Supporting Information

Faigenbaum-Golovin et al. 10.1073/pnas.1522200113

Introduction

The main goal of the current research was to estimate the minimal number of authors involved in the scripting of the Arad corpus. To deal with this issue, we had to differentiate between authors of different inscriptions. Although relevant algorithms have been proposed in the past (e.g., ref. 34 for incised lapidary texts), our experience shows that most of the solutions are tailor-made for specific corpora. The poor state of preservation of the Arad First Temple period ostraca, and the high variance of their cursive texts of mundane nature, presented difficulties that none of the available methods could overcome (see Fig. 2). Therefore, novel image processing and machine learning tools had to be developed.

The input for our system is the digital images of the inscriptions. The algorithm involves two preparatory stages, leading to a third step that estimates the probability that two given inscriptions were written by the same author. All of the stages are fully automatic, with the exception of the first, semiautomatic, preparatory step. The basic steps of the algorithm are as follow:

i) Restoring characters via approximation of their composing strokes, represented as a spline-based structure, and estimated by an optimization procedure (for further details see *Description of the Algorithm*, *Character Restoration*).
ii) Feature extraction and distance calculation: creation of feature vectors describing the characters' various aspects (e.g., angles between strokes and character profiles); calculating the distance (similarity) between characters (see *Description of the Algorithm*, *Feature Extraction and Distance Calc*ulation).
iii) Testing the hypothesis that two given inscriptions were written by the same author. Upon obtaining a suitable P value (the significance level of the test, denoted as P), we reject the hypothesis of a single author and accept the competing proposition of two different authors; otherwise, we remain undecided (see *Description of the Algorithm*, *Hypothesis Testing*).

The next section will present an in-depth description of each of the stages. This will be followed by an experimental section that describes the application of our algorithm to both modern and ancient texts. We verify the validity of our approach by applying the algorithm to modern texts (with a number of contemporary texts written by individuals known to us).

Description of the Algorithm

Character Restoration. The state of preservation of most ostraca is poor at best. After more than two and a half millennia buried in the ground, the inscriptions are often blurry, partially erased, cracked, and stained. However, to analyze the script, clear black and white ("binary") images are required. Theoretically, such depictions of the inscriptions do exist, in the form of manually created facsimiles (drawings of the ostraca), created by epigraphic experts. However, these have been shown to be influenced by the prior knowledge and assumptions of the epigrapher (32). A potential solution for this problem could have been provided by automatic binarization procedures from the domain of image processing. Unfortunately, in our experimentations, various binarization methods produced unsatisfactory results (12).

We finally substituted these initial attempts with a semiautomatic approach of individual character restoration. Restoring a character is equivalent to reconstructing its strokes, which are the character's building blocks, and then combining them. Accordingly, henceforth we will discuss the problem of stroke restoration rather than complete character reconstruction. Stroke restoration aims at imitating the reed pen's movement using several manually sampled key points. An optimization of the pen's trajectory is performed for all intermediate sampled points, taking into account information from the noisy character image. A short mathematical description of the procedure follows; for more details and analysis see ref. 14.

A stroke could be referred to as a 2D piecewise smooth curve $(x(t),y(t))$, depending on the parameter $t\in[a,b]$. However, such a representation ignores the stroke's thickness, which is related to the stance of the writing pen toward the document (in our case, a potshard) and to the characteristics of the pen itself. In the case of Iron Age Hebrew, it is well accepted that the scribes used reed pens, which have a flat, rather than pointed, top. This fact makes the writing thickness even more essential to the process of stroke restoration. Therefore, we denote the stroke as a set-valued function:

$$S(t)=\left\{(p,q)|(p-x(t))^2+(q-y(t))^2\le r(t)^2\right\}\quad t\in[a,b],$$

where $x(t)$ and $y(t)$ represent the coordinates of the center of the pen at t, and $r(t)$ stands for the radius of the pen at t (Fig. S1). The corresponding stroke curve is thus

$$\gamma(t)=(x(t),\,y(t),r(t))\;t\in[a,b],$$

whereas the skeleton of the stroke will accordingly be the curve

$$\beta(t)=(x(t),y(t))\quad t\in[a,b]\,.$$

We note that our model of a written stroke is an approximation, because in reality the top of the reed pen was not necessarily a perfect circle.

