An Open-Source Implementation of the CMPS Algorithm for Assessing Similarity of Bullets

by Wangqian W. Ju and Heike Hofmann

Abstract In this paper, we introduce the R package **cmpsR**, an open-source implementation of the Congruent Matching Profile Segments (CMPS) method developed at the National Institute of Standards and Technology (NIST) for objective comparison of striated tool marks. The functionality of the package is showcased by examples of bullet signatures that come with the package. Graphing tools are implemented in the package as well for users to assess and understand the CMPS results. Initial tests were performed on bullet signatures generated from two sets of 3D scans in the Hamby study under the framework suggested by the R package **bulletxtrctr**. New metrics based on CMPS scores are introduced and compared with existing metrics. A measure called Sum of squares ratio is included, and how it can be used for evaluating different scans, metrics, or parameters is showcased with the Hamby study data sets. An open-source implementation of the CMPS algorithm makes the algorithm more accessible, generates reproducible results, and facilitates further studies of the algorithm such as methods comparison.

Introduction

In this paper we present an open-source implementation of the algorithm of the Congruent Matching Profile Segments (CMPS) method. Chen et al. (2019) developed the CMPS method for "objective comparison of striated tool marks" and demonstrated its use in some examples of comparing bullet signature correlations. Although Chen et al. (2019) conceptually describe the CMPS algorithm in their paper, the authors did not release an implementation of their method. Thus, our effort here is to introduce the CMPS method to the R (R Core Team, 2012) community and provide an open-source, publicly available implementation of the algorithm to use, review, and improve. Our implementation is made available as part of the R package **cmpsR** on CRAN.

According to the Uniform Crime Reporting Program of FBI (Federal Bureau of Investigation), "more than 76 percent (76.7) of the homicides for which the FBI received weapons data in 2020 involved the use of firearms" (United States Department of Justice, Federal Bureau of Investigation., a). At the same time, the number of murder victims jumped between 2019 and 2020 by more than 23 percent (23.4) to 17754 (United States Department of Justice, Federal Bureau of Investigation., b). This increase is unprecedented and highlights the important role that firearm examination plays in all these cases. An important task of firearm examination is to answer the question of whether two pieces of evidence come from the same source or whether a piece of evidence matches a sample obtained from a specific firearm. Here, in particular, we are interested to determine whether two bullets were fired from the same gun barrel. Assessing the similarity between two bullets is based on a comparison of striation marks acquired during the firing process as bullets are propelled through the barrel. Current state of art sees firearms examiners make an assessment of similarity based on a visual comparison, generally, using a comparison microscope (AFTE Criteria for Identification Committee, 1992). This practice has been criticized for its lack of objectivity and the associated problem of determining valid error rates (President's Council of Advisors on Science and Technology, 2016).

A report published by the Committee on Identifying the Needs of the Forensic Sciences of the National Research Council (2009) states that "[m]uch forensic evidence-including, for example, bite marks and firearm and toolmark identification—is introduced in criminal trials without any meaningful scientific validation, determination of error rates, or reliability testing to explain the limits of the discipline." To overcome those criticisms and concerns, researchers have been making an effort to build databases and develop frameworks and algorithms that bring an objective and quantitative assessment into the field. The database of Zheng (2016) with digital 3D topographic scans from various studies by Brundage (1998), Hamby et al. (2009), Hamby et al. (2019) and others, provides a great resource for researchers. Algorithms that can be validated and tested and generate quantitative results are developed for 2D and 3D surface texture (Song et al., 2005), tool marks (Chumbley et al., 2010), and striae on Land Engraved Areas (LEAs) of bullets (Hare et al., 2017, Krishnan and Hofmann (2019), Chen et al. (2019)). The notion of bullet signatures is one of the results of these efforts, and we will use it to explain the CMPS algorithm and showcase the **cmpsR** implementation. Bullet signatures can be extracted from bullet land engraved areas (LEAs), typically one for each land engraved area, and how those bullet signatures are extracted from scans will be discussed in the Background section. Bullet signatures capture striation marks on the bullet in a numeric format and therefore serve as the foundation for

algorithms. These scans played an important role in bringing objectivity into the field. In the following sections we will: review the background of bullet signature comparisons, discuss how we followed the idea described by Chen et al. (2019) for the implementation of the CMPS algorithm, propose new metrics based on the CMPS score and a principled evaluation framework for algorithmic results comparison, and present results of applying our implementation to real data.

Background

Hamby data set

The datasets we worked with come from the James Hamby Consecutively Rifled Ruger Barrel Study (Brundage, 1998; Hamby et al., 2009, 2019), in particular, Hamby set 252 and Hamby set 44. For each Hamby set, a total of 35 bullets is fired from (the same) ten consecutively manufactured Ruger P-85 pistol barrels. Two bullets are fired from each barrel, making up a set of 20 reference bullets. An additional 15 bullets are fired from these ten barrels in a fashion unknown to the study participant. The aim of the Hamby Study was to have firearms examiners identify which barrel each of the 15 questioned bullets was fired from. The Ruger P-85 barrels are traditionally rifled barrels with six grooves and lands as shown in Figure 1. During the firing process, grooves and lands are engraved on a bullet. Firearms examiners use striation marks on land engraved areas (LEAs) for their visual comparison. For algorithmic purposes, 3D topographical images of land engraved areas were obtained and stored in x3p format (XML 3-D Surface Profile). The x3p format provides a standard way of exchanging 2D and 3D profile data. It conforms to the ISO5436-2 standard adopted by the OpenFMC (Open Forensic Metrology Consortium), a group of firearm forensics researchers who contributes to the establishment of best practices of using metrology in forensic science. Hamby set 252 was scanned using a NanoFocus lens at 20x magnification with the scan resolution being 1.5625 μm × 1.5625 μm per pixel. Hamby set 44 was scanned at the Roy J Carver High-Resolution microscopy lab at Iowa State. These scans were acquired with a Sensofar Confocal Light Microscope at 20x magnification for a nominal resolution of 0.645 μm × 0.645 μm per pixel. Both Hamby set 252 and Hamby set 44 are publicly available from the NIST Ballistics Database Project (Zheng, 2016).

Extracting signatures from LEA scans

The automated framework for extracting signatures from x3p files used in this paper was proposed by Hare et al. (2017) and is implemented in the R packages **x3ptools** (Hofmann et al., 2020) and **bulletxtrctr** (Hofmann et al., 2019). **x3ptools** is a package to read, write, and generally, process x3p files. The **bulletxtrctr** package implements a pipeline for extracting and comparing signatures from scans of land-engraved areas.

Figure 2 gives an overview of all of the steps in the process from scan to signatures. Figure 2(a) shows a rendering of a 3D scan of a bullet land engraved area. The raised portion of the surface on the left and right of the scan are parts of the adjacent groove engraved areas (GEAs), the middle area shows a land-engraved area (LEA) with well expressed striation marks. The first step of obtaining the bullet signature is to extract a cross-section at a fixed height along the land engraving.

The thin white horizontal line in Figure 2(a) indicates which cross-section was identified by the algorithm to represent the LEA; Figure 2(b) shows the corresponding cross-sectional view. Groove engraved areas are removed from the analysis as indicated by the vertical blue lines in Figure 2(c) and Figure 2(d)). A non-parametric LOESS smooth (Cleveland et al., 1991) is fitted to capture the bullet curvature (Figure 2(e)) and, finally, the **bullet signature** (Figure 2(f)) is obtained as residuals of the cross-section and the fitted smooth. Note, that in Chen et al. (2019) bullet signatures are referred to as bullet profiles. However, to avoid confusion, we distinguish the notion of bullet signatures from bullet profiles. Bullet profiles are shown in panels (b), (c), and (d) of Figure 2, while Figure 2(f) shows the corresponding bullet signature. Identifying the groove engraved areas correctly is fundamental for a correct down-stream analysis of the signatures. To allow for a human-in-the middle inspection and intervention we have provided an interactive web-application, implemented as R Shiny App (Chang et al., 2021), **bulletinspectR** to identify and correct those errors. An example of the extraction process with corresponding code and parameter settings can be found in the "Supplementary materials". Note that the process of extracting signatures might be different from the one used in Chen et al. (2019) because no code or parameter settings are made available publicly.

