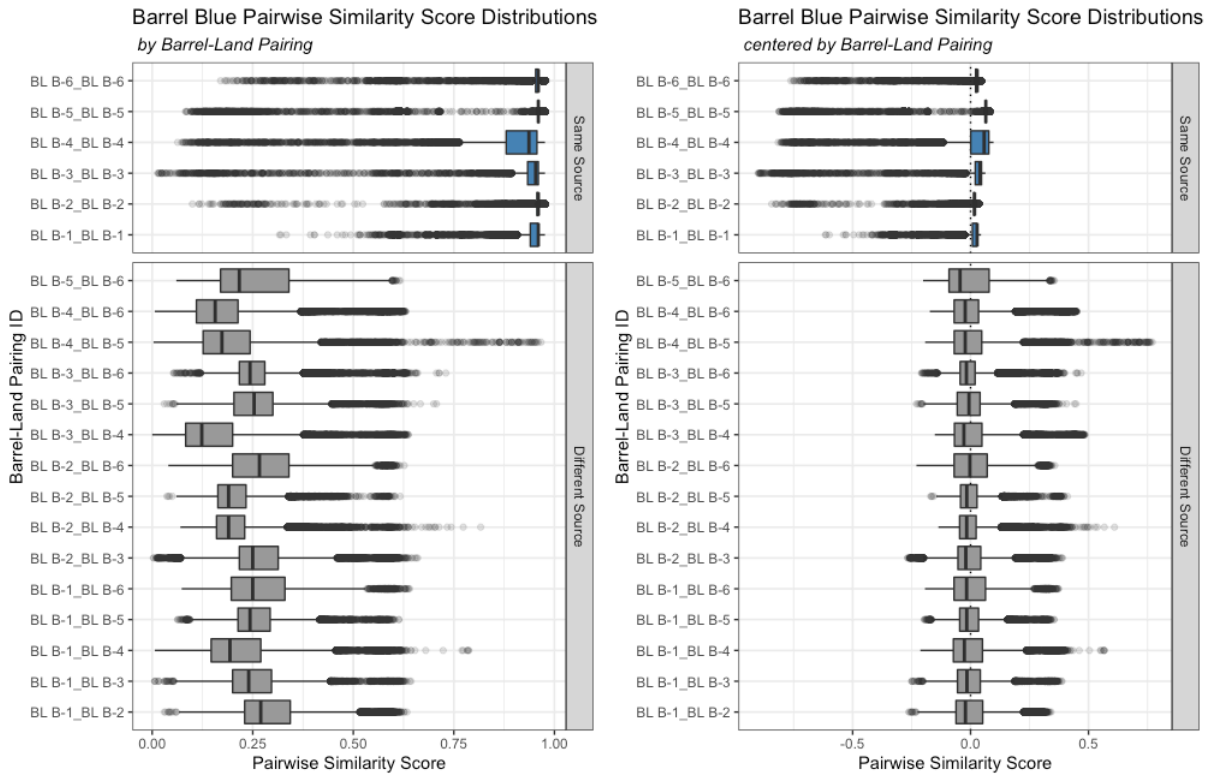

Figure 3.19 Barrel Blue pairwise score distributions by barrel-land pairing. Distributions differ by both mean and shape based on barrel-land. (Left) Raw score distributions. (Right) Distributions centered by barrel-land mean score.

In addition, we also interact the barrel-land pairing index $(L)'$ with each of our study factor pairing random effects. This modeling choice is made for a similar reason as the inclusion of a location interaction at the signature level; the study factor pairings alone do not estimate the quantity of interest. Consider the distribution of pairwise scores for Barrel Blue shown in Figure 3.20, which are centered by barrel-land pairing. When grouped by bullet pairing, the distributions differ by both location and spread by barrel-land pairing, which suggests that including barrel-land pairing as an interaction with bullet pairing will more effectively model variability structures present in pairwise similarity scores. We therefore include barrel-land pairing as an interaction with each study factor in our model.

Our resulting model is then:

$$\begin{aligned} z_{(L)'(j)'(k)'(m)'(n)'} &= \text{ Random Forest score comparing two signatures} \\ &= \mu_{(L)'} + b_{(L)'(j)'} + o_{(L)'(k)'} + d_{(L)'(m)'} + bo_{(L)'(j)'(k)'} + \\ &= \quad bd_{(L)'(j)'(m)'} + od_{(L)'(k)'(m)'} + bod_{(L)'(j)'(k)'(m)'} + e \end{aligned} \tag{3.8}$$

with a fixed, unknown measurement average for Barrel-Land pairing $\mu_{(L)'}$, and random effects

$$\begin{aligned} b_{(L)'(j)'} &\quad \text{for Bullet pairing } (j)' \text{ by Barrel-Land pairing } (L)', \\ &\qquad \text{following a } N(0, \sigma^2_{(b)'}) \\ o_{(L)'(k)'} &\quad \text{for Operator pairing } (k)' \text{ by Barrel-Land pairing } (L)', \\ &\qquad \text{following a } N(0, \sigma^2_{(o)'}) \\ d_{(L)'(m)'} &\quad \text{for Device pairing } (m)' \text{ by Barrel-Land pairing } (L)', \\ &\qquad \text{following a } N(0, \sigma^2_{(d)'}) \end{aligned}$$

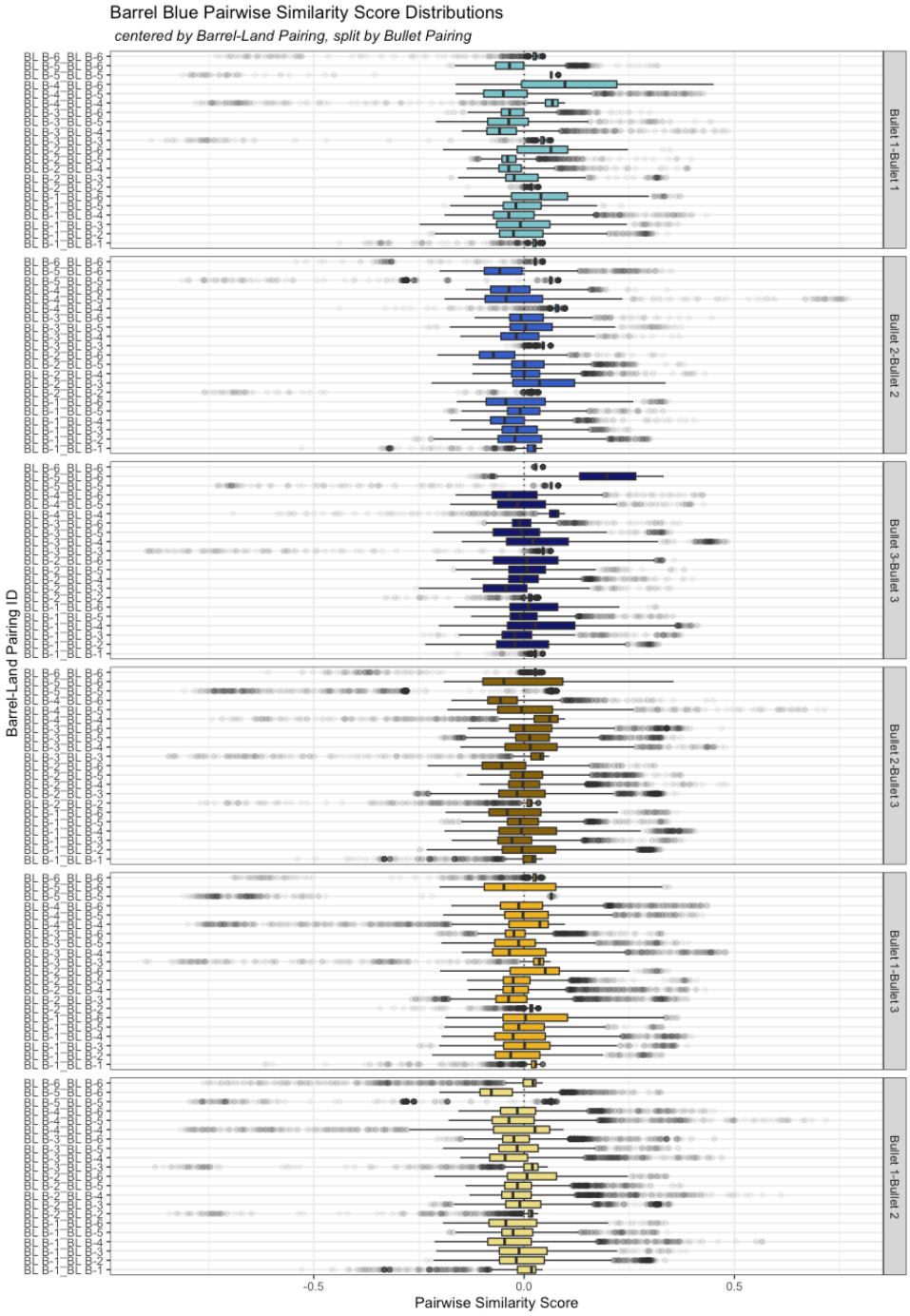

Figure 3.20 Pairwise score distributions for Barrel Blue, centered by barrel-land pairing, and split by bullet pairing. Distributions by bullet pairing vary by barrel-land pairing, which suggests including barrel-land pairing as an interaction with bullet pairing will more effectively model variability structures present in pairwise similarity scores.

$bo_{(L)'(j)'(k)'}$ for Bullet pairing $(j)'$-Operator pairing $(k)'$
by Barrel-Land pairing $(L)'$, following a $N(0, \sigma^2_{(bo)'})$

$bd_{(L)'(j)'(m)'}$ for Bullet pairing $(j)'$-Device pairing $(m)'$
by Barrel-Land pairing $(L)'$, following a $N(0, \sigma^2_{(bd)'})$

$od_{(L)'(k)'(m)'}$ for Operator pairing $(k)'$-Device pairing $(m)'$
by Barrel-Land pairing $BL)'$, following a $N(0, \sigma^2_{(od)'})$

$bod_{(L)'(j)'(k)'(m)'}$ for Bullet pairing $(j)'$-Operator pairing $(k)'$-
-Device pairing $(m)'$ by Barrel-Land pairing $(L)'$,
following a $N(0, \sigma^2_{(bod)'})$

$e_{(L)'(j)'(k)'(m)'(n)'}$ is error across repetitions, following a $N(0, \sigma^2)$.

The subsampled and phased approach used for signatures is not required for the pairwise model specified above, due to the difference in response structure and modeling approach. However, for each barrel type, we will fit three models: a model using only same-source barrel-land pairings, a model using only different-source pairings, and a model which uses all barrel-land pairings. As Figure 3.19 demonstrates, we observe structural differences between scores for same-source barrel-land pairings and score distributions for different-source barrel-land pairings. This is by design; the random forest algorithm used is designed to predict a probability of "same-source" class membership. As such, we expect same-source pairs to generally have a high, dense distribution of scores while different-source pairs have a lower and more diffuse score distribution.

We will report results from the three separate models – same source, different source, all pairings – using the model structure in Equation 3.8 for each of the three barrels in our study: Orange, Blue, and Pink.

3.4 Results

All estimated model parameters reported in the following results were fit using the `lmer` function from the `lme4` package in `R` (Bates et al., 2015), using the default optimization algorithm and Restricted Maximum Likelihood (REML) estimation.

We also consider, in each set of model results, the difference in estimated model parameters when removing LEAs that have tank rash. Tank rash, made of abrasions on the surface of a bullet due to contact with a water recovery tank, is described in section 1.5 and shown visually in Figure 1.8. LEAs with tank rash are often deemed unsuitable for comparison by forensic examiners as the striation pattern is partially or mostly destroyed. We identify 4 barrel-lands – 3 from Barrel Orange and one from Barrel Pink – for which at least one LEA has tank rash. Tank rash may occur on a barrel-land for one bullet, but not the other two. In both signature-level and pairwise modeling efforts, we fit models with all LEAs included as well as models where tank rash LEAs have been excluded from consideration. The resulting estimates from both sets of models are reported and discussed in the following.

3.4.1 Two Dimensional Signatures

The extracted signatures used as data were extracted using the data processing steps detailed in section 1.6. GEA data identification and removal (Groove ID) was done by manual identification, as to not bias the results using one of the proposed – but less accurate – automated methods discussed in Chapter 2.

In the following, we report estimates for the following parameters in each of the three barrels in our study:

- $\hat{\sigma}_b$: estimated bullet-by-location effect

- $\hat{\sigma}_o$: estimated operator-by-location effect
- $\hat{\sigma}_d$: estimated machine-by-location effect
- $\hat{\sigma}_{bo}$: estimated bullet-by-operator-by-location effect
- $\hat{\sigma}_{bd}$: estimated bullet-by-machine-by-location effect
- $\hat{\sigma}_{od}$: estimated operator-by-machine-by-location effect
- $\hat{\sigma}_{bod}$: estimated bullet-by-operator-by-machine-by-location effect
- $\hat{\sigma}$: estimated measurement repetition effect (residual standard error)

The reported estimates resulted from fitting ten phased models, and are thus reported as the mean estimate across ten phases. The range of estimates across ten phases is also reported as a band of uncertainty around our estimates. They are not formal confidence intervals for the effects, but represent our uncertainty in estimation due to the subsampling approach.

In addition, we also calculate and report two summary parameters, commonly used in Gauge R&R studies:

- $\hat{\sigma}_{repeatability} = \hat{\sigma}$, which represents a measure of measurement repeatability for measurements taken under the same environmental conditions; and
- $\hat{\sigma}_{reproducibility} = \sqrt{\hat{\sigma}_b^2 + \hat{\sigma}_o^2 + \hat{\sigma}_d^2 + \hat{\sigma}_{bo}^2 + \hat{\sigma}_{bd}^2 + \hat{\sigma}_{od}^2 + \hat{\sigma}_{bod}^2}$, which represents a measurement reproducibility for measurements taken under different environmental conditions.

Fixed effect estimates μ_i are not of inferential interest and are therefore not included in reported results.

3.4.1.1 Barrel Orange

Barrel Orange contains 3 bullets all fired from one Ruger P-85 barrel. Seven operators scanned five repetitions of each of 6 LEAs on each bullet on two machines (Sneox1 and Sneox2). One operator scanned three repetitions of each of 6 LEAs on each bullet on two machines. This resulted in ~228 repetitions of each of 6 barrel-lands (O-1, O-2, O-3, O-4, O-5, O-6), with some minor imbalance due to issues described in subsubsection 3.2.2.1. A total of 1368 scans of Barrel Orange data were gathered and are utilized in the pooled model.

The resulting parameter estimates for model random components are reported in Table 3.3. Results are reported as the mean estimated parameter across ten phased models, as well as the minimum and maximum estimates. The distribution of results across the ten phased models is also displayed as boxplots in Figure 3.21. All results are reported in microns (μm).

The estimated bullet effect, σ_b^2, and residual error (repetition effect), σ^2, are the largest in magnitude for Barrel Orange across all six barrel-lands as well as the pooled model. The pooled model results in estimates which fall in the middle of estimated values from each of the six individual barrel-land models, and have a smaller range of estimated values. This is likely to due to increase sample size in the pooled model as well as pooling of data; random effects are calculated using all available groups and as such we expect they will fall between extremes estimated when groups are separated.

Barrel-Land O-6 has higher estimates for the bullet effect, operator effect (σ_o^2) and residual error than other barrel-land models and the pooled model. This can be partially attributed to the presence of tank rash on two of the three bullets that came in contact with Barrel-Land O-6.

Tank rash was identified on Barrel-Lands O-6, O-4, and O-1, though it is most pronounced for Barrel-Land O-6 as two of the three bullets had observable tank rash. Barrel-Lands O-1

Table 3.3 Barrel Orange signature-level model estimates for variance components. Mean estimate from 10 phased models (min estimate, max estimate).

Barrel-Land	σ_b	σ_o	σ_d	σ_{bo}
O-1	0.51 (0.46, 0.57)	0.07 (0.06, 0.08)	0.04 (0.00, 0.06)	0.14 (0.12, 0.17)
tank rash excluded	*0.24 (0.20, 0.30)*	*0.08 (0.07, 0.10)*	*0.09 (0.07, 0.11)*	*0.09 (0.05, 0.13)*
O-2	0.44 (0.32, 0.55)	0.08 (0.05, 0.11)	0.08 (0.08, 0.09)	0.04 (0.00, 0.08)
O-3	0.55 (0.44, 0.62)	0.00 (0.00, 0.02)	0.08 (0.07, 0.10)	0.13 (0.11, 0.15)
O-4	0.80 (0.65, 0.98)	0.00 (0.00, 0.03)	0.03 (0.00, 0.06)	0.18 (0.16, 0.20)
tank rash excluded	*0.16 (0.12, 0.19)*	*0.06 (0.06, 0.08)*	*0.05 (0.03, 0.07)*	*0.10 (0.07, 0.11)*
O-5	0.45 (0.38, 0.53)	0.06 (0.00, 0.08)	0.06 (0.03, 0.07)	0.27 (0.23, 0.44)
O-6	1.30 (1.27, 1.36)	0.20 (0.09, 0.28)	0.05 (0.00, 0.10)	0.38 (0.32, 0.44)
tank rash excluded	*0.33 (0.15, 0.46)*	*0.05 (0.00, 0.11)*	*0.03 (0.00, 0.06)*	*0.04 (0.00, 0.10)*
Pooled	0.70 (0.66, 0.74)	0.09 (0.00, 0.06)	0.05 (0.00, 0.10)	0.21 (0.20, 0.24)
tank rash excluded	*0.44 (0.37, 0.51)*	*0.06 (0.05, 0.07)*	*0.08 (0.07, 0.08)*	*0.16 (0.14, 0.17)*

Barrel-Land	σ_{bd}	σ_{od}	σ_{bod}	σ
O-1	0.07 (0.00, 0.10)	0.03 (0.00, 0.05)	0.00 (0.00, 0.00)	0.53 (0.52, 0.54)
tank rash excluded	*0.04 (0.03, 0.04)*	*0.04 (0.03, 0.06)*	*0.01 (0.00, 0.05)*	*0.32 (0.29, 0.34)*
O-2	0.00 (0.00, 0.00)	0.02 (0.00, 0.06)	0.06 (0.00, 0.09)	0.55 (0.52, 0.59)
O-3	0.04 (0.03, 0.09)	0.02 (0.00, 0.04)	0.02 (0.00, 0.05)	0.42 (0.39, 0.46)
O-4	0.04 (0.00, 0.09)	0.02 (0.00, 0.10)	0.14 (0.08, 0.18)	0.59 (0.56, 0.61)
tank rash excluded	*0.01 (0.00, 0.02)*	*0.07 (0.06, 0.09)*	*0.01 (0.00, 0.05)*	*0.39 (0.35, 0.44)*
O-5	0.00 (0.00, 0.00)	0.00 (0.00, 0.00)	0.00 (0.00, 0.00)	0.61 (0.51, 0.76)
O-6	0.03 (0.00, 0.05)	0.04 (0.00, 0.19)	0.00 (0.00, 0.00)	1.92 (1.90, 1.95)
tank rash excluded	*0.02 (0.00, 0.05)*	*0.01 (0.00, 0.04)*	*0.01 (0.00, 0.04)*	*0.27 (0.33, 0.42)*
Pooled	0.02 (0.00, 0.05)	0.01 (0.00, 0.06)	0.00 (0.00, 0.00)	0.87 (0.86, 0.88)
tank rash excluded	*0.00 (0.00, 0.11)*	*0.00 (0.00, 0.03)*	*0.00 (0.00, 0.02)*	*0.48 (0.46, 0.52)*