Borrowing the idea of minimizing an energy functional (35, 36), we produce an analytic reconstruction of a stroke with respect to a given image $I(p,q)$ $((p,q)\in[1,N]\times[1,M])$. This reconstructed stroke $S^*(t)$ is defined as corresponding to the stroke curve $\gamma^*(t)$, minimizing the following functional:

$$F[\gamma(t)]=c_1\int_a^b\frac{G_I(t)}{r(t)^2}dt+c_2\int_a^b\frac{1}{\sqrt{r(t)}}dt+c_3\sum_{j=0}^{J-1}\int_{t_j+\varepsilon}^{t_{j+1}-\varepsilon}|K(\dot{x},\dot{y},\ddot{x},\ddot{y})|\,dt$$

$$\gamma^*(t)=\underset{\gamma(t)}{\operatorname{argmin}}F[\gamma(t)],$$

where $G_I(t)=\sum_{(p,q)\in S(t)}I(p,q)$ is the sum of the gray level values of the image I inside the disk $S(t)$; $\gamma(t_j)=(x(t_j),y(t_j),r(t_j))\;j=0,\ldots,J$ are manually sampled points on the stroke curve $\gamma(t)$, with respect to the natural parameter t; $\dot{x},\ddot{x}$ and $\dot{y},\ddot{y}$ denote the first and second derivatives of x and y; $K(\dot{x},\dot{y},\ddot{x},\ddot{y})=(\dot{x}\ddot{y}-\dot{y}\ddot{x})/(\dot{x}^2+\dot{y}^2)^{3/2}$ stands for the curvature of the skeleton of the stroke $\beta(t)$; $0<c_1,c_2,c_3,\varepsilon\in\mathbb{R}$ are parameters, set to $c_1=2,c_2=2{,}000,c_3=50,\varepsilon=0.01$ in our experiments.

The reconstruction is subject to initial and boundary conditions at (*a*) the beginning and end of strokes; (*b*) intersections of strokes; (*c*) significant extremal points of the curvature; and (*d*) points with no traces of ink. These conditions are supplied by manual sampling.

The energy minimization problem described above is solved by performing gradient descent iterations on a cubic-spline

representation of the stroke (for more details see ref. 14). The end product of the reconstruction is a binary image of the character, incorporating all its strokes.

Fig. S2 presents a restoration of an entire character, stroke by stroke. It can be seen that although the original character image contains several erosions (Fig. S2*A*), the reconstructed strokes (Fig. S2*C*) look both smooth and complete, and their union results in a clear letter, adhering to the character image (Fig. S2*D*).

Feature Extraction and Distance Calculation. Commonly, automatic comparison of characters relies upon features extracted from the characters' binary images. In this study, we adapted several well-established features from the domains of computer vision and document analysis. These features refer to aspects such as the character's overall shape, the angles between strokes, the character's center of gravity, as well as its horizontal and vertical projections. Some of these features correspond to characteristics commonly used in traditional paleography (21).

The feature extraction process includes a preliminary step of the characters' standardization. The steps involve rotating the characters according to their line inclination, resizing them according to a predefined scale, and fitting the results into a padded (at least 10% on each side) square of size $a_L \times a_L$ (with $L = 1, ..., 22$ the index of the alphabet letter under consideration). On average, the resized characters were 300 × 300 pixels.

Subsequently, the proximity of two characters can be measured using each of the extracted features, representing various aspects of the characters. For each feature, a different distance function is defined (to be combined at a later stage; discussed below).

Table S1 provides a list of the features and distances we use, along with a description of their implementation details. Some of the adjustments (e.g., replacement of the L_2 norm with the L_1 norm) were required due to the large amount of noise present in our medium.

After the features are extracted, and the distances between the features are measured, there arises a challenge of combining the various distances. Several combination techniques [e.g., AdaBoost (37) and Bag of Features (38)] were considered. Unfortunately, boosting-related methods are unsuitable due to the lack of training statistics, and the Bag of Features performed poorly in preliminary experiments using a modern handwritten character dataset (details regarding this dataset are given below). Hence, we developed a different approach for combining the distances.

Our main idea was to consider the distances of a given character from all of the other characters, with respect to all of the features under consideration (i.e., two characters closely resembling each other ought to have similar distances from all other characters). Namely, they will both have small distances from similar characters and large distances from dissimilar characters. This observation leads to a notion of a generalized feature vector (defined here for the first time to our knowledge).