Conceptual idea of CMPS

Most algorithms for comparing striation marks are based on the digitized signatures and produce a similarity score (Song et al., 2005, Chumbley et al. (2010), Hare et al. (2017), Krishnan and Hofmann (2019)). The congruent matching profile segments (CMPS) algorithm, developed by Chen et al. (2019) for "objective comparison of striated tool marks", is one such algorithm. The algorithm's main idea is to take a set of consecutive and non-overlapping basis segments from the comparison and for each segment find the "best" registration position on the reference (the other bullet signature) with respect to their cross-correlation values. From a comparison of these registration positions, a **congruent**
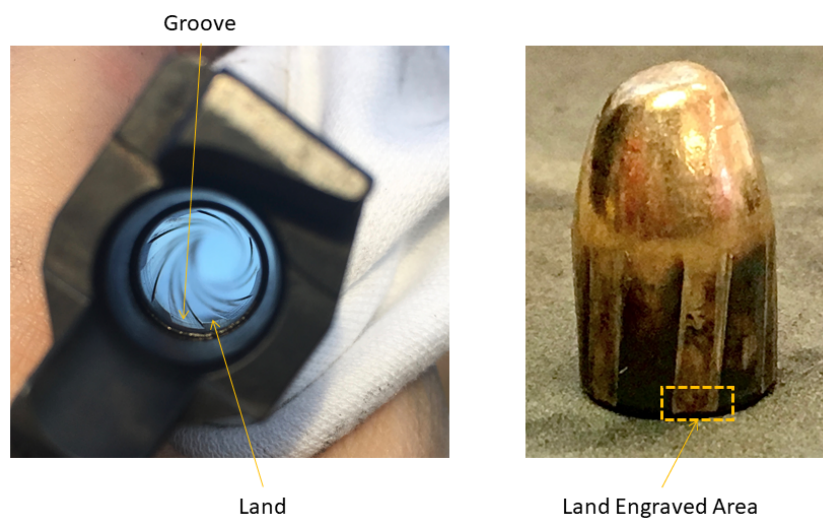
Figure 1: Photo of a traditionally rifled gun barrel (left) and a fired bullet (right).
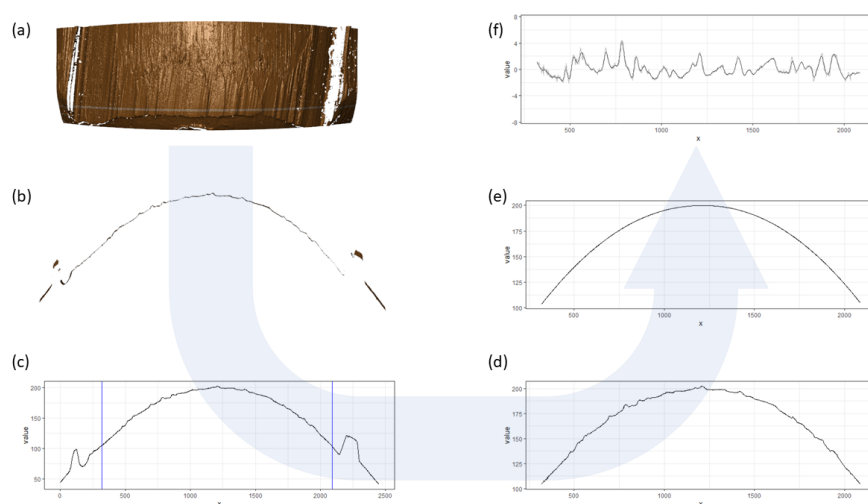

Figure 2: A framework of obtaining a bullet signature. (a) rendering from the 3D topographic scan of a land engraved area (LEA). The selected crosscut location is indicated by a thin white horizontal line. (b) view of the cross-section of the land engraved area at the white line in (a). (c) the crosscut data plotted in 2D; blue vertical lines indicate the position of left and right grooves. (d) the crosscut data after chopping the left and right grooves. (e) the fitted curvature using LOESS. (f) after removing the curvature from the crosscut data, the bullet signature is obtained

registration position is identified, and the number of basis segments taking the congruent registration position is the CMPS score. Note that researchers in Chen et al. (2019) refer to the origin of basis segments as the reference, but in this paper we refer to it as the comparison signature. High CMPS scores are achieved between more similar signatures and are therefore indicative of a same-source pair. Low scores between pairs of signatures are attributed to different source pairs. However, a specific threshold of the CMPS score to distinguish between same-source and different-source comparisons is not provided in Chen et al. (2019), instead the threshold depends on the underlying structure of the data and the choice of parameters. In a legal setting this variability is problematic because it allows for situations in which experts could choose parameters based on whether they are witnesses for the defense or the prosecution. Further research is needed to understand how to determine optimal threshold settings. The CMPS algorithm can assist firearm examiners with drawing a conclusion about the source of a comparison pair. Thus, in this paper we present an open-source implementation of the CMPS algorithm in the R package **cmpsR** available from both CRAN and Github. This publicly available implementation calculates the CMPS score of a comparison using the following code:

```
# install.packages("cmpsR")

library(cmpsR)
data(bullets)

sig1 <- bullets$sigs[[2]]$sig
sig2 <- bullets$sigs[[9]]$sig
sig3 <- bullets$sigs[[10]]$sig

cmps_result_KM <- extract_feature_cmps(sig1, sig2)
cmps_result_KNM <- extract_feature_cmps(sig1, sig3)
```

In this example, the comparison between `sig1` and `sig2`, two signatures coming from the same source (a known-match comparison), gets a CMPS score of 17; the comparison between `sig1` and `sig3`, two signatures coming from different sources (a known non-match comparison), gets a CMPS score of 1.

We also implemented graphing tools for users to better understand these results as well as the algorithm itself.

The section "Implementation" will go through the algorithm and show how to use the **cmpsR** package. A further example that illustrates the main points is also included. In the section on 'Evaluation metrics" we introduce new CMPS metrics that summarize land-level CMPS scores and a sum of squares ratio that can be used to evaluate algorithmic results. The section "Results" presents the results of evaluating the **cmpsR** package using Hamby set 252 and Hamby set 44. The results from Hamby set 252 are used to verify that our implementation is, at least qualitatively, comparable to the algorithm described in Chen et al. (2019). Results from Hamby 44 show the need for a further investigation of the parameter choices even in the case of bullets fired from the same barrels. The last section covers some final discussion and conclusions.

Implementation

Algorithm

Conceptually, the CMPS algorithm consists of three main steps:

1. **cut the comparison signature into consecutive, non-overlapping and equal-length basis segments:** The command `get_segs(x,len=50)` implements this step: it takes bullet signature `x` in the format of a numeric vector and cuts it into consecutive, non-overlapping and equal-length segments of length `len`, which are referred to as "basis segments".

2. **identify candidate positions:** For each basis segment a set of candidate registration positions on the comparison signature is identified based on the segment's similarity to the reference signature. In a first step, the cross-correlation function of the segment to the reference is calculated, then a number of positions with high correlation values are identified as candidate positions. In case multiple segment lengths are considered, the length of each basis segment is expanded (by default it is doubled) and these two steps are repeated. Only when candidate positions coincide (or are similar enough), they are considered further. Figure 5 and Figure 6 illustrate these ideas.

   - **Calculate the cross-correlation curve:** Calculate the cross-correlation curve between a basis segment x and the reference signature y using the function `get_ccf4(x,y,...)` as shown in Figure 5b. The position indicates the lag by which a basis segment is moved

with respect to its original placement. A position is considered "good" if it results in a peak in the cross-correlation between the basis segment and the reference.

- **Correlation peaks:** Two strategies referred to as "multi-peak inspection" and "multi-peak inspection at different segment lengths" in Chen et al. (2019) are used for identifying positions of correlation peaks as candidate positions. The latter is also called "multi-segment lengths strategy". The parameter `npeaks_set` in `extract_feature_cmps(...)` determines which strategy to use and the number of candidate positions:

    - If `npeaks_set` is an integer vector of length 1, for example `npeaks_set = 5`, the positions of the top five peaks in the cross-correlation curve are identified as candidate positions for registration.
    - If `npeaks_set` is an integer vector of length more than 1, for example `npeaks_set = c(5,3,1)`, the multi-segment lengths strategy will be used: calculate the cross-correlation function between a basis segment and the reference and identify positions of the top five peaks; double the segment length to a specified value, re-calculate the cross-correlation function, and identify three peaks; repeat this process and identify a single peak in the newly computed cross-correlation function. Figure 6 shows an example of three levels of basis segment 6 and their corresponding cross-correlation curves and identified peaks. Note that in Chen et al. (2019) the segment length is doubled at each level of a basis segment, but in the present implementation users are allowed to choose the segment length at each level.
    - `get_ccr_peaks(comp,segments,seg_outlength,nseg = 1,npeaks = 5)` computes the cross-correlation curve between a basis or increased segment and the reference signature and finds peaks in the cross-correlation curve. The number of peaks detected is equal to `npeaks`, which is an integer. `segments`, `seg_outlength`, and `nseg` determines the segment in the cross-correlation computation, and `comp` gives the reference signature. If the multi-segment lengths strategy is used, then `get_ccr_peaks(...)` is called in a `lapply()` for each level of the basis segment. The resulting list is called `ccr_list`.

- **multi-segment lengths strategy:** with the multi-segment lengths strategy being used, a position is identified as a candidate position for registration and is called as a "consistent correlation peak" if it results in a top peak in the cross-correlation curve with a tolerance zone determined by `Tx` in all segment levels. Note that in Chen et al. (2019), a segment at its largest scale (highest level) always identifies one peak, but we do not have this requirement in our implementation.

    - the function `get_seg_scale(segments,nseg,out_length)` is used to obtain the (potentially increased) version of a basis segment. `segments`, which is a list containing all basis segments generated by the function `get_segs(...)` in step 1, and `nseg` are used to determine the basis segment to be increased. `out_length` specifies the length of the output segment.
    - `get_ccp(ccr_list,Tx = 25)` tries to identify the "consistent correlation peak". `ccr_list` is the result of `lapply()` and `get_ccr_peaks(...)`, and `Tx` determines the size of a tolerance zone used in identifying the consistent correlation peak. `get_ccp(...)` returns `NULL` if there is no consistent correlation peak.

3. **determine the congruent registration position:** A candidate position "receives" votes from basis segments that identify it or a close position within a tolerance zone of `Tx` as a candidate position in step 2. Votes for all candidate positions are tallied, and the position with the highest number of votes gets chosen as the *congruent registration position*, indicating that most of the basis segments find their highly similar counterpart in the reference signature in terms of correlation at this registration position. In case of ties the middle position is taken as the congruent registration position. Basis segments with a congruent registration position are called "congruent matching profile segments" (CMPS). The total number of CMPS is the CMPS score of the comparison. `get_CMPS(input_ccp,Tx = 25)` is the function that tallies the votes and determines the congruent registration position and congruent matching profile segments (CMPS).

