Submitted to the Annals of Applied Statistics

A CLUSTERING METHOD FOR GRAPHICAL HANDWRITING COMPONENTS AND STATISTICAL WRITERSHIP ANALYSIS[†]

By Nicholas S. Berry[‡,§], Amy M. Crawford [‡]

Iowa State University[‡] *and Berry Consultants, LLC*[§]

1. Introduction. Many fields rely on the ability to parse and process handwritten text, and in recent years there has been a shift towards the automation of this processing. At its heart, automatic handwriting processing is the task of converting an image of handwriting into usable data. For us, this data serves as the foundation for an analysis of the handwriting. When done algorithmically, handwriting analysis usually falls into one of two categories. One common analysis objective is to recognize the characters written on a page. Another is a determination of who wrote the sample, which is called writer identification. For writer identification, the goal does not concern investigating *what* is written, but the *way* in which it is written.

Within the writer identification framework, it is important to make the distinction between authorship and writership. Authorship analyses often use word choice or punctuation in a document (for example Rosen-Zvi et al. [16] or Seroussi, Zukerman and Bohnert [18]) and can involve a mix of the two objectives described above. Writership identification is limited to an analysis of the shapes that a writer emits via their practiced writing style.

One large and active area of writership identification is in forensic statistics. Such analyses can be used to determine the source of a piece of handwritten evidence, e.g. a bank robbery note or a bomb threat. Traditionally, handwriting analysis of this nature is done by trained forensic practitioners. In practice, a practitioner comes to a conclusion on a decision scale such as the 9−point scale of ASTM Standard E1658-08 [4] where terms like "identification," "probable," and "elimination," are used when making a comparison between a document from an unknown writer and one of a known writer.

This paper is concerned with automating and quantifying portions of the forensic handwriting examination process. In particular, we seek to parse a handwritten document into data, filter these data into clusters with similar characteristics, and explore how a writer's propensity for creating forms that falls into these clusters can be used in a statistical model.

There is existing software for handwriting analysis of this nature. The proprietary product FLASH ID® (Sciometrics LLC, Chantilly, VA, USA) searches a database for

[*]The authors would like to thank Drs. Alicia Carriquiry and Danica Ommen for their guidance, edits, and feedback.

[†]This work was partially funded by the Center for Statistics and Applications in Forensic Evidence (CSAFE) through Cooperative Agreement #70NANB15H176 between NIST and Iowa State University, which includes activities carried out at Carnegie Mellon University, University of California Irvine, and University of Virginia.

Keywords and phrases: clustering, Bayesian, hierarchical modeling, forensic statistics, handwriting

closest writing matches and gives results based on a scoring system. The WANDA workbench [6] and CEDAR-FOX [14] systems have tools to facilitate both automated and interactive document examination, as well as database management frameworks.

Our work does not rely on any character recognition techniques, and is independent of the context on the page. Other work of this nature has been done. For example, in their 2007 paper Bulacu and Schomaker use graphemes that are normalized to a 30×30 pixel image and overlaid to extract pixel shade differences. A standard clustering algorithm yields a codebook, and grapheme distribution across that codebook is one of six features used to identify writers in downstream writer verification analysis. Miller et al. [13], whose methods are "substantially similar to those used in the proprietary product FLASH ID®", group graphemes in a deterministic fashion, and measurement comparisons are made within those groups.

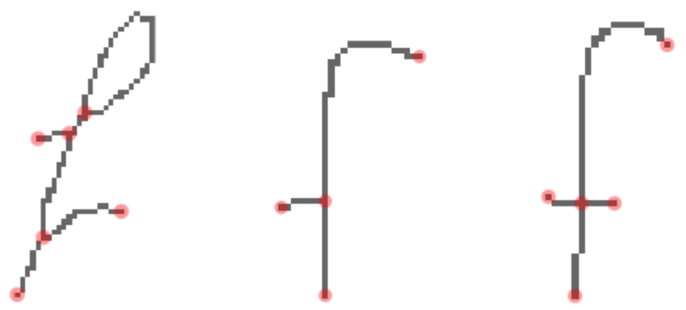
In this paper, we address grouping grapheme-like structures through a K-means type clustering algorithm that relies on a distance measure designed specifically for handwritten structures. We develop our own deterministic grouping method, similar to that of Miller et al. [13], for comparison. Examples of group separation via deterministic and our proposed dynamic grouping methods are given in Section 1.1.

Our document analysis pipeline begins by processing scanned handwritten documents, and segmenting the writing into small graphical structures which we call glyphs. Glyphs are the smallest units of writing that we consider, and they often, but not always, correspond to Roman characters. We cluster glyphs using measures based on the similarities of their major physical attributes. These groupings are more descriptive and repeatable for writers than deterministic groupings, because of their robustness to small structural differences. This dynamic grouping better templates a writer, and when used as data for a Bayesian hierarchical model, improves writer identification results over those of deterministic groupings.

1.1. *A Motivating Example.* Consider two writers. Suppose that writer A favors formal cursive, so they tend to make loops when forming characters such as 'l' and 'f'. Conversely, writer B uses a more broken writing style and tends to make a single stroke 'l' and loopless segments in an 'f'.

Ideally, when writer A's documents are processed their glyphs would be assigned to groups that are characterized by a higher rate of cursive style loops. When writer B's glyphs are considered, their distribution over the groups should differ from that of writer A, since writer B will have fewer glyphs that are characterized by loops, and more that are assigned to groups embodying simple stroke glyphs.

Figure 1 shows three example 'f's from two writers. The characters in Figures 1b and 1c are simple block 'f's (like writer B), while Figure 1a shows a cursive style 'f' with a loop near the top (like writer A). Deterministic and clustering based groupings of these three 'f's are provided in their captions as D and C, respectively. Notice that when compared to Figure 1c, 1b is missing an appendage. This results in different deterministic group assignments for the two letters despite other clear structural similarities. This is unfortunate for an algorithm attempting to differentiate writer A from B based on the grouping.



(a) Writer A
D = 1, C = 1

(b) Writer B
D = 2, C = 2

(c) Writer B
D = 3, C = 2

Fig 1: Three glyphs whose assignment will contribute to characterizing a writer's style. D is deterministic group assignment. C is cluster group assignment.

Under our clustering method, both Figures 1b and 1c are assigned to the same cluster (C=2). Their overall structure is dominating and the small incidental edge that is missing from 1b does not force a separation between groups. This results in a partitioning of the 'f's that can be used to help identify each character's writer. Of course, this is a pathological example with only one character type. In a real example there is a variety of shapes to group and the writer identification is based on an aggregation of small gains across many groups, rather than the hard division described here.

1.2. *Paper Organization.* Section 2 discusses the document processing pipeline for taking a handwritten document from a scanned image to usable data (glyphs). A deterministic grouping method is presented as a direct result of the glyph structures that arise from processing. We introduce our clustering algorithm for glyphs in Section 3. To address our motivating problem of writer identification, the method is applied to writers from the Computer Vision Lab (CVL) handwriting dataset [9] in Section 4. We compare writership analysis results for the dynamic cluster groupings to results using the same model and writers, but based on deterministic groupings.

2. Document Processing.

The pipeline for document pre-processing and data extraction begins with a scanned handwritten document and results in a set of *glyphs*. Glyphs are small pieces of connected ink that serve as individual observations for the analyses that follow. Processing is done using the R [20] package *handwriter*, available on github `https://github.com/CSAFE-ISU/handwriter`. This R package, written by Nick Berry for the Center for Statistics and Applications in Forensic Evidence (CSAFE) at Iowa State University, provides a toolkit for handwritten document processing.

2.1. *Pre-Processing.* In the *handwriter* pipeline, pre-processing begins with binarization to convert an image to pure black and white. Colored images are turned to grayscale using the linear combination $0.2126R + 0.7152G + 0.0722B$ on each pixel. Otsu's binarization method [15] is applied to the grayscale images to assign each pixel to one of two groups in a fashion that maximizes between-class variance. The groups are appropriately normalized to black and white.

Cleaning is done by *handwriter* using Stentiford and Mortimer [19]'s procedure, which introduces a set of masks that aim to isolate and correct spurious pixels, fill white holes likely caused by mistakes in binarization, and widen holes that were likely intentionally made during writing. On the clean binary image we use the Zhang-Suen thinning algorithm [25], to reduce the writing to a one pixel wide skeleton structure that maintains the shape and connections of the original. By choosing to work with the writing skeleton we sacrifice all information about the width of lines. However, thinning facilitates our ability to identify structural components in the handwritten images.

Figure 2a gives an example of a binary image displayed in gray with its thinned, pixel wide counterpart overlaid in black. The overall shape of the written text remains intact and key structural features (like terminal pixels, intersections, and the pixel wide paths connecting them) are easier to detect. Section 2.2 expands on the idea of using simple structural elements of the skeleton to extract requisite information from the documents.

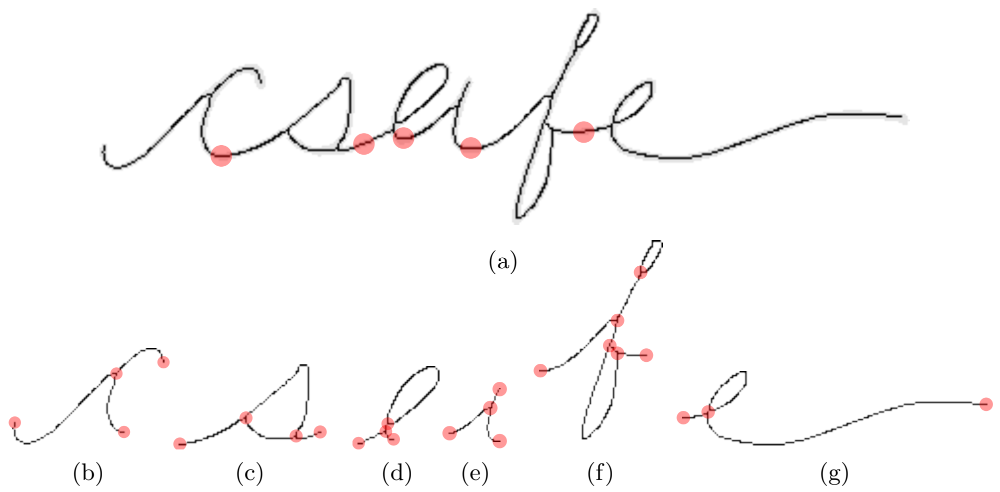


Fig 2: The word "csafe" taken from a processed document. (a) Binarized pen strokes in grey, pixel wide skeleton overlaid in black. Red dots denote glyph breakpoints of the connected writing. (b)-(g) Each glyph shown separately where red dots show nodes (endpoints and intersections).