The generalized feature vector is defined by the following procedure (for each letter $L = 1, ..., 22$ in the alphabet). First, we define a distance matrix for each feature. For example, the SIFT distance matrix is

$$U_{SIFT} = \begin{pmatrix} D_{SIFT}(1,1) & \cdots & D_{SIFT}(1,J_L) \\ \vdots & \ddots & \vdots \\ D_{SIFT}(J_L,1) & \cdots & D_{SIFT}(J_L,J_L) \end{pmatrix} = \begin{pmatrix} - & \vec{u}^{1}_{SIFT} & - \\ & \vdots & \\ - & \vec{u}^{J_L}_{SIFT} & - \end{pmatrix},$$

where J_L represents the total number of characters, $D_{SIFT}(i,j)$ is the SIFT distance between characters i and j, and $\vec{u}^{i}_{SIFT} = (D_{SIFT}(i,1) \cdots D_{SIFT}(i,J_L))$ is the vector of SIFT distances between the character i and all of the others.

In addition, we denote the SD of the elements of the matrix U_{SIFT} by $\sigma_{SIFT} = std\{D_{SIFT}(i,j) | (i,j) \in \{1,...,J_L\} \times \{1,...,J_L\}\}$. Matrices of all of the other features ($U_{Zernike}$,U_{DCT}, and so forth) and their respective SDs ($\sigma_{Zernike}$, σ_{DCT}, etc.) are calculated in a similar fashion.

Therefore, each character k is represented by the following vector (of size $7 \cdot J_L$), concatenating the respective normalized row vectors of the distance matrices:

$$\vec{u}_k = \left(\frac{\vec{u}^{k}_{SIFT}}{\sigma_{SIFT}} || \frac{\vec{u}^{k}_{Zernike}}{\sigma_{Zernike}} || \frac{\vec{u}^{k}_{DCT}}{\sigma_{DCT}} || \frac{\vec{u}^{k}_{Kd-tree}}{\sigma_{Kd-tree}} || \frac{\vec{u}^{k}_{Proj}}{\sigma_{Proj}} || \frac{\vec{u}^{k}_{L1}}{\sigma_{L1}} || \frac{\vec{u}^{k}_{CMI}}{\sigma_{CMI}} \right) \in \mathbb{R}^{7 \cdot J_L}.$$

In this fashion, each character is described by the degree of its kinship to all of the characters, using all of the various features.

Finally, the distance between characters i and j is calculated according to the Euclidean distance between their generalized feature vectors:

$$chardist(i,j) = \left\| \vec{u}_i - \vec{u}_j \right\|_2.$$

The main purpose of this distance is to serve as a basis for clustering at the next stage of the analysis.

Hypothesis Testing. At this stage we address the main question raised above: What is the probability that two given texts were written by the same author? Commonly, similar questions are addressed by posing an alternative null hypothesis H_0 and attempting to reject it. In our case, for each pair of ostraca, the H_0 is both texts were written by the same author. This is performed by conducting an experiment (detailed below) and calculating the probability ($P \in [0,1]$) of an affirmative answer to H_0. If this event is unlikely ($P \leq 0.2$), we conclude that the documents were written by two different individuals (i.e., reject H_0). However, if the occurrence of H_0 is probable ($P > 0.2$), we remain agnostic. We reiterate that in the latter case we cannot conclude that the two texts were in fact written by a single author.

The experiment, which is designed to test H_0, is composed of several substeps (illustrated in Fig. S3):

i) Initialization: We begin with two sets of characters of the same letter type (e.g., *alep*), denoted A and B, originating from two different texts (Fig. S3*A*).

ii) Character clustering: The union $A \cup B$ is a new, unlabeled set (Fig. S3*B*). This set is clustered into two classes, labeled I and II, using a brute-force (and not heuristic) implementation of k-means ($k = 2$). The clustering uses the generalized feature vectors of the characters, and the distance *chardist*, defined above (Fig. S3*C*).

iii) Cluster labels consistency: If $|I| > |II|$, their labels are swapped.

iv) Similarity to cluster I: For each of the two original sets, A and B, the maximal proportion of their elements in class I (their "similarity" to class I) is defined as

$$MP_I = \max\left\{ \frac{|A \cap I|}{|A|}, \frac{|B \cap I|}{|B|} \right\}.$$

v) Counting valid combinations: We consider all of the possible divisions of $A \cup B$ into two classes i and ii, s.t. $|i| = |I|$. The number of such valid combinations is denoted by NC.

vi) Significance level calculation: The P value is calculated as

$$P = \frac{|\{i \,|\, MP_i \geq MP_I\}|}{NC}.$$

That is, P is the proportion of valid combinations with at least the same observational MP. This is analogous to integrating over a tail of a probability density function.

The rationale behind this calculation is based on the scenario of two authors (negation of H_0). In such a case, we expect the k-means clustering to provide a sound separation of their characters (Fig. S3*D*), that is, *I* and *II* would closely resemble *A* and *B* (or *B* and *A*). This would result in MP_I being close to 1. Furthermore, the proportion of valid combinations with $MP_i \geq MP_I$ will be meager, resulting in a low *P*. In such a case, the H_0 hypothesis would be justifiably rejected.