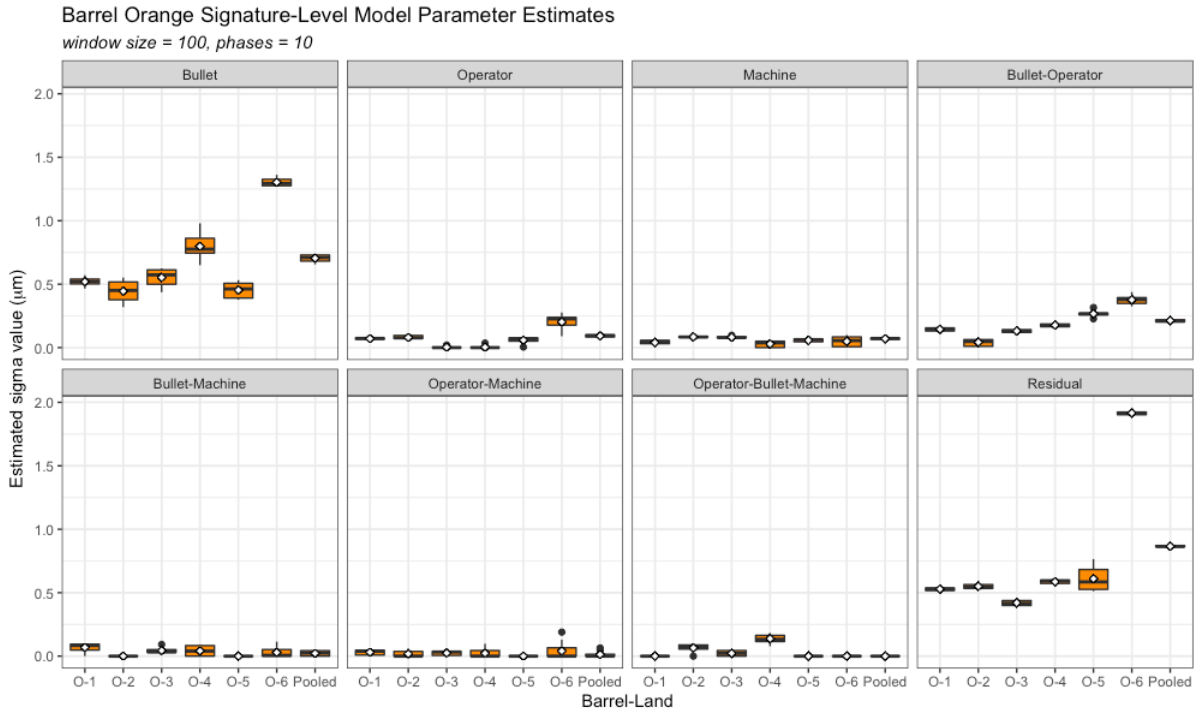

Figure 3.21 Resulting distributions of parameter estimates across ten phases of the subsampling random effects model, fit to Barrel Orange. Ten phases of models were calculated for each land in Barrel Orange individually, as well as a pooled dataset including all six lands in Barrel Orange.

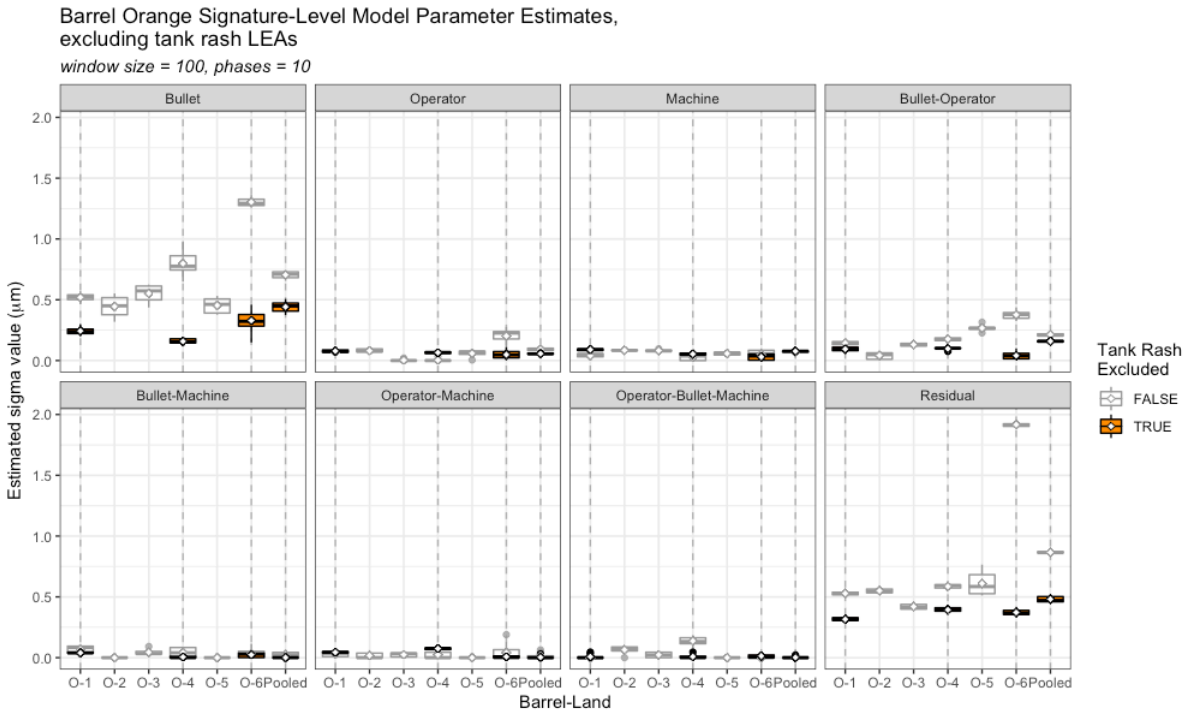

Figure 3.22 Resulting distributions of parameter estimates across ten phases of the subsampling random effects model after removing LEAs with tank rash from Barrel Orange. Boxplots in grey show initial model estimates, while colored boxplots show updated estimates after removing LEAs with tank rash.

and O-4 each had one bullet with tank rash. Signature-level models which filter out tank rash LEAs are shown in Table 3.3 along with the initial model results. A visual depiction of the change in model estimates after removing LEAs with tank rash is shown in Figure 3.22. Bullet effect is reduced for all three barrel-lands as well as the pooled model. It is important to note that the bullet effect will naturally be reduced after excluding tank rash LEAs due to the fact that there are less bullet levels for those barrel-lands. For Barrel-Land O-6, the operator effect as well as bullet effect were also reduced. The residual error standard deviation was reduced for all four affected models. The reduction shown in Figure 3.22 and Table 3.3 demonstrates the impact that tank rash LEAs have on signature-level repeatability and reproducibility; signature measurements are much more unpredictable and variable when tank rash is present on a LEA.

Table 3.4 Barrel Orange signature-level model estimates for summary quantities, $\sigma_{repeatability}$ and $\sigma_{reproducibility}$.

Barrel-Land	$\sigma_{repeatability}$	$\sigma_{reproducibility}$
O-1	0.53	0.55
tank rash excluded	*0.32*	*0.29*
O-2	0.55	0.47
O-3	0.42	0.58
O-4	0.59	0.83
tank rash excluded	*0.39*	*0.22*
O-5	0.61	0.53
O-6	1.92	1.37
tank rash excluded	*0.37*	*0.34*
Pooled	0.87	0.75
tank rash excluded	*0.48*	*0.48*

Summary values $\sigma_{repeatability}$ and $\sigma_{reproducibility}$ are reported in Table 3.4. Individual barrel-land models result in $\sigma_{repeatability}$ values between 0.42 μm and 1.92 μm, and $\sigma_{reproducibility}$ values between 0.47 μm and 1.37 μm. The pooled data model results in a $\sigma_{repeatability}$ estimate of 0.87 μm and a $\sigma_{reproducibility}$ estimate of 0.75 μm. After excluding tank rash LEAs, pooled $\sigma_{repeatability}$ is 0.48 μm and pooled $\sigma_{reproducibility}$ is 0.48 μm.

The values of $\sigma_{repeatability}$ and $\sigma_{reproducibility}$ for Barrel Orange are of similar magnitudes; this implies that a similar amount of measurement variability can be attributed to changes in environment as can be attributed to repetition in the same environment. However, it is worth noting that differences by bullet make up most of the contribution to $\sigma_{reproducibility}$, and those differences represent physical differences in how the barrel marks each land on each bullet. Across Barrel Orange, when environmental conditions are held constant, signature measurements vary with a residual standard deviation of approximately 0.48 μm (less than one half of one micron), meaning differences in measurement across repetitions are relatively small.

Table 3.5 Barrel Pink signature-level model estimates for variance components. Mean estimate from 10 phased models (min estimate, max estimate).

Barrel-Land	σ_b	σ_o	σ_d	σ_{bo}
P-1	0.81 (0.68, 0.94)	0.24 (0.12, 0.31)	0.01 (0.00, 0.04)	0.18 (0.12, 0.24)
P-2	0.93 (0.78, 1.07)	0.10 (0.00, 0.55)	0.25 (0.22, 0.30)	0.89 (0.75, 0.98)
P-3	0.79 (0.67, 0.92)	0.15 (0.12, 0.21)	0.07 (0.00, 0.16)	0.08 (0.00, 0.15)
P-4	0.51 (0.49, 0.54)	0.18 (0.10, 0.25)	0.07 (0.03, 0.11)	0.21 (0.18, 0.24)
P-5	0.49 (0.35, 0.61)	0.08 (0.00, 0.16)	0.10 (0.00, 0.16)	0.07 (0.00, 0.21)
P-6	1.23 (1.05, 1.36)	0.18 (0.14, 0.24)	0.17 (0.08, 0.26)	0.56 (0.40, 0.71)
tank rash excluded	*0.54 (0.40, 0.74)*	*0.16 (0.00, 0.24)*	*0.13 (0.10, 0.16)*	*0.39 (0.21, 0.57)*
Pooled	0.85 (0.82, 0.91)	0.17 (0.13, 0.28)	0.15 (0.12, 0.18)	0.48 (0.40, 0.54)
tank rash excluded	*0.71 (0.64, 0.76)*	*0.16 (0.12, 0.28)*	*0.14 (0.12, 0.15)*	*0.44 (0.39, 0.50)*

Barrel-Land	σ_{bd}	σ_{od}	σ_{bod}	σ
P-1	0.00 (0.00, 0.00)	0.03 (0.00, 0.10)	0.00 (0.00, 0.00)	1.25 (1.19, 1.31)
P-2	0.19 (0.12, 0.26)	0.22 (0.00, 0.33)	0.00 (0.00, 0.00)	1.83 (1.77, 1.87)
P-3	0.17 (0.14, 0.22)	0.11 (0.07, 0.15)	0.04 (0.00, 0.19)	0.93 (0.86, 0.98)
P-4	0.01 (0.00, 0.07)	0.04 (0.00, 0.08)	0.00 (0.00, 0.00)	0.77 (0.70, 0.83)
P-5	0.06 (0.00, 0.13)	0.17 (0.07, 0.27)	0.10 (0.00, 0.27)	0.89 (0.85, 0.92)
P-6	0.27 (0.24, 0.31)	0.19 (0.10, 0.27)	0.09 (0.00, 0.28)	1.60 (1.49, 1.72)
tank rash excluded	*0.01 (0.00, 0.05)*	*0.11 (0.00, 0.17)*	*0.06 (0.00, 0.14)*	*0.97 (0.82, 1.05)*
Pooled	0.16 (0.14, 0.19)	0.17 (0.13, 0.21)	0.00 (0.00, 0.00)	1.29 (1.26, 1.34)
tank rash excluded	*0.11 (0.08, 0.13)*	*0.15 (0.12, 0.19)*	*0.00 (0.00, 0.00)*	*1.16 (1.14, 1.19)*

3.4.1.2 Barrel Pink

Barrel Pink contains 3 bullets all fired from one Ruger LCP barrel. Seven operators scanned five repetitions of each of 6 LEAs on each bullet on two machines (Sneox1 and Sneox2). One operator scanned three repetitions of each of 6 LEAs on each bullet on two machines. This resulted in ~228 repetitions of each of 6 barrel-lands (P-1, P-2, P-3, P-4, P-5, P-6), with some minor imbalance due to issues described in subsubsection 3.2.2.1. A total of 1368 scans of Barrel Pink data were gathered and are utilized in the pooled model.

The resulting parameter estimates for model random components are reported in Table 3.5. Results are reported as the mean estimated parameter across ten phased models, as well as the minimum and maximum estimates. The distribution of results across the ten phased models is also displayed as boxplots in Figure 3.23. All results are reported in microns (μm).

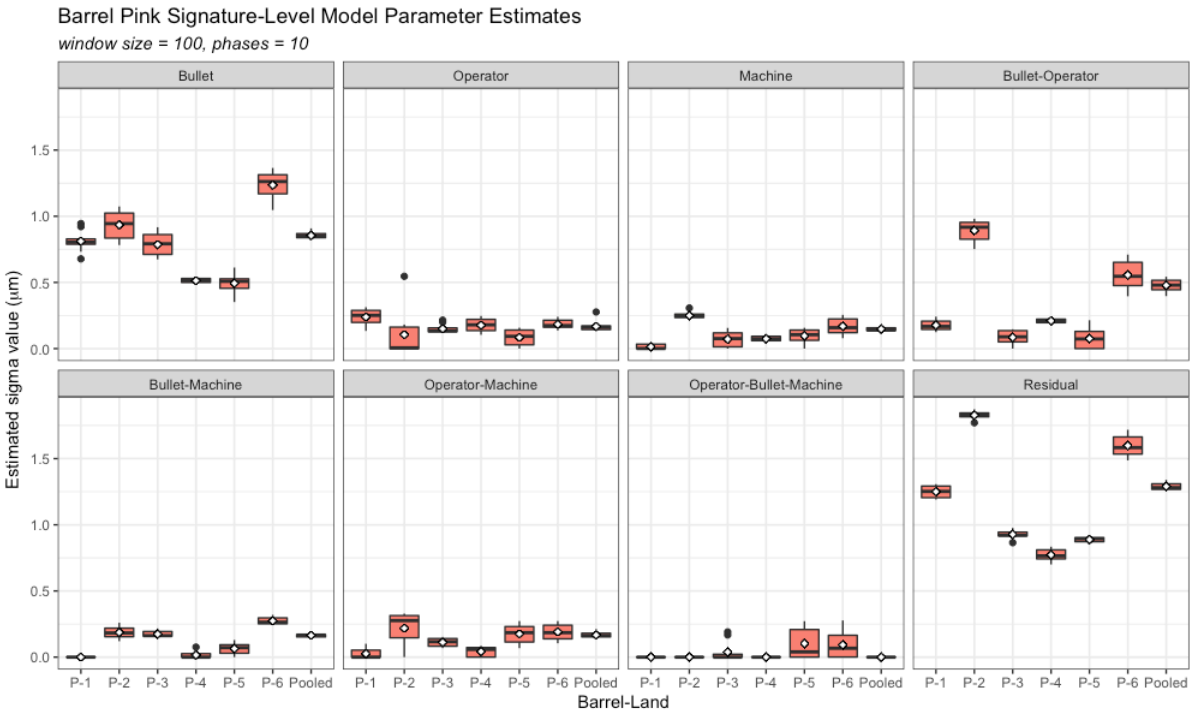

Figure 3.23 Resulting estimates from ten phases of the subsampling random effects model, fit to Barrel Pink. Ten phases of models were calculated for each land in Barrel Pink individually, as well as a pooled dataset including all six lands in Barrel Pink.

The estimated bullet effect, σ_b^2, and residual error (repetition effect), σ^2, are the largest in magnitude for Barrel Pink across all six barrel-lands as well as the pooled model, just as they were for Barrel Orange. The pooled model results in estimates which fall in the middle of estimated values from each of the six individual barrel-land models, and have a smaller range of estimated values; this is consistent with Barrel Orange results.

Barrel-Land P-6, which has tank rash on Bullet 3, has higher estimates for the bullet effect and residual error than most other barrel-land models and the pooled model. Signature-level models for Barrel-Land P-6 and the pooled signature data which filter out tank rash LEAs are shown in Table 3.5 along with the initial model results. A visual depiction of the change in model estimates after removing LEAs with tank rash is shown in Figure 3.24. Bullet effect and Residual error are both reduced for Barrel-Land P-6 as well as the pooled model.

Barrel-Land P-2 does not have identified tank rash but does have the largest estimated Machine effect, Bullet-Operator interaction effect, and Residual error. We do not have a quantifiable explanation for this phenomenon; however, we will discuss one likely source of increased variability for Barrel Pink in general in the Discussion.

Summary values $\sigma_{repeatability}$ and $\sigma_{reproducibility}$ are reported in Table 3.6. Individual barrel-land models result in $\sigma_{repeatability}$ values between 0.77 μm and 1.95 μm, and $\sigma_{reproducibility}$ values between 0.56 μm and 1.42 μm. The pooled data model results in a $\sigma_{repeatability}$ estimate of 1.29 μm and a $\sigma_{reproducibility}$ estimate of 1.03 μm. After excluding the P-6 tank rash LEA, pooled $\sigma_{repeatability}$ is 1.16 μm and pooled $\sigma_{reproducibility}$ is 0.88 μm.