2.2. *Data Extraction.* After a document has gone through pre-processing, the *handwriter* package decomposes the skeletonized writing into glyphs. Consider the ith glyph of a document as an attributed graph $\mathcal{G}_i$ with a set of vertices $\mathcal{V}_i$ made up of the terminal pixels and locations of intersecting lines of the glyph. The set of edges $\mathcal{E}_i$ represents paths in glyph i that connect elements of $\mathcal{V}_i$. For the glyph representing an "x", shown in Figure 3, the red dots make up the set $\mathcal{V}_i$, and each of the four paths (individualized by numbers) are elements of the set $\mathcal{E}_i$. These graph structures, similar to those in [17, 13, 24, 8], will be the basis for evaluating the structural styles of a writer.

We start by breaking the writing down into a set of connected pieces of ink. In cursive writing this generally corresponds to a connected word, and in disjoint print, a single character. Individual connected ink blots are then candidates to be decomposed further into glyphs.

handwriter makes the decision on which ink blots to break by looking through each edge in an ink blot and breaking those that satisfy a set of rules designed to estimate the intended character breaks within that ink blot.

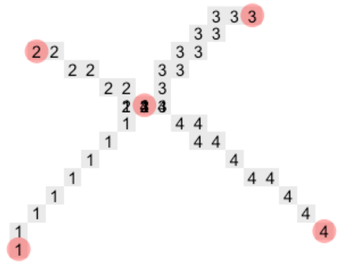


Fig 3: The letter 'X' processed by *handwriter*. Numbers show different paths. Red dots (if in color) indicate nodes. Note that the middle node is actually a merging of two close by nodes.

When breaking on an edge, if that edge has a clear trough (lowest point), then it is broken at the base of that trough, and if not then it is broken at its midpoint. Breakpoints of the word "csafe" in Figure 2a are shown in red.

After breaking, the document has been fully decomposed into a disjoint set of glyphs $\{\mathcal{G}_i\}$, each with their respective vertices and edges. Finally, the vertices in each glyph are systematically ordered so that we can compare vertices across glyphs in sections that follow. With these ordered verices and edges between them, an adjacency matrix for each glyph is constructed.

It is worth mentioning that we only use *handwriter* for document processing, but it also has feature extraction capabilities such as finding centroids, slants, and other measurables for each glyph. These features are undeniably important for forensic handwriting analysis, but will not be used for the clustering that is of focus here, and are thus omitted.

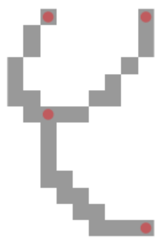
2.3. *Deterministic Adjacency Grouping.*

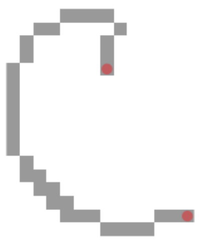
We previously suggested that by establishing a grouping system for glyphs, we could assess the rate at which a writer produces glyphs of each group, and use those rates to characterize a writer's style. The first grouping method, which we will use for baseline comparison with our clustering method, uses only the edge connectivity of a glyph. Glyphs with identical adjacency matrices are placed together in a group. We call this the **adjacency grouping** method, and it is readily available after processing by *handwriter*. This deterministic grouping method is sensitive to small changes in glyph structure, because small incidental pen strokes change the adjacency matrix of a glyph, and thus the adjacency group assignment. Since this method is so sensitive to small differences between the glyphs, the number of resulting groups is very large. Walch and Gantz [24], Gantz, J Miller and Walch [8], Saunders et al.

[17], Miller et al. [13] utilize, in part, a similar method, which they call the “isocode”.

We analyze 162 documents from the CVL database, and *handwriter* partitions these documents into a total of 52,451 glyphs. The number of resulting unique adjacency groups is 1,636. The two most common adjacency groups, shown in Figure 4 with glyph examples, account for approximately 60% of all glyphs in the CVL documents.

Advantages of this grouping method come from the strict structural similarity imposed by the identical adjacency matrices between the members of a group. Within a group, each glyph has the same number of edges and vertices, allowing for one-to-one comparisons between structural components of the graphs. Gantz, J Miller and Walch [8] leverage this advantage. On the other hand, the required strict similarity means that glyphs with minor differences are not grouped together, are never compared, and potentially leave valuable information unused.


(a)


(b)

$$\begin{array}{c|cccc|} & a & b & c & d \\ \hline a & 0 & 1 & 1 & 1 \\ b & 1 & 0 & 0 & 0 \\ c & 1 & 0 & 0 & 0 \\ d & 1 & 0 & 0 & 0 \\ \hline \end{array}$$

(c)

$$\begin{array}{c|cc|} & a & b \\ \hline a & 0 & 1 \\ b & 1 & 0 \\ \hline \end{array}$$

(d)

Fig 4: 4a shows an example of the most common graph structure. 4b shows an example of the next most common. (c) and (d) show the corresponding adjacency matrices.

3. Clustering. Rather than assign groups through deterministic means, we propose a dynamic and flexible K-means based grouping method that is tolerant to incidental pen strokes. This allows glyphs with similar, but slightly different, graphical structures to be captured in the same group.

All K-means type clustering algorithms hinge on the ability to calculate two essential quantities: a distance (or discrepancy, or similarity) measure and a measure of center. Neither of these measures are readily or easily defined for glyphs. Section 3.1 summarizes the notion of an edit distance used in the general graph framework. Then, for glyphs, which are a type of attributed graph, we develop a novel distance measure and a mean calculation. Finally, we outline an accompanying clustering algorithm yielding a dynamic glyph grouping methodology.

3.1. *Edit Distances.* An existing graph discrepancy measure, which helps to motivate ours, is the error-correcting graph matching of Messmer and Bunke [12]. Their distance measure for two graphs is the cost associated with a sequence of steps necessary to transition from one graph to the other, called the edit distance. This graph edit distance is related to the simpler concept used to quantify the difference in two strings [10]. To obtain the string edit distance, operations are sequentially applied to a string S_1 until it matches another string S_2. The available edit operations are *Change*, *Insert*, and *Delete*. The resulting distance calculation can be posed and solved as a dynamic programming problem [23].

In the graph edit distance calculation, like with strings, the goal is to sequentially apply edit operations to transition one graph to another. The necessary operations for graph editing are conceptually the same as in the string context, except they can be applied to both edges and vertices. Each edit operation is assigned a cost, and the difference between two graphs is the sum of the minimal cost edit sequence between them.

Graphs that are compared using this method have labelled vertices, but generally do not have edge attributes such as lengths or curvatures, or vertex attributes like locations. The graphs that describe handwritten glyphs do have these attributes, which will be leveraged in their distance calculations. In the following section, we exploit edge attributes to develop a distance measure for two glyphs and a mean calculation for a set of glyphs. These measures will emulate aspects of the graph edit distance measure, with costs corresponding to the magnitude of the changes necessary to transition one glyph into the other.

3.2. *Distance Measure for Glyphs.* There are distinct differences in how distances are approached for glyphs, rather than general graphs. First, only edges of the glyphs are taken into account. There is no cost directly associated with vertex edits. Every edge has two vertices, so differences in the vertices of the glyphs can be reflected by instead altering edges. Second, the attributes of the edges will be used to determine the cost associated with each change.

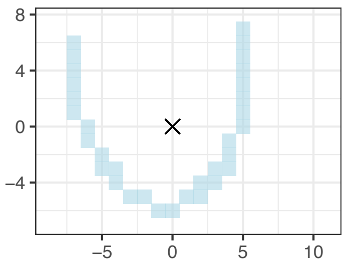
Each glyph is completely described by its collection of edges, so the task of calculating glyph distances can be simplified to calculating edge distances. We present a distance measure for two edges, then provide the mechanism for computing full glyph distances by combining edge distances.

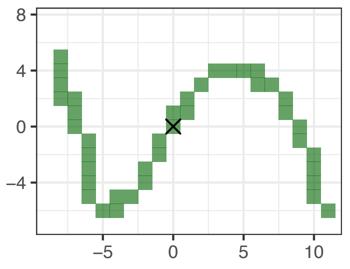
3.2.1. *Distance between Two Edges.* To develop the edge distance measure we keep in mind that the distance between glyphs should be, at least in part, characterized by their structures, while allowing glyphs with similar but not identical structures to be grouped together.

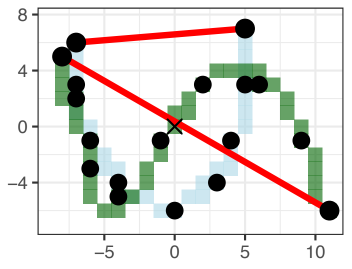
For two edges, the edge distance calculation is comprised of three component distances, d_{loc}, d_{sld}, and d_{sh}, capturing the difference in **endpoint locations**, difference in the **straight-line distance** between edge endpoints, and a rough estimation of the difference in edge **shapes** respectively. Each of these components will be addressed in turn in Section 3.2.2.

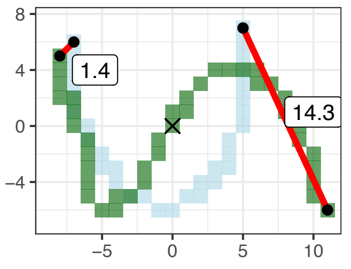
3.2.2. *Edge Distance Calculation.* To allow for meaningful location comparisons of edges, we first align glyphs on the 2-dimensional coordinate system with their centers of gravity overlapping on the origin. Figures 5a and 5b give single edge glyphs, each with their centroids anchored at the origin.