Note that there are several parameters in the CMPS algorithm that are left to the users to decide, such as the length of basis segments `seg_length` in step 1, the number of peaks `npeaks_set` identified on each level in step 2, and the length of the tolerance zone `Tx` in both step 2 and 3. In our implementation of the CMPS algorithm we used the parameters given in the original CMPS paper (Chen et al., 2019) as the default values for these parameters. However, the authors state that no cross-validation has been done - there might be also issues with respect to resolution of the scans. Further research is needed, until then users are advised to think of default values as starting values and consider

alternatives. The main function that combines all steps in the CMPS algorithm described above is called `extract_feature_cmps(...)`. Here we present it with its default parameters.

```
extract_feature_cmps(
  x,
  y,
  seg_length = 50,
  Tx = 25,
  npeaks_set = c(5, 3, 1),
  include = NULL,
  outlength = NULL
)
```

The function `extract_feature_cmps` allows for the following input from users besides the previously discussed parameters `seg_length`, `npeaks_set`, and `Tx`:

- `x` and `y` are two signatures: `x` serves as the comparison signature (which will be divided into basis segments) and `y` is the reference signature;
- `include` determines the format of the function result. Besides the `CMPS_score`, other aspects of the comparison help in understanding how the `CMPS_score` is computed. By default `include` is set to `NULL` and only the `CMPS_score` is returned; further results are included when `include` is (an abbreviation of) one of or a vector of the following strings: "nseg", "congruent_pos", "congruent_seg_idx", "segments", "parameters", and "full_result". If include is specified as "full_result" (or its abbreviation), the output includes everything listed below.

    - `nseg`: the number of basis segments from the comparison signature; this is also the highest possible CMPS score of the comparison;
    - `congruent_pos`: the congruent registration position;
    - `congruent_seg_idx`: the indices of all congruent matching profile segments;
    - `ccp_list`: a list showing identified candidate positions of all basis segments;
    - `pos_df`: a data frame containing all candidate positions and their respective number of votes;
    - `segments`: a list containing all basis segments;
    - `parameters`: a list containing all input arguments of `extract_feature_cmps`;

- `outlength` specifies the segment length of a basis segment at each level under the multi-segment lengths strategy. By default `outlength` is set to `NULL`, indicating that a basis segment should double its segment length at the next level and conforming to the description in Chen et al. (2019) .

In the remainder of the paper we showcase the use of the **cmpsR** functionality on some examples and present the results of applying it to two datasets.

Installation

The **cmpsR** package is publicly available from CRAN and can be installed by

```
install.packages("cmpsR")
```

Moreover, its development version is also available from Github and can be installed by

```
# install.packages("remotes")
remotes::install_github("willju-wangqian/cmpsR")
```

An example

The **cmpsR** package contains a simple example to illustrate the basic usage of the package. The data in this example are twelve bullet signatures obtained from two bullets in Hamby set 252 (Hamby et al., 2009). The procedure for generating signatures from high-resolution 3D topographic scans of bullet lands used here follows the methodology described in Hare et al. (2017) (as discussed above). The two bullets under consideration are known to have been fired from the same gun barrel, so for the 36 pairwise land-by-land comparisons, six comparisons are from same-source pairs (known matches) while thirty are from different-source pairs (known non-matches). To access the example data, we use

```
library(cmpsR)
data(bullets)
```
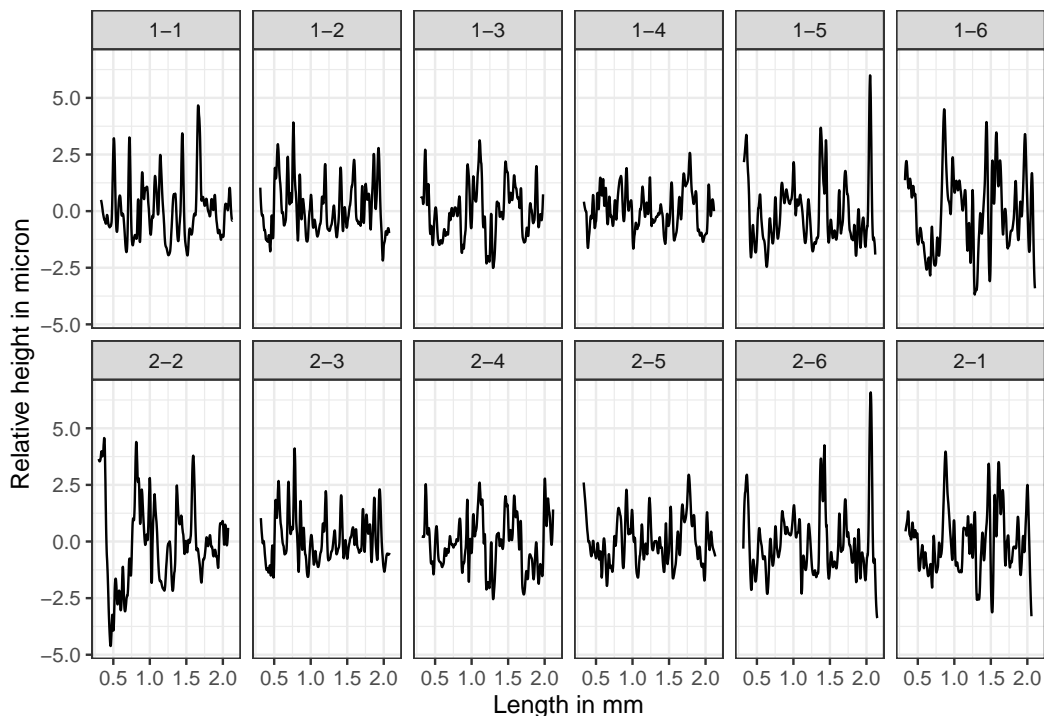
Figure 3: Signatures of all lands of bullet 1 in the top row, and of bullet 2 in the bottom row. Signatures in the second row are ordered to be in phase with the signatures above, i.e. matching signatures are displayed on top of each other.

`bullets$sigs` is a list of twelve numeric vectors corresponding to the twelve bullet signatures shown in Figure 3. `bullets$source` contains the URLs to the corresponding x3p file containing the topographic scan from the NIST Ballistics Toolmark Research Database (Zheng, 2016).

The signatures of Land 4 of Bullet 1 and Land 5 of Bullet 2 are stored in objects `sigs2` and `sigs1`, respectively. This comparison consists of a pair of signatures that are known to be a match – a KM (known match) comparison. We compute the CMPS score using two versions of the CMPS algorithm:

```
sigs1 <- bullets$sigs[bullets$bulletland == "2-5"][[1]]
sigs2 <- bullets$sigs[bullets$bulletland == "1-4"][[1]]

# compute cmps

# algorithm with multi-peak insepction at three different segment levels
cmps_with_multi_scale <-
  extract_feature_cmps(sigs1$sig, sigs2$sig,
    npeaks_set = c(5, 3, 1), include = "full_result"
  )

# algorithm with multi-peak inspection at the basis scale only
cmps_without_multi_scale <-
  extract_feature_cmps(sigs1$sig, sigs2$sig,
    npeaks_set = 5, include = "full_result"
  )
```

In the first example, `npeaks_set` is a vector of three integers, i.e. the algorithm uses the multi-segment lengths strategy to create the result object `cmps_with_multi_scale`. For `cmps_without_multi_scale` each basis segment is linked to the top 5 candidate positions. We use `include = "full_result"` to capture all results. In this example, the CMPS is 9 when using multiple segments, and 12 when using a single segment. As discussed in Chen et al. (2019), using multi-segment lengths strategy can reduce the number of false positives when identifying candidate positions; however, any score based method is walking the line between false positives and false negatives. As the number of false positives is reduced the number of false negatives might rise. More discussion and comparisons between the two versions of the CMPS algorithm will be presented in later sections. Note, that the multi-segment lengths method is slower because the algorithm is run once for each segment length.

Visualize and understand CMPS results

We also implemented graphing tools for visualizing results of the CMPS algorithm. The goal is to provide users with tools to inspect each of the basis segments and to help them have a better understanding of how the algorithm works. Figure 4 shows the plots generated by the first graphing function, `cmps_signature_plot()`, and continues with the example above. `cmps_signature_plot()` takes the output of `extract_feature_cmps(...,include = "full_result")` and returns a list of 5 elements. It creates an overall impression of how the comparison signature aligns with the reference signature at the congruent registration position.

- The first element is a plot called `segment_shift_plot`, shown in Figure 4a. On this plot the reference signature is drawn as a black line, congruent matching profile segments from the comparison signature are overlaid in red at the congruent registration position.

- The second plot is called `signature_shift_plot`, shown in Figure 4b. This visual presents both the comparison signature and the reference signature. The comparison signature is aligned with the reference signature based on the congruent registration position. Congruent matching profile segments are highlighted by solid red lines.