In the opposite scenario of a single author:

- If a sufficient number of characters is present, there is an arbitrary low probability of receiving clustering results resembling *A* and *B*. In a common case, the MP_I will be low, which will result in high *P*.
- Alternatively, if the number of characters is low, the clustering may result in a high MP_I by chance. However, in this case *NC* would be low, and the *P* will remain high.

Either way, in this scenario, we will not be able to reject the H_0 hypothesis.

Notes:

- We assume that each given text was written by a single author. If multiple authors wrote the text, both H_0 and its negation should be altered. We do not cover such a case.
- In substep *iii*, the swapping is performed for regularization purposes, because the measurement on substep *iv* is not symmetric. Substep *iii* verifies that *I* is a minority class, and thus the value of $MP_I = 1$ is achieved only if the clustering resembles the original sets *A* and *B*.
- In cases where $|I| = |II|$ (substep *iii*), the results of substeps *iv*–*vi* can be affected by swapping the classes. To avoid such infrequent inconsistencies, we perform the calculations for both alternatives, and choose the lower *P*.
- Note that in any case, the definition of *P* in substep *vi* results in $P > 0$.
- Not every text provides a sufficient amount of characters for every type of letter in the alphabet. In our case, we do not perform comparisons for sets *A* and *B* such that: $|A| = 1$ & $|B| \leq 6$ or $|B| = 1$ & $|A| \leq 6$ or $|A| = 2$ & $|B| = 2$.

As specified, substeps *i*–*vi* are applied to one specific letter of the alphabet (e.g., *alep*) present (in sufficient quantities) in the pair of texts under comparison. However, we can often gain additional statistical significance if several different letters (e.g., *alep*, *he*, *waw*, etc.) are present in the compared documents. In such circumstances, several independent experiments are conducted (one for each letter), resulting in corresponding *P*s. We combine the different values into a single *P* via the well-established Fisher method (ref. 33; in case no comparison can be conducted for any letter in the alphabet, we assign $P = 1$). This end product represents the probability that H_0 is true based on all of the evidence at our disposal.

Experiment Details and Results

Our experiments were conducted on two large datasets. The first is a set of samples collected from contemporary writers of Modern Hebrew (www-nuclear.tau.ac.il/~eip/ostraca/DataSets/Modern_Hebrew.zip). This dataset allowed us to test the soundness of our algorithm. It was not used for parameter-tuning purposes, however, because the algorithm was kept as parameter-free as possible. The second dataset contained information from various Arad Ancient Hebrew ostraca, dated to *ca*. 600 BCE, described in detail in the main text (www-nuclear.tau.ac.il/~eip/ostraca/DataSets/Arad_Ancient_Hebrew.zip). Following are the specifications and the results of our experiments for both datasets.

Modern Hebrew Experiment. The handwritings of 18 individuals $i = 1, ..., 18$ were sampled. Each individual filled in a Modern Hebrew alphabet table consisting of 10 occurrences of each letter, out of the 22 letters in the alphabet (the number of letters and their names are the same as in Ancient Hebrew; see Fig. S4 for a table example). These tables were scanned and their characters were segmented. For a complete dataset of the characters, see www-nuclear.tau.ac.il/~eip/ostraca/DataSets/Modern_Hebrew.zip.

From this raw data, a series of "simulated" inscriptions were created. Owing to the need to test both same-writer and different-writer scenarios, the data for each writer were split. Furthermore, to imitate a common situation in the Arad corpus, where the scarcity of data is prevalent (Table S3), each simulated inscription used only three letters (i.e., 15 characters, 5 characters for each letter). In total, 252 inscriptions were "simulated" in the following manner:

- All of the letters of the alphabet except for *yod* (because it is too small to be considered by some of the features) were split randomly into seven groups (three letters in each group) $g = 1, ..., 7$: *gimel*, *het*, *resh*; *bet*, *samek*, *shin*; *dalet*, *zayin*, *ayin*; *tet*, *lamed*, *mem*; *nun*, *sade*, *taw*; *he*, *pe*, *qop*; *alep*, *waw*, *kap*.
- For each writer *i*, and each letter belonging to group *g*, five characters were assigned into simulated inscription $S_{i,g,1}$, with the rest assigned to $S_{i,g,2}$.