The value of $\sigma_{repeatability}$ is slightly larger than $\sigma_{reproducibility}$ for Barrel Pink. This implies that a larger proportion of measurement variability can be attributed to repetitions in the same environment than changes in environment. The primary contributors to $\sigma_{reproducibility}$ in terms of magnitude are Bullet and Bullet-Operator interaction variability. For Barrel Pink, this means that differences across bullets and differences in how operators stage those bullets are the main

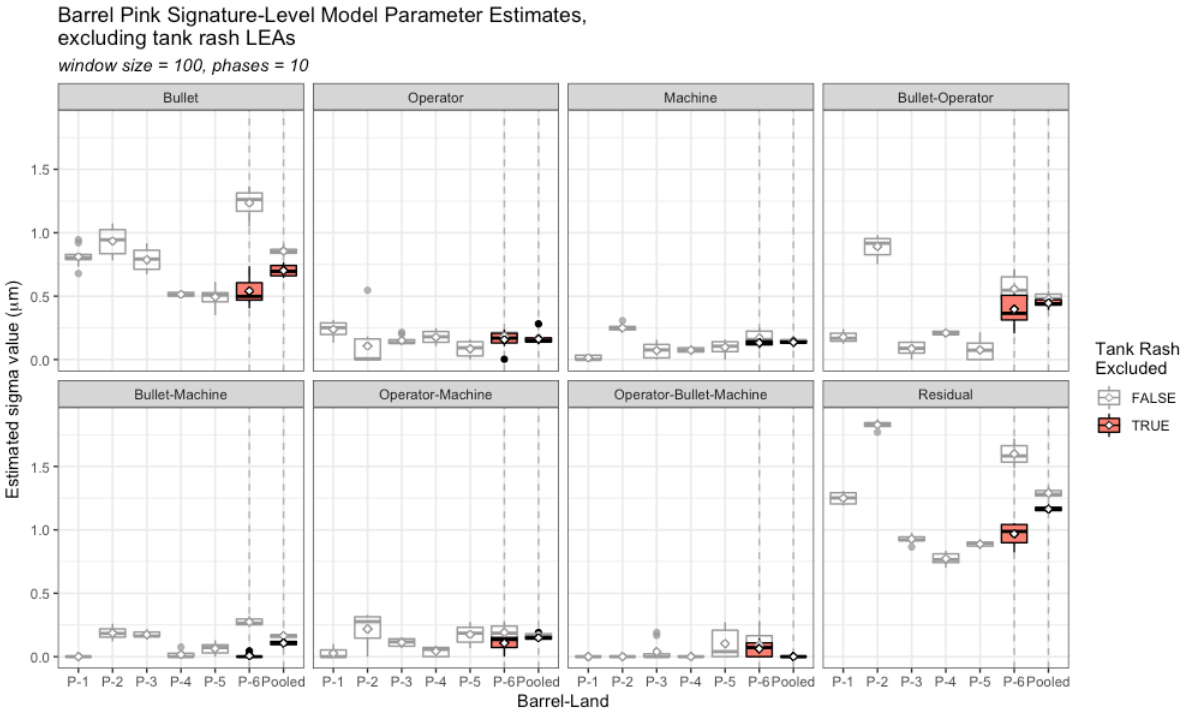

Figure 3.24 Resulting distributions of parameter estimates across ten phases of the subsampling random effects model after removing one LEA with tank rash from Barrel Pink. Boxplots in grey show initial model estimates, while colored boxplots show updated estimates after removing the LEA with tank rash.

Table 3.6 Barrel Pink signature-level model estimates for summary quantities, $\sigma_{repeatability}$ and $\sigma_{reproducibility}$.

Barrel-Land	$\sigma_{repeatability}$	$\sigma_{reproducibility}$
P-1	1.25	0.86
P-2	1.83	1.35
P-3	0.93	0.84
P-4	0.77	0.59
P-5	0.89	0.56
P-6	1.60	1.42
tank rash excluded	*0.99*	*0.71*
Pooled	1.29	1.03
tank rash excluded	*1.16*	*0.88*

sources of reproducibility variance. Across Barrel Pink, when environmental conditions are held constant, signature measurements vary with a residual standard deviation of approximately 1.16 μm (less than one and one fifths microns). Differences in measurement across repetitions are larger than those for Barrel Orange, but are still relatively small in terms of microns.

3.4.1.3 Barrel Blue

Barrel Blue contains 3 bullets all fired from one Beretta 92 F/FS barrel. Five operators scanned five repetitions of each of 6 LEAs on each bullet on two machines (Sneox1 and Sneox2). This resulted in ~150 repetitions of each of 6 barrel-lands (B-1, B-2, B-3, B-4, B-5, B-6), with some minor imbalance due to issues described in subsubsection 3.2.2.1. A total of 900 scans of Barrel Blue data were gathered and are utilized in the pooled model.

The resulting parameter estimates for model random components are reported in Table 3.7. Results are reported as the mean estimated parameter across ten phased models, as well as the minimum and maximum estimates. The distribution of results across the ten phased models is also displayed as boxplots in Figure 3.25. All results are reported in microns (μm).

The estimated bullet effect, σ_b^2, and residual error (repetition effect), σ^2, are the largest in magnitude for Barrel Blue across all six barrel-lands as well as the pooled model, just as

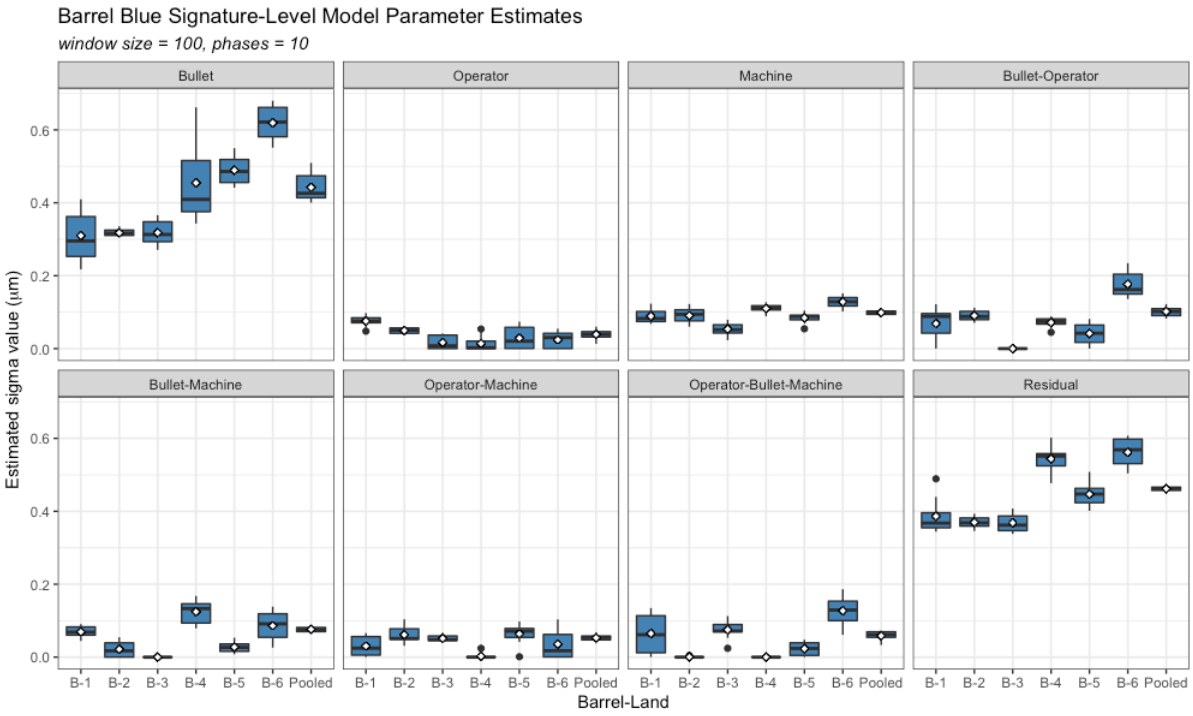

Figure 3.25 Resulting estimates from ten phases of the subsampling random effects model, fit to Barrel Blue. Ten phases of models were calculated for each land in Barrel Blue individually, as well as a pooled dataset including all six lands in Barrel Blue.

Table 3.7 Barrel Blue signature-level model estimates for variance components. Mean estimate from 10 phased models (min estimate, max estimate).

Barrel-Land	σ_b	σ_o	σ_d	σ_{bo}
B-1	0.31 (0.22, 0.41)	0.07 (0.05, 0.10)	0.09 (0.07, 0.12)	0.07 (0.00, 0.12)
B-2	0.32 (0.30, 0.33)	0.05 (0.04, 0.06)	0.09 (0.06, 0.12)	0.09 (0.08, 0.11)
B-3	0.32 (0.27, 0.37)	0.02 (0.00, 0.04)	0.05 (0.02, 0.08)	0.00 (0.00, 0.00)
B-4	0.45 (0.34, 0.66)	0.01 (0.00, 0.05)	0.11 (0.09, 0.13)	0.07 (0.04, 0.09)
B-5	0.49 (0.44, 0.55)	0.03 (0.00, 0.07)	0.08 (0.05, 0.10)	0.04 (0.00, 0.08)
B-6	0.62 (0.55, 0.68)	0.02 (0.00, 0.05)	0.13 (0.10, 0.15)	0.18 (0.14, 0.23)
Pooled	0.44 (0.40, 0.51)	0.04 (0.01, 0.06)	0.10 (0.09, 0.11)	0.10 (0.08, 0.12

Barrel-Land	σ_{bd}	σ_{od}	σ_{bod}	σ
B-1	0.07 (0.04, 0.09)	0.03 (0.00, 0.07)	0.06 (0.00, 0.13)	0.39 (0.34, 0.49)
B-2	0.02 (0.00, 0.05)	0.06 (0.03, 0.10)	0.00 (0.00, 0.00)	0.37 (0.35, 0.39)
B-3	0.00 (0.00, 0.00)	0.05 (0.04, 0.07)	0.07 (0.02, 0.11)	0.37 (0.34, 0.39)
B-4	0.12 (0.08, 0.17)	0.00 (0.00, 0.02)	0.00 (0.00, 0.00)	0.54 (0.48, 0.60)
B-5	0.03 (0.01, 0.05)	0.06 (0.00, 0.10)	0.02 (0.00, 0.05)	0.45 (0.40, 0.51)
B-6	0.09 (0.02, 0.14)	0.03 (0.00, 0.10)	0.13 (0.06, 0.19)	0.56 (0.50, 0.61)
Pooled	0.08 (0.06, 0.09)	0.05 (0.04, 0.07)	0.06 (0.03, 0.07)	0.46 (0.45, 0.47)

they were for Barrels Orange and Pink. The pooled model results in estimates which fall in the middle of estimated values from each of the six individual barrel-land models; this is consistent with Barrel Orange and Barrel Pink results.

There were no identified LEAs with tank rash in Barrel Blue, so the results presented in Table 3.7 and Figure 3.25 are the "final" set of results. The magnitude of effects for Barrel Blue is lower than that of effects for Barrel Pink. The magnitude is also lower than the initial set of Barrel Orange results; however, after excluding tank rash LEAs in Barrel Orange, the magnitude of Bullet and Residual effects are similar for Barrels Orange and Blue.

This can also be seen in $\sigma_{repeatability}$ and $\sigma_{reproducibility}$, reported in Table 3.8. Individual barrel-land models result in $\sigma_{repeatability}$ values between 0.39 μm and 0.56 μm, and $\sigma_{reproducibility}$ values between 0.35 μm and 0.68 μm. The pooled data model results in a $\sigma_{repeatability}$ estimate of 0.46 μm and a $\sigma_{reproducibility}$ estimate of 0.48 μm, which are very similar to the pooled results for Barrel Orange after tank rash exclusion.

Table 3.8 Barrel Blue signature-level model estimates for summary quantities, $\sigma_{repeatability}$ and $\sigma_{reproducibility}$.

Barrel-Land	$\sigma_{repeatability}$	$\sigma_{reproducibility}$
B-1	0.39	0.35
B-2	0.37	0.35
B-3	0.37	0.34
B-4	0.54	0.49
B-5	0.45	0.50
B-6	0.56	0.68
Pooled	0.46	0.48

The values of $\sigma_{repeatability}$ and $\sigma_{reproducibility}$ are similar for Barrel Blue. This implies that a similar amount of measurement variability is attributed to repetitions in the same environment as well as repetitions across changes in environment. The primary contributor to $\sigma_{reproducibility}$ in terms of magnitude is the Bullet effect. Across Barrel Blue, when environmental conditions are held constant, signature measurements vary with a residual standard deviation of approximately 0.46 μm (less than one half of one micron). Differences in measurement across repetitions are similar to those for Barrel Orange.

3.4.2 Pairwise Similarity Scores

The pairwise similarity score values used as data in the following results were calculated using the Hare et al. (2017) random forest, with pairwise features calculated and similarity scores predicted using functions in the `bulletxtrctr` package in `R`. Details about this process can be found in section 1.6.

All possible pairwise combinations of signatures within a barrel were identified, and pairwise scores were calculated for use in the model. One aspect to clarify is that we were careful to keep track of the lands included in each pair in order to ensure we don't duplicate a pair of signatures. This concept, depicted in Figure 3.26, is important to ensure scores comparing the same two paired signatures are not duplicated. Duplication could artificially deflate estimated variances components by increasing the sample size but including duplicated identical scores.

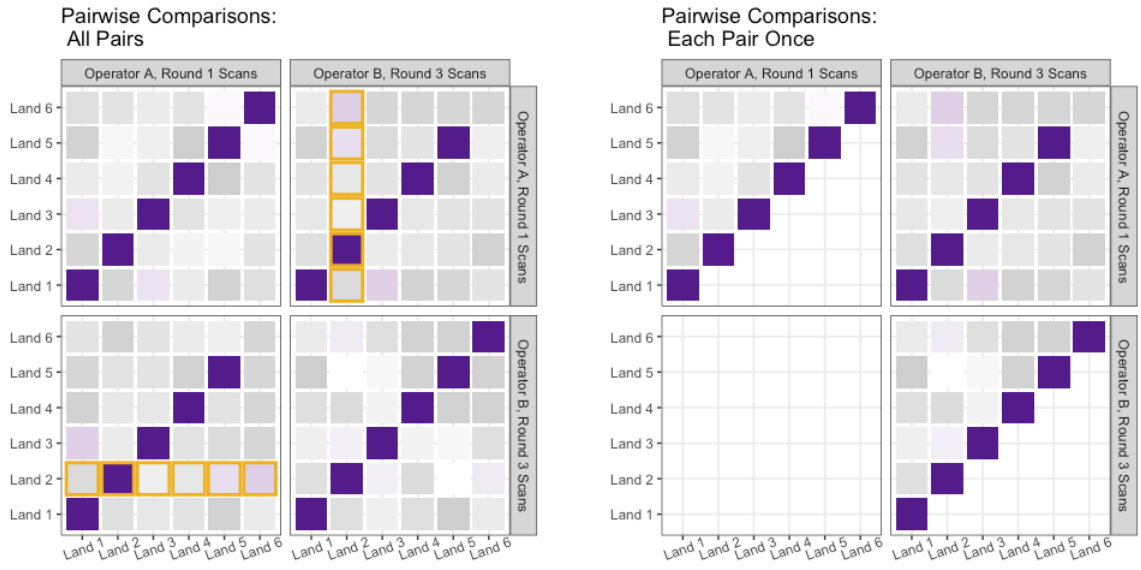

Figure 3.26 Illustration of the difference between including all pairwise comparisons, regardless of pairing ordering (left), and only including each pair once (right). The highlighted row and column in the (left) image represent the same six paired comparisons. Dark purple, on-diagonal comparisons are same-source pairs, while gray, off-diagonal comparisons are different-source pairs.

This pairing scheme results in $\frac{n(n+1)}{2}$ pairs for each test set made of n signatures.

The full distribution of pairwise scores for each of the three barrels is shown in Figure 3.27. Results from the pairwise models defined in Equation 3.8 are presented for each barrel.

As discussed in the previous section, we also consider two sets of models: (1) models which include all barrel-lands in the set of pairwise comparisons, and (2) models which remove any comparisons involving LEAs that have tank rash. Tank rash LEAs would be deemed unsuitable for comparison by forensic examiners, so it is important to examine pairwise results that don't include comparisons with tank rash LEAs.

Finally, it is important to note that pairwise scores are on a $(0, 1)$ scale; therefore, model parameter estimates are not in terms of microns as they are for signatures.

3.4.2.1 Barrel Orange

The Barrel Orange signature data consists of 1368 unique scans. When all scans are compared to one another using the sample scheme in Figure 3.26, the result is over 930,000 paired comparisons, with over 100,000 same-source comparisons and over 700,000 different-source comparisons. Random effect parameter estimates from the model defined in Equation 3.8 are reported for (1) all same-source pairings, (2) all different-source pairings, and (3) all pairings in Table 3.9, supplemented with updated results when tank rash LEAs are excluded. There are just under 600,000 paired comparisons remaining after removing all comparisons with tank rash LEAs.

The Bullet effect and Residual error have the largest magnitudes, similar to the signature results for all three barrels. After removing tank rash LEAs from the pool of comparisons, the Bullet effect is decreased; however, it remains larger in magnitude than most other effects, aside from the Residual error. The same source model consistently has larger parameter esti-

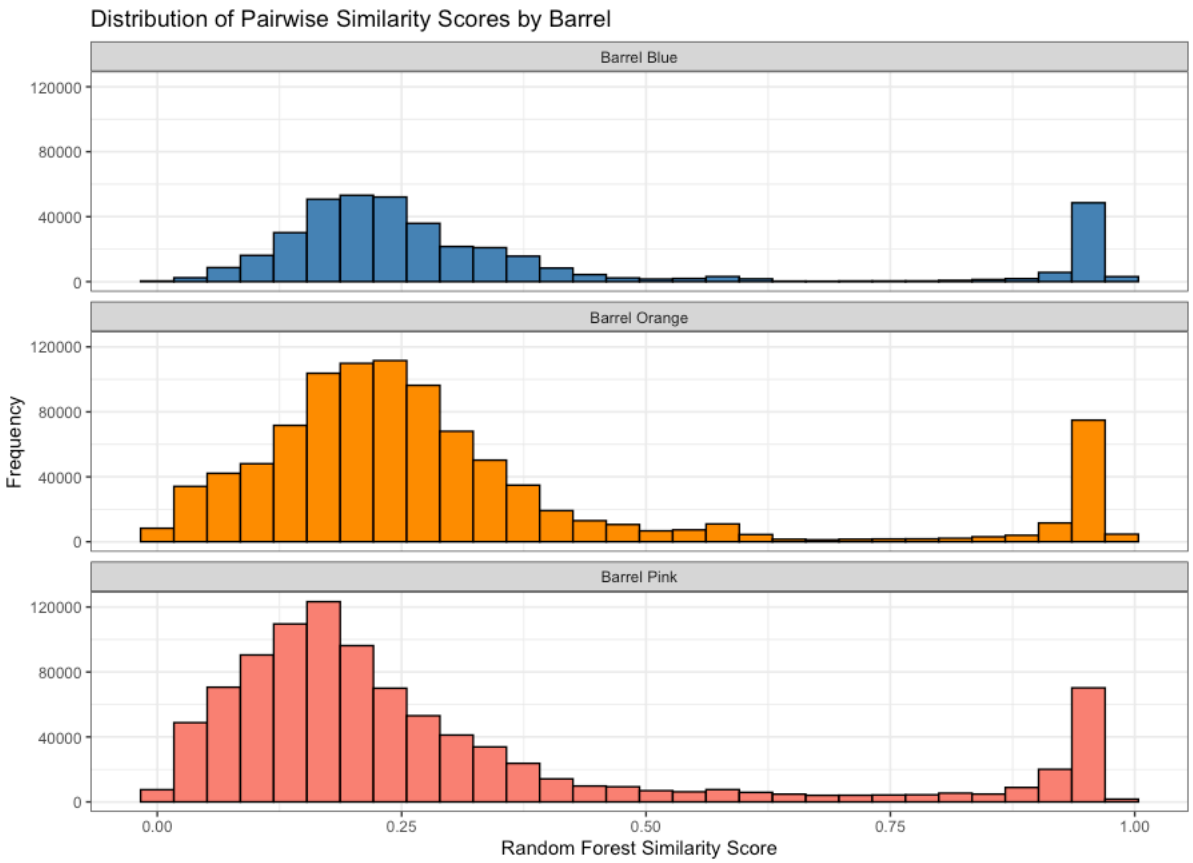

Figure 3.27 Pairwise similarity score distributions for each of the three barrels in our study.

Table 3.9 Barrel Orange pairwise-level model estimates for variance components.

