Code Finder: Scores or Probabilities

Robert M. Haralick
Computer Science, Graduate Center
City University of New York
365 Fifth Avenue
New York, NY 10016
haralick@netscape.net

1 Introduction

In this note we analyze the score calculation done in the Code Finder program for determining its relation to the probability that a cluster of ELSs in a table of given size would happen by chance.

Our first order of business is to say what probability means. For us probability must always be associated with an experiment. The experiment begins with an experimental protocol that has

1. *a priori* specification of the key words
2. a monkey text population
3. an ELS skip search specification
4. a resonance cylinder size specification
5. a procedure by which a compactness score value for a text can be computed

In the experiment, a text is randomly sampled from a specified population of texts, called the monkey text population to indicate that whatever effect is thought to be occuring with the Torah text it is certainly not occuring in the texts of the monkey text population. In accordance with a given experimental protocol, a table of ELSs is constructed and a score C measuring the compactness of the table is computed. The probability we are interested in is the probability that a randomly sampled text from the population will have a table whose compactness score C is smaller than or equal to C_0: $Prob(C \leq C_0)$, where C_0 is the compactness score observed in the Torah text.

This probability can be determined two ways. One way is to determine the probability analytically exactly analogous to the kind of analytic computation of drawing five cards at random in a poker game and having a full house. In the case of small combinatorial situations, this way is tractable. In the general situation, particularly with regard to Torah code tables, it is difficult if not tractable.

The second way is to estimate the probability by a Monte Carlo experiment. In the Monte Carlo experiment, a given number, N, of texts are sampled from the specified monkey text population. For each of the randomly sampled texts, a table of ELSs is constructed in accordance with the experimental protocol and the compactness values $C_1, \ldots, C_N$ of the tables from the N texts is computed. Then $Prob(C \leq C_0)$ is estimated by

$$Prob(C \leq C_0) = \frac{\#\{n \in \{0, 1, \ldots, N\} \mid C_n < C_0\}}{N} \tag{1}$$

the fraction of texts whose compactness value is less than or equal to that observed in the Torah text. Hence the probability $Prob(C \leq C_0)$ means the probability that a randomly sampled text from the text population will have a table that is as compact or more compact than the table obtained from the Torah text.

There are two conditions that must be satisfied by the Monte Carlo experiment in order for the fraction computed in equation (1) to have the meaning of being a probability. The first is that the entire experiment protocol must be specified in advance before the experiment is done, thereby guaranteeing that no information gained from the Torah text itself is used to define any part of the protocol.

The second is the condition of symmetry or uniformity. Simply stated it is that whatever procedure is carried out on the Torah text to establish a compactness value C_0 must be done exactly the same way to compute a compactness value C from each text sampled from the monkey text population.

The Code Finder program attempts an analytic calculation. We will go through the details of this calculation explaining exactly what probability the program intends to compute and how that differs from the correct probability it ought to be computing. We will show that the Code Finder score calculation produces a number that can be biased orders of magnitude too small compared to the probability that the Monte Carlo experiment would

produce. Recall that it is the Monte Carlo experiment that defines what the probability means.

2 The Code Finder Calculation

The Code Finder calculation is based on a monkty text population of random letter permuted texts. The program lets the user specify a Torah text such as Genesis, or the Five Books, or any one book of the Tanach, or the entire Tanach. The program then lets the user set a fixed minimum and maximum skip specification, for each key word. The user provides a list of key words. Then the program finds all the ELSs satisfying the skip specification of the given list of key words in the specified text. Next the user does some interactive manipulation attempting to construct in some undefined sense the smallest table having at least one ELS of the key words. In terms of a completely specified algorithmic procedure, this constitutes a weak link. But it is in fact not the cause of the difficulty of the Code Finder calculation.