(a) The black line shows the comparison signature; each red line segment shows one congruent matching profile segment.

```
sig_plot <- cmps_signature_plot(
  cmps_with_multi_scale
)
sig_plot$segment_shift_plot
```
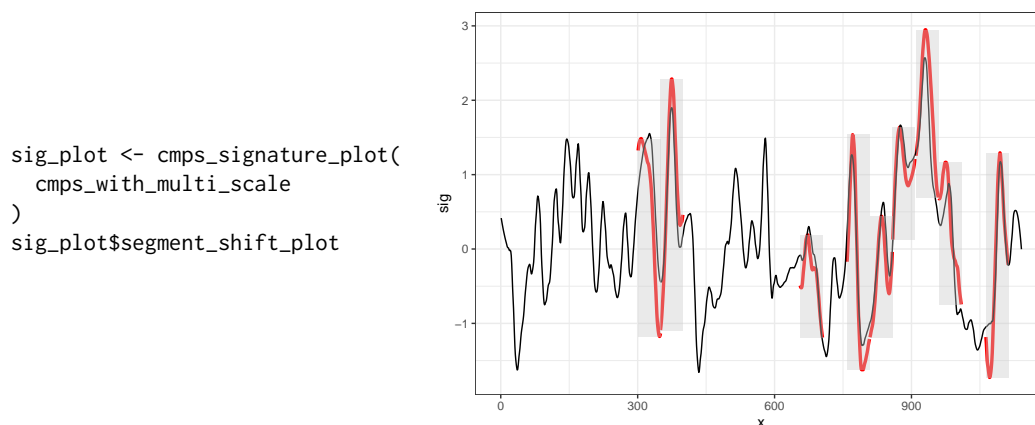

(b) The black line shows the reference signature; the red line shows the comparison signature. Solid part shows the congruent matching profile segments, and the dashed part shows segments that do not agree with the congruent registration position.
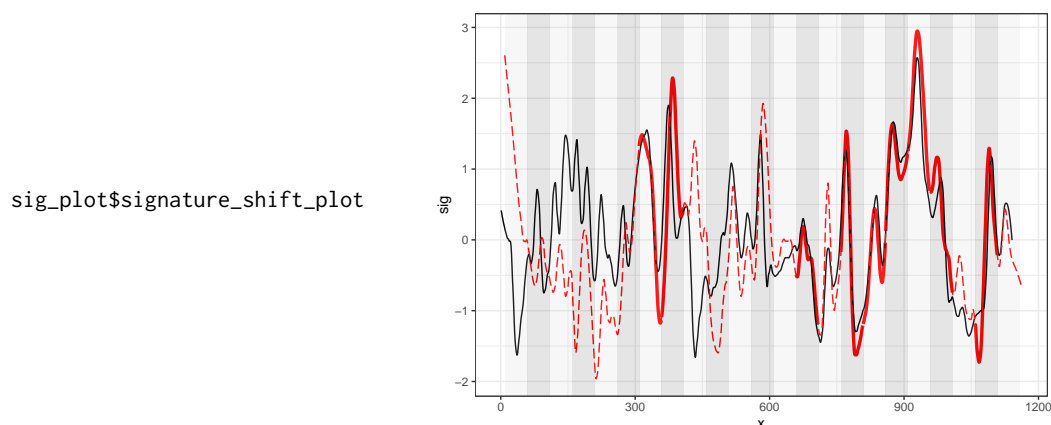
```
sig_plot$signature_shift_plot
```


Figure 4: Two visual outputs of cmps_signature_plot()

- Other elements of this list are `seg_shift` and `sig_shift`. `sig_shift` gives the congruent registration position, while `seg_shift` is a data frame showing the congruent matching profile segments and their identified candidate position closest to the congruent registration position.

```
sig_plot$seg_shift
```

```
#>    seg_idx seg_shift
#> 7       7        0
#> 8       8       -1
#> 14     14        5
#> 16     16        8
#> 17     17        8
#> 18     18        8
#> 19     19        9
#> 20     20        9
#> 22     22       12
```

While `cmps_signature_plot()` focuses on the signature level, `cmps_segment_plot()` focuses on the segment level. It provides the "full result" of `extract_feature_cmps()`, but also takes an argument, `seg_idx`, indicating which segment should be inspected. When checking `sig_plot$seg_shift` we notice that segment number 6 is not one of the congruent matching profile segments. We can therefore set `seg_idx = 6` in `cmps_segment_plot()` and investigate the reason why this segment disagrees with the congruent registration position.

For each segment scale, we have two plots: `segment_plot` and `scale_ccf_plot`, as shown in Figure 5 for the example of segment number 6:

- Figure 5a is the `segment_plot` for basis segment 6 at level one (in its original length). We used `npeaks_set = c(5,3,1)` in `extract_feature_cmps()` when calculating the CMPS score. Therefore the top five peaks are identified in the cross-correlation curve at level one. Segment 6 is plotted at the positions where these 5 peaks are identified with dashed lines in the `segment_plot`. The solid thick black line shows the segment at its original position (which in this example is very close to the actual registration position).
- Figure 5b is the `scale_ccf_plot` of basis segment 6 at level one. It shows the cross-correlation curve computed by the reference signature and the level-one basis segment 6. The five highest peaks are marked by dots on the curve.

Additionally, users can have more insights about why segment 6 is not a congruent matching profile segment if we put the `segment_plot` and `scale_ccf_plot` of all three segment levels together, as shown in Figure 6 with the help of `ggpubr::ggarrange()`.

In Figure 6, the red vertical dashed line indicates the congruent registration position. We can see that the basis segment 6 does obtain a peak near the congruent registration position at level two and level three, respectively; however, this position doesn't give one of the five highest peaks at level one. As a result, segment 6 fails to identify the consistent correlation peak (ccp) and fails to become one of the congruent matching profile segments according to the multi-segment lengths strategy. The identified top five peaks at level one are also examples of "false positive" peaks. The "true positive" peak (the peak within the tolerance zone of the congruent registration position) is identified at level two and three by increasing the segment length, which justifies the usage of multi-segment lengths strategy.

Evaluation metrics

### Metrics based on CMPS scores

The CMPS algorithm measures the similarity between two signatures resulting in a similarity score of a land-to-level comparison. Bullets fired from traditionally rifled barrels have multiple land and groove engraved areas. Here, we are working with bullets fired from Ruger P85 barrels with six lands and grooves. A comparison of two bullets therefore involves 36 land-to-land comparisons, resulting in 36 CMPS scores (as shown in Figure 7). In order to obtain a single similarity score of a bullet-level comparison, we need to summarize these 36 CMPS scores. Two similarity metrics for bullet-level comparisons have been introduced in the literature (Chen et al., 2019): $\text{CMPS}_{\max}$ and $\overline{\text{CMPS}}_{\max}$. $\text{CMPS}_{\max}$ is the highest CMPS score obtained among all land-level comparisons, while $\overline{\text{CMPS}}_{\max}$ is the highest possible mean CMPS score of land-level comparisons that are in the same phase:

In general, we assume each bullet has n land engravings (in our case $n = 6$). Let c_{ij} denote the CMPS score of a comparison between bullet 1 land i and bullet 2 land j, for $i, j = 1, \ldots, n$. Let $\mathcal{P}_k$ denote bullet land pairs in phase k for $k = 0, \ldots, n - 1$, and

$$\mathcal{P}_k = \{(i, j) : i = 1, \ldots, n; \ j = (i + k) \bmod n\} \tag{1}$$

, where mod denotes the modulo operation. For example, $\mathcal{P}_1 = \{(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 1)\}$ when $n = 6$. Let k^* denote the index of the highest phase.

(a) segment_plot for segment 6 at level one. The original position of segment 6 is indicated by the solid black line. Positions where the segment achieves the 5 highest cross-correlations are indicated by the dashed line segments.

```
seg_plot <- cmps_segment_plot(
  cmps_with_multi_scale,
  seg_idx = 6
)
seg_plot[[1]]$segment_plot
```
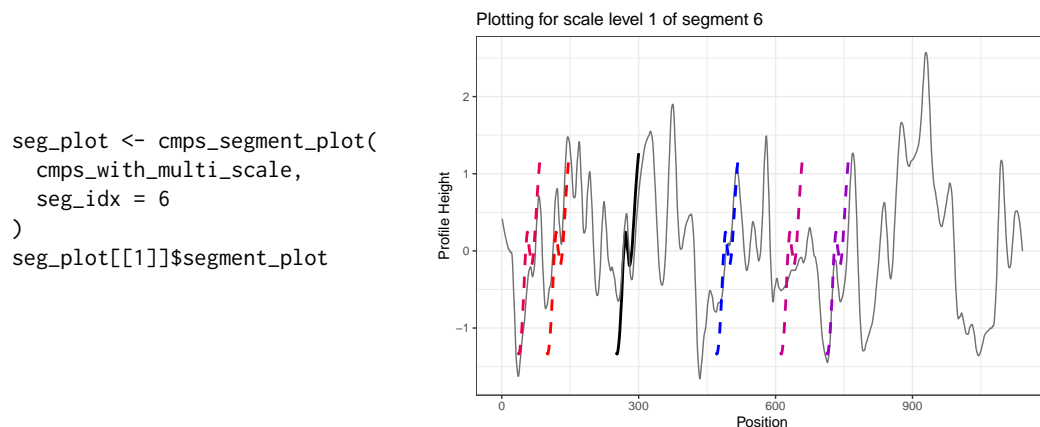

(b) scale_ccf_plot shows the cross-correlation curve between the reference signature and segment 6 at level one. The five highest peaks are marked by dots. The vertical red dashed line indicates the congruent registration position; the green dashed line shows a peak position in the highest segment level; the blue dashed lines show the tolerance zone around the green dashed line. We can see that none of the five highest peaks at level one falls within the tolerance zone, indicating that there is no consistent correlation peak or a candidate position identified by basis segment 6 under the multi-segment lengths strategy. Thus, the basis segment 6 doesn't vote for the congruent registration position and is not a cmps.
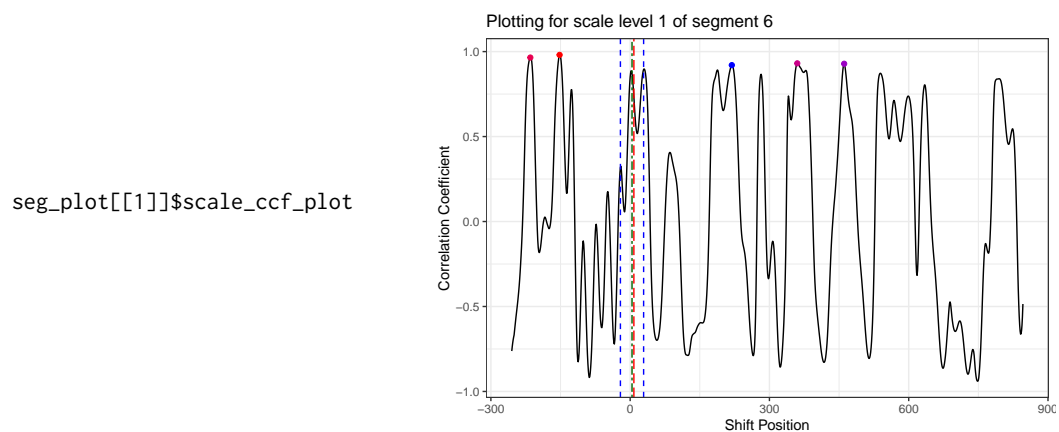
```
seg_plot[[1]]$scale_ccf_plot
```