Model	$\sigma_{(b)}$	$\sigma_{(o)}$	$\sigma_{(d)}$	$\sigma_{(bo)}$
Same-Source	0.212 (0.154, 0.272)	0.027 (0.022, 0.032)	0.020 (0.011, 0.027)	0.050 (0.047, 0.052)
tank rash excluded	*0.067 (0.045, 0.088)*	*0.030 (0.025, 0.036)*	*0.023 (0.013, 0.034)*	*0.042 (0.039, 0.045)*
Different-Source	0.035 (0.029, 0.040)	0.011 (0.010, 0.012)	0.003 (0.000, 0.005)	0.016 (0.015, 0.016)
tank rash excluded	*0.034 (0.027, 0.041)*	*0.010 (0.008, 0.011)*	*0.000 (0.000, 0.003)*	*0.015 (0.014, 0.016)*
All Pairings	0.117 (0.101, 0.133)	0.017 (0.016, 0.019)	0.011 (0.008, 0.014)	0.029 (0.028, 0.030)
tank rash excluded	*0.045 (0.038, 0.054)*	*0.018 (0.016, 0.019)*	*0.011 (0.009, 0.015)*	*0.025 (0.025, 0.026)*

Model	$\sigma_{(bd)}$	$\sigma_{(od)}$	$\sigma_{(bod)}$	σ
Same Source	0.013 (0.011, 0.016)	0.022 (0.020, 0.024)	0.003 (0.000, 0.007)	0.159 (0.159, 0.160)
tank rash excluded	*0.009 (0.006, 0.012)*	*0.029 (0.027, 0.032)*	*0.000 (0.000, 0.007)*	*0.155 (0.154, 0.156)*
Different-Source	0.008 (0.007, 0.009)	0.007 (0.007, 0.008)	0.003 (0.002, 0.004)	0.104 (0.104, 0.104)
tank rash excluded	*0.008 (0.006, 0.009)*	*0.006 (0.005, 0.006)*	*0.005 (0.005, 0.006)*	*0.087 (0.087, 0.087)*
All Pairings	0.010 (0.009, 0.011)	0.013 (0.012, 0.013)	0.004 (0.003, 0.005)	0.115 (0.115, 0.115)
tank rash excluded	*0.008 (0.007, 0.009)*	*0.015 (0.014, 0.016)*	*0.003 (0.000, 0.004)*	*0.103 (0.103, 0.103)*

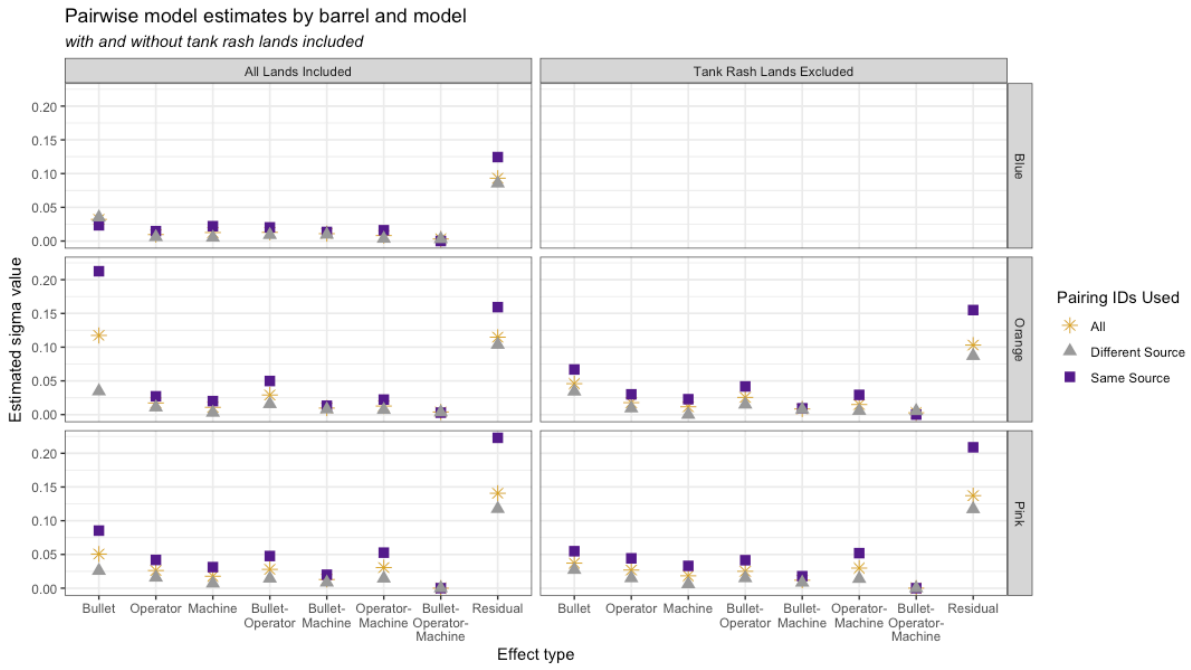

Figure 3.28 Estimated model components for each of the three barrels, split between models which include all lands and models which filter out lands with identified tank rash. Barrel Blue does not have any tank rash lands. Three estimates are shown for each parameter in each barrel; one for a model including all pairwise comparisons, one for a model including all same-source comparisons, and one for a model including all different-source comparisons.

Table 3.10 Barrel Orange pairwise-level estimates for $\sigma_{repeatability}$ and $\sigma_{reproducibility}$.

Barrel	Model	$\sigma_{repeatability}$	$\sigma_{reproducibility}$
Orange	Same-Source	0.159	0.222
	tank rash excluded	*0.155*	*0.092*
	Different-Source	0.104	0.041
	tank rash excluded	*0.087*	*0.040*
	All Pairings	0.115	0.124
	tank rash excluded	*0.103*	*0.059*

mates than the different source model, and the model which includes all comparisons results in parameters which typically fall between the two models. This effect can also be seen visually in Figure 3.28. The larger estimates of variability for the same-source model may be due to a smaller sample size; however, it may also be attributable to major left skew of same-source pairwise scores. Same-source scores are typically very high and close to one (as shown in Figure 3.19), but there are many low-scoring outliers where something within the pairwise comparison went awry. These low outliers can significantly increase variability seen across study levels.

Summary values $\sigma_{repeatability}$ and $\sigma_{reproducibility}$ are shown in Table 3.10 for both the full set of models and the set of models which exclude LEAs with tank rash. The estimated values are lower when tank rash LEAs are excluded. In addition, after excluding tank rash LEAs, $\sigma_{repeatability}$ is greater than $\sigma_{reproducibility}$ for all three models, with the model for all pairwise comparisons having a $\sigma_{repeatability}$ of 0.103 and a $\sigma_{reproducibility}$ of 0.059. Since pairwise scores live on a range of (0,1) and mean by barrel-land pairing is taken into account, these values demonstrate a non-trivial amount of variability associated with pairwise scores for Barrel Orange.

3.4.2.2 Barrel Pink

The Barrel Pink signature data consists of 1368 unique scans. When all scans are compared to one another using the sample scheme in Figure 3.26, the result is over 930,000 paired compar-

Table 3.11 Barrel Pink pairwise-level model estimates for variance components, with bootstrap 95% confidence intervals.

Model	$\sigma_{(b)}$	$\sigma_{(o)}$	$\sigma_{(d)}$	$\sigma_{(bo)}$
Same-Source	0.085 (0.063, 0.107)	0.042 (0.035, 0.050)	0.031 (0.016, 0.046)	0.048 (0.045, 0.050)
tank rash excluded	*0.055 (0.039, 0.070)*	*0.044 (0.036, 0.051)*	*0.033 (0.018, 0.048)*	*0.041 (0.039, 0.044)*
Different-Source	0.026 (0.022, 0.030)	0.016 (0.014, 0.017)	0.007 (0.004, 0.010)	0.014 (0.014, 0.015)
tank rash excluded	*0.027 (0.023, 0.031)*	*0.015 (0.013, 0.016)*	*0.006 (0.003, 0.010)*	*0.015 (0.014, 0.016)*
All Pairings	0.051 (0.044, 0.058)	0.026 (0.024, 0.029)	0.018 (0.013, 0.022)	0.028 (0.027, 0.029)
tank rash excluded	*0.037 (0.032, 0.042)*	*0.027 (0.024, 0.029)*	*0.018 (0.014, 0.023)*	*0.025 (0.024, 0.026)*

Model	$\sigma_{(bd)}$	$\sigma_{(od)}$	$\sigma_{(bod)}$	σ
Same Source	0.020 (0.016, 0.024)	0.053 (0.049, 0.056)	0.000 (0.000, 0.008)	0.223 (0.222, 0.224)
tank rash excluded	*0.018 (0.014, 0.021)*	*0.052 (0.048, 0.056)*	*0.000 (0.000, 0.008)*	*0.209 (0.208, 0.210)*
Different-Source	0.008 (0.007, 0.010)	0.014 (0.014, 0.015)	0.000 (0.000, 0.000)	0.118 (0.117, 0.118)
tank rash excluded	*0.008 (0.007, 0.010)*	*0.014 (0.013, 0.015)*	*0.000 (0.000, 0.000)*	*0.117 (0.117, 0.118)*
All Pairings	0.013 (0.012, 0.014)	0.030 (0.029, 0.032)	0.000 (0.000, 0.003)	0.141 (0.140, 0.141)
tank rash excluded	*0.012 (0.011, 0.013)*	*0.030 (0.029, 0.031)*	*0.000 (0.000, 0.003)*	*0.137 (0.137, 0.137)*

isons, with over 100,000 same-source comparisons and over 700,000 different-source comparisons. Random effect parameter estimates from the model defined in Equation 3.8 are reported for (1) all same-source pairings, (2) all different-source pairings, and (3) all pairings in Table 3.11, supplemented with updated results when the one Pink tank rash LEA is excluded. There are over 800,000 paired comparisons remaining after removing all comparisons with the tank rash LEA.

The Residual error has the largest magnitude for Barrel Pink. After removing the tank rash LEA from the pool of comparisons, the Bullet effect is decreased in all three models. The Bullet effect has a similar magnitude to the Operator-Machine interaction effect. The same source model consistently has larger parameter estimates than the different source model, and the model which includes all comparisons results in parameters which typically fall between the two models; this is consistent with patterns in Barrel Orange. The Barrel Pink residual error is larger in magnitude than the other two barrels, shown in Figure 3.28.

Summary values $\sigma_{repeatability}$ and $\sigma_{reproducibility}$ are shown in Table 3.12 for both the full set of models and the set of models which exclude LEAs with tank rash. $\sigma_{repeatability}$ is greater

Table 3.12 Barrel Pink pairwise-level estimates for $\sigma_{repeatability}$ and $\sigma_{reproducibility}$.

Barrel	Model	$\sigma_{repeatability}$	$\sigma_{reproducibility}$
Pink	Same-Source	0.223	0.124
	tank rash excluded	*0.209*	*0.104*
	Different-Source	0.118	0.038
	tank rash excluded	*0.117*	*0.039*
	All Pairings	0.141	0.074
	tank rash excluded	*0.137*	*0.064*

than $\sigma_{reproducibility}$ for all three models, regardless of whether the tank rash LEA was excluded. The model for all pairwise comparisons results in a $\sigma_{repeatability}$ of 0.137 and a $\sigma_{reproducibility}$ of 0.064. This suggests a larger proportion of the pairwise score variability is due to repetition of pairing levels than differences across pairing levels. Since pairwise scores live on a range of $(0, 1)$ and mean by barrel-land pairing is taken into account, these values demonstrate a non-trivial amount of variability associated with pairwise scores for Barrel Pink.

3.4.2.3 Barrel Blue

The Barrel Blue signature data consists of 900 unique scans. When all scans are compared to one another using the sample scheme in Figure 3.26, the result is approximately 400,000 paired comparisons, with over 60,000 same-source comparisons and over 300,000 different-source comparisons. Random effect parameter estimates from the model defined in Equation 3.8 are reported for (1) all same-source pairings, (2) all different-source pairings, and (3) all pairings in Table 3.13 and depicted visually in Figure 3.28.

The Residual error has the largest magnitude for Barrel Blue, following a similar pattern as Barrels Orange and Pink. The same source model has larger parameter estimates than the different source model, with the exception of the Bullet effect and Bullet-Operator-Machine interaction effect. The model with all comparisons results in parameters which typically fall between the two models; this is consistent with patterns in Barrels Orange and Pink. The

Table 3.13 Barrel Blue pairwise-level model estimates for variance components, with bootstrap 95% confidence intervals.

Model	$\sigma_{(b)}$	$\sigma_{(o)}$	$\sigma_{(d)}$	$\sigma_{(bo)}$
Same-Source	0.023 (0.016, 0.029)	0.015 (0.010, 0.018)	0.022 (0.013, 0.034)	0.020 (0.014, 0.018)
Different-Source	0.035 (0.029, 0.041)	0.006 (0.005, 0.007)	0.005 (0.003, 0.007)	0.010 (0.009, 0.010)
All Pairings	0.032 (0.028, 0.037)	0.009 (0.008, 0.011)	0.013 (0.009, 0.015)	0.013 (0.012, 0.013)

Model	$\sigma_{(bd)}$	$\sigma_{(od)}$	$\sigma_{(bod)}$	σ
Same Source	0.013 (0.011, 0.016)	0.016 (0.014, 0.018)	0.000 (0.000, 0.000)	0.124 (0.124, 0.126)
Different-Source	0.010 (0.009, 0.010)	0.004 (0.003, 0.004)	0.003 (0.002, 0.004)	0.085 (0.085, 0.086)
All Pairings	0.011 (0.010, 0.012)	0.008 (0.007, 0.009)	0.003 (0.002, 0.004)	0.093 (0.093, 0.093)

Table 3.14 Barrel Blue pairwise-level estimates for $\sigma_{repeatability}$ and $\sigma_{reproducibility}$.

Barrel	Model	$\sigma_{repeatability}$	$\sigma_{reproducibility}$
Blue	Same-Source	0.124	0.046
	Different-Source	0.085	0.038
	All Pairings	0.093	0.040

Barrel Blue residual error is the smallest in magnitude of the three barrels, which can be seen in Figure 3.28.

Summary values $\sigma_{repeatability}$ and $\sigma_{reproducibility}$ are shown in Table 3.14 for all three models. $\sigma_{repeatability}$ is greater than $\sigma_{reproducibility}$ for all three models. The model for all pairwise comparisons results in a $\sigma_{repeatability}$ of 0.093 and a $\sigma_{reproducibility}$ of 0.040. This suggests a larger proportion of the pairwise score variability is due to repetition of pairing levels than differences across pairing levels. These values are the lowest of the three barrels; however, they still demonstrate a non-trivial level of variability present in the pairwise score data.

3.5 Conclusions

Estimated residual standard deviations and bullet effects for both pairwise and signature-level data are largest for Barrel Pink, and smaller for Barrels Orange and Blue. This relationship suggests that Barrel Pink measurements, and therefore pairwise scores, are more variable than those from Barrels Orange and Blue.

There also exists a consistent pattern of relationships between study factor parameter magnitude. Bullet effect is the largest in magnitude for all barrels and both signature-level and pairwise-level results. Bullet-Operator interaction effect is also large for some individual barrel-land models, and the pooled model for Barrel Pink. Most of the other estimated study effects are relatively small in magnitude. This relationship between effect sizes suggests that differences in pattern by bullet are larger than differences in how those patterns translate to a 3D surface profile when captured on differing machines and by differing operators. The land-specific increase in Bullet-Operator interaction effect (e.g., for Barrel Pink-Land 6) suggests that when there are large differences between bullet structures, there are also differences in how operators stage or handle tricky or different structures.

One of the ways we observe this interaction in the data is use of lighting settings. Operators are directed to employ the "x10" setting when the default lighting settings do not adequately capture large changes in height values, such as deep striae. The difference in resulting 3D surface profiles is shown in Figure 3.29. The setting is most commonly needed in this study on Barrel Pink, the Houston set 3 bullets. The frequency with which operators employed the x10 setting by barrel is shown in Figure 3.30, which demonstrates that it differs by both barrel and operator.

The lighting setting disparity may help explain the increased variability across Barrel Pink. However, there is also barrel-land to barrel-land variation, as individual barrel-land model estimates differ at the signature-level, and same-source and different-source pairwise model estimates differ. The "pooled" signature model provides an overall barrel-level variability estimate of striation measurement variability. The "all pairings" pairwise model provides a barrel-level variability estimate of similarity score variability.

From a numerical standpoint, estimated standard deviations are larger for signature-level data than for pairwise-level data. However, the difference in response value structure dictates the opposite interpretation: pairwise-level results, which exist on a $(0, 1)$ range, have estimated

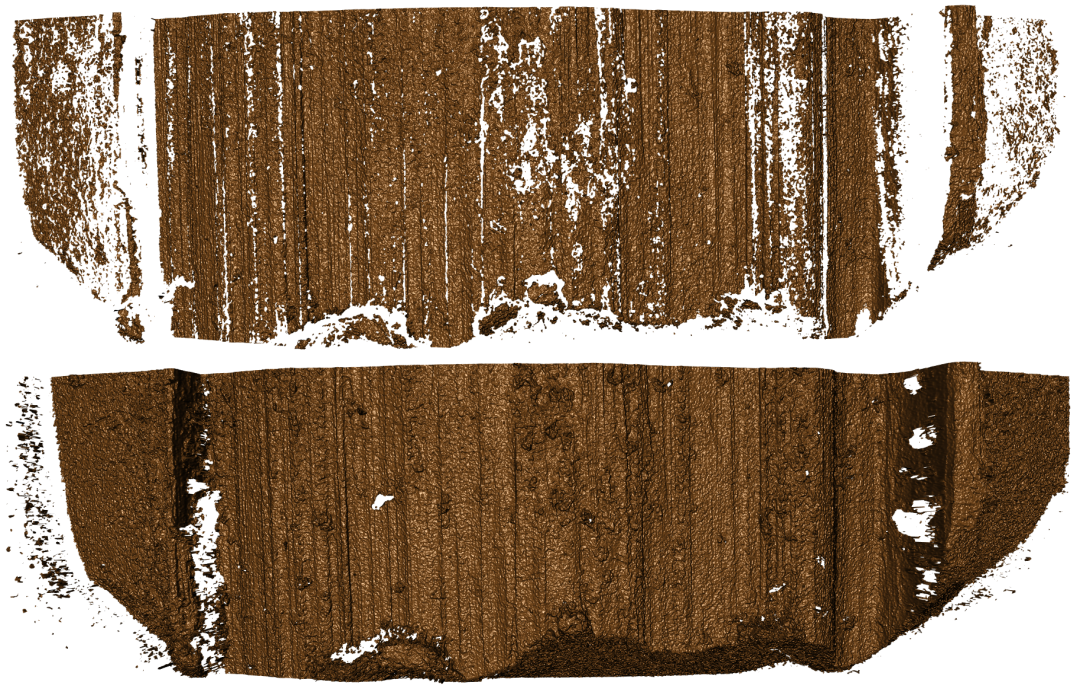

Figure 3.29 Example of two scans of the same LEA (Barrel Pink - Land 2) by the same operator using different light settings. (Top) Rendering of surface profile captured without using the x10 setting. (Bottom) Rendering of surface profile captured using the x10 setting. Failure to employ the x10 setting for LEAs with deep striae can sometimes result in surface profiles with more missing data, typically in straition 'valleys' as pictured here.