Note that the starting and ending points of edges are labeled arbitrarily, so the edge comparison will need to be considered in both traversal directions. When end point ordering is relevant for the three component calculations, a plus sign subscript, d_+, indicates a distance component taken in the original assignment order of the paths, and a minus sign subscript, d_-, indicates that the order of the second path is considered in reverse. Choosing the correct order of comparison will be addressed in Equation 3.3 after all three of the edge distance components have been established. Only the positive traversal comparisons are pictured in Figure 5, as that turns out to the be the proper comparison direction for these two edges.

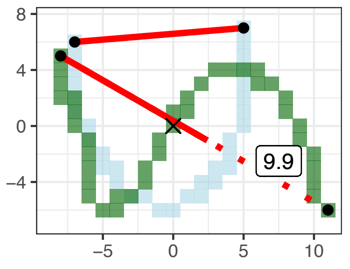

(a) Glyph 1


(b) Glyph 2

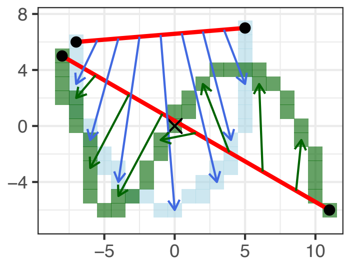

(c) Glyph components

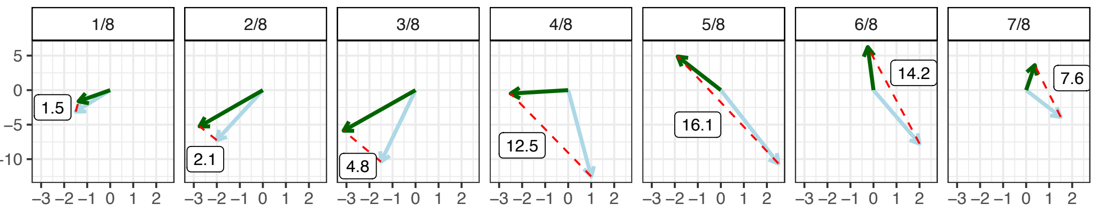


(d) Location distance component, d_{loc}



(e) Difference in straight line distance, d_{sld}


(f) Shape point calculation



(g) Seven pieces of shape distance component, d_{sh}

Fig 5: Edge distance measure calculation companion figures.

We begin by defining the endpoint location component of the edge distance, d_{loc}, which only requires the pixel positions of the end points of the paths. Denote the position of the first terminal pixel, the starting point of an edge, e, as p_1^e. The second terminal pixel position, the end point of the edge, is p_2^e. These locations are recorded as (x, y) points on the coordinate plane. The points connected by red lines in Figure 5c represent the edge end points.

Let $||\cdot||_2$ denote the Euclidean norm. Then, the location component is defined as, $d_{loc}(e_1, e_2) = \{d_{loc+}(e_1, e_2),\ d_{loc-}(e_1, e_2)\}$, where

$$d_{loc+}(e_1,e_2) = \min\Big\{||p_1^{e_1} - p_1^{e_2}||_2,\ ||p_2^{e_1} - p_2^{e_2}||_2\Big\}$$
$$d_{loc-}(e_1,e_2) = \min\Big\{||p_1^{e_1} - p_2^{e_2}||_2,\ ||p_2^{e_1} - p_1^{e_2}||_2\Big\}.$$

In this formula, any two paths that either begin or end at the same pixel location will have a $d_{loc} = 0$ regardless of their shapes, angles, or lengths. Figure 5d shows $||p_1^{e_1} - p_1^{e_2}||_2 = 1.4$ and $||p_2^{e_1} - p_2^{e_2}||_2 = 14.3$, resulting in $d_{loc+}(e_1, e_2) = 1.4$.

The second and third components will make use of the straight line that connects p_1^e and p_2^e in an edge, e, which we denote as ℓ^e. The red line segments in Figure 5c depict ℓ^{e_1} and ℓ^{e_2} for the two example single edge glyphs.

The second distance component concerns the difference in the lengths of ℓ^{e_1} and ℓ^{e_2}. This component adds a penalty reflecting the difference in the straight line displacement of the endpoints in each edge. Define this straight line distance component $d_{sld}(e_1, e_2)$ as

$$d_{sld}(e_1,e_2) = \Big|\ ||\ell^{e_1}||_2 - ||\ell^{e_2}||_2\ \Big|.$$

In this case there is no need to make this comparison in both directions since the specification of the straight line between end points for an edge, ℓ^e is unchanged by the ordering of the terminal pixel locations d_1^e and d_2^e. The diagonal segment in Figure 5e depicts this component for our example edges. The solid portion has the same length as the shorter segment, and the dotted portion, with a length of 9.9, represents the difference in the distances of ℓ^{e_1} and ℓ^{e_2}.

For the third component of edge distance, we define a set of points that capture the shape of an edge. Denote the seven points on the edge e that cut it into eight equal length pieces as $q_1^e, q_2^e, q_3^e, \ldots, q_7^e$. Similarly, place seven equally spaced points on the straight line that connects the edge endpoints and call them $q_1^\ell, q_2^\ell, q_3^\ell, \ldots, q_7^\ell$. Then, subtract each of the line points from each of the corresponding edge points to obtain seven calculated shape points that are a rough representation of the shape of edge e. These shape points are $s_1^e = q_1^e - q_1^\ell, \ldots, s_7^e = q_7^e - q_7^\ell$.

The shape points capture the direction and magnitude of the edge's deviation from the straight line ℓ^e. If an edge is relatively straight (think the number one), we do not expect to see much deviation from ℓ^e. If an edge has more curvature (think the number five) we expect more deviation of the edge from the straight line.

We take d_{sh} as the average difference in the corresponding shape points of e_1 and e_2. Here, we certainly need to consider both the forward (d_{sh+}) and backward (d_{sh-})

orderings of the shape points. The shape contribution to the edge distance measure $d_{sh}(e_1, e_2) = \{d_{sh+}(e_1, e_2),\ d_{sh-}(e_1, e_2)\}$ is

$$d_{sh+}(e_1, e_2) = \frac{1}{7}\Big[||s_1^{e_1} - s_1^{e_2}||_2 + \cdots + ||s_7^{e_1} - s_7^{e_2}||_2\Big] \tag{3.1}$$

or

$$d_{sh-}(e_1, e_2) = \frac{1}{7}\Big[||s_1^{e_1} - s_7^{e_2}||_2 + \cdots + ||s_7^{e_1} - s_1^{e_2}||_2\Big]. \tag{3.2}$$

In 5f, vectors are drawn from the line points to the edge points, depicting the formulation of the shape points s_1 through s_7 for each edge. Figure 5g shows the same vectors, pointing at each of the seven shape points for each edge. The distance between shape points in each panel of Figure 5g are taken (shown next to the red lines), and the average of those 7 distances make up the shape component of the edge distance. Here,

$$d_{sh+} = \frac{1}{7}(1.5 + 2.1 + 4.8 + 12.5 + 16.1 + 14.2 + 7.6) = 8.4$$

Now, with all three edge comparison components (d_{loc}, d_{sld}, and d_{sh}) in-hand, the final step is to combine them. We must take care to scale the edge distances so edges that make up a large proportion of their glyphs are weighted more heavily. This gives way to an edge distance calculation that is robust to differences in small edges, but demands that large edges be similar. This is a favorable property since larger edges generally dominate the structure of a glyph, and to find similar glyphs, we seek similar dominating structures. The small edge distance down weighting also provides tolerance by preventing small edges from drastically influencing the distance measure. The amount that an edge distance is down weighted is based on the average proportion of the two glyphs that the edges e_1 and e_2 make up.

The distance measure between two edges $d(e_1, e_2)$ is the final culmination of the three comparison components and weighting results. It is given as

$$d(e_1, e_2) = \frac{1}{2}\left[\frac{|e_1|}{\sum_{i=1}^{|\mathcal{E}_1|}|e_i^{\mathcal{E}_1}|} + \frac{|e_2|}{\sum_{i=1}^{|\mathcal{E}_2|}|e_i^{\mathcal{E}_2}|}\right] \times \min\{d_{loc+} + d_{sld} + d_{sh+}, d_{loc-} + d_{sld} + d_{sh-}\}, \tag{3.3}$$

where we let $e_i^{\mathcal{E}_j}$ denote the i^{th} edge in $\mathcal{E}_j$, the set of edges comprising glyph j. The leading fraction in (3.3) is the weighting component, where $|e_i|$ denotes the length of path i and $|\mathcal{E}_j|$ is the total number of edges in glyph j. The minimum statement of Equation 3.3 is the mechanism for choosing the direction of edge comparison that yields the smallest distance measure.

The edge distance calculation relates to fundamentals of the graph edit distance. The endpoint location component d_{loc} emulates an edit operation that shifts an edge in space. The cost of this edit is equivalent to the distance that an edge must travel to align itself with its counterpart's nearest endpoint. The straight-line distance component d_{sld} measures the difference in displacement of the edge endpoints. This is comparable to stretching or compressing edges as an edit operation. The final component d_{sh} represents the amount that two paths have to be straightened, curved, or twisted in order to match each other.

3.2.3. *Glyph Distance Measure from Edge Calculations.* In Figure 5 the edge distance is equal to the glyph distance since each glyph has only one edge. For more complicated glyphs, combining edge distances to form a complete graph distance happens in two phases. First, we handle potential differences in edge counts between two glyphs. Then, the optimal matching between the edge sets is chosen via evaluation of all available edge distance comparisons. The edge distances for the resulting matches are summed to form the final glyph distance measure.