Figure 5: Two plots used to investigate basis segment 6 at level one

```
library(ggpubr)

ggarrange(
  plotlist =
    unlist(seg_plot,
      recursive = FALSE
    ),
  ncol = 2,
  nrow = 3
)
```
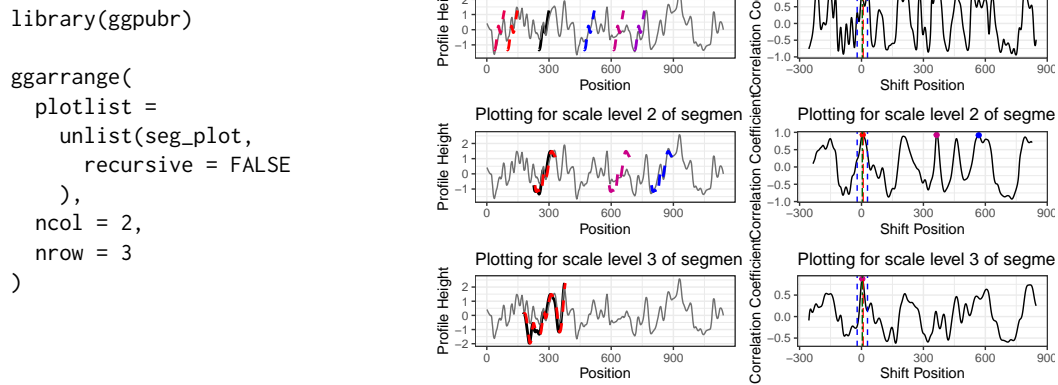
Figure 6: Put segment_plot and scale_ccf_plot of all three levels together. We are identifying five highest peaks at level one, three peaks at level two, and one peak at level three since npeaks_set = c(5, 3, 1). The highest peak position at level three is marked by the green dashed line across all segment levels. However, the highest peak on level three does not coincide with any of the top five highest peaks at level one. This indicates that there is no consistent correlation peak or a candidate position for basis segment 6 under the multi-segment lengths strategy.

With that, the two measures to evaluate accuracy used in Chen et al. (2019) are defined as

$$\mathrm{CMPS}_{\mathrm{max}} = \max_{i,j} c_{ij} \text{ , and} \tag{2}$$

$$\overline{\mathrm{CMPS}}_{\mathrm{max}} = \frac{1}{n} \sum_{(i,j) \in \mathcal{P}_{k^*}} c_{ij} \text{ , where} \tag{3}$$

$$k^* = \arg\max_{k} \left[\frac{1}{n} \sum_{(i,j) \in \mathcal{P}_{k}} c_{ij} \right] \tag{4}$$

We can continue with the example used in previous sections. `bullets` contains bullet signatures of two bullets, `bullet1` and `bullet2`. As mentioned before, each bullet has 6 land engravings, resulting in 6 bullet signatures. Thus, there are 36 pairwise bullet signature comparisons, resulting in 36 c_{ij} values in total. We use multi-segment lengths strategy with default parameters to compute these CMPS scores, and the result is shown in Figure 7. We can see that in this example,

$$\mathrm{CMPS}_{\mathrm{max}} = \max_{i,j} c_{ij} = 17$$

and since bullet lands in phase $\mathcal{P}_1$ gives the highest mean CMPS score ($k^* = 1$), we have

$$\begin{aligned}
\overline{\mathrm{CMPS}}_{\mathrm{max}} &= \frac{1}{6} \sum_{(i,j) \in \mathcal{P}_1} c_{ij} \\
&= \frac{1}{6} (c_{12} + c_{23} + c_{34} + c_{45} + c_{56} + c_{61}) \\
&= \frac{1}{6} (3 + 17 + 14 + 10 + 15 + 16) \\
&= 12.5
\end{aligned}$$

However, both $\mathrm{CMPS}_{\mathrm{max}}$ and $\overline{\mathrm{CMPS}}_{\mathrm{max}}$ consider only relatively high CMPS scores and ignore the rest. So we introduce a new metric based on CMPS scores called $\overline{\mathrm{CMPS}}_{\mathrm{diff}}$, which is the difference between $\overline{\mathrm{CMPS}}_{\mathrm{max}}$ and the mean of all other CMPS scores. With our notation above, we have:

$$\overline{\mathrm{CMPS}}_{\mathrm{diff}} = \left[\frac{1}{n} \sum_{(i,j) \in \mathcal{P}_{k^*}} c_{ij} \right] - \left[\frac{1}{n(n-1)} \sum_{(i,j) \notin \mathcal{P}_{k^*}} c_{ij} \right] \tag{5}$$
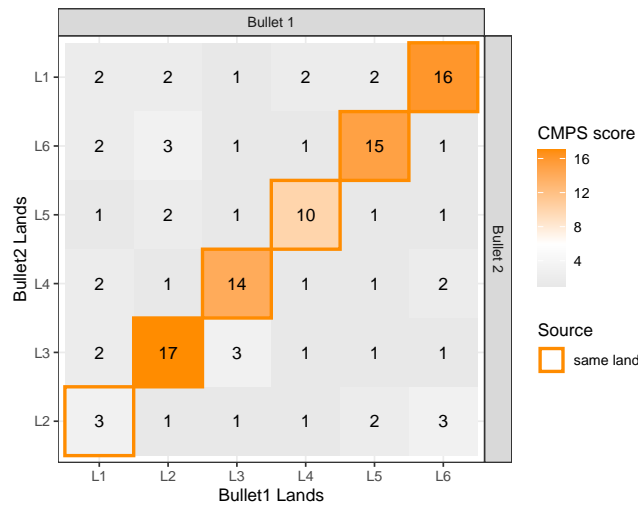
Figure 7: CMPS scores of all 36 pairwise bullet signature comparisons for two bullets. Land engraving pairs generated by the same land (KM comparisons) are highlighted. Note that in this example the axis along Bullet 2 starts with Land 2. This corresponds to Phase 1 in equation (1).

$\overline{\text{CMPS}}_{\text{diff}}$ highlights the difference between CMPS scores of matching and non-matching comparisons. If two bullets are non-matching, all 36 CMPS scores are expected to be small with relatively the same values, resulting in a $\overline{\text{CMPS}}_{\text{diff}}$ value close to 0. For the example above, $\overline{\text{CMPS}}_{\text{diff}} = 12.5 - 1.53 = 10.97$

Scaled CMPS scores

Another issue with the CMPS score is that the highest possible CMPS score (the total number of basis segments) might differ across comparisons (as shown in Figure 8(a)) due to different lengths of bullet signatures and different lengths of basis segments specified by the parameter. A CMPS score of 5 might indicate a non-match if the highest possible CMPS score is 30 but indicate a match if the highest possible CMPS score is 6. Thus, we introduce the scaled CMPS score, denoted as c_{ij}^*. Let s_{ij} denote the highest possible CMPS score or the total number of basis segments, then the scaled CMPS score c_{ij}^* is defined as the ratio of raw score and maximum score:

$$c_{ij}^* = \frac{c_{ij}}{s_{ij}} \tag{6}$$

The scaled CMPS scores of the above example are shown in Figure 8(b). Compared to the original CMPS scores, scaled scores have values within the interval $[0,1]$ regardless of the length of the basis segments and therefore make a comparison of values possible across different parameter choices. Similarly to the original CMPS scores, we will denote the scaled CMPS scores adjusted for out-of-phase background values by $\overline{\text{CMPS}^*}_{\text{diff}}$. For example, $\overline{\text{CMPS}^*}_{\text{diff}} = 0.498$ for Figure 8(b)

Sum of squares ratio

The "sum of squares ratio" quantifies how well two groups of values separate. Let n_T denote the total number of observations, n_k denote the number of observations in group k, and y_{kl} denote the l-th observation in group k, for $k = 1, 2$ and $l = 1, \ldots, n_k$. Let $\bar{y}_{k.} = \frac{1}{n_k} \sum_{l=1}^{n_k} y_{kl}$ denote the mean value in group k and $\bar{y}_{..} = \frac{1}{n_T} \left(\sum_k \sum_{l=1}^{n_k} y_{kl} \right)$ denote the mean value of all observations. Consider the following model:

$$y_{kl} = \mu_k + e_{kl} \tag{7}$$

where μ_k is the true mean of group k and e_{kl} is a random effect of different observations. Then we can define the sum of squares ratio V as:

$$V = \frac{\sum_k n_k (\bar{y}_{k.} - \bar{y}_{..})^2}{\sum_k \sum_l^{n_k} (y_{kl} - \bar{y}_{k.})^2}. \tag{8}$$