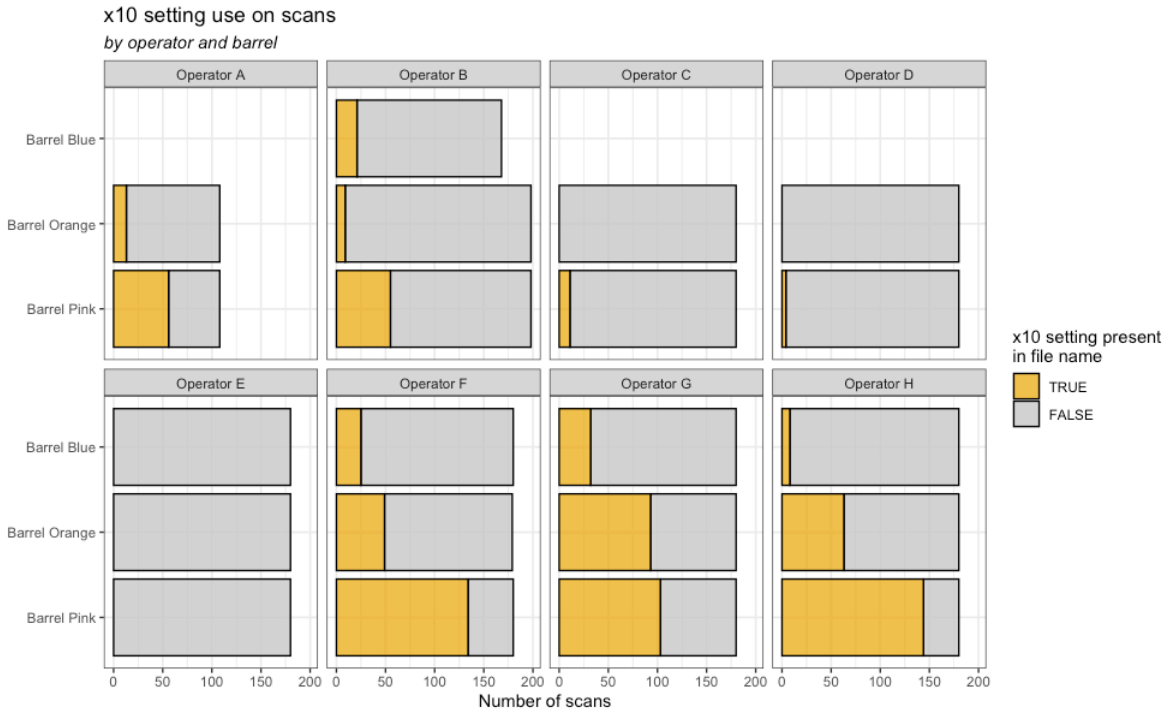

Figure 3.30 Number of scans where operators employed the 'x10' setting, a microscope light setting they are directed to employ when the default microscope lighting cannot adequately capture deep straie. There are differences in use of the x10 setting by operator as well as barrel. Barrel Pink has the deepest striae and therefore has the largest proportion of scans that were captured using the x10 setting.

repeatability and reproducibility measures near 0.1. Signature-level results, which exist in terms of microns (μm) and represent deviations from average signature height at a location x_i, have estimated repeatability and reproducibility measures near 0.5 for Orange and Blue and 1.0 for Pink, where striae structure themselves are 5-10 microns in height. A larger standard deviation in pairwise similarity scores is more concerning given the implication of an incorrect decision based on a pairwise similarity score, whereas deviations in measured heights for a signature represent variability in measurement precision.

The signature-level results therefore provide evidence for relative reproducibility of measured signature heights, particularly due to the majority of reproducibility variability originating from differences in bullet structure. Differences due to operator and machine contribute minimally to variability at the signature level. The same is true for the pairwise results.

We therefore conclude that the measurement process which takes a physical LEA, creates a measured 3D surface profile, and extracts a 2D signature has a high level of reproducibility, with the largest amount of variability introduced by differences in engraving patterns on different bullets, and minimal variability introduced by differences in scanning environment such as bullet and operator. Where differences do exist, they typically interact with a bullet effect, leading us to infer that how operators adjust scanning procedure across bullets and bullet types can impact the reliability of the data obtained.

We also conclude that the largest source of variability in pairwise similarity scores originates from differences across bullets, but there is also a large amount of unexplained variability in the residual structure. Differences in operators and machines contribute minimally to pairwise score variability. This structure of results provides support for employing a bullet-to-bullet score which incorporates the pairwise scores from all 6 LEAs to come up with a bullet-level similarity score. Aggregating data at the bullet level may reduce the pairwise-level variability observed in our results.

3.6 Discussion

Neither the signature-level results nor the pairwise results exist in a vacuum. Both data structures exist within a data pipeline, and as such their reproducibility depends on several other components within and outside of that process.

3.6.1 Pipeline Processing

The extracted signatures used for the signature-level model rely on multiple non-study-related components. They are processed using both an automated "Crosscut ID" as well as a "Groove ID". For the purposes of this study, Groove ID was done by manual identification by the authors. While that increases accuracy of GEA data removal, it also introduces human decision-making into the data processing steps. The "Crosscut ID" function also depends on the parameter settings, and is a relatively ad-hoc approach which uses a maximum cutoff for percentage of missing values and a minimum cutoff for cross-correlation. That function can cause averaged profiles – and therefore extracted signatures – to differ for otherwise similar scans, if the chosen crosscut for one scan is in a much different area than for another. Just as Groove ID is studied in Chapter 2, Crosscut ID is a step in the data process that should be studied and improved. In addition, all parameters used throughout the process – from LOESS spans to extract the signature, to random forest parameters – can have an effect.

Outside of the data processing pipeline, bullet curvature can also play a role in creating low-scoring same-source scores. If a bullet is angled incorrectly when staged by an operator, or the bullet curvature is slightly warped, features can be distorted. When this occurs, two striation patterns that should look very similar can end up receiving a very low score due to misrepresented data structures.

As with all data processes, the output depends on the input. In the case of our variability study, we can see how the process performs with the current approach proposed by Hare et al. (2017). We determine that given varying input by operators and microscopes, the process is reproducible. However, there are spots of vulnerability where the algorithm needs to be improved upon and reinforced.

3.6.2 Outcomes and Future Work

We have developed a logical approach to measuring repeatability and reproducibility for a complex data structure that is applied within a large-scale data analysis pipeline and completed a study of measurement variability on 3D surface profiles of bullet LEAs. The modeling approach, while not novel in parameter estimation, is novel in approach and application to a new data type. The measurements calculated provide a novel look at the reproducibility of a proposed process for automation of a forensic evidence type.

In addition, the variability estimates reported from the study provide two major insights for the data collection process. First, the study outcomes can be used as training materials to help operators avoid common pitfalls as well as ensure they are using the correct lighting and settings when faced with a "tricky" bullet. Secondly, the parameter estimates estimated here can be used to provide an indicator of a problematic barrel-land or a low quality scan. Scans that fall outside of predetermined ranges for scan variability from other scans can be flagged as having a potential quality issue such as lighting, tank rash, or crosscut misidentification.

Future work should consider two major approaches: (1) investigating aggregated bullet-to-bullet scores and whether they improve the repeatability and reproducibility of pairwise matching scores, and (2) investigating vulnerable spots within the data process such as processing steps and parameter settings. The work presented here provides a framework for future studies of the latter; with well-defined models to measure variability at two stages of data,

processing steps between and around those data stages can be manipulated and the resulting effect on data can be measured.

CHAPTER 4. A FRAMEWORK FOR ADAPTIVE COMPUTATIONAL REPRODUCIBILITY IN A CHANGING SOFTWARE PACKAGE LANDSCAPE

4.1 Introduction

Any data analysis process can be conceptualized as a modularized, linear pipeline of sequential actions applied to data. In a functional data pipeline, each action may depend on underlying code from statistical software. For data pipelines implemented in `R` or `Python`, the data pipeline actions often rely on code in the form of other users' contributed packages.

The language models of both R and Python allow and fundamentally rely on including user-developed packages to enhance the language's data science capabilities. The languages themselves provide a framework for object manipulation and function syntax, as well some basic data structures. However, many common data manipulation and modeling abilities are available to language users due to supplementary packages rather than the base language itself.

The package development framework, as discussed in section 1.2, means many packages develop and change over time. This is doubly true for packages which are widely used and have users who report bugs and functionality requests to package developers. Changes by developers can adversely affect results of code which utilizes those packages as part of an automated or semi-automated data pipeline. Collaborators on one project using different versions of a myriad of packages are also subject to code differences in different package versions on each machine.

We have already seen in the previous two chapters how interconnected the steps of a data process can be, and how dramatically changes in one step of the process can alter results obtained at another step. However, neither of the two previous studies focused on computational changes (i.e., changes in underlying code).

For a complex data analysis process such as the automated bullet matching pipeline, there are many externally-developed R packages involved as well as many researcher-developed functions which together complete data manipulation and analysis steps. There is a complex network of code and package dependencies which exists in parallel to the statistical dependence of each step in the data pipeline on all decisions made at all the previous steps.

Dependence on developed packages leaves the bullet analysis pipeline in a vulnerable position; it can be affected by code changes by researchers within the bullet analysis team, external code changes in packages upon which the researchers' tools are built, and differences in utilized package versions across team members.

The large-scale bullet analysis pipeline is just one example. The R package landscape is constantly changing, which leaves statisticians, data scientists, and other researchers vulnerable to computational reproducibility foibles.

The benefits of utilizing innovative packages often outweigh the risks introduced by a shifting package landscape. However, there is still a need for both a framework and tools to navigate the changing landscape of analyzing data in R.

We propose here a framework and accompanying tools for managing the ever-changing problem that is computational reproducibility in R. We demonstrate the functionality of our developed tools available in R as the `manager` package, and present two case studies in managing computational reproducibility when changes occur to packages in use.

4.2 Defining a Framework for Computational Reproducibility

Differences in package versions and functionality are an inevitability for many popular R packages, particularly those that are part of the `tidyverse`, a popular suite of packages which are subject to continual changes and improvements (Wickham, 2017). R packages are not automatically updated; it is completely up to the user to install any non-base packages and update those packages over time. Therefore, it is more common than not that any two users will have a different set of packages installed or different package versions. One user may update a package and find its functionality differs from a previous version, or two users on different machines may find the functionality of one package differs between them.

When disparities between functionality exist, it indicates that there are two different versions of a package involved. Some previous reproducibility approaches in R such as the `packrat` and `checkpoint` packages (Ushey et al. (2018), Ooi et al. (2020)) provide package versioning to assist in *static* reproducibility; they allow users to keep track of package versions in use in a project or in a script, and continue to use that same library of package versions. This assists in identical re-production of results over time and on different machines. However, the reproducible product exists at a static point in time for package versions.

We propose conceptualizing computational reproducibility in R as an *adaptive* reproducibility problem. Adaptive reproducibility focuses on actively tracking changes within a data analysis process and ensuring users are aware of changes that may affect results of a particular process. In our view, tools which emphasize adaptivity are the key to successful reproducibility both over time and across teams or machines.

This is an advantageous approach for several reasons. First, while there may be a host of *nominal* package version differences between two machines or two time points, the list of *practical* package version differences may be much smaller. Practical version differences are differences in code functionality or syntax which have the potential to actually change the

outcome of code implementation. Second, pinpointing specific package functions or objects which are different between two sets of package versions allows direct determination of whether those pinpointed "practical" changes are used in the project or script in question. A function may practically differ between two versions of a package in use, but that function itself may not be involved in the process at hand. If that is the case, it allows collaborators to do one of two things: (1) safely use two different versions of a package to achieve the same result, or (2) upgrade and utilize the newest version of the package on both machines knowing that the result will not be affected due to changes over time. Option (2) is often desirable as package updates often include bug fixes or improved code implementation.

On the other hand, if the identified difference across versions will potentially or surely impact resulting code and if users have an identified list of those impactful changes, they can either (1) ensure versioning methods are employed to reproduce code using the same package version across machines and over time, (2) take steps to adapt the data analysis process to work with a new version, or (3) robustify the process to account for differences in functionality. This certainly does not enumerate all possible approaches users can take to address identified differences, nor should it. Adaptive computational reproducibility, just like all reproducibility approaches, differs by project and user. The key is that using an *adaptive* approach allows users to identify meaningful package changes and choose their own approach to address those disparities within their own projects.

We approach adaptive computational reproducibility using a two-step approach. First, we identify the extent of package dependency within a project or process by taking a project-level or script-level *inventory* of a set of packages. When considered early in a project, steps can be taken to reduce the level of package dependency involved with a process in the first place. If the package inventory is overly complex, users may consider re-coding certain aspects of the process using base code or an alternative package.

The second step introduces the adaptability aspect; after time has passed, or when another user on a separate machine needs to implement the process, a second inventory can be taken and changes within the package inventory can be automatically identified prior to re-implementation of the process or script. Identified differences in the package inventory can be used to subsequently identify whether any code utilized within the process or script will be affected.

The adaptive aspect of computational reproducibility is therefore achieved by comparing static versions of a project inventory captured at different moments in time. The comparison of static versions requires that the reproducibility effort be initiated *before* it is needed; capturing an initial static inventory of a project's dependency network is required in order for the inventory to be later compared to an updated version or a version on another user's machine. However, following this approach provides a greater level of detail and confidence when identifying and addressing changes in the project's package dependency network.

We next present tools which we developed to assist users in implementing this two-stage approach to adaptive computational reproducibility. These tools are available for use in R as part of the `manager` package.

4.3 Capturing and Visualizing a Package Inventory

The first set of tools is meant to achieve the first step in adaptive adaptive reproducibility: enumerating the network of package dependencies and assessing its size and complexity. We achieve this using the `take_inventory()` and `plot_inventory()` functions.

Consider the conceptual data process shown in Figure 4.1. Suppose that a script requires the use of the five packages name in Steps 2-5 (`tidyr`, `stringr`, `purrr`, `dplyr`, and `randomForest`). Those five packages are considered the *direct* (or *explicit*) package dependencies. However, there exists a larger network of packages that are *hidden* (or *implicit*) dependen-

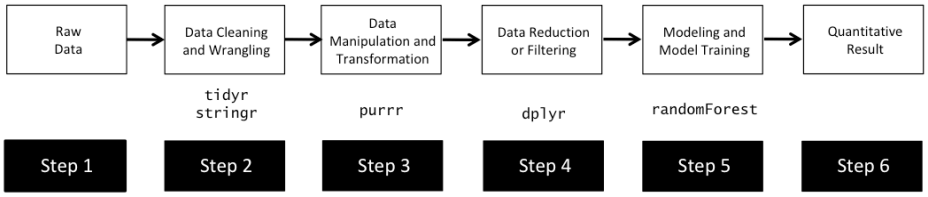

Figure 4.1 Example data analysis process with five packages used to accomplish the data analysis task.

cies because the set of direct dependencies rely on those packages. The hidden dependencies may not be directly required by the user within their script or project, but carry the potential to impact any functions in other packages which rely on their code.

To enumerate this dependency structure, we use the `take_inventory()` function. The `take_inventory()` function takes a vector of R package names as an argument and returns a nested `data.frame` object which contains all explicit and implicit packages as well as their objects. The full set of packages included are identified using an iterative level-by-level approach:

1. Objects from all explicitly called packages are cataloged saved at "Level 0".

2. All packages explicitly listed as Depends or Imports to Level 0 packages are identified as "Level 1 packages". Objects from each package are cataloged and saved. Any Level 1 packages which are already in the set of Level 0 packages are not included at Level 1, as they have already been cataloged once at the top level.

3. Repeat Step 2, increasing the package level with each repetition, until there are no new packages in the list of Depends and Imports from the current level.

The level-by-level scheme results in an exhaustive list of all packages involved – either explicitly or implicitly – when the "Level 0" packages are called by a user as well as a level-by-level set of dependency connections. It does not provide a complete network of all package dependency relationships. The `pkgnet` package can be used to identify a complete package

dependency network for a single package (Burns et al., 2019). We focus here instead on the extent of dependence for a project or script, which typically involves multiple explicit package dependencies and many implicit package dependencies.

The `plot_inventory()` function takes an inventory object (as returned by `take_inventory()`) as an argument and returns a `ggplot` visualization of the dependency structure, in the form of a level-by-level "tree". For the list of five example packages shown in Figure 4.1, the resulting inventory is visualized as a tree using `plot_inventory()` in Figure 4.2. The following code generated Figure 4.2:

```
library(devtools)
devtools::install_github("kiegan/manager")
library(manager)

inventory_example <- take_inventory(packages =
                        c("tidyr", "stringr", "purrr", "dplyr", "randomForest"))
plot_inventory(inventory_object = inventory_example)
```

Note that while the packages do not have to themselves be called via a `library()` statement in order for the code to run, the packages do need to be installed on the computer in use for the function to include them in the inventory.

We observe several things in Figure 4.2: first, the interconnectedness of multiple packages involved in our process. `tidyr`, `stringr`, `purrr`, and `dplyr` – which are all part of the `tidyverse` suite – share multiple Level 1 package dependencies. Second, we can see that there are in fact multiple levels of hidden dependency involved when the five packages in the example are called. Finally, we see that when these five packages are invoked by a user, the user is implicitly invoking a total of thirty unique packages, with six of those packages considered part

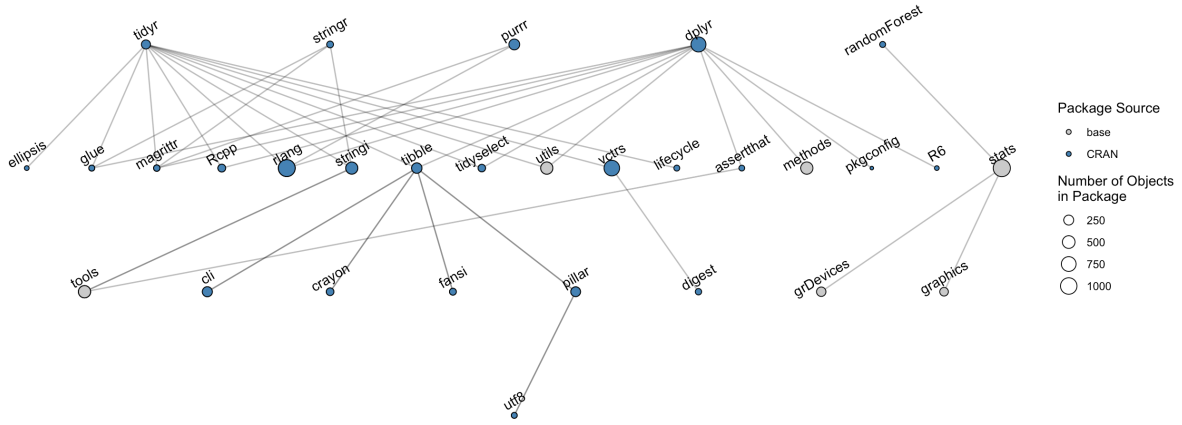

Figure 4.2 Dependency tree extracted when taking an inventory of the packages `tidyr`, `stringr`, `purrr`, `dplyr`, and `randomForest`. The top row denotes Level 0, the explicit packages called. The second, third, and fourth rows denote Level 1, 2, and 3 layers of implicit dependencies. There are a total of thirty packages in the dependency tree, with six base packages and twenty-four non-base packages.

of the "base" suite in R version 3.6.0. That leaves twenty-four packages that are subject to change.