To equalize the number of edges between two glyphs, dummy edges are added to the graph with smaller $|\mathcal{E}|$ until the number of edges is equal. These dummy paths do not have any physical structure, and thus cannot be compared with real edges in the usual way (using endpoints, shape points, etc.). Instead, the distance between a real path and a dummy path is assigned solely based on the length of the real path. For a real edge e_r and a dummy edge e_d, we define the distance as

$$\begin{aligned} d(e_r, e_d) &= \frac{1}{2}\left[\frac{|e_r|}{\sum_{i=1}^{|\mathcal{E}_r|}|e_i^{\mathcal{E}_r}|} + \frac{0}{\sum_{i=1}^{|\mathcal{E}_2|}|e_i^{\mathcal{E}_2}|}\right]|e_r|^2 \\ &= \frac{1}{2}\left[\frac{|e_r|^3}{\sum_{i=1}^{|\mathcal{E}_r|}|e_i|}\right]. \end{aligned} \tag{3.4}$$

This calculation uses the same weighting factor as in Equation 3.3 with $|e_2| = 0$ and the distance component set to the squared length of the real path, e_r.

Including the dummy edge comparisons, we make $\max(|\mathcal{E}_1|, |\mathcal{E}_2|)^2$ unique pairwise edge distance calculations for $\mathcal{G}_1$ and $\mathcal{G}_2$. We match the edges in $\mathcal{G}_1$ with edges in $\mathcal{G}_2$ in a one-to-one fashion such that the total edge distance is as small as possible. This task is a constrained minimization optimization problem, which is solved via linear programming.

We now give a formal expression for the complete glyph distance measure. Define $I^* = \left\{i_1^*, \ldots, i^*_{\max\{|\mathcal{E}_1|,|\mathcal{E}_2|\}}\right\}$ to be the set of indices that reorder the edges of $\mathcal{G}_2$ to reflect the optimal minimum distance matching with respect to the original ordering of $\mathcal{G}_1$. The distance between the two glyphs is

$$D(\mathcal{G}_1, \mathcal{G}_2) = \sum_{i=1}^{\max\{|\mathcal{E}_1|,|\mathcal{E}_2|\}} d(e_i^{\mathcal{E}_1}, e_{I_i^*}^{\mathcal{E}_2}). \tag{3.5}$$

3.3. *Weighted Mean of Glyphs.* We begin by defining a weighted mean of two glyphs. Once this procedure has been established, it can be applied in sequence to produce the mean of a larger set of glyphs by iteratively taking weighted means with decreasing weight on each newly introduced glyph. Like the glyph distance measure relied on combining individual edge distances, the glyph mean calculation will rely on combining individually calculated edge means. This procedure requires that the edges of two glyphs have been matched such that they provide a minimum glyph distance calculation as described in Section 3.2.3.

With the exception of dummy edges, an edge is characterized by just its end points, shape points, and length, so the weighted mean of two edges is constructed as the weighted mean of each of these components. The heavier the weight on the first letter, the closer all of the mean edge components will be to that letters end points, shape points, and length.

Figures 6a-6f demonstrate a weighted mean calculation in the simple situation in which both glyphs have one edge. Notice that as the weight on the blue glyph grows, the endpoints, seven shape points, and length of the red mean are pulled toward the blue glyph.

When one of the edges in the mean calculation is a dummy edge, the mean must be calculated slightly differently. A real edge that is matched with a dummy edge lends its spatial components to the mean edge, so the resulting mean has the same end points and shape points as the real edge. However, the length of the resulting edge is a weighted average of the length of the real edge and 0, down weighting that edges importance in the means.

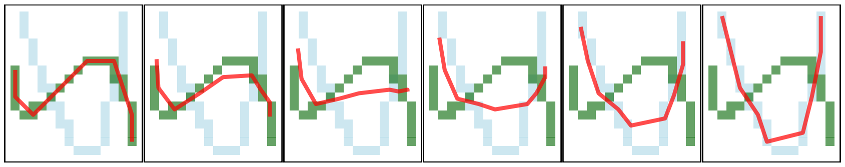
Figures 6g-6l show the weighted mean of two glyphs that have a different number of edges. In this example, the crossing on the green 'f' is matched with the crossing edge on the right side of the blue 'f' as result of the linear programming routine. The dummy edge appears in Figure 6h when the blue glyph begins to accumulate some weight, where it is matched with the remaining real edge. Although the images do not show it, the length of that edge is down-weighted between the length of the blue edge it represents and 0 (the length of the dummy edge).

Due to repeated down weighting of an edge across a set of glyphs, it is possible that the length of certain edges gets very small. We delete any edge from the mean graph that is shorter than 1 pixel long in the sense of weighted length. This prevents the number of edges in the mean graph from being equal to the largest glyph in the set. Figures 6m-6x give a look at a more complex comparison where dummy edges and pruning both play a role as the weightings progress towards $p = 1$.

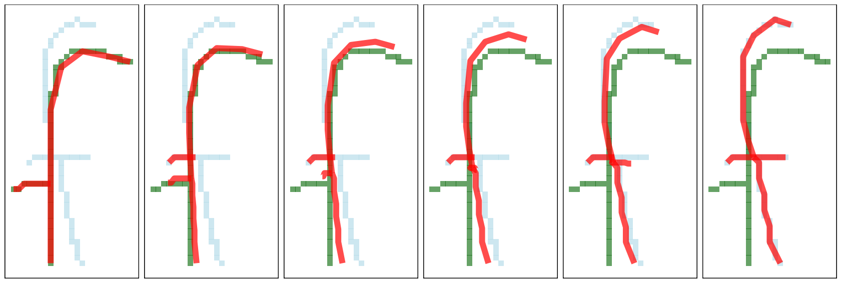
With this process we can calculate the mean of two glyphs for any value of the weight, p. Now consider a larger set of glyphs, say $\mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$. Define a function to calculate the weighted mean $w'(\mathcal{G}_1, \mathcal{G}_2, p)$ where p is the $[0, 1]$ weight put on $\mathcal{G}_1$. Then, the mean of the first two elements of $\mathcal{G}$ is $m_2 = w'(\mathcal{G}_1, \mathcal{G}_2, 0.5)$. To incorporate the third element into the mean, we must take care that the included glyphs are each weighted evenly in m_3, so $m_3 = w'(m_2, \mathcal{G}_3, \frac{2}{3})$. For each subsequent mean calculation the update formula is $m_i = w'(m_{i-1}, \mathcal{G}_i, \frac{i-1}{i})$. Applying this update formula iteratively across a set of glyphs provides a formula for calculating a set mean.

While this set mean does a good job of summarizing a set of similar glyphs, it has some properties that are sub-optimal when encountering a set of widely varying glyphs. The most egregious is that the set mean depends on the order in which the glyphs are introduced into the update formula. This is due to the path matching step of the algorithm, and is unavoidable. We address this shortcoming in the coming section by modifying the clustering measure of center.

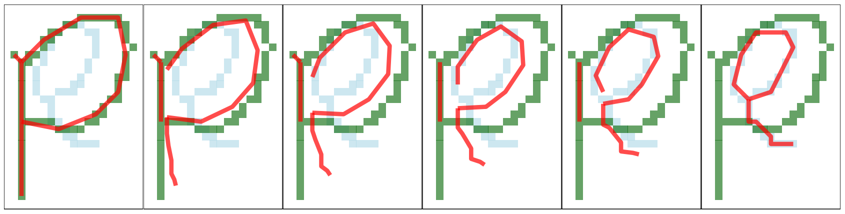
3.4. *K−means Type Algorithm.* Prior to implementing the K-means framework, we consider what an outlier may look like in our data format. The presence of outliers can have undesirable effects on cluster groupings [22]. There will be certainly be glyphs that



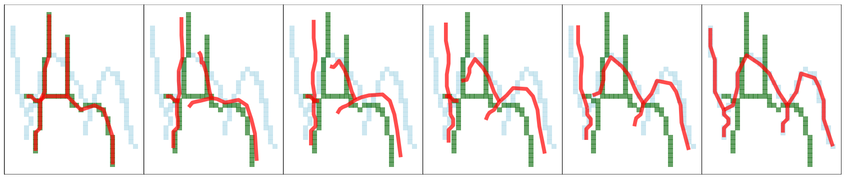
(a) $p = 0$ (b) $p = .2$ (c) $p = .4$ (d) $p = .6$ (e) $p = .8$ (f) $p = 1$



(g) $p = 0$ (h) $p = .2$ (i) $p = .4$ (j) $p = .6$ (k) $p = .8$ (l) $p = 1$



(m) $p = 0$ (n) $p = .2$ (o) $p = .4$ (p) $p = .6$ (q) $p = .8$ (r) $p = 1$

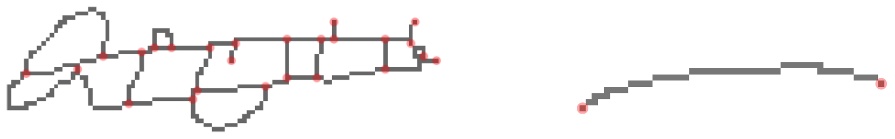


(s) $p = 0$ (t) $p = .2$ (u) $p = .4$ (v) $p = .6$ (w) $p = .8$ (x) $p = 1$

Fig 6: Figures showing weighted means of two letters. The red line represents the weighted mean, and the weighting component, p, indicates the amount of weight placed on the blue glyph in each panel. For example, Figure 6a shows the mean weighted completely on the 'n', then left to right transitions from 'n' to 'v', and finally 6f shows the mean weighted completely on the 'v'.

are very different from all of the others (crossed out words, for example). Additionally, there may be sets of similar glyphs that do not occur frequently enough to form their own cluster, but are not close to any of the other cluster centers. Such observations are considered outliers in the setting.

Figure 7 shows two examples of unconventional glyph types that will be far from every cluster mean. Figure 7a is a large complex glyph that is not repeatable, and Figure 7b is a simple, but uncommon structure in handwriting. When K-means encounters an observation very far from any cluster it may either pull the center of a cluster toward it, thereby summarizing the rest of the cluster poorly, or it may create its own cluster with only that point. Neither of these options are desirable.


(a) A word that was crossed out, and so was not broken apart as per the breaking rules used in *handwriter*.

(b) A long horizontal line, which is a rare character in writing.

Fig 7: Two very different glyphs that will be considered part of an outlier group during the clustering routine.