The numerator of the sum of squares ratio V quantifies the variation between the two groups, while the denominator quantifies the variation within each group. The sum of squares ratio V can be used
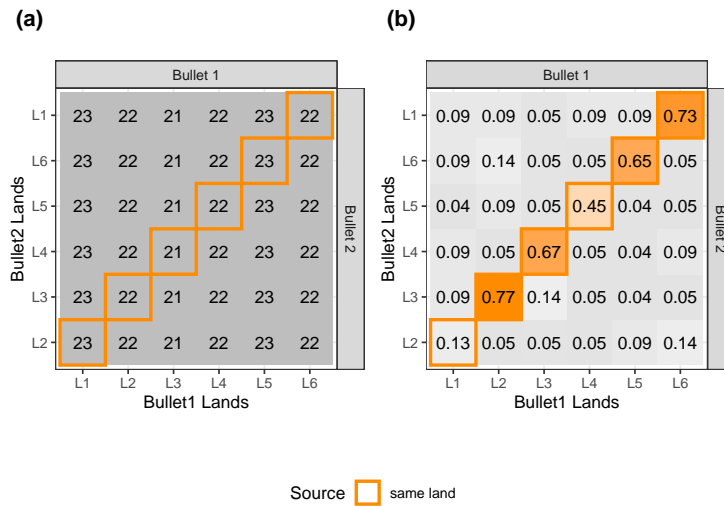
Figure 8: Plot (a) shows the highest possible CMPS scores (the total number of basis segments) for the 36 comparisons. (b) shows the scaled CMPS scores for the 36 comparisons.

as an index for evaluating scans, metrics, and different sets of parameters if the same data set is being used. Some examples will be presented in the following section. If we impose the normality and independence assumptions on the random effects e_{kl}, the sum of squares ratio V becomes a scaled F-statistic with degrees of freedom of $k-1$ and $n_T - k$ and $F = \frac{n_T - k}{k - 1} V$. If we want to compare different data sets, stating that a certain setup can achieve better separation on one data set than another, we can scale the sum of squares ratio V and obtain the F-statistic and obtain the corresponding p-value as an index for comparison.

Using the sum of squares ratio V as an evaluation metric, we are able to construct a pipeline that aims to find the optimal parameter values for the CMPS algorithm by maximizing the sum of squares ratio. Note that other measures of an algorithm, such as the accuracy and the AUC (Area Under the Curve), are also important and useful. But when algorithms achieve 100 percent accuracy and AUC value of 1, we need other measures such as the sum of squares ratio to further distinguish the performance of algorithms. In the following section, we will use the sum of squares ratio to compare the CMPS metrics introduced earlier and investigate effects of different parameter settings.

Results

As presented in the work of Chen et al. (2019), researchers applied the CMPS method to scans of one of the Hamby sets. While it is not explicitly stated in the paper, we presume this to be Hamby 252, as only those scans were publicly available at the time. In order to show that our implementation of the CMPS algorithm is able to reproduce the results in Chen et al. (2019) and be used for other data sets, we applied our implementation to both Hamby set 252 and Hamby set 44. For the purpose of reproducibility, we present how we obtained bullet signatures from the Hamby set data and include code in the appendix: for both Hamby 252 and Hamby 44, we started with scans in the form of x3p files in the database. Following the framework proposed by Hare et al. (2017), we used the same set of parameters, removed damaged bullet scans, obtained bullet signatures for each bullet land engraving, and removed outliers in bullet signatures. Note that researchers of Chen et al. (2019) applied the CMPS algorithm to bullet signatures as well but used a framework different from ours. However, since their work is not open-source, we were not able to follow their framework and were only able to reproduce the results for Hamby set 252 qualitatively.

Hamby 252

Figure 9 shows the distribution of CMPS_{max} and $\overline{\text{CMPS}}_{\text{max}}$ after we applied the CMPS algorithm to Hamby set 252 with the multi-segment lengths strategy. The parameters we used in `extract_feature_cmps` for the CMPS algorithm are:

```
extract_feature_cmps(
  x, y,
  seg_length = 50,
  Tx = 25,
  npeaks_set = c(5, 3, 1),
  include = "nseg"
```
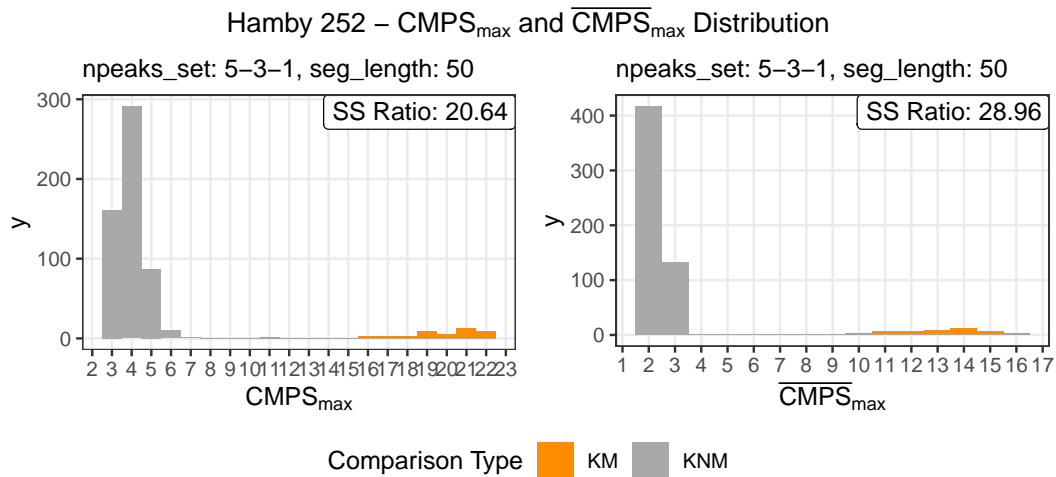
Figure 9: Distribution of $CMPS_{max}$ and $\overline{CMPS}_{max}$ for Hamby 252; outliers are removed in bullet signatures; `seg_length = 50`, `Tx = 25`, `npeaks_set = c(5,3,1)`

```
)
```

As noted above, the CMPS scores we found here are not exactly the same as those presented in Chen et al. (2019) since we were not able to follow their framework, but the results presented in Figure 9 are qualitatively equivalent to those presented in Chen et al. (2019), showing a clear separation between scores based on comparisons from known matches (KM) and scores from comparisons of known non-matches (KNM) for both $CMPS_{max}$ and $\overline{CMPS}_{max}$.

Additionally, to mimic the parameters used in Chen et al. (2019), we set `seg_length = 50` and `Tx = 25` to make sure that each basis segment has length of 78.125 μm and the tolerance zone is ± 39.0625 μm (one unit represents 1.5625 μm for Hamby set 252).

The sum of squares ratios are 20.64 and 28.96 for $CMPS_{max}$ and $\overline{CMPS}_{max}$, respectively. This indicates, that even though scores from $CMPS_{max}$ for known-match comparisons are larger than scores from the averaged version of $\overline{CMPS}_{max}$, these scores achieve a better separation between the two groups of comparisons.

Hamby 44

Similar procedures are applied to Hamby set 44, and Figure 10 shows the distribution of $CMPS_{max}$ and $\overline{CMPS}_{max}$, respectively. The parameters used in `extract_feature_cmps` are:

```
extract_feature_cmps(
  x, y,
  seg_length = 61,
  Tx = 30,
  npeaks_set = c(5, 3, 1),
  include = "nseg"
)
```

Since the resolution of Hamby 44 is set to 1.29 μm per unit, we make `seg_length = 61` and `Tx = 30` to ensure that the setup of Hamby set 44 is similar to that of Hamby set 252, resulting in basis segments of 78.69 μm and the tolerance zone of ± 38.7 μm.

As shown in Figure 10, again, we are able to see a clear separation between the known match comparisons and the known non-match comparisons, even though the separation is relatively small compared with that of Hamby set 252, which is also indicated by the sum of squares ratios. For this specific set of parameters, the sum of squares ratios are 8.87 and 10.64 for $CMPS_{max}$ and $\overline{CMPS}_{max}$, respectively. This might suggest that we could enlarge the separation in terms of the sum of squares ratio by using other CMPS metrics and other sets of parameters.