Twenty-four packages being vulnerable to change is a natural overestimate, as there are many non-"base" packages whose functionality is unlikely to change. However, this exposes just how complex package dependency structures can get very quickly with a small number of explicitly required packages.

The combination of taking an inventory and visualizing an inventory using the tools in `manager` allows a user to quickly assess the complexity of their package dependency structure. This assessment can facilitate strategic decisions by users to remove a dependency or re-write code to remove some package vulnerabilities, which achieves the first goal of adaptive computational reproducibility: defining and refining the scope of package dependence.

4.4 Comparing Inventories Over Time or Across Users

The second step in adaptive computational reproducibility, comparing static versions of inventories to identify code differences over time or across machines, can also be accomplished using two tools provided in `manager`: `take_inventory()` and `compare_inventory()`. These two tools work together using a key part of the package inventory captured by `take_inventory()`: strategically-placed object identifiers that facilitate object comparison over time.

MD5 checksums are strings of 32 hexadecimal digits calculated by a cryptographical hash function. MD5 checksums serve as an algorithmically-generated summary of an object. In R, these can be calculated using the `digest` package (Eddelbuettel et al., 2020). Using a checksum to summarize the contents of an object is a natural choice for easy comparison of two functions or objects in a package; it is computationally much faster to compare two hexadecimal strings and determine whether they match than two compare two entire functions or objects and determine whether they match.

The inclusion of checksums at strategic locations in an inventory also allows for comparison of aggregated meta-information at high levels. If two users have the same exact version of a package, there is no need to compare the objects within those versions and their contents can be filtered out of the comparison process early. However, if the users have different package versions, a closer look at package object differences is important and those packages need to be kept for further comparison.

We implement the MD5 algorithm using the `digest()` function in the `digest` package and save checksums for the following sets of information when taking an inventory:

1. **Package meta-information**: package name, package source (CRAN, GitHub, local, base), package version, and GitHub repo and commit if a GitHub package.

2. **Package dependencies**: the list of package dependencies (depends and imports) for each package.

3. **Object within a package**: each object within each package. Objects can be exported functions, non-exported functions, data objects, or other formats. Objects are deparsed before the checksum is calculated[1].

4. **Parameter information**: small `data.frame` of parameter names, types, and default values for functions.

5. **Object meta-information**: each object's meta-information, including the name, whether it is exported, whether it is "exported", "non-exported", or "data", and the object-level checksum and parameter-level checksum.

This set of checksums facilitates computationally quick comparison across two inventories, by iteratively filtering out packages and package objects whose summarizing checksums are identical. This process is implemented in the `compare_inventory()` function in the `manager` package. `compare_inventory()` takes the following arguments:

`inventory1`: An inventory object, as returned from the `take_inventory()` command.

`inventory2`: A second inventory object, as returned from the `take_inventory()` command. `inventory1` and `inventory2` should typically be from different points in time or different users, for comparison.

`object_types`: A vector of object types to include in the comparison. Object types within the inventory are split into three categories: `exported`, `non-exported`, and `data`. Default is `"exported"` objects only, but can be any of `c("exported", "non-exported", "data")`.

[1]Note that deparsing does not account for compiled C or C++ functions which include a machine-specific address for the compiled function. Future work aims to address this issue.

`report_type`: A vector of reporting preferences. `"table"` returns a data.frame with one row for each object that differs between the two inventories. `"summary"` returns concatenated text summaries of identified differences by package. `"objects"` returns the "table" but also includes the full deparsed objects for inventory1 and inventory2, as well as additional meta-information.

`summary_file`: A character filepath with the location to store difference summary information. Default is `""`, which will return the text summary in the R console.

The object returns a list with elements `"table"` and `"objects"` as specified by the `report_type` parameter. `"table"` is a `tibble` containing a row for each object that differs between the two inventories, and the following columns:

`package_name`: character.

`object_name`: character.

`object_presence`: character. A description of whether the object is present in each inventory. Some objects are added or removed in different versions.

`object_changes`: character. A description of whether the object differs, its parameters differ, or both.

`obj_same`: logical. `TRUE` if objects are identical in both inventories (i.e., if their checksums are identical).

`params_same`: logical. `TRUE` if object parameter data is identical in both inventories (i.e., if the parameter data checksums are identical).

To demonstrate the output returned by `compare_inventory`, we use an example. Two inventories of the five packages used in Figure 4.1 and Figure 4.2 were taken by two different

users on two different machines on the same day. We will refer to them as "User A" and "User B", who captured inventories 1 and 2, respectively. We compare them using the code below:

```
## run on one machine
userA_inventory <- take_inventory(packages =
                        c("tidyr", "stringr", "purrr", "dplyr", "randomForest"))
## run on second machine
userB_inventory <- take_inventory(packages =
                        c("tidyr", "stringr", "purrr", "dplyr", "randomForest"))


compare_AB <- compare_inventory(inventory1 = userA_inventory,
                                inventory2 = userB_inventory,
                                report_types = c("table", "summary", "objects"),
                                summary_file = "compare_AB.txt")
```

```
head(compare_AB$table)
#> # A tibble: 6 x 6
#>   package_name object_name object_presence object_changes obj_same
#>   <chr>        <chr>       <chr>           <chr>          <lgl>
#> 1 cli          is_ansi_tty Object in both~ Objects chang~ FALSE
#> 2 digest       digest      Object in both~ Objects chang~ FALSE
#> 3 dplyr        contains    Object in both~ Both object a~ FALSE
#> 4 dplyr        ends_with   Object in both~ Both object a~ FALSE
#> 5 dplyr        enquos      Object in both~ Objects chang~ FALSE
#> 6 dplyr        everything  Object in both~ Both object a~ FALSE
#> # ... with 1 more variable: params_same <lgl>
head(compare_AB$objects[,c("package_name", "object_name",
                           "function_text_inv1", "function_text_inv2")])
#> # A tibble: 6 x 4
```

```
#>   package_name object_name function_text_inv1 function_text_inv2
#>   <chr>        <chr>       <list>             <list>
#> 1 cli          is_ansi_tty <chr [7]>          <chr [14]>
#> 2 digest       digest      <chr [68]>         <chr [68]>
#> 3 dplyr        contains    <chr [9]>          <chr [9]>
#> 4 dplyr        ends_with   <chr [12]>         <chr [10]>
#> 5 dplyr        enquos      <chr [10]>         <chr [10]>
#> 6 dplyr        everything  <chr [4]>          <chr [4]>
```

`"objects"` is a `tibble` containing a row for each object that differs. It contains all the same columns as `"table"`, as well as more detailed object information, including:

`package_source_inv1`: character. Source of the package in inventory1 (CRAN, GitHub, local, base).

`package_source_inv2`: character. Source of the package in inventory2 (CRAN, GitHub, local, base).

`package_version_inv1`: character. Version of the package in inventory1.

`package_version_inv2`: character. Version of the package in inventory2.

`package_gitrepo_inv1`: character. GitHub repository for package in inventory1. `NA` if CRAN, base, or local package.

`package_gitrepo_inv2`: character. GitHub repository for package in inventory2. `NA` if CRAN, base, or local package.

`package_gitcommit_inv1`: character. GitHub commit for package in inventory1. `NA` if CRAN, base, or local package.

`package_gitcommit_inv2`: character. GitHub commit for package in inventory2. `NA` if CRAN, base, or local package.

`function_text_inv1`: nested character vector. Deparsed text of function or object in inventory1.

`function_text_inv2`: nested character vector. Deparsed text of function or object in inventory2.

`fun_params_inv1`: nested data.frame with columns `param_name`, `param_type`, and `param_default`. `NA` if object is not a function.

`fun_params_inv2`: nested data.frame with columns `param_name`, `param_type`, and `param_default`. `NA` if object is not a function.

`obj_checksum_inv1`: character. MD5 checksum of `function_text_inv1`.

`obj_checksum_inv2`: character. MD5 checksum of `function_text_inv2`.

`params_checksum_inv1`: character. MD5 checksum of `fun_params_inv1`.

`params_checksum_inv2`: character. MD5 checksum of `fun_params_inv2`.

The reporting style of "one row per object" allows users to get a quick overview using either the `table` or `objects` elements, or by reading the text `summary`. We include key information describing the nature of the object difference in both the `table` and `objects` frames to assist users in both summarizing differences and filtering to focus on a particular type of difference.

For example, the `object_presence` variable reported gives three possible options:

(1) Object in Inventory 1 but not Inventory 2

(2) Object in Inventory 2 but not Inventory 1

(3) Object in both Inventories

This categorization provides useful information. If a user wants to compare a new inventory (Inventory 2) to an older version of the inventory (Inventory 1), and the object is in Inventory 2 but not Inventory 1, that suggests that the object is an *addition* to the package. In that case, a script or process in question which utilized Inventory 1 package versions wouldn't be impacted, as it wouldn't have been able to call that object.

Conversely, if an object is in Inventory 1 but not in Inventory 2, that suggests it has been *removed* from the package. This could adversely impact any code which relies on that function. Complete removal of objects is uncommon; however, when comparing inventories across two users, it is not uncommon for one user to have a newer version of one package but an older version of a different package, as compared to a second user. In this scenario, we see both object "additions" and object "removals", which are most likely the *absence of a new addition* rather than a true "removal". If one user writes code which relies on a newer set of functions, it is important to identify that discrepancy.

Reporting the presence of package objects in one or both packages allows users to narrow down problematic differences specific to their own needs on a project, and whether they are comparing across users or with themselves over time.

We include the `object_changes` descriptive column for similar reasons. It has three possible options:

(1) Object dropped or added (corresponding to options (1) and (2) in `object_presence`)

(2) Objects changed but parameters unchanged

(3) Both object and parameters changed. Object change may be due to parameter changes.

We consider parameters changed separately from the object changing due to default parameters being widely used for many packages. If the default parameters change, that is important to know. Note that there is not an option for "object has changed but parameters have not." This is due to the fact that when considering a package object such as a function, the default parameter vector is included in the function, so it is difficult to separate changes in the parameters from changes in the object. One future direction is to improve our tools to more clearly separate elements of a package object for comparison.

The `objects` element adds the deparsed objects and parameter data to the key descriptive information provided in `table` in order for users to directly investigate differences in objects:

```
# investigate a single object:
# "ends_with" from the "dplyr" package
compare_AB$objects[4,c("package_name", "object_name",
                       "function_text_inv1", "function_text_inv2")]
#> # A tibble: 1 x 4
#>   package_name object_name function_text_inv1 function_text_inv2
#>   <chr>        <chr>       <list>             <list>
#> 1 dplyr        ends_with   <chr [12]>         <chr [10]>

# version in inventory 1
compare_AB$objects[4,"function_text_inv1"][[1]]
#> [[1]]
#>  [1] "function (match, ignore.case = TRUE, vars = peek_vars()) "
#>  [2] "{"
#>  [3] "    stopifnot(is_string(match), !is.na(match), nchar(match) > "
#>  [4] "        0)"
#>  [5] "    if (ignore.case) "
```

```
#>  [6] "        match <- tolower(match)"
#>  [7] "    n <- nchar(match)"
#>  [8] "    if (ignore.case) "
#>  [9] "        vars <- tolower(vars)"
#> [10] "    length <- nchar(vars)"
#> [11] "    which_vars(match, substr(vars, pmax(1, length - n + 1), length))"
#> [12] "}"


# version in inventory 2
compare_AB$objects[4,"function_text_inv2"][[1]]
#> [[1]]
#>  [1] "function (match, ignore.case = TRUE, vars = peek_vars(fn = \"ends_with\")) "
#>  [2] "{"
#>  [3] "    check_match(match)"
#>  [4] "    if (ignore.case) {"
#>  [5] "        vars <- tolower(vars)"
#>  [6] "        match <- tolower(match)"
#>  [7] "    }"
#>  [8] "    length <- nchar(vars)"
#>  [9] "    flat_map_int(match, ends_with_impl, vars, length)"
#> [10] "}"
```

These tools provide functionality for users to quickly and automatically identify and categorize differences in package objects between any two given inventories. On a project-by-project basis, they can use the results from combining `take_inventory()` and `compare_inventory()` to identify differences between the inventory of packages used to run a script, process, or project.

4.5 Identifying Script Vulnerabilites

Often, the list of differences will be large between two inventories, but the majority of objects which have differences will not be involved in a data process or script. We also provide a tool to assist in further identifying differences which can impact a particular script.

The `script_check()` function takes two major arguments. The first is a `compare_object`, which is the comparison object resulting from comparing two inventories using `compare_inventory()`. The second is a filepath to a desired R script that needs to be checked for use of functions that differ between the two inventories. There are three additional arguments:

`is_R_script`: Logical. Default is `TRUE`, which assumes the provided script is a .R file. If set to `FALSE`, `knitr::purl()` is applied to convert file to raw R script prior to parsing. This accommodates the use of .Rmd and .Rnw files, for example.

`include_specials`: Logical. Default is `FALSE`, which will exclude special characters such as the `magrittr` pipe `%>%` from being considered when checking the script.

`summary_file`: Character path to a text file to save the summary output. Default is `""`, which will print the summary in the console.

The function returns a text summary, either in the console or written to a text file as specified using `summary_file`, of objects that are different between the two inventories and appear in the script in question. It provides the lines in the script at which the object is called. This tool is meant to help users pinpoint locations where their script can be potentially affected by changes in package objects across users or over time, so that they may focus on checking those specific objects. It will save users the time and energy of manually searching their own scripts for affected actions applied to data. In addition, it simplifies version control and collaboration concerns in many cases. If a script in question used by multiple users or

being re-run at a later date does not utilize any of the functions or objects that differ between the two inventories, the user can safely use either version of the inventory to run the script.

Future work to expand this functionality will include automatically "mapping" a script to determine at which point a script is affected, and whether it affects subsequent portions of the script.

4.6 Case Studies

In order to get a sense of both the utility of our tools and the importance of considering computational reproducibility issues, we highlight two R packages as case studies. The first package, `tidyr`, is a widely-used data cleaning package that is actively under development and has experienced functionality and syntax changes within the last year (Wickham and Henry, 2019). The second package, `bulletxtrctr`, is an in-development package used and developed by a group of researchers to complete a particular type of data analysis (Hofmann et al., 2019a).

4.6.1 `tidyr` example

The `tidyr` package is a widely-used data cleaning and tidying package which is part of the `tidyverse` suite of packages (Wickham and Henry (2019), Wickham (2017)). `tidyr` provides functions which allow users to easily manipulate data from a variety of data structures into formats which work best for their own analysis. Some of these functions, such as `spread()` and `gather()`, focus on converting data between a "long" format and a "wide" format. A "long" format typically provides one row per observation (or response variable), while a "wide" format may provide multiple observations within one row. Other functions, such as `nest()` and `unnest()`, focus on grouping or ungrouping data by one or more key variables, in order to apply actions to each set of data separately.

`tidyr` and the functions it provides, while widely used, are subject to changes by the developers. One recent example is the replacement of the `spread()` and `gather()` functions by two new functions: `pivot_wider()` and `pivot_longer()`[2]. However, the package developers have not removed the `spread()` and `gather()` functions, simply provided an updated alternative. Users can still safely use the same `spread()` and `gather()` functions even after updating the package.

In other cases, changes are made to the syntax and functionality of pre-existing functions. To investigate the impact this has on R code, we can study the differences between two versions of the package. We consider two inventories gathered by the same user on the same machine: first, an inventory with `tidyr` version 0.8.3, which was released on March 2, 2019[3]. The second inventory was taken after the user re-installed the package at version 1.0.0, released September 13, 2019, which is just over six months later. An overview of both dependency trees for that user is shown in Figure 4.3. The dependency trees have the same number and set of dependencies, but the structure differs in that version 1.0.0 includes more packages at "Level 1", the set of packages which are explicitly listed as Depends or Imports in the package description.

After applying `compare_inventory()` to the two inventories, we get the summary shown in Figure 4.4, which shows 48 object differences between the two versions. While some differences would not affect pre-existing code, such as the addition of the `pivot_longer()` and `pivot_wider()` functions, others are not so benign. Consider the `nest()` function, for example. The `ChickWeight` data, a dataset included in R `datasets`, is a data set which has measured weights of chicks from an experiment where chicks were fed multiple diets (R Core Team, 2020). When using `tidyr` version 0.8.3, the following code produces a result of nested subsets of data for each "Diet", with columns named "Diet" and "data":

[2] https://github.com/tidyverse/tidyr/releases/tag/v1.0.0
[3] https://github.com/tidyverse/tidyr/releases/tag/v0.8.3