To accommodate outliers, we implement the method of Gan and Ng [7]. A cutoff distance is derived from the average distance from each observation to its cluster mean, and used to decide which are too far from any center. Such observations are grouped as a set of outliers, meaning they will not contribute to any cluster center calculations. The number of outliers is capped by a parameter, but the number of classified outliers is learned based on the data. In the case of no outliers the modified algorithm simplifies to standard K-means.

The clustering framework of Forgy [5] and Lloyd [11], the simplest formulation of K-means, starts with a set of initial clusters and iterates through two steps until convergence. First, each point in a dataset is assigned to the cluster whose center is closest to it, then the cluster centers are recalculated with all of the points that were just assigned to that cluster. Rather than use the cluster means calculated as in Section 3.3, we increase the stability of the clustering algorithm by using the cluster *exemplar* as the center for the group reassignment step. A cluster exemplar is simply the data point closest to the calculated cluster mean. The complete algorithm is as follows.

Outlier tolerant $K-$means Algorithm for Glyphs:.

Consider a set of observed glyphs $\boldsymbol{X} = \{X_1, \ldots, X_n\}$, a pre-specified number of clusters K, a set of initial cluster exemplars $\mathcal{C} = \{\mathcal{C}_1, \ldots \mathcal{C}_K\}$, a maximum number of outliers n_o, and a parameter γ which is part of the calculation that controls the allowable distance from a center to an observation before it is classified as an outlier. Also define the variable T_o to be the distance threshold by which outliers are determined. T_o will be learned and adjusted within each iteration of the algorithm, but the initial value of $T_o = \infty$. The cluster assignment of observation i is φ_i, where $\varphi_i = -1$ denotes that the i^{th} observation is an outlier.

Iterate through the following steps until the cluster assignments do not change:

1. Assign each glyph $X_i \in \boldsymbol{X}$ to the cluster whose exemplar is nearest to X_i with respect to the distance measure in Section 3.2.

$[\varphi_i = \text{argmin}_{j=1,\dots,K} \{D(X_i, \mathcal{C}_j)\}]$

2. Remove the cluster assignment of X_i and call it an outlier (set $\varphi_i = -1$) if its distance to its cluster exemplar, $D(X_i, \mathcal{C}_{\varphi_i})$, is larger than T_o. If more than n_o observations would be outliers, only reassign the observations with the n_o largest distances.
3. Calculate the mean of the glyphs in each cluster as in Section 3.3. Update $\mathcal{C}_1, \dots \mathcal{C}_K$ to be the X_i closest to each calculated mean (exemplars). Outlier glyphs do not contribute the mean calculations.
   $\left[\mathcal{C}_j = \text{argmin}_{X_i|\varphi_i=j} D(X_i, \text{Mean}(\{X_m \mid \varphi_m = j\})\right]$
4. Update T_o with

$$T_o = \frac{\gamma}{n - \sum_{i=1}^{n} \mathbb{I}(\varphi_i = -1)} \sum_{i=1}^{n} \mathbb{I}(\varphi_i \neq -1) D(X_i, \mathcal{C}_{\varphi_i})$$

When the algorithm has converged, each glyph is assigned to the cluster with the nearest center, with the exception of the outlier cluster. Like all versions of K-means, only a local optimum is guaranteed. Running the algorithm with different starting values is suggested to try to reach the global optimum.

The input parameters n_o and γ are set with the default values of $n_o = .25n$ and $\gamma = 3$, which allows for up to 25% of the observations to be called outliers and requires a distance 3 times the average within cluster distance to call an observation an outlier. For this algorithm, K is fixed before running, and throughout the remainder of the paper, K will be assumed to be known, so no K selection criteria are discussed.

4. Application.

We demonstrate the use of our clustering method in a forensic handwriting analysis context using a subset of the publicly available CVL handwriting database [9] to perform a writership analysis on a set of documents with known origins. The data set is a collection of 162 writing samples by 27 writers, and 5 prompts for each writer. Immediately, we set aside the fourth writing sample of each writer to create a holdout set of documents. These will serve as questioned documents in the writer identification exercise to come. The documents are cropped to the smallest bounding box containing writing and down sampled to contain a maximum of 1750000 pixels without changing the aspect ratio of the document. Then, *handwriter* is used to extract glyphs from each of the documents.

4.1. *Cluster Assignments.*

For computational purposes, we use the glyphs extracted from one document for each writer to perform clustering, effectively creating a cluster template from a diverse collection of writing. After the clusters are created, the glyphs that were not used during clustering are sorted into groups based on their distances to the established cluster exemplars.

The training glyphs are grouped into $K = 40$ clusters. We consider more clusters than letters in the alphabet because there are a variety of forms for many letters that often correspond to glyphs. Supplementary materials provide results for analyses given other reasonable values of K.

Figure 9 shows the 40 exemplars that result from the clustering, ordered based on the number of observed glyphs that fall into each cluster, and Figure 10 shows the calculated means for the clusters in the same order. While the exemplars are used for functional purposes, the combination of these two measures of center provides an interesting point of view to investigate behaviors of the clusters built from handwritten documents. For brevity, we will not discuss each of the 40 clusters individually. Instead, we note some general trends and focus on specific clusters of interest.

Most of the highly populated clusters are very compact, and the exemplars are good descriptors for the rest of the glyphs. Many of these clusters describe very simple glyphs like vertical segments and single path glyphs, such as those representing ‘c’s or the stem of an ‘i’.

Clusters that are comprised of more complicated glyphs have more natural variability both within and between writers. This can be seen in the less populated clusters like 9ab and 9an. Looking at the mean of the clusters, rather than the exemplar, can reveal patterns in the clusters of complex glyphs. Figure 10al, for example, reveals that in the complex cluster shown in Figure 9al, the common structural component is the loop at the bottom of the glyph. In the displayed mean, the dots above the loop signify that there are more edges in the glyph, but they are not important in grouping glyphs into that cluster (recall that small edges come with very small weights in the distance measure). Figure 10an reveals a similar characteristic for the cluster in 9an.

The complex clusters are examples of how this glyph grouping method is tolerant to small deviations from an overall dominating structure. Many of the glyphs in those complex clusters do not have the same adjacency matrix (or even number of edges) and would never be considered in the same group under the deterministic grouping method.

A large number of the clusters align directly with known letter types. Cluster 9m is largely ‘f’s, cluster 9r contains ‘n’s, clusters 9s, 9w, and 9aa are groups of ‘e’s, clusters 9ac and 9aj contain ‘o’s, and so on. If each occurrence of a character type were grouped into a single cluster, then we would be counting the frequency in which letters appear in the writing prompts, rather than capturing a characteristic of handwriting forms. As a mini identification example, temporarily ignoring all the other glyphs groups, the rate at which the writer emits ‘e‘ glyphs into 9s, 9w, or 9aa is information that sets them apart from other writers.

We expected that large, non-repeatable, complex glyphs would be called outliers. This is true to an extent, but many of the complicated glyphs got a cluster to accommodate them. The cluster shown in Figure 9ab represents this unexpected, but not unwelcome cluster type. From Figure 10ab we observe that the mean is essentially a large set of points scattered around the character space, which absorbs the glyphs with a large number of edges, making it very diverse. A large proportion of the outliers are actually simple graphs, pictured in Figure 8, that do not line up with any of the existing cluster types.

Only certain types of writers are likely to produce the large simple glyphs that comprise the outlier cluster. This makes the frequency with which outlier glyphs are observed a useful piece of information for writer identification. Thus, we include it as the 41$^{\text{st}}$ cluster in the analysis that follows.

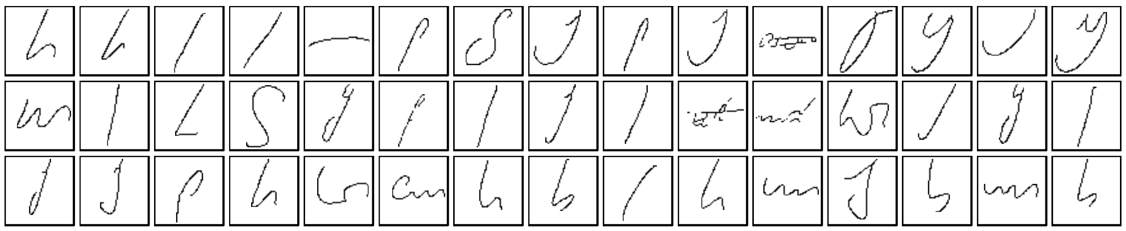


Fig 8: This set of glyphs represent the observations categorized as outliers. Some of the outliers are very complex glyphs, while more are large, simple glyphs. The general trend for outliers is that they are large, simple glyphs.