In this fashion, for constant *i* and *g*, we can test whether our algorithm arrives at wrong rejection of H_0 for $S_{i,g,1}$ and $S_{i,g,2}$ (FP indicates "false-positive" error; 18 writers and 7 groups producing 126 tests in total). Additionally, for constant *g*, $1 \leq i \neq j \leq 18$, and $b, c \in \{1,2\}$, we can test whether our algorithm fails to correctly reject H_0 for $S_{i,g,b}$ and $S_{j,g,c}$ (FN indicates "false-negative" error $[(18 \times 17)/2] \times 7 \times 2 \times 2 = 4{,}284$ tests in total).

The results of the Modern Hebrew experiment are summarized in Table S2. It can be seen that in modern context the algorithm yields reliable results in ~98% of the cases (about 2% of both FP and FN errors). These results signify the soundness of our algorithmic sequence. The successful and significant results on the Modern Hebrew dataset paved the way for the algorithm's application on the Arad Ancient Hebrew corpus.

Arad Ancient Hebrew Experiment. As specified in the main text, the core experiment addresses ostraca from the Arad fortress, located on the southern frontier of the kingdom of Judah. These inscriptions belong to a short time span of a few years, *ca*. 600 BCE, and are composed of army correspondence and documentation.

The texts under examination are 16 ostraca: 1, 2, 3, 5, 7, 8, 16, 17, 18, 21, 24, 31, 38, 39, 40, and 111. Ostraca 17 and 39 contain writing on both sides of the potshard and were treated as separate texts (17a and 17b and 39a and 39b), resulting in 18 texts under examination. As stated in the algorithm description, we assume that each text was written by a single author. A short summary of the content of the texts can be seen in Table 1.

The seven letters we used were *alep, he, waw, yod, lamed, shin*, and *taw*, because they were the most prominent and simple to restore. In the abovementioned ostraca, out of the 670 deciphered characters of these types in the original publication (6), 501 legible characters were restored, based upon computerized images of the inscriptions. These images were obtained by scanning the negatives taken by the Arad expedition (courtesy of the Israel Antiquities Authority and the Institute of Archaeology of Tel Aviv University). After performing a manual quality assurance procedure (verifying the adherence of the restored characters to the original image; Fig. S2*D*), 427 restored characters remained. The resulting letters' statistics for each text are summarized in Table S3. For a complete dataset of the characters, see www-nuclear.tau.ac.il/~eip/ostraca/DataSets/Arad_Ancient_Hebrew.zip. In addition, a comparison between several specimens of the letter *lamed* is provided in Fig. S5.

We reiterate that our algorithm requires a minimal number of characters to compare a pair of texts. For example, when we compared ostraca 31 and 38, the letters in use were *he* (7:1 characters), *waw* (6:2 characters), and *yod* (4:2 characters). The three independent tests respectively yielded $P = 0.125$, $P = 0.25$, and $P = 1$. Their combination through Fisher's method resulted in the final value of $P = 0.327$, not passing the preestablished threshold. Therefore, in this case, we remain agnostic with respect to the question of common authorship. However, the comparison of texts 1 and 24 used all possible letters, *alep*, *he*, *waw*, *yod*, *lamed*, *shin*, and *taw*, resulting in *P*s of 0.559, 0.00366, 0.375, 0.119, 0.0286, 0.429, and 0.0769, respectively. The combined result was $P = 0.003$, passing the threshold of 0.2. Therefore, in the latter case, we reject the H_0 hypothesis and conclude that these texts were written by two different individuals.

The complete comparison results are summarized in Table 1. We can observe six pairwise distinct "quadruplets" of texts: (*i*) 7, 17a, 24, and 40; (*ii*) 5, 17a, 24, and 40; (*iii*) 7, 18, 24, and 40; (*iv*) 5, 18, 24, and 40; (*v*) 7, 18, 24, and 31; and (*vi*) 5, 18, 24, and 31. The existence of no less than six such combinations indicates the high probability that the corpus indeed contains at least four different authors. As specified in the main text, additional (contextual) considerations can raise this number up to at least six distinct writers. Among these, the different authors of the prosaic lists of names in ostraca 31 and 39 were most likely located at the tiny fort of Arad, implying the composition by authors who were not professional scribes. For the full implications of our results, see the main text.

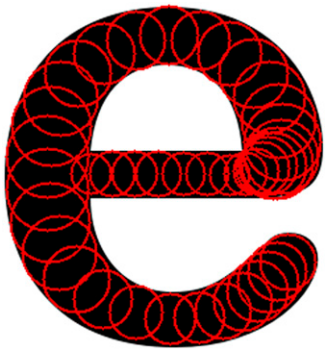

Fig. S1. The Latin character "e" as unification of discs. The discs painted in red over the character were created using the stroke restoration algorithm.