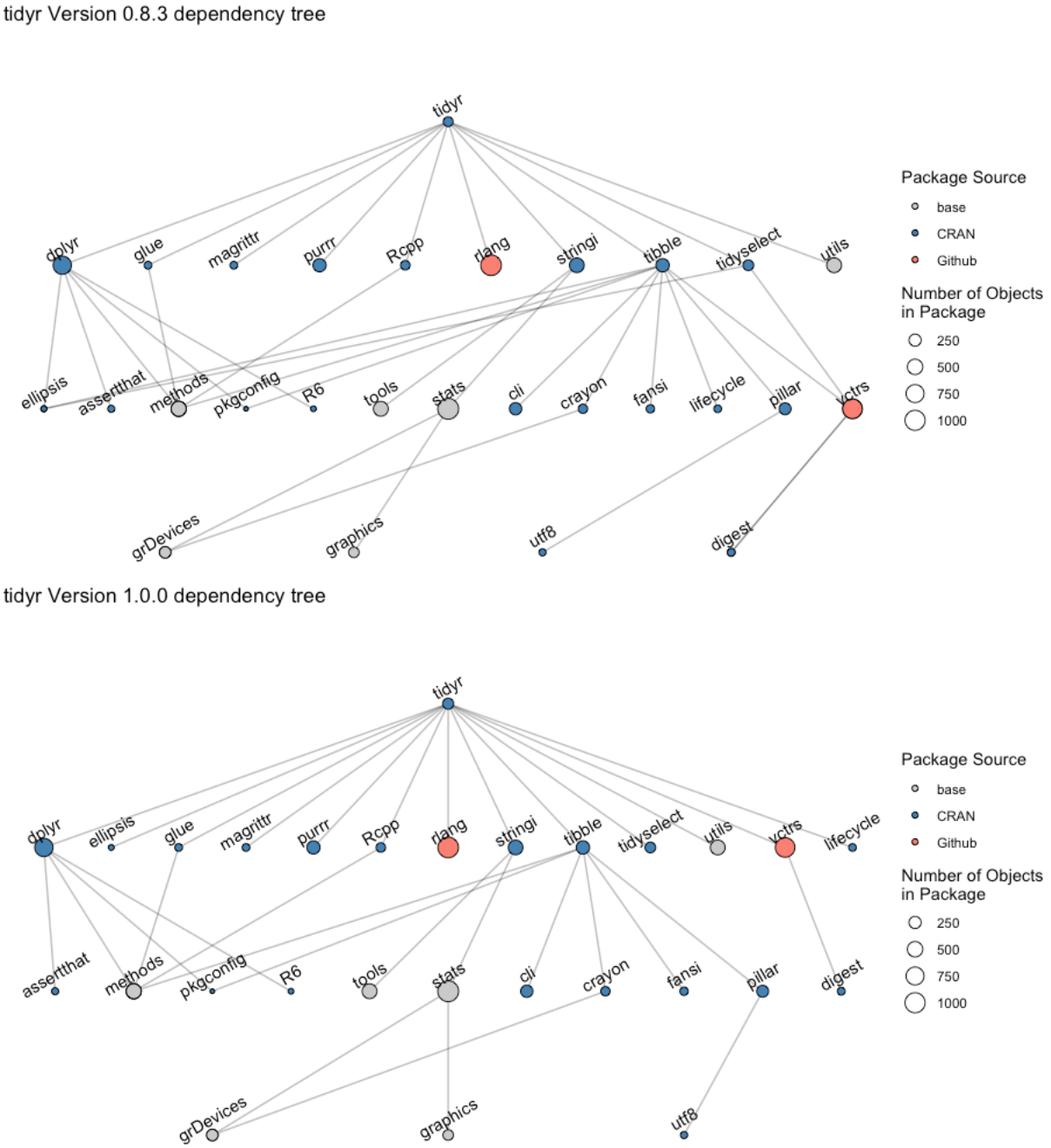

Figure 4.3 Dependency trees for versions 0.8.3 (top) and 1.0.0 (bottom) of the `tidyr` package, taken on one computer by one user before and after installing version 1.0.0. The number of implicit dependencies is the same for the two versions, but there are more Level 1 dependencies for version 1.0.0, which demonstrates an increased number of packages included in the Depends or Imports for the package as part of the version update.

```
The 'tidyr' package has 48 object differences between the two inventories. Differences were
identified in the following objects (** denotes parameter changes):

1. as_tibble
2. build_longer_spec
3. build_wider_spec
4. chop
5. complete_
6. contains
7. crossing
8. crossing_
9. drop_na
10. drop_na_
11. ends_with
12. everything
13. expand_
14. expand_grid
15. extract
16. extract_
17. fill**
18. gather
19. hoist
20. last_col
21. matches
22. nest**
23. nest_**
24. nest_legacy
25. nesting
26. num_range
27. one_of
28. pack
29. pivot_longer
30. pivot_longer_spec
31. pivot_wider
32. pivot_wider_spec
33. replace_na
34. separate
35. separate_rows
36. starts_with
37. tibble
38. tidyr_legacy
39. tribble
40. unchop
41. unite**
42. unnest**
43. unnest_**
44. unnest_auto
45. unnest_legacy
46. unnest_longer
47. unnest_wider
48. unpack
```

Figure 4.4 Text summary resulting from the `compare_inventory()` function when comparing the inventory taken with version 0.8.3 and the inventory taken with version 1.0.0 of the `tidyr` package.

```
ChickWeight %>% nest(-Diet) %>% glimpse()
```

```
#> Rows: 4
#> Columns: 2
#> $ Diet <fct> 1, 2, 3, 4
#> $ data <list> [<data.frame[220 x 3]>, <data.frame[120 x 3]>, <dat...
```

Using the version of `nest()` in `tidyr` version 1.0.0, however, produces a different result:

```
ChickWeight %>% nest(-Diet) %>% glimpse()
```

```
#> Rows: 4
#> Columns: 2
#> $ ...1 <fct> 1, 2, 3, 4
#> $ data <list> [<data.frame[220 x 3]>, <data.frame[120 x 3]>, <dat...
```

in which the `Diet` column is named `"...1"`[4]. In any script which later calls the column name `Diet` for any reason whatsoever, the code will not run as it will not be calling the correct column.

This is a difference which is easy to remedy; simply by updating the syntax used inside the `nest()` function or by applying the `group_by()` function from the `dplyr` package before applying `nest()`, the difference in result is resolved. However, the differing result demonstrates two major points: first, that changes over time to widely-used, common functions can affect the outcome of a data analysis process or the ability of a script to run successfully. Secondly, it demonstrates that a framework for addressing computational reproducibility is sorely needed in the changing R package landscape, and our `manager` tools provide a very useful approach to managing those changes.

[4]It is worth noting that in the R console, the version 1.0.0 code displayed here will throw an error and fail to return a result, but an RMarkdown document will complete the nest() task but with the incorrect name. This is true as of April 15, 2020, using R version 3.6.0 in RStudio.

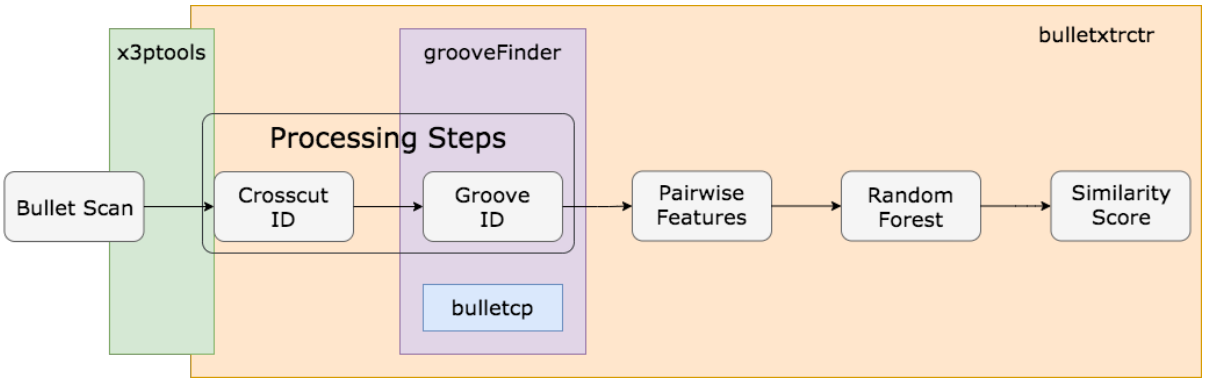

Figure 4.5 The bullet data analysis process described in Hare et al. (2017) and the four packages used to accomplish the data analysis task. `x3ptools` is used to work with the x3p data format and process the data. `grooveFinder` and `bulletcp` are used for a processing step, Groove ID. `bulletxtrctr` is used throughout the data process, including the Crosscut ID processing step and the pairwise feature creation and comparison.

We next study a less widely-used package that changes more frequently.

4.6.2 `bulletxtrctr` example

The `bulletxtrctr` package is package in development that aims to perform a specific type of analysis; namely, automated comparisons of forensic bullet evidence (Hofmann et al., 2019a). The package is primarily used for data analysis by a team of faculty and graduate students at the Center for Statistics and Applications in Forensic Evidence (CSAFE). Functions within the package are under development by several members of the team and therefore are subject to change regularly often. The package, once further cemented, is meant to provide an open-source set of tools for anyone to analyze bullet data. The `bulletxtrctr` package also interacts quite heavily and relies on several other packages in development by the team: `x3ptools` (Hofmann et al., 2019b), `grooveFinder` (Hofmann et al., 2019c), and `bulletcp` (Garton, 2020).

A high-level overview of the data analysis process implemented by the suite of bullet packages is depicted in Figure 4.5. The `bulletxtrctr` package directly relies on `x3ptools` for reading in and transforming x3p data. It also directly relies on the `grooveFinder` and `bulletcp` packages to complete the Groove ID processing step.

We can also see the dependence on the three additional bullet packages in the dependency tree shown in Figure 4.6, as well as the complex dependence structure imposed by the use of `bulletxtrctr`. 100 implicit package dependencies are involved when `bulletxtrctr` is utilized. We can therefore assess the vulnerability of the data analysis process shown in Figure 4.5 by studying inventories from several members of the team.

Inventories were cataloged for five members of the team, with one member of the team taking inventories on three separate machines – a laptop, a desktop, and a research server with an RStudio interface. We study the resulting set of seven inventories to examine differences in versions across the team.

We can first examine the versions of the bullet project packages each team member has. As all users are using either GitHub or local versions of each of the four packages (`bulletxtrctr`, `x3ptools`, `grooveFinder`, and `bulletcp`). We created a series of identifiers for which Git commit each user's package was downloaded at; the "newest" (recent Git commit), "middle" (older Git commit), and "oldest" (the oldest Git commit version anyone on the team has for that package), as well as "local" (using a version in development on their computer which has not been installed from GitHub). We display these coded versions and how the differ by user and machine in Figure 4.7.

We observe several patterns in Figure 4.7. First, we see that all seven user-machine combinations have the same version of the `bulletcp` package. This package was not under active development for several months preceding the capture of these inventories, so this is not

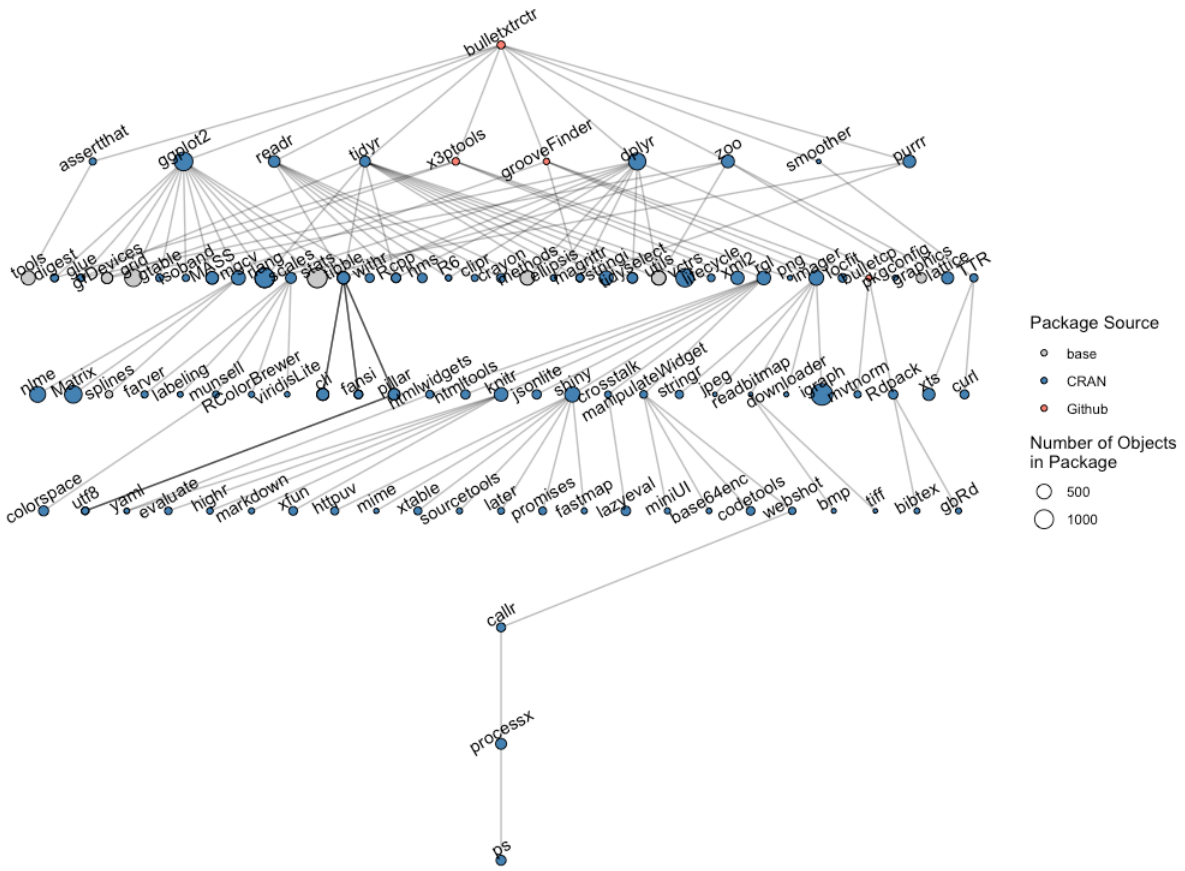

Figure 4.6 Dependency tree of the `bulletxtrctr` package. `x3ptools`, `grooveFinder`, and `bulletcp` are all shown, and all four packages originate from GitHub. The tree is very complex, with 100 total implicit dependencies.

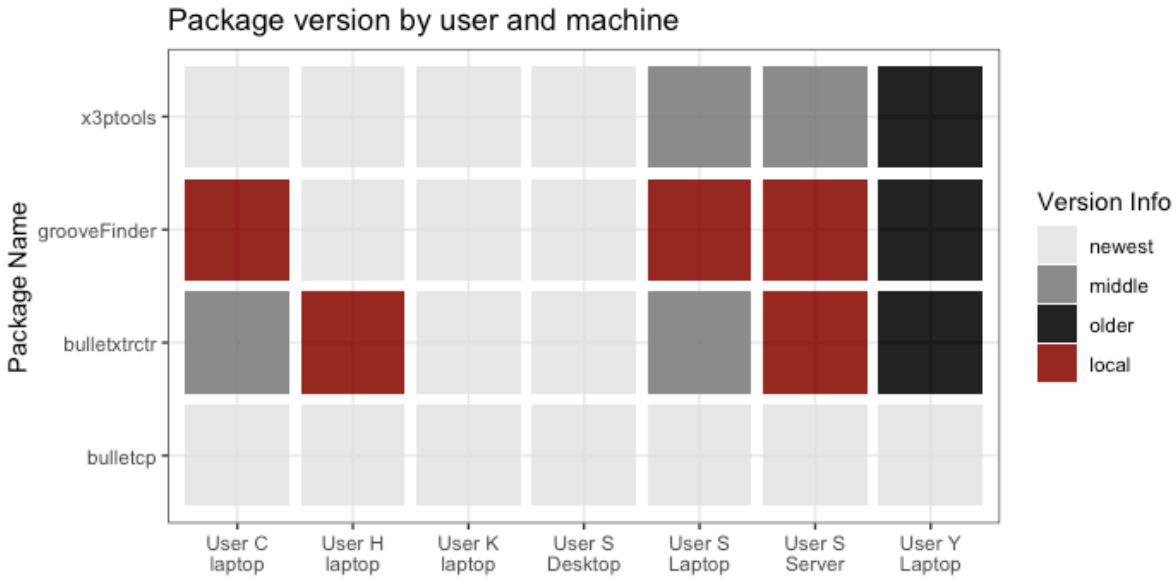

Figure 4.7 Package version breakdown by user, machine, and bullet package. Two tiles with the same color for the same package indicates that those two users or machines have the same exact version of the package. The exception is the local copies, shown in red, which all differ by user and machine.

surprising. In addition, a recent bug fix prompted re-installation of the newest version for many team members.

We also note that there are local versions of `bulletxtrctr` or `grooveFinder` on several users' machines, which indicates local package development. Local package use typically occurs when a user is in the process of changing or adding functionality. Actively worked-on objects may differ from GitHub versions as well as other local copies on other machines.

Additionally, there are only two user machines which have the exact same set of the four packages: "User K-Laptop" and "User S-Desktop". All other machines have at least one version difference from all other machines.

This introductory look at just package versions by GitHub commit and local copy shows that the team is vulnerable to many potential differences, particularly when there are different versions of two or three packages. This look also does not take into account the other 96 packages included in the dependency tree, only the four most explicitly involved in the data analysis and code development process.

In total, 73 of the other 96 packages have at least one version difference between the seven machines. We can look at a sampling of 20 of those 73 packages, shown in Figure 4.8, to get a sense for how versions differ across users.

Figure 4.8 demonstrates that versions of some packages differ very little, such as the `zoo` package. Three users have one version of `zoo` and four users have another; those two versions are also close in name – 1.8-6 and 1.8-7. The `ggplot2` package, on the other hand, exists in five different versions across seven machines, with three versions being GitHub versions all at different commits!

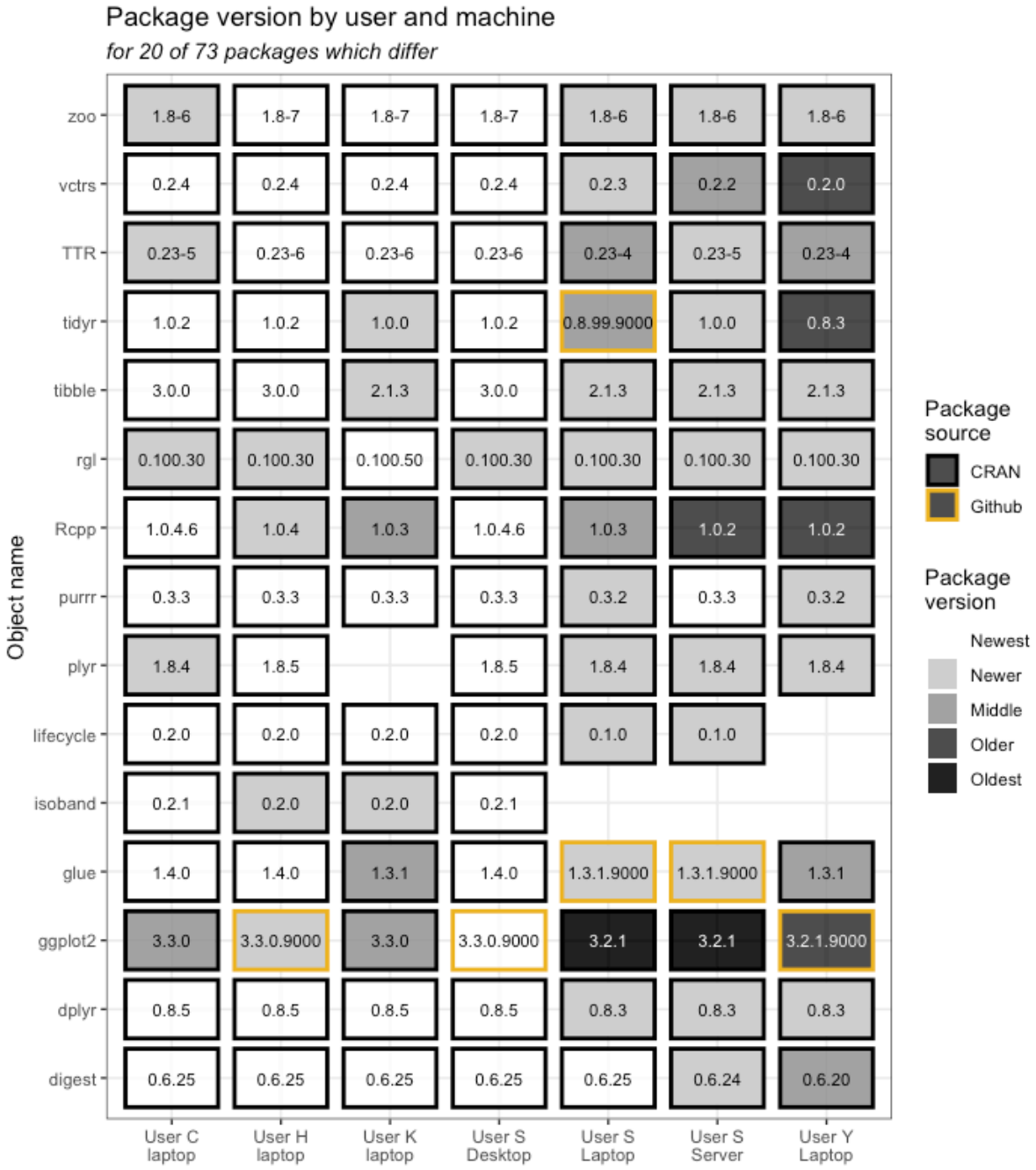

Figure 4.8 Package version breakdown by user, machine, and source (GitHub vs. CRAN) for twenty of the 73 packages in the `bulletxtrctr` dependency tree which differ across seven machines of interest. Two tiles with the same color for the same package indicates that those two users or machines have the same exact version of the package. Lighter colors represent newer versions of the package, while darker colors represent older versions.