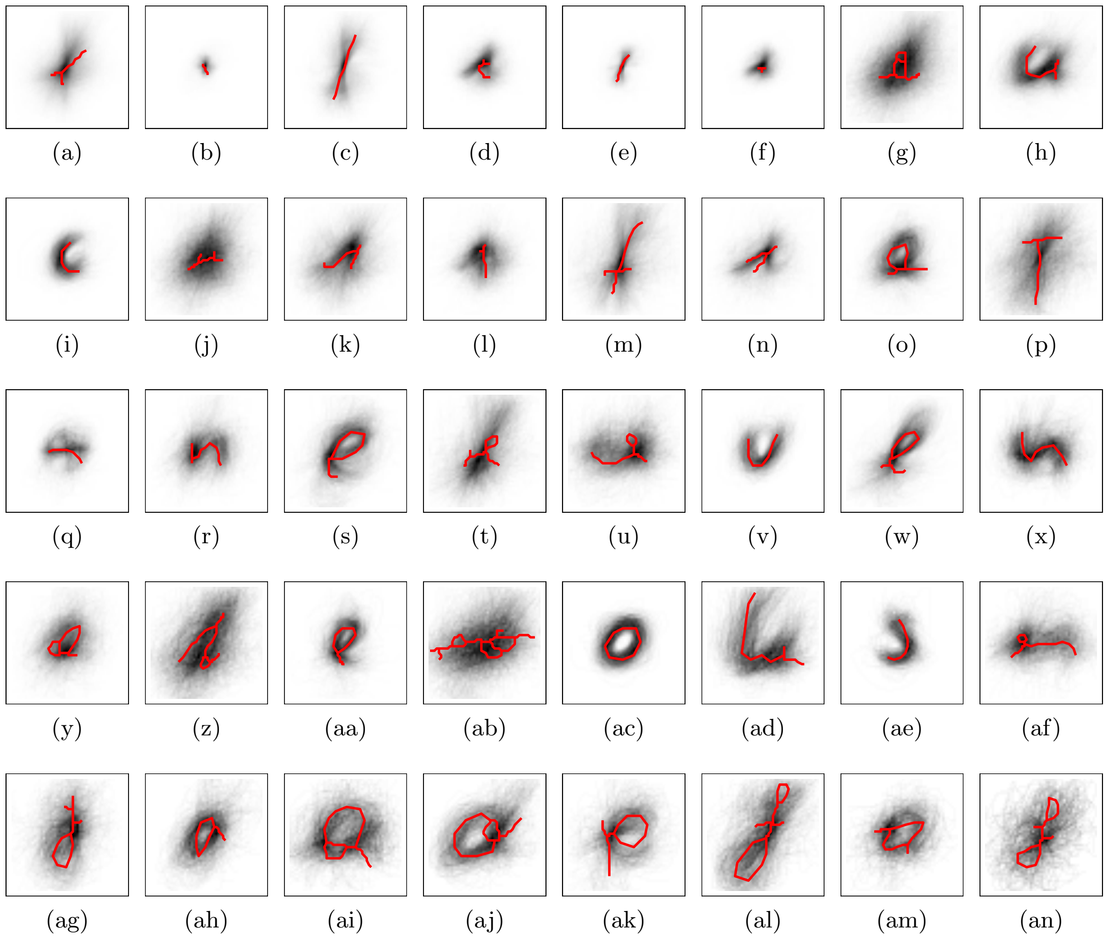


Fig 9: Each subplot represents 1 of the 40 clusters obtained for the 162 CVL documents. The black background figures are the glyphs which were grouped into the cluster. The exemplars are shown in red on top of the glyphs.

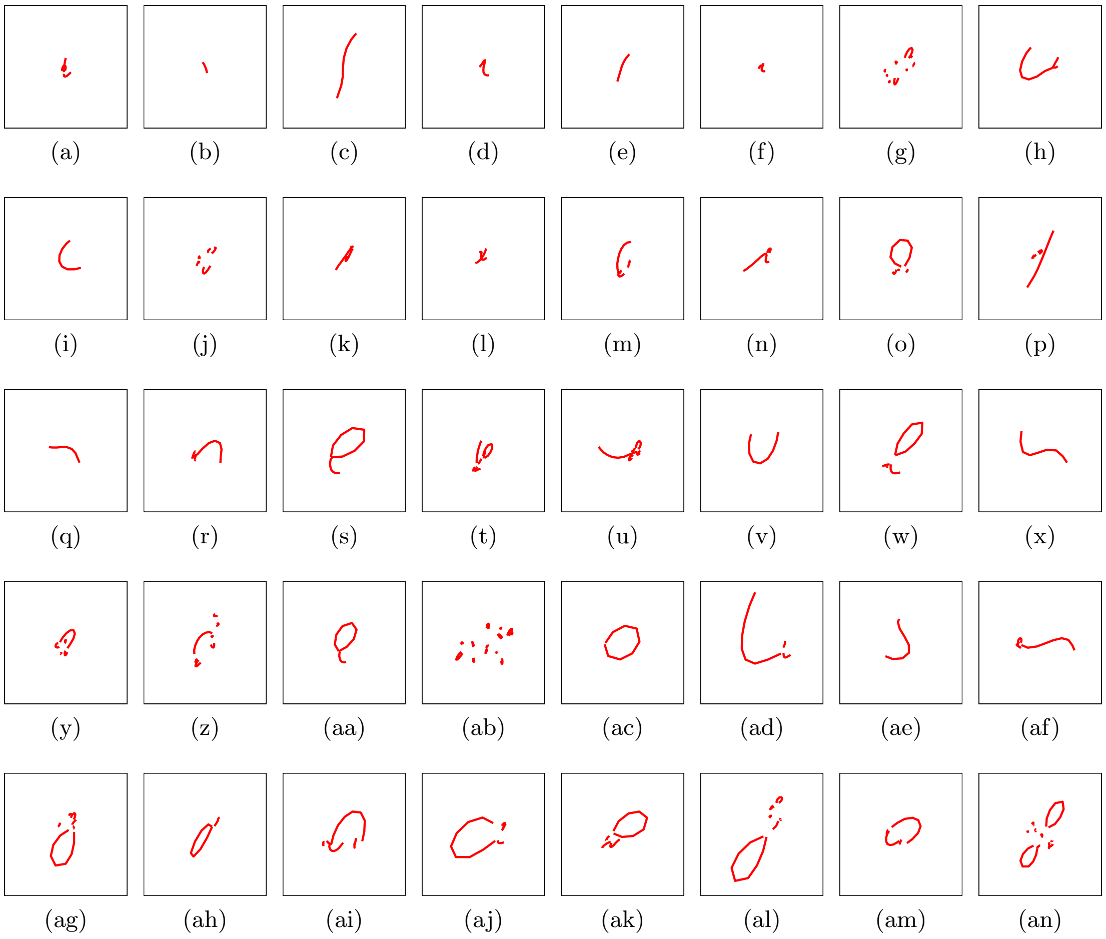


Fig 10: These plots show the raw calculated mean for each of the clusters. The panes are ordered to match the exemplars in Figure 9. These means indicate generally important characteristics of each cluster.

4.2. *Writer Identification.* There are 135 documents in the training data set (5 prompts for each of the 27 writers), and 27 documents in the holdout dataset (1 prompt for each of the 27 writers). We will index model elements from 1 to 27 for the sake of visuals and tables, but we note that the writer identification names from CVL are not consecutive and span the range of 1 to 50.

For each document, we tally the number of glyphs that fall into each of the 41 clusters. A Bayesian hierarchical model for this count data is developed. The notion of gathering handwriting information in "bins" has been used before (Bhardwaj et al. [1] use such data with a Latent Dirichlet Allocation model and Saunders et al. [17] with a variety of classifiers). The desired result of our model is a probabilistic conclusion about writer identification for each of the held out documents. We compare model results from our dynamic cluster groupings with results using the same model, but with the deterministic adjacency grouping method discussed in Section 2.3.

4.2.1. *Model Formulation.* Let $Y_{w(d),c}$ be the number of glyphs assigned to cluster c for training document d nested in writer w where,

- $c = \{1, \ldots, K+1\}$, where $K+1$ arises from the fixed number of clusters plus the group of outliers, and the cluster ordering is the same as in Figures 9 and 10,
- $w = 1, \ldots, 27$, and
- $d = 1, \ldots, 5$.

Then, $\boldsymbol{Y}_{w(d)} = \{Y_{w(d),1}, \ldots, Y_{w(d),K}, Y_{w(d),K+1}\}$ characterizes the number of glyphs assigned to each cluster for document $w(d)$. We consider these cluster assignments as samples from a multinomial distribution with a writer specific parameter vector, $\boldsymbol{\pi}_w$, that captures rate at which a writer emits glyphs to each cluster. Let $\boldsymbol{\pi}_w = \{\pi_{w,1}, \ldots, \pi_{w,K+1}\}$ denote the simplex of size $K+1$ for writer w from a Dirichlet distribution. The $K+1$ hyperparameters of the Dirichlet distribution, denoted as $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_{K+1}\}$, are assigned independent gamma priors. The mathematical formulation of the model is

$$\begin{aligned} \mathbf{Y}_{w(d)} &\overset{ind}{\sim} \mathit{Multi}(\boldsymbol{\pi}_w), \\ \boldsymbol{\pi}_w &\overset{ind}{\sim} \mathit{Dirichlet}(\boldsymbol{\alpha}), \\ \alpha_c &\overset{iid}{\sim} \mathit{Gamma}(a = 3,\ b = 0.25). \end{aligned} \tag{4.1}$$

Markov chain Monte Carlo (MCMC) estimates were obtained using the *rstan* R package [21]. In the analysis that follows, 3000 samples were collected after a burn-in period of 2000. Denote these samples as $\boldsymbol{\alpha}^{(m)}$ and $\boldsymbol{\pi}_w^{(m)}$, for $m = 1, \ldots, M (= 3000)$.

Figure 11 shows posterior densities of the $\pi_{w,c}$ components for all writers and seven of the $K+1$ clusters. With overlap in every pane, there is no cluster that can completely separate the writers. However, considering all of the $\boldsymbol{\pi}_w$ simultaneously, a writer's style can be sufficiently captured and characterized by the rate in which their glyphs are assigned to various clusters. The bottom pane of Figure 11 shows the posterior samples for the probability of observing outlier glyphs for each writer. Overlap of the densities in the outlier pane resemble those in the traditional cluster panes. The separation of writers observed in the outlier densities is again a suggestion that there is identifying information carried in the frequency of outliers.

4.2.2. *Prediction.* We use the model defined in Equation 4.1 to evaluate a holdout document written by unknown writer, w^*. The holdout document is processed by the *handwriter* R package to extract glyphs. Every glyph is assigned to a cluster, and the number of glyphs in each cluster is tallied to obtain a multinomial response vector for the new document $\boldsymbol{Y}^*_{w^*} = \{Y_{w^*,1} \ldots Y_{w^*,K+1}\}$. We drop the document nested within writer notation, $w(d)$, here since we consider only one questioned document at a time.

We use the posterior predictive distribution to calculate posterior probabilities of writership for each of the 27 writers in the training data. Let $\boldsymbol{\pi}_{w'}^{(m)}$ be the m^{th} MCMC sample of the multinomial parameter vector for a particular writer in the training data set, w'.