Comparing CMPS metrics and parameters

We investigated the effects of different sets of parameters at the example of both Hamby sets 252 and 44. More specifically, we investigated the separation achieved using the **cmpsR** implementation under various segment lengths (controlled by the parameter `seg_length`). Specifically, we fixed the parameter `npeaks_set` that controls the number of peaks at each segment level to be `npeaks_set = c(5,3,1)` and modified the parameter `seg_length`. Figure 11 shows that the default values of `seg_length` for Hamby 252 and Hamby 44 (50 for Hamby 252 and 61 for Hamby 44, which represent
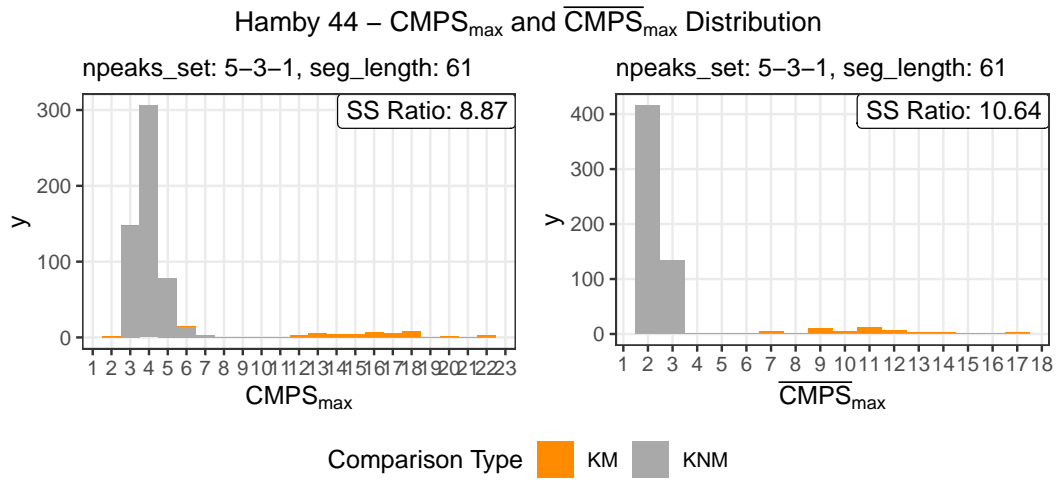
Figure 10: Distribution of $CMPS_{max}$ and $\overline{CMPS}_{max}$ for Hamby 44; outliers are removed in bullet signatures; `seg_length = 61`, `Tx = 30`, `npeaks_set = c(5,3,1)`
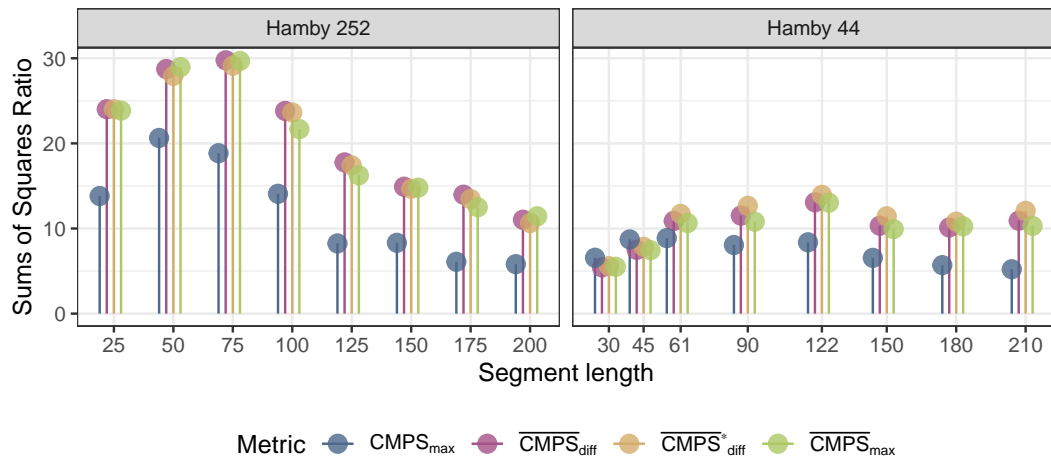

Figure 11: Comparison of results from the CMPS algorithm based on different basis segment lengths (parameter `seg_length`). Only the $CMPS_{max}$ metric suggests that the default values for the basis segment length result in the best separation. Better separation is achieved based on the modified CMPS metrics, inclusing the newly suggested ones. For Hamby 252 these metrics agree on a segment length of 75, and a segment length of 122 for Hamby 44 yields better results
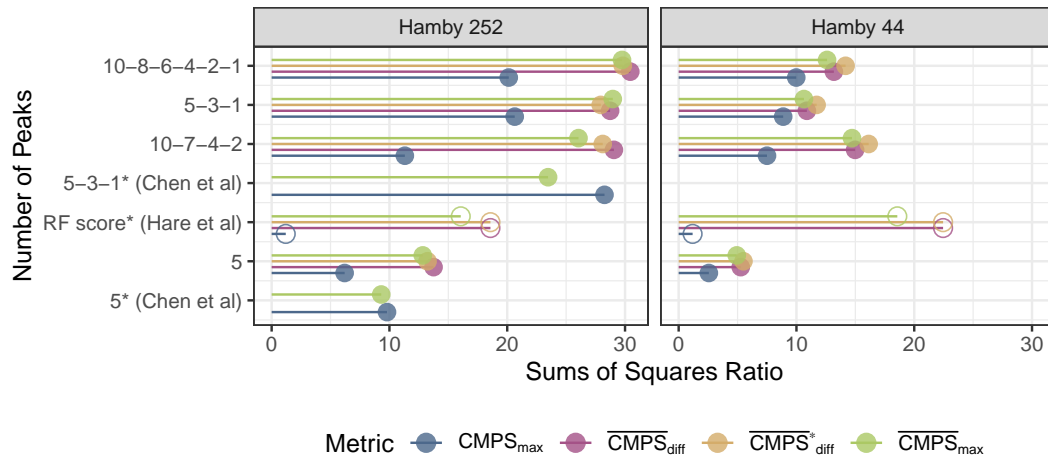
Figure 12: Comparison of CMPS results based on different strategies of number of peak selections. Starred results compare CMPS performance with results published in the literature. Results for the random forest score are represented with circles because the metrics are computed not based on the CMPS scores, but the random forest scores with the same logic. Since random forest scores lie within the interval $[0, 1]$, scaling the random forest scores will not change the results.

78.125 μm and 78.69 μm, respectively) result in high values of the sum of squares ratio, no matter which CMPS metrics are used for an evaluation. However, we also see, that for Hamby set 252 a basis segment length of 75 is a better choice than the default segment length; a basis segment length of 122 results in a higher value of sum of squares ratio for Hamby 44.

Figure 12 shows results of the CMPS algorithm using different strategies for identifying peaks in the correlation structure between signatures. Basis segment length is fixed to the default level for this evaluation. As can be seen in Figure 12, the default value of npeaks_set (npeaks_set = c(5,3,1)) leads to promising results in terms of the sum of squares ratio; however, other choices of npeaks_set match and exceed this sum of squares ratio value.

As seen before, the results suggest that the CMPS_{max} metric produces the least amount of separation compared with the other three CMPS metrics. They also suggest that the two newly proposed metrics, $\overline{\text{CMPS}}_{diff}$ and $\overline{\text{CMPS}}^*_{diff}$, lead to equally good or even better results as $\overline{\text{CMPS}}_{max}$. Because $\overline{\text{CMPS}}^*_{diff}$ relies on a scaled version of CMPS scores it is more comparable to other similarity scores and summarizes the out-of-phase background CMPS scores, making it superior to the other CMPS metrics.

What we can also observe in Figure 11 and Figure 12 is that the values of the sum of squares ratio for Hamby 44 is lower than those for Hamby 252. This might be because determining the source is a harder task for Hamby 44 than for Hamby 252, but also suggests that the choice of parameters also depends on the resolution or the scanning process of the data set. The same set of parameters might work for one data set, but not work equally well for another.

The purpose of the results shown in Figure 11 and Figure 12 is not to determine the "best" parameters for the CMPS algorithm, but to show that the sum of squares ratio can be used as an evaluation measure to compare different parameters, metrics, or scans. A pipeline that maximizes the sum of squares ratio might help researchers determine the set of parameters that work best for their data. But a large database that is representative is what we really need in order to fully understand and cross-validate the parameters of the CMPS algorithm.

Comparing with original results and the random forest model

Chen et al. (2019) present histograms of CMPS_{max} and $\overline{\text{CMPS}}_{max}$ for npeaks_set = 5 and npeaks_set = c(5,3,1) with (presumably) Hamby 252. The values in these histograms allow us to calculate the sum of squares ratios and include the results in Figure 12 as well. They are marked by an asterisk at the top right corner in Figure 12. The sum of squares ratios we obtained for npeaks_set = 5 and npeaks_set = c(5,3,1) is slightly higher than those obtained from the histograms of Chen et al. (2019). It's curious to see that for the Hamby 252 results published in Chen et al. (2019) the CMPS_{max} metric achieves values of the sum of squares ratio higher than those achieved by the $\overline{\text{CMPS}}_{max}$ metric.

Since the researchers of Chen et al. (2019) did not use $\overline{\text{CMPS}}_{diff}$ or $\overline{\text{CMPS}}^*_{diff}$, and they did not apply the CMPS algorithm to the Hamby set 44, we were not able to compare those results.

In Figure 12, we also included the sum of squares ratios computed from the random forest scores (Hare et al., 2017) for different metrics. The random forest model presented in Hare et al. (2017) was

trained at the Center for Statistics and Applications in Forensic Evidence (CSAFE) and is publicly available in the R package **bulletxtrctr** (Hofmann et al., 2019). Similarly to the CMPS algorithm, this random forest model produces a score to quantify the similarity of a land-by-land comparison. The RF scores lie in an interval of [0,1] making them most comparable to the scaled CMPS scores. We applied the logic of CMPS_{max}, $\overline{\text{CMPS}}_{max}$, and $\overline{\text{CMPS}}_{diff}$ to the random forest scores and obtained results of the random forest model for different metrics. As shown in Figure 12, the random forest model does not achieve the sum of squares ratio as high as those achieved by the top CMPS algorithms for Hamby 252, but it does achieve better results for Hamby 44. This might suggest that an inclusion of the CMPS score as an additional feature in the random forest model training might be beneficial for an overall separation to determine source.

For details about the implementation, examples and results, please refer to the "Supplementary materials".

Conclusion

In this paper we present the **cmpsR** package, an open-source implementation of the Congruent Matching Profile Segments (CMPS) algorithm (Chen et al., 2019), and apply it to two datasets in Hamby study (Hamby et al., 2009) to show its potential for further research. The CMPS algorithm was proposed by NIST in 2019 and was made for objective tool marks comparisons. We introduce the basic logic of the CMPS algorithm and lay out its implementation in the **cmpsR** package. We also showcase the functionality of the **cmpsR** package with a small dataset example that is included in the package. In the **cmpsR** package we implement some graphing tools for users to visualize results and to gain a better understanding of both the algorithm and the results.

Additionally, we propose two new metrics based on the CMPS scores and compare the new metrics with the existing metrics. We also introduce a principled evaluation framework of algorithmic results using a measure based on the sum of squares ratio. We showcase the implementation with two datasets in Hamby study (Hamby set 252 and Hamby set 44) and compare the CMPS algorithm using different sets of parameters and different metrics, following the evaluation framework based on the sum of squares ratio. The results obtained are promising: we were both able to reproduce the results in Chen et al. (2019) qualitatively and achieve a clear separation between the known match (KM) comparisons and known non-match (KNM) comparisons in another bullet study. However, the comparisons among different sets of parameters suggest that the optimal choice for parameter settings vary between different datasets.