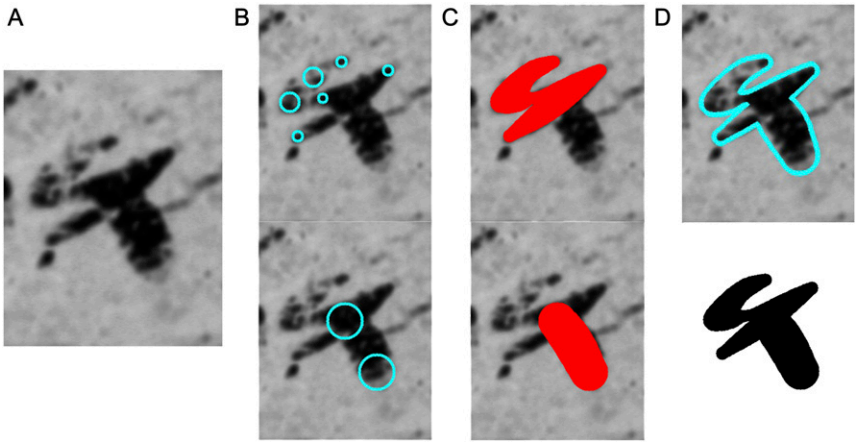

Fig. S2. Example of a semiautomatic stroke restoration of the character *waw* from Arad ostracon 24. (*A*) Image of the character to be reconstructed. (*B*) Manually sampled key points (of top and bottom strokes, respectively). (*C*) The semiautomatic stroke restorations (of top and bottom strokes, respectively). (*D*) The reconstructed character (*Top*: the contour of the reconstructed character overlaid on top of the original image; *Bottom*: the binary image of the restored character). Images are courtesy of the Institute of Archaeology, Tel Aviv University, and of the Israel Antiquities Authority.

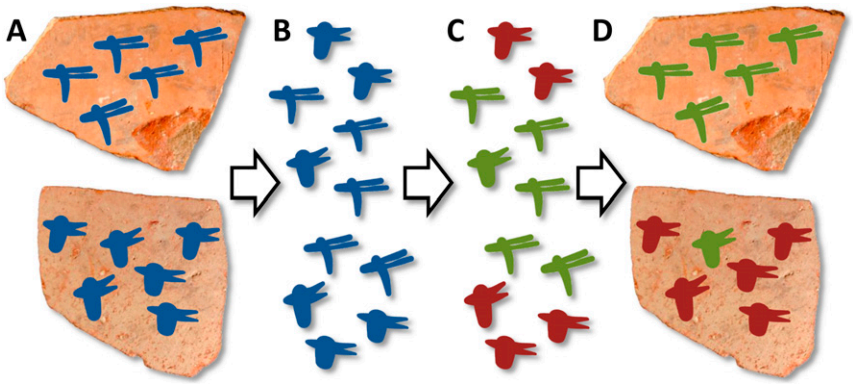

Fig. S3. Artificial illustration of H_0 rejection experiment (containing only *alep* letters). (*A*) Two compared documents. (*B*) Unifying their sets of characters. (*C*) Automatic clustering. (*D*) The clustering results vs. the original documents. Images are courtesy of the Institute of Archaeology, Tel Aviv University, and of the Israel Antiquities Authority.