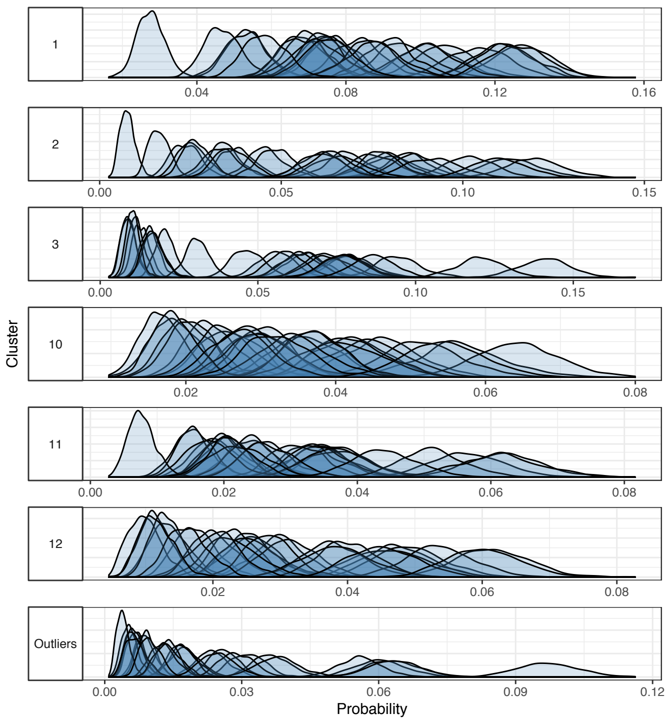


Fig 11: This plot shows densities of the posterior samples for $\boldsymbol{\pi}$. Within a pane there is a density included for each writer. The top three panes of Figure 11 present the cluster proportion estimates for the three largest clusters. That is, the posterior distributions for $\pi_{w,1}$, $\pi_{w,2}$, and $\pi_{w,3}$, $\forall w = 1, \ldots, 27$. The next panes, labelled 10, 11, 12, and Outliers, show the same for less frequently occurring clusters.

Then the posterior probability of writership for writer w' on the new document $\boldsymbol{Y}^*_{w^*}$ at MCMC iteration m is

$$p^{(m)}_{w'} = \frac{Multi(\boldsymbol{Y}^*_{w^*}; \boldsymbol{\pi}^{(m)}_{w'})}{\sum_{w_i=1}^{27} Multi(\boldsymbol{Y}^*_{w^*}; \boldsymbol{\pi}^{(m)}_{w_i})}. \tag{4.2}$$

Performing this calculation for every known training writer yields the probability vector

$$\boldsymbol{p}^{(m)} = \{p^{(m)}_{w_1}, \ldots, p^{(m)}_{w_{27}}\}, \tag{4.3}$$

and we summarize the probability of writership for writers over all MCMC samples as

$$\bar{\boldsymbol{p}} = \{\bar{p}_{w_1}, \ldots, \bar{p}_{w_{27}}\}, \tag{4.4}$$

where $\bar{p}_{w_i} = \frac{1}{M}\sum_{m=1}^{M} p^{(m)}_{w_i}$. In addition to the mean, we calculate lower and upper 90% credible bounds for the p_{w_i}. This posterior predictive procedure was used to evaluate each of the 27 holdout documents.

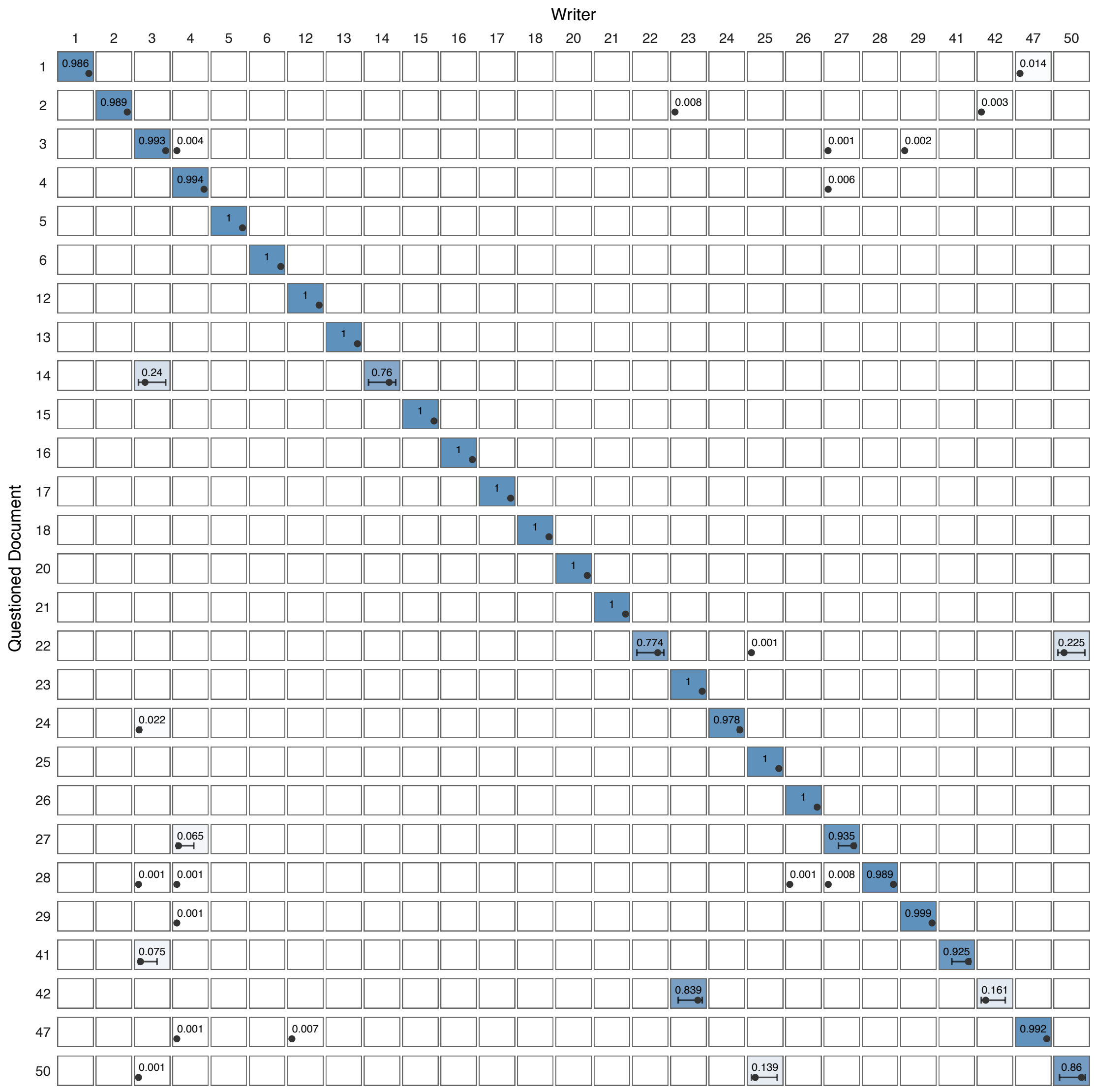


Fig 12: Results of the posterior predictive analysis defined in Equations (4.2) - (4.4) for each of the 27 holdout documents in which glyphs were grouped by our **clustering algorithm**. Columns represent known writers. There is one row to summarize each questioned document analysis with the true writer labeled on the left-hand side. Cells in the row are colored and labeled by elements of their $\bar{\boldsymbol{p}}$ vector defined in Equation (4.4). Rows are independent and sum to one, as all of the posterior writership probability is spread across the writers. Diagonal cells indicate the correct identification for the questioned documents. A point for the average probability, and 90% credible interval band is plotted below the cell label. Cell contents are omitted for any cell whose average probability rounds to 0 in three decimal places.

Figure 12 provides a graphical summary of results for the holdout documents using our glyph clustering method. Each row presents the summary of the probabilities assigned to each of the 27 possible writers for a given questioned document. The diagonal, where the known writer of the questioned document matches the training set writer, indicates a correct identification. We place more than 0.5 of the average posterior probability on the true writer for all but one of the questioned documents. The holdout document written by writer 42 only places 0.161 average probability on the true author, while writer 23 gets the rest. Despite this, the credible interval bands in this row are wide for both highlighted cells, indicating the model acknowledges the uncertainty associated with the questioned document for writer 42. The supplementary material includes figures similar to 12, but with $K = 30$ and $K = 60$ to cluster instead of 40. Results are similar.

The deterministic adjacency groupings of Section 2.3 are used in the same modeling structure. The rigid nature of these groupings are such that approximately 60% of training glyphs fall into two of 1636 adjacency groups. This pile-up renders some of the glyph group assignment information less valuable, since it tends to happen for most writers. In addition, many groups include only one or two glyphs in them. In order to make the deterministic grouping results comparable to that of cluster grouping, we take the sparsely populated adjacency groups and collapse them into two new groups, making 40 total adjacency groups. We call the two new groups "moderate" and "rare" to describe their membership populations.

Figure 13 reflects similar information as Figure 12, but the modeling results are products of the adjacency grouping method described in Section 2.3. Compared to the cluster grouping results of Figure 12, it is clear that there is more posterior probability assigned off-diagonal when the deterministic grouping is used.

Under the adjacency grouping method the model places more than 0.5 of the average probabilities on the true writer for only 23 of the 27 holdout documents. We should note that in a forensic statistics context, one would never use the cutoff of 0.5 probability for actual evidence as corresponding conclusions would be fraught with false positives. We use it here to make simple comparisons between the grouping method results. Many of the on-diagonal cells in the deterministic grouping results reflect probabilities between $0.5 - 0.8$, in which case we would probably not be comfortable making an identification. The adjacency grouping handles the questioned document for writer 42 poorly, which is the document that the clustering-based method also struggles to identify.