Note, that the main difference between Hamby sets 44 and 252 is that they were taken with different resolution scanning devices. The bullets for Hamby 44 and Hamby 252 are fired from the same ten consecutively rifled P-85 Ruger barrels. The difference in parameter settings for optimal differentiation between same-source comparisons and different-source comparisons is therefore quite surprising. As next steps we need validation studies, similar to Vanderplas et al. (2020) - trying out the CMPS algorithm on firearms and ammunition combinations that are different to test the limits of the algorithm. The evaluation framework we proposed based on the sum of squares ratio will facilitate such studies and other validation of the algorithm.

Comparisons of the R implementation of the CMPS algorithm with the random forest model proposed by Hare et al. (2017) suggest that adding the CMPS score as an additional feature in the random forest model might add further separation between known match (KM) comparisons and known non-match (KNM) comparisons.

The open-source implementation of the CMPS algorithm provided by the **cmpsR** package is just one step towards in the framework of open science. Open science is particularly important to fields like forensic science where transparency and accuracy are critical to fair justice. Open-source implementations not only allow a peer-review but also facilitate further research, such as parameter cross-validations, method comparisons, and the development of statistical methods for modeling KM and KNM CMPS score distributions, which can then be used for error-rate estimations and fair applications that were called for by the PCAST (President's Council of Advisors on Science and Technology, 2016) report. However, having open-source implementations is not enough for open science. In order to actually build the world of open science, many other efforts, such as open evaluation, open access publication, open educational resources, etc., are still needed. The database built and maintained by the National Institute of Standards and Technology is a good example of open data. What we are aiming for is an open system that is able to collect open results, compare multiple algorithms using multiple datasets, and evaluate algorithmic variation and accuracy. The evaluation metric we proposed in this paper can be used to compare different algorithms or even different datasets and is a snippet of this open system. But the core of this open system is the open science culture and contributions of the community.

Supplementary materials

The zip file "supplementary-files.zip" contains R scripts for reproducing all aspects of this paper.

Data not included in "supplementary-files.zip"

The processed data of Hamby set 252 and Hamby set 44 are stored in two `.rds` files:

- `BulletSignatures252.rds`
- `BulletSignatures44.rds`

And the ground truth of Hamby set 252 is provided in `StudyInfo.xlsx`

Due to the size of the file, these processed data are not included in the "supplementary-files.zip", but the links are provided for download.

In order to reproduce the results in the paper, please download "BulletSignatures252.rds", "BulletSignatures44.rds", and "StudyInfo.xlsx" and save them in a folder named "bullet_signatures_etc". And place the folder "bullet_signatures_etc" into the folder of all the R scripts of "supplementary-files.zip".

Please check out the folder structure of the reproducible folder for reference.

File description

- `hamby*_result_generator.R`: these R scripts take the processed data of Hamby set 252 and Hamby set 44, generate preliminary results used in the paper, and save the results in the `.csv` format in the folder `data-csv`.
- `rds_generator.R`: this R script takes the generated `.csv` files and produce `CMPSpaper_results.rds`. `CMPSpaper_results.rds` is identical to `data/CMPSpaper_results.rds` and is used generate figures and other results presented in the paper.
- `func_collection.R`: this R script contains some helper functions used by other R scripts.

Data included in "supplementary-files.zip"

- `data-csv`: this folder contains `.csv` files we generated using R scripts `hamby*_result_generator.R`
- `CMPSpaper_results.rds`: this is the `.rds` file we generated using R script `rds_generator.R`

These data can be used as reference or example results of the reproducible codes

- `csafe_rf2.rds`: this `.rds` file contains the random forest model used in this paper

Bibliography

AFTE Criteria for Identification Committee. Theory of identification, range striae comparison reports and modified glossary definitions. *AFTE Journal*, 24(3):336–340, 1992. [p1]

D. Brundage. The Identification of Consecutively Rifled Gun Barrels. *AFTE Journal*, 30(3):438–444, 1998. [p1, 2]

W. Chang, J. Cheng, J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges. *shiny: Web Application Framework for R*, 2021. URL https://CRAN.R-project.org/package=shiny. R package version 1.6.0. [p2]

Z. Chen, W. Chu, J. A. Soons, R. M. Thompson, J. Song, and X. Zhao. Fired bullet signature correlation using the Congruent Matching Profile Segments (CMPS) method. *Forensic Science International*, 305: Article 109964, (10 pages), 2019. ISSN 0379-0738. doi: \url{https://doi.org/10.1016/j.forsciint.2019.109964}. URL https://www.sciencedirect.com/science/article/pii/S0379073819303767. [p1, 2, 4, 5, 6, 7, 9, 11, 13, 14, 16, 17]

L. S. Chumbley, M. D. Morris, M. J. Kreiser, C. Fisher, J. Craft, L. J. Genalo, S. Davis, D. Faden, and J. Kidd. Validation of tool mark comparisons obtained using a quantitative, comparative, statistical algorithm: Validation of tool mark comparisons. *Journal of forensic sciences*, 55(4):953–961, 2010. ISSN 0022-1198. doi: https://doi.org/10.1111/j.1556-4029.2010.01424.x. [p1, 2]

W. S. Cleveland, E. Grosse, and W. M. Shyu. Local regression models. In J. M. Chambers and T. J. Hastie, editors, *Statistical Models in S*, chapter 8, pages 309–376. Chapman and Hall/CRC, Boca Raton, Florida, 1991. [p2]

Committee on Identifying the Needs of the Forensic Sciences of the National Research Council. Strengthening Forensic Science in the United States: A Path Forward. https://www.ncjrs.gov/pdffiles1/nij/grants/228091.pdf, 2009. [p1]

J. E. Hamby, D. J. Brundage, and J. W. Thorpe. 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, 2009. [p1, 2, 6, 17]

J. E. Hamby, D. J. Brundage, N. D. K. Petraco, and J. W. Thorpe. A Worldwide Study of Bullets Fired From 10 Consecutively Rifled 9MM RUGER Pistol Barrels—Analysis of Examiner Error Rate. *Journal of Forensic Sciences*, 64(2):551–557, 2019. ISSN 1556-4029. doi: 10.1111/1556-4029.13916. [p1, 2]

E. Hare, H. Hofmann, and A. Carriquiry. Automatic matching of bullet land impressions. *Ann. Appl. Stat.*, 11(4):2332–2356, 12 2017. doi: 10.1214/17-AOAS1080. [p1, 2, 6, 13, 16, 17]

H. Hofmann, S. Vanderplas, and G. Krishnan. *bulletxtrctr: Automatic Matching of Bullet Striae*, 2019. URL https://heike.github.io/bulletxtrctr/. R package version 0.2.0. [p2, 17]

H. Hofmann, S. Vanderplas, G. Krishnan, and E. Hare. *x3ptools: Tools for Working with 3D Surface Measurements*, 2020. URL https://github.com/heike/x3ptools. R package version 0.0.3. [p2]

G. Krishnan and H. Hofmann. Adapting the Chumbley Score to Match Striae on Land Engraved Areas (LEAs) of Bullets. *J Forensic Sci*, 64(3):728–740, May 2019. [p1, 2]

President's Council of Advisors on Science and Technology. Report on forensic science in criminal courts: Ensuring scientific validity of feature-comparison methods. https://obamawhitehouse.archives.gov/sites/default/files/microsites/ostp/PCAST/pcast_forensic_science_report_final.pdf, 2016. [p1, 17]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL http://www.R-project.org/. ISBN 3-900051-07-0. [p1]

J. Song, L. Ma, E. Whitenton, and T. Vorburger. 2d and 3d surface texture comparisons using autocorrelation functions. In *Measurement Technology and Intelligent Instruments VI*, volume 295 of *Key Engineering Materials*, pages 437–440. Trans Tech Publications Ltd, 8 2005. doi: 10.4028/www.scientific.net/KEM.295-296.437. [p1, 2]

United States Department of Justice, Federal Bureau of Investigation. Crime in the United States, 2019, a. URL https://ucr.fbi.gov/crime-in-the-u.s/2019/crime-in-the-u.s.-2019/tables/expanded-homicide-data-table-8.xls. Expanded Homicide Data Table 8. [p1]

United States Department of Justice, Federal Bureau of Investigation. Expanded Homicide Tables, 2020, b. URL https://s3-us-gov-west-1.amazonaws.com/cg-d4b776d0-d898-4153-90c8-8336f86bdfec/CIUS/downloads/2020/expanded-homicide-2020.zip. Data available through the link, not published to the website yet. [p1]

S. Vanderplas, M. Nally, T. Klep, C. Cadevall, and H. Hofmann. Comparison of three similarity scores for bullet lea matching. *Forensic Science International*, 308:110167, 2020. ISSN 0379-0738. doi: https://doi.org/10.1016/j.forsciint.2020.110167. URL https://www.sciencedirect.com/science/article/pii/S0379073820300293. [p17]

X. A. Zheng. NIST Ballistics Toolmark Research Database (NBTRB), 2016. URL https://tsapps.nist.gov/NRBTD. [p1, 2, 7]

Wangqian W. Ju
Department of Statistics
Center for Statistics and Applications in Forensic Evidence
Iowa State University
2438 Osborn Dr
Ames, IA 50011
https://github.com/willju-wangqian
ORCiD: 0000-0002-9977-377X
wju@iastate.edu

Heike Hofmann
Department of Statistics
Center for Statistics and Applications in Forensic Evidence
Iowa State University
2438 Osborn Dr
Ames, IA 50011
https://github.com/heike
ORCiD: 0000-0002-9079-593X
hofmann@iastate.edu