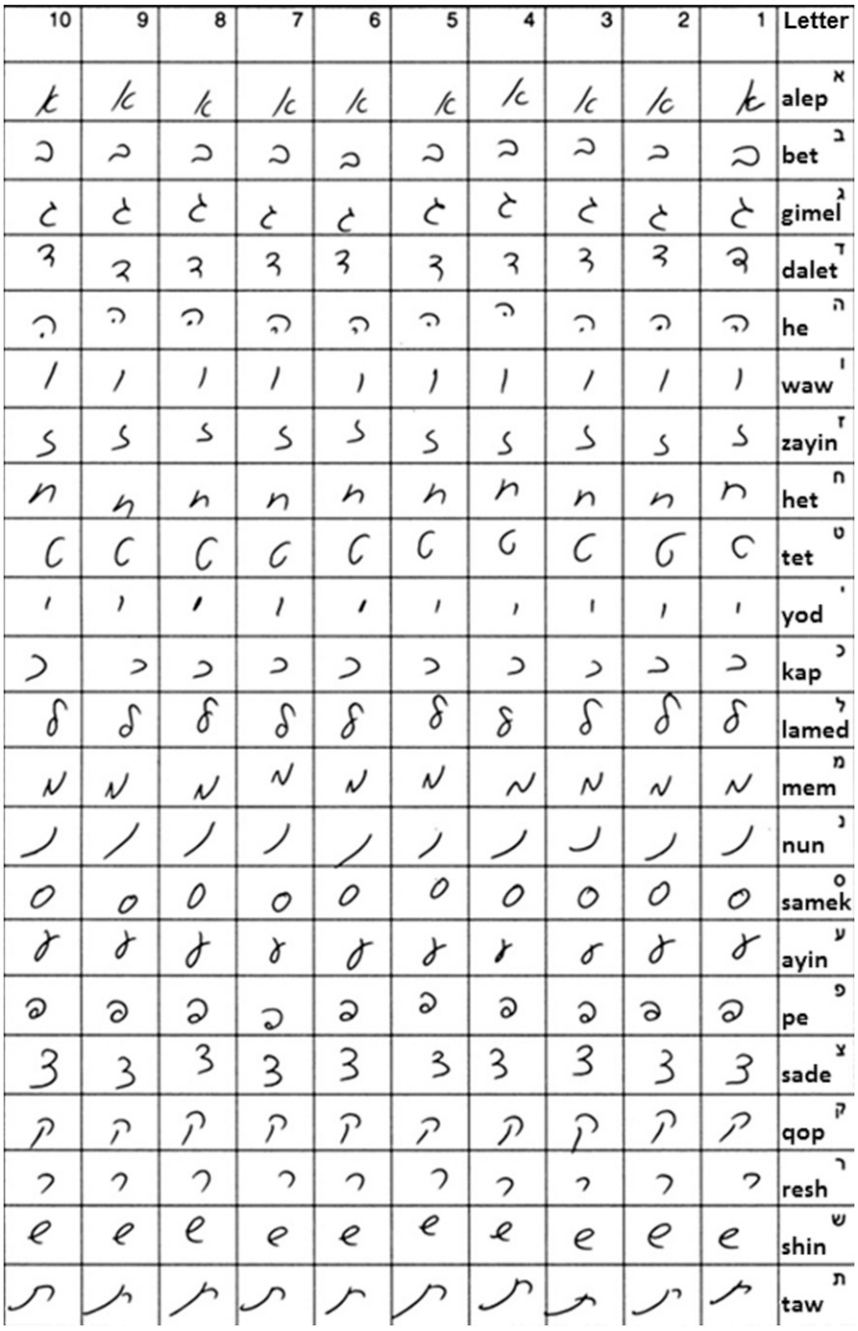

Fig. S4. An example of a Modern Hebrew alphabet table, produced by a single writer (with 10 samples of each letter).

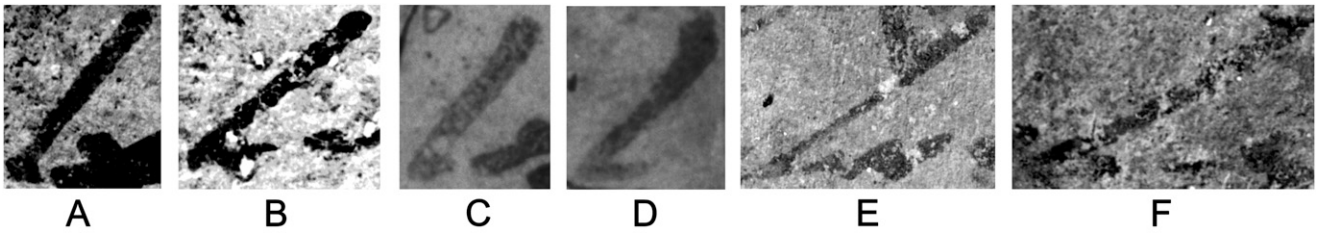

Fig. S5. Comparison between several specimens of the letter *lamed*, stemming from Arad 1 (*A* and *B*), Arad 7 (*C* and *D*), and Arad 18 (*E* and *F*). Note that our algorithm cannot distinguish between the author of Arad 1 and the author of Arad 7, or the authors of Arad 1 and Arad 18. However, Arad 7 and Arad 18 were probably written by different authors ($P = 0.015$ for the letter *lamed* and $P = 0.004$ for the whole inscription, combining information from different letters). Images are courtesy of the Institute of Archaeology, Tel Aviv University, and of the Israel Antiquities Authority.