5. Discussion.

In this paper, we create a clustering algorithm for grouping small pieces of handwriting, and use an individuals propensity for creating glyphs that belong to particular clusters to identify the writer of questioned source documents. Development of the clustering algorithm hinges on defining measures of distance and center for handwritten graphs. Our glyph distance measure emulates a graph edit distance measure while leveraging additional attributes that the glyphs possess. On the CVL handwriting dataset we create clusters for glyphs derived from handwriting samples and show that writers can be characterized and identified by examining the frequency with which they emit glyphs to the various clusters. The glyph assignments serve as observations for a multinomial-dirichlet hierarchical model, and posterior predictive of writership for a set

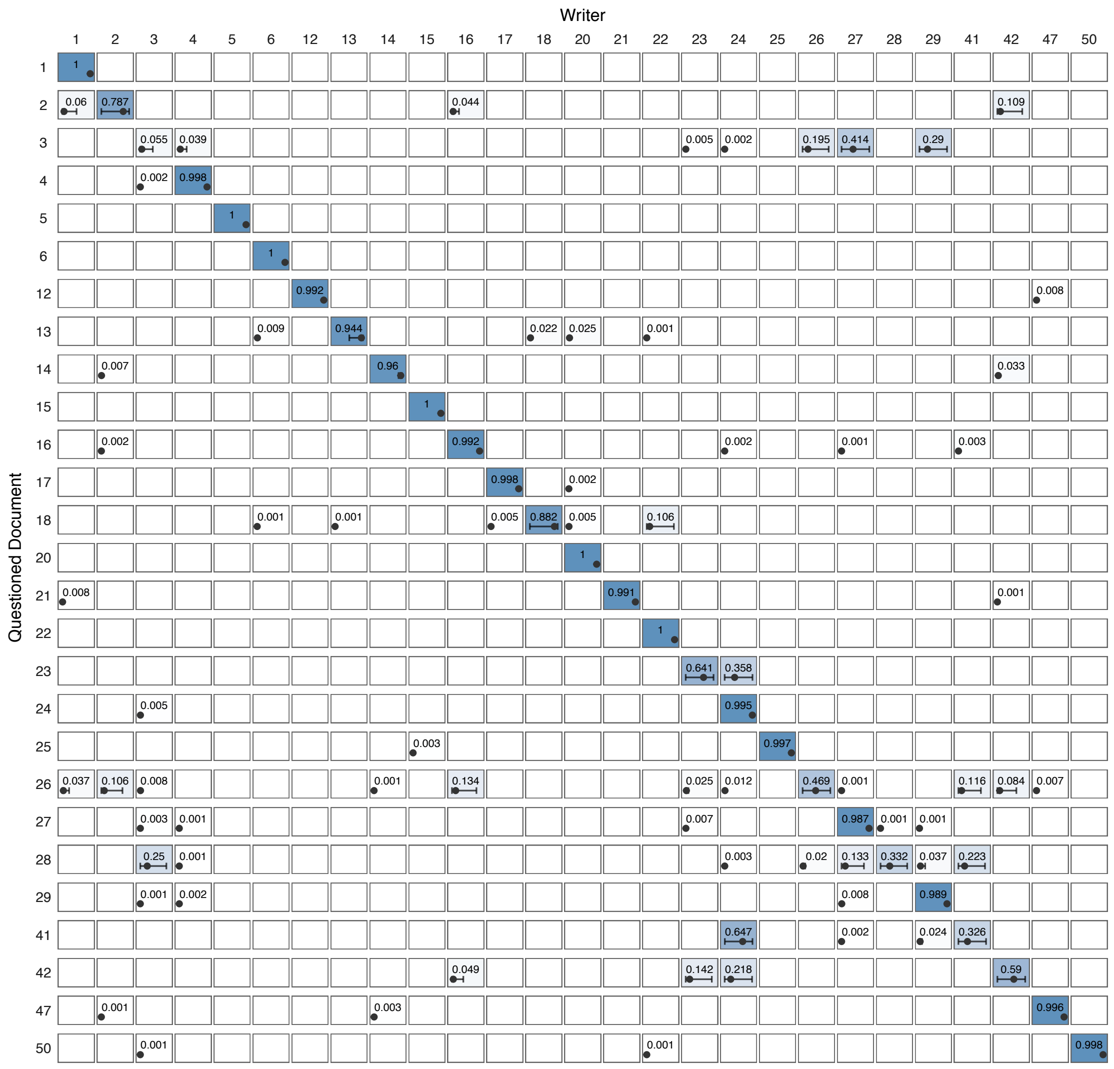


Fig 13: Results of the posterior predictive analysis defined in Equations (4.2) - (4.4) for each of the 27 holdout documents using the **deterministic adjacency grouping** of glyphs. Columns represent known writers. There is one row to summarize each questioned document analysis with true writer labeled on the left-hand side. Cells in the row are colored and labeled by elements of their $\bar{\boldsymbol{p}}$ vector defined in Equation (4.4). Rows are independent and sum to one, as all of the posterior writership probability is spread across the writers. Diagonal cells indicate the correct identification for the questioned documents. A point for the average probability, and 90% credible interval band is plotted below the cell label. Cell contents are omitted for any cell whose average probability rounds to 0 in three decimal places.

of questioned documents is discussed. We compare these identification results to those obtained using deterministic grouping of the glyphs, showing that our cluster assignments improve upon the easily available, but volatile, deterministic assignments and proffers an improvement in model results.

In addition to the clustering algorithm, we describe the procedure by which a document is processed and turned to data. The *handwriter* R package reads, processes, and parses a document into glyphs, the observations we work with during clustering. This processing takes the extremely complex data source, and makes usable information readily available in the form of attributed graphs. The clustering and writership analysis as we did it would not be possible without this extraction.

In the handwriting application there are times when the clustering method produces surprising results. In particular, the mean calculation for diverse clusters results in non-intuitive centers. For diverse clusters, the mean tends to shrink towards the centroid of the observations, and the order that the observations are introduced to the weighted mean calculation can have small yet meaningful impacts on the resulting measure of center. This issue is mitigated in this paper by using exemplars rather than the raw weighted mean as the measure of center for the K-means algorithm, but finding a way to calculate a stable mean for any cluster type could improve and simplify our algorithm.

As with every distance measure, the one presented in this paper favors certain properties over others in assessing similarity. It would be interesting, and easy, to implement distance measures that prioritize other features of handwriting. For example, distance measures that do not take edge shape into account would prioritize edge location more heavily. On the other hand, discounting the location of an edge within a glyph would lead to a measure that cares only about how similar the edge shapes are and not at all about where the edges lie in space. The distance measure that we implement is used because it leans evenly on many types of distance without over emphasizing any specific aspects of the glyphs.

References.

[1] Bhardwaj, A., Reddy, M., Setlur, S., Govindaraju, V. and Ramachandrula, S. (2010). Latent dirichlet allocation based writer identification in offline handwriting. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems* 357–362. ACM.

[2] Bulacu, M. and Schomaker, L. (2007). Text-independent writer identification and verification using textural and allographic features. *IEEE transactions on pattern analysis and machine intelligence* **29** 701–717.

[3] Csardi, G. and Nepusz, T. (2006). The Igraph Software Package for Complex Network Research. *InterJournal* **Complex Systems** 1695.

[4] ASTM Standard E1658-08 (2008). Standard terminology for expressing conclusions of Forensic Document Examiners. *ASTM International* 1–2.

[5] Forgy, E. (1965). Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classifications. *Biometrics* **21** 768-780.

[6] Franke, K., Schomaker, L., Veenhuis, C., Taubenheim, C., Guyon, I., Vuurpijl, L., van Erp, M. and Zwarts, G. (2004). WANDA: A generic Framework applied in Forensic Handwriting Analysis and Writer Identification. In *IEEE Computer Society. 9th IWFHR.*

[7] Gan, G. and Ng, M. K.-P. (2017). K-Means Clustering with Outlier Removal. *Pattern Recognition Letters* **90** 8-14.

[8] Gantz, D., J Miller, J. and Walch, M. (2005). Multi-Language Handwriting Derived Biometric Identification.

[9] Kleber, F., Fiel, S., Diem, M. and Sablatnig, R. (2013). CVL-DataBase: An Off-Line Database for Writer Retrieval, Writer Identification and Word Spotting. In *2013 12th International Conference on Document Analysis and Recognition* 560-564.

[10] Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* **10** 707.

[11] Lloyd, S. (1982). Least Squares Quantization in PCM. *IEEE Transactions on Information Theory* **28** 129-137.

[12] Messmer, B. T. and Bunke, H. (1998). A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20** 493-504.

[13] Miller, J. J., Patterson, R. B., Gantz, D. T., Saunders, C. P., Walch, M. A. and Buscaglia, J. (2017). A Set of Handwriting Features for Use in Automated Writer Identification. *Journal of Forensic Sciences* **62** 722-734.

[14] N. Srihari, S. and Shi, Z. (2004). Forensic Handwritten Document Retrieval System. 188-194.

[15] Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics* **9** 62-66.

[16] Rosen-Zvi, M., Griffiths, T., Steyvers, M. and Smyth, P. (2004). The Author-topic Model for Authors and Documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence. UAI '04* 487–494. AUAI Press, Arlington, Virginia, United States.

[17] Saunders, C. P., Davis, L. J., Lamas, A. C., Miller, J. J. and Gantz, D. T. (2011). Construction and Evaluation of Classifiers for Forensic Document Analysis. *The Annals of Applied Statistics* **5** 381-399.

[18] Seroussi, Y., Zukerman, I. and Bohnert, F. (2011). Authorship Attribution with Latent Dirichlet Allocation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning. CoNLL '11* 181–189. Association for Computational Linguistics, Stroudsburg, PA, USA.

[19] Stentiford, F. W. M. and Mortimer, R. G. (1983). Some New Heuristics for Thinning Binary Handprinted Characters for OCR. *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-13** 81-84.

[20] R Core Team (2018a). *R: A Language and Environment for Statistical Computing.*

[21] Stan Development Team (2018b). *RStan: The R Interface to Stan.*

[22] Tseng, G. C. and Wong, W. H. (2005). Tight Clustering: A Resampling-Based Approach for Identifying Stable and Tight Patterns in Data. *Biometrics* **61** 10-16.

[23] Wagner, R. A. and Fischer, M. J. (1974). The String-to-String Correction Problem. *Journal of the ACM* **21** 168-173.

[24] Walch, M. A. and Gantz, D. T. (2004). Pictographic Matching: A Graph-Based Approach towards a Language Independent Document Exploitation Platform. In *HDP '04*.

[25] Zhang, T. Y. and Suen, C. Y. (1984). A Fast Parallel Algorithm for Thinning Digital Patterns. *Commun. ACM* **27** 236-239.