The version differences shown in Figure 4.7 and Figure 4.8 are also only demonstrating *nominal* differences, without having accounted for *practical* differences in functions and objects across versions.

However, we can easily identify differing object versions across machines utilizing the object checksums and parameter checksums stored in each inventory for each user.

For the four bullet-focused packages (`bulletxtrctr`, `x3ptools`, `grooveFinder`, and `bulletcp`), we identify 19 functions which differ in at least one inventory, and present the extent of differing object versions in Figure 4.9. Because some function differences are local changes, we do not grade objects on a "newest" to "oldest" scale as we did with package versions, since local copies may all be equally "as new" but still differ. The resulting differences in Figure 4.9 demonstrate several phenomena.

First, we note that the functions in `x3ptools` differ only for one user, who has an older version of the package. Secondly, we see that object versions and parameter versions differ the most for functions involving the "Groove ID" step: `get_grooves_hough` and `get_grooves_lasso` in `grooveFinder` and `cc_locate_grooves` in `bulletxtrctr`. This suggests active development on those methods by one or more team members; this is indeed the case.

We again note that there are only two machines which have the same versions of all of these functions; "User K-Laptop" and "User S-Desktop". What this means practically for the research team is that those two machines could run a data analysis script using these packages and return the same result; however, that is also with the stipulation that differences in other implicit dependencies don't affect the analysis. Any other pair of machines in this set which run a data analysis using these functions are more vulnerable to receiving different results on the same data. The most concerning result we see is that there are six unique combinations of these 19 functions across seven machines.

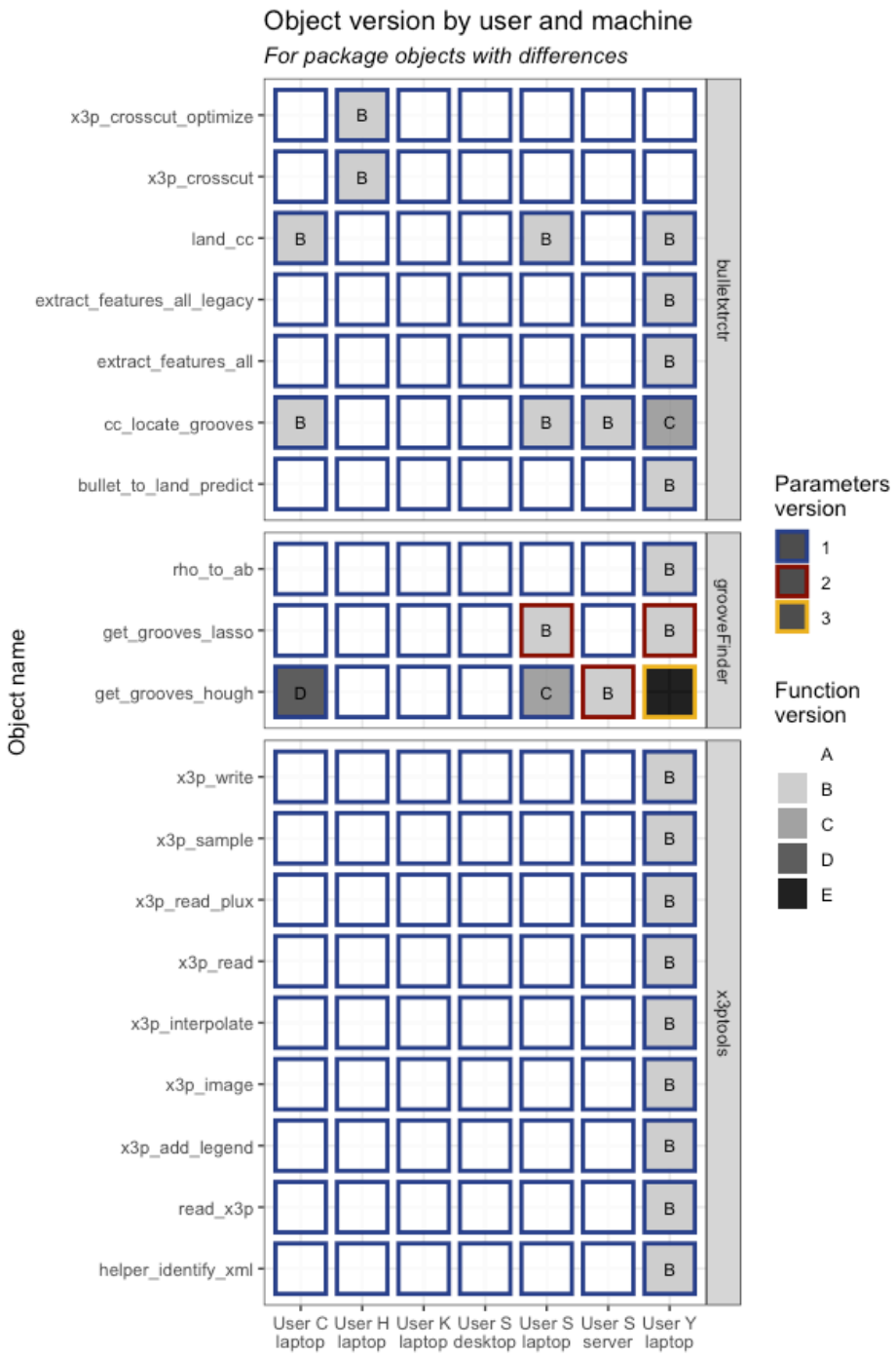

Figure 4.9 Differences in versions for 19 objects in `bulletxtrctr`, `x3ptools`, and `grooveFinder`. `bulletcp` did not have any identified differences. Function differences are denoted by color and label, not in a particular order. Parameter differences are denoted by differing colors outlining each tile. Versions B, C, and D for functions are labeled to assist in visually separating greyscale values.

Practically speaking, the team would need to identify some of the specific changes to determine whether the difference could result in a real impact on results; in many cases, these differences may be minimal or not affect the end result at all. In other cases, like the parameter difference in the `get_grooves_lasso` function, the data would almost surely be altered for users with different versions of the parameters. Being able to identify where package functions differ allows the team to more effectively assess version control issues and ensure future analyses are not impacted.

Without the tools in the `manager` package, identifying and summarizing the differences among users on the `bulletxtrctr` research team would have been much more time consuming and difficult. Additionally, the adaptive framework which allows comparisons *across* machines and users increases confidence in results when distributing work across team members and machines. If differences can be enumerated and addressed, team members will not have to question whether results may differ due to underlying code changes.

4.7 Conclusion

The framework we propose for adaptive computational reproducibility focuses on a two-step approach of first assessing the level of dependence a script or project may have to other code packages and subsequently comparing multiple package inventories to pinpoint specific changes in underlying code. This framework, which differs from many other static reproducibility methods, allows for both greater flexibility in reproducibility approaches as well as higher confidence in reproducibility of results when collaborating across machines or users.

In addition, the adaptive framework allows users to identify changes present within new package updates and adjust accordingly, which more effectively facilitates staying "up to date" with new packages and innovative data analysis approaches. However, it also allows users to identify whether updates will impact a script or process and therefore safely use multiple

versions across multiple machines if some machines have newer versions which others do not. This is advantageous for users on machines or servers who may need administrative approval to install new package versions; if an old version will complete the task in the same way, there is no need to update.

Enumerating the extent of package dependency on a project also gives users an eye-opening look at just how vulnerable their projects are to the changing package landscape, and can give users the opportunity to reduce their level of dependency and robustify their process from the start.

The tools provided through the `manager` package facilitate this framework of adaptive reproducibility management and pave the way for users to assess and address their own levels of reproducibility. They also give users the ability to identify changes that may affect their code and handle discrepancies ahead of time, as we saw with both the `tidyr` case study and the `bulletxtrctr` case studies. Being aware of the differences in `tidyr`'s `nest()` or the 19 differing functions in the bullet packages gives users the chance to patch scripts, update versions, or fix code before it becomes a problem.

Our approach to both the framework and tools presented here is a novel one, and we believe it will be very useful for many R users. The tools in the `manager` package are still under development, and we expect a multitude of added functionality in the future to streamline the adaptive computational reproducibility process further and expand its applicability. However, the framework alone provides a new way to consider computational reproducibility in R. With the growing R community and the extensive list of packages and package versions that exist both on CRAN and GitHub, this framework and tools will only become more useful over time.

CHAPTER 5. CONCLUDING REMARKS

The study of statistics and data analysis inherently involves variation. Data analysis assumes randomness and noise as part of everyday life and measurement. That assumption drives the way Statisticians approach research; all results are measured with uncertainty and levels of confidence, rather than in absolutes. It is precisely that same approach to uncertainty that drives the need for the study of reproducibility in data analysis processes.

Reproducibility in statistics research involves both statistical reproducibility and computational reproducibility. Statistical reproducibility often involves the impact of human decision-making throughout different steps of a data analysis process, while computational reproducibility often involves underlying software utilized or developed by researchers.

We have throughout this dissertation analyzed both aspects of reproducibility in statistics research, and proposed frameworks for studying both. Using a large-scale data analysis project and methodology, the automated comparison of bullet evidence, we studied the impact of a processing method decision in Chapter 2 and the variability of the data collection process in Chapter 3. We also studied the complex and often overlooked network of R package dependencies in statistics projects and the vulnerability of projects to changes in the R package landscape in Chapter 4.

One of the most impactful lessons learned from this work is the sore lack of statistical reproducibility research in the field of Statistics; our approach to Gauge R&R in Chapter 3 for data which is not a single response measurement is new. There is a larger library of prior work in computational reproducibility, where the `knitr` package provides reproducibility in the form

of including source code with generated text documents, and the `packrat` and `checkpoint` packages, among others, provide approaches to static reproducibility. Our framework for adaptive computational reproducibility in Chapter 4 is a new approach, and one we believe provides many advantages; however, it is not the only approach.

Our results and case studies in Chapters 2, 3, and 4 show that data analysis processes are sensitive to human decision-making in many ways, including use of R packages and use of data processing method. They also demonstrate that a well-designed data analysis process, such as the bullet analysis pipeline, produces results which are relatively robust to varying input.

There is much to be done in the field of reproducibility as it applies to data analysis processes. Our study of one large-scale pipeline merely scratches the surface of possible research in this field. Future work will include improvements to the tools in the `manager` package to assist in managing computational reproducibility, as well as applying the same statistical reproducibility ideas utilized in Chapters 2 and 3 to other aspects of the bullet pipeline and other data analysis processes.

REFERENCES

AFTE Glossary (1998). Theory of identification as it relates to toolmarks. *AFTE Journal*, 30(1):86–88.

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., and Iannone, R. (2020). *rmarkdown: Dynamic Documents for R*. R package version 2.1.

Bachrach, B. (2002). Development of a 3d-based automated firearms evidence comparison system. *Journal of Forensic Sciences*, 47(6):1253–1264.

Bachrach, B. (2006). A statistical validation of the individuality or guns using 3d images of bullets. NIJ Report.

Baggerly, K. A. and Coombes, K. R. (2009). Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology. *The Annals of Applied Statistics*, 3(4):1309–1334.

Baggerly, K. A., Morris, J. S., and Coombes, K. R. (2004). Reproducibility of SELDI-TOF protein patterns in serum: comparing datasets from different experiments. *Bioinformatics*, 20(5):777–785.

Baldwin, D., Bajic, S., Morris, M., and Zamzow, D. (2014). A study of false-positive and false-negative error rates in cartridge case comparisons. USDOE Technical Report.

Bates, D., Mächler, M., Bolker, B., and Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48.

Biasotti, A. A. (1959). A statistical study of the individual characteristics of fired bullets. *Journal of Forensic Sciences*, 4(1):34–50.

Buckheit, J. B. and Donoho, D. L. (1995). *WaveLab and Reproducible Research*, pages 55–81. Springer New York, New York, NY.

Buja, A., Asimov, D., Hurley, C., and McDonald, J. A. (1988). Elements of a viewing pipeline for data analysis. In *Dynamic Graphics for Statistics*. Wadsworth, Inc.

Burns, B., Lamb, J., and Qi, J. (2019). *pkgnet: Get Network Representation of an R Package*. R package version 0.4.0.

Chu, W., Song, T., Vorburger, J., Yen, J., Ballou, S., and Bacharach, B. (2010). Pilot study of automated bullet signature identification based on topography measurements and correlations. *Journal of Forensic Sciences*, 55(2):341–47.

Chu, W., Thompson, R. M., Song, J., and Vorburger, T. V. (2013). Automatic identification of bullet signatures based on consecutive matching striae (cms) criteria. *Forensic Science International*, 231(1-3):137–41.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836.

De Kinder, J. and Bonifanti, M. (1999). Automated comparison of bullet striations based on 3d topography. *Forensic Science International*, 101:85–93.

De Kinder, J., Prevot, P., Pirlot, M., and Nys, B. (1998). Surface topology of bullet striations: an innovating technique. *AFTE Journal*, 30(2):294–299.

Eddelbuettel, D., Lucas, A., Tuszynski, J., Bengtsson, H., Urbanek, S., Frasca, M., Lewis, B., Stokely, M., Muehleisen, H., Murdoch, D., Hester, J., Wu, W., Kou, Q., Onkelinx, T., Lang, M., Simko, V., Hornik, K., Neal, R., Bell, K., de Queljoe, M., Suruceanu, I., and Denney, B. (2020). *digest: Create Compact Hash Digests of R Objects*. R package version 0.6.25.

Fadul, T., Hernandez, G., Stoiloff, S., and Gulati, S. (2013). An empirical study to improve the scientific foundation of forensic firearm and tool mark identification utilizing consecutively manufactured glock ebis barrels with the same ebis pattern. NIJ Report.

Friedman, J., Hastie, T., Tibshirani, R., Simon, N., Narasimhan, B., and Qian, J. (2018). glmnet: Lasso and elastic-net regularized generalized linear models.

Garton, N. (2020). *bulletcp: Automatic Groove Identification via Bayesian Changepoint Detection*. R package version 1.0.0.

Gentleman, R. and Lang, D. T. (2007). Statistical analyses and reproducible research. *Journal of Computational and Graphical Statistics*, 16(1):1–23.

Hamby, J. E., Brundage, D. J., and Thorpe, J. W. (2009). The identification of bullets fired from 10 consecutively rifled 9mm ruger pistol barrels: A research project involving 507 participants from 20 countries. *AFTE Journal*, 41(2):99–110.

Hare, E., Hofmann, H., and Carriquiry, A. (2017). Automatic matching of bullet land impressions. *The Annals of Applied Statistics*, 11:2332–2356.

Hofmann, H., Vanderplas, S., and Krishnan, G. (2019a). *bulletxtrctr: Automatic Matching of Bullet Striae*. R package version 0.2.0.

Hofmann, H., Vanderplas, S., Krishnan, G., and Hare, E. (2019b). *x3ptools: Tools for Working with 3D Surface Measurements*. R package version 0.0.2.9000.

Hofmann, H., Vanderplas, S., Rice, K., Garton, N., and Roiger, C. (2019c). *grooveFinder: Identify Locations of Grooves in Land Engraved Areas of Bullet Scans*. R package version 0.0.1.

ISO 5436-2:2012(en) (2012). Geometrical product specifications (GPS) – Surface texture: Profile method; Measurement standards – Part 2: Software measurement standards. Standard, International Organization for Standardization, Geneva, CH.

Knuth, D. E. (1984). Literate Programming. *The Computer Journal*, 27(2):97–111.

Leisch, F. (2003). Sweave and beyond: Computations on text documents. In Hornik, K., Leisch, F., and Zeileis, A., editors, *Proceedings of the 3rd International Workshop on Distributed Statistical Computing, March 20-22, 2003, Technische Universität Wien, Vienna, Austria*. ISSN 1609-395X.

Ma, L., Song, J., Whitenton, E., Zheng, A., Vorburger, T., and Zhou, J. (2004). Nist bullet signature measurement system for rm (reference material) 8240 standard bullets. *Journal of Forensic Sciences*, 49(4):649–59.

Madelin, G., Babb, J. S., Xia, D., Chang, G., Jerschow, A., and Regatte, R. R. (2012). Reproducibility and repeatability of quantitative sodium magnetic resonance imaging in vivo in articular cartilage at 3 t and 7 t. *Magnetic Resonance in Medicine*, 68(3):841–849.

Marwick, B. (2016). Computational Reproducibility in Archaeological Research: Basic Principles and a Case Study of Their Implementation. *Journal of Archaeological Method and Theory*, pages 1–28.

McClarin, D. S. (2015). Adding an objective component to routine casework: Use of confocal microscopy for the analysis of 9mm caliber bullets. *AFTE Journal*, 47(3):161–170.

National Research Council (2009). *Strengthening Forensic Science in the United States: A Path Forward*. The National Academies Press, Washington, D.C.

Ooi, H., de Vries, A., and Microsoft (2020). *checkpoint: Install Packages from Snapshots on the Checkpoint Server for Reproducibility*. R package version 0.4.8.

Petraco, N. and Chan, H. (2012). Application of machine learning to toolmarks: Statistically based methods for impression pattern comparisons.

President's Council of Advisors on Science and Technology (2016). Forensic science in criminal courts: Ensuring scientific validity of feature-comparison methods.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Riva, F. and Champod, C. (2014). Automatic comparison and evaluation of impressions left by a firearm on fired cartridge cases. *Journal of Forensic Sciences*, 59(3):637–47.

Rossini, A. (2001). Literate statistical analysis. In Hornik, K. and Leisch, F., editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing, March 15-17, 2001, Technische Universität Wien, Vienna, Austria*. ISSN 1609-395X.

Sandve, G. K., Nekrutenko, A., Taylor, J., and Hovig, E. (2013). Ten simple rules for reproducible computational research. *PLOS Computational Biology*, 9(10):1–4.

Tai, X. H. and Eddy, W. F. (2018). A fully automatic method for comparing cartridge case images,. *Journal of Forensic Sciences*, 63(2):440–448.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Ushey, K., McPherson, J., Cheng, J., Atkins, A., and Allaire, J. (2018). *packrat: A Dependency Management System for Projects and their R Package Dependencies*. R package version 0.5.0.

Vardeman, S. B. and VanValkenburg, E. S. (1999). Two-way random-effects analyses and gauge r&r studies. *Technometrics*, 41(3):202–211.

Wickham, H. (2017). *tidyverse: Easily Install and Load the 'Tidyverse'*. R package version 1.2.1.

Wickham, H. and Henry, L. (2019). *tidyr: Tidy Messy Data*. R package version 1.0.0.

Xie, Y. (2020). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.28.

Xie, Y., Allaire, J., and Grolemund, G. (2018). *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida. ISBN 9781138359338.