Table S1. Features and distances used in our algorithm

Feature (ref.)	Feature implementation details	Distance implementation details
SIFT (28)	For each character j, we use the normalized SIFT descriptors $\vec{d}_i \in \mathbb{R}^{128}$ (with $\|\vec{d}_i\|_2 = 1$) and the spatial locators $\vec{l}_i \in [1, a_L]^2$ for at most 40 significant key points $k_i = (\vec{d}_i, \vec{l}_i)$, according to the original SIFT implementation. The resulting feature is a set $f_j^{SIFT} = \{k_i\}_{i=1}^{40}$.	The distance between f_1^{SIFT} and f_2^{SIFT} is determined as follows: *i*) For each key point $k_i^1 \in f_1^{SIFT}$, find a matching key point $m_i^2 \in f_2^{SIFT}$ s. t. $m_i^2 = \underset{(d_j^2, l_j^2) \in f_2^{SIFT}}{\arg\min}\, dist(k_i^1, k_j^2)$; where $dist(k_i^1, k_j^2) = \arccos(\langle d_i^1, d_j^2 \rangle) \cdot \|l_i^1 - l_j^2\|_2^2$. Thus, our definition augments the original SIFT distance by adding spatial information. *ii*) The one-sided distance is $D_{SIFT}^{1,2} = \underset{i}{median}\{dist(k_i^1, m_i^2)\}$. *iii*) The final distance is $D_{SIFT}(1,2) = \frac{D_{SIFT}^{1,2} + D_{SIFT}^{2,1}}{2}$.
Zernike (29)	An off-the-shelf (39) implementation was used. Zernike moments up to the fifth order were calculated.	$D_{Zernike}$ is the L_1 distance between the Zernike feature vectors.
DCT	MATLAB (R2009a) default implementation was used.	D_{DCT} is the L_1 distance between the DCT feature vectors.
K_d-tree (30)	An off-the-shelf (40) implementation was used. Both orders of partitioning are used (first height, then width, and vice versa)	$D_{Kd-tree}$ is the L_1 distance between the K_d-tree feature vectors.
Image projections (31)	The implementation results in cumulative distribution functions of the histogram on both axes.	D_{Proj} is the L_1 distance between the projections' feature vectors; this is similar to the Cramér–von Mises criterion (which uses L_2 distance).
L1	Existing character binarizations.	D_{L1} is the L_1 distance between the character images.
CMI (32)	Existing character binarizations, with values in $\{0,1\}$.	The CMI computes a difference between the averages of the foreground and the background pixels of $\Im$, marked by a binary mask M, $CMI(M, \Im) = \mu_1 - \mu_0$, where $\mu_k = mean\{\Im(p,q) | M(p,q) = k\}$ $k = 0,1$ In our case, given character binarizations B_1, B_2, the one-sided distance is $D_{CMI}^{1,2} = 1 - CMI(B_1, B_2)$. The final distance is $D_{CMI}(1,2) = \frac{D_{CMI}^{1,2} + D_{CMI}^{2,1}}{2}$.

Table S2. Results of the Modern Hebrew experiment

Group of letters (corresponding to g-index of simulated inscriptions)	False positive (FP out of all same-writer comparisons)	False negative (FN out of all different-writer comparisons)	False positive, % (FP out of all same-writer comparisons)	False negative, % (FN out of all different-writer comparisons)
Gimel, het, resh	0/18	8/612	0	1.31
Bet, samek, shin	1/18	5/612	5.56	0.82
Dalet, zayin, ayin	1/18	18/612	5.56	2.94
Tet, lamed, mem	0/18	22/612	0	3.59
Nun, sade, taw	0/18	3/612	0	0.49
He, pe, qop	0/18	16/612	0	2.61
Alep, waw, kap	1/18	11/612	5.56	1.80
Total	3/126	83/4,284	2.38	1.94

The percentages of false-positive and false-negative errors are about 2% each.

Table S3. Letter statistics for each text under comparison

	Alphabet letters						
Text	Alep	He	Waw	Yod	Lamed	Shin	Taw
1	4	5	3	7	3	3	8
2	6	3	3	5	3	1	7
3	2	4	5	4	4	3	3
5	5	3	1	3	4	2	4
7	1	2	1	4	6	8	5
8	2	1	2	1	4	4	2
16	6	3	9	5	10	3	2
17a	2	4	2	2	2	1	2
17b		1		2	1	1	2
18	2	4	4	5	6	6	3
21	5	4	6	6	12	5	2
24	9	10	5	8	4	4	7
31	3	7	6	4	1	1	
38	1	1	2	2	2	1	
39a	3	3	3	5	2	1	1
39b	3	1	1	4	1		
40	4	5	3	4		3	2
111	4	3	3	3	1	3	2