Once the user has constructed a table, the user has a list of the ELSs that the table contains. Associated with each ELS is its absolute skip. The Torah code effect has been hypothesized to occur at the smaller ranked skip lengths and therefore, the compactness score function should put more weight on those ELS with relatively smaller skips. Let us see how Code Finder does this.

In a text of length Z the number N of possible placements an ELS of a non-symmetric key word of length L can have with absolute skip lengths of 1 through skip length D, is given by

$$N(Z, L, D) \;=\; D * (2Z - (L-1) * (D+1))$$

This is the same calculation done in the 1994 Witztum, Rips, and Rosenberg paper. See the appendix for the details.

In the letter permuted text population, the ELS placement probability for ELSs of a word whose letters are $< \alpha_1, \ldots, \alpha_K >$ is given by

$$p \;=\; \prod_{k=1}^{K} p(\alpha_k)$$

where $P(\alpha_k)$ is the probability that letter α_k occurs in the text. Hence, the expected number of ELSs in a randomly sampled text from the letter permuted text population is $E = pN$.

In producing what Code Finder interprets as the odds of the table being one that could have happened by chance, the expected number that the Code Finder program provides is not based on the maximum skip specification of the search, but on the absolute skips of the ELSs occurring in the Torah code table. As we will discuss later, this makes the interpretation of the score that Code Finder associates with a table, to be one that has no proper interpretation as the odds that the table could have arisen by chance.

We continue on with the Code Finder calculation. Code Finder sets D to the absolute value of the ELS skip. The Code Finder program converts this ELS expected number to an R-value defined by

$$R = log(1/E) \tag{2}$$

This is the R-value as originally defined by Dr. Alex Rotenberg in his SofSof Torah code program. Thus, ELSs having small expected number, small being less than one, will have R-values larger than 1. The R-value was defined by Dr. Alex Rotenberg for the purpose of associating a score with an ELS that would be a measure of the degree to which the ELS was a small skip ELS.

Let us consider an example to illustrate these calculations. We will use three key words, the first key word being the axis key word. The key words are Mega Terror, Prayer, and Mega Attack. Searching for ELSs with a maximum absolute skip of 30,000 and interactively finding the smallest area table we obtain the table shown in Figure 1.

Mega Terror	**מגה טרור**
Prayer	**תפילה**
Meta Attack	**פגוע מגא**

Figure 1: **The cylinder size is 27083 and this is the smallest area table having an ELS of each of the key words.**

Keyword	Skip	p	N	E	R
מנהטרור	−27083	1.098312×10^{-10}	1.00778×10^{10}	1.01686	−.04409
תפילה	−6	6.255486×10^{-7}	3.04787×10^{6}	2.28794	−.035944
פנועמנא	5	2.037707×10^{-11}	4.266934×10^{6}	6.210666×10^{-5}	4.20686

Figure 2: **Shows the intermediate calculations for the R-value. p is the probability that the characters of an ELS will match the characters of a text in a random placement. N is the number of placements of an ELS of the given skip. $E = pN$ is the expected number of times that in a random letter permuted text a random placement of an ELS of with absolute skip less than the absolute skip of the ELS found in the Torah text, will match the characters of the text in the placement. R is the Rotenberg R-value, which is the log of $1/E$.**

To understand the table of Figure 2, we consider the following experiment. We construct a population of texts, each text being a letter permutation of the Torah text. This means that we repeatedly take the Torah text and randomly shuffle its letters, adding the randomly shuffled Torah text to the monkey text population. We do this for a very large number of times. Then we randomly choose a monkey text from the population. We take the absolute skip 27083 ELS of מנהטרור and successively place it in every possible position we can and then check whether each of the letters of the ELS match the letters of text over which they are positioned. We count the number of times all the letters match. Then we repeat the experiment randomly sampling another text from the monkey text population and obtain a count of the number of times all the letters of the ELS match among all the possible placements of the ELS. We do this for a large number of times, say, M times. If we were to take the arithmetic average of all the counts we obtained, as M larger and larger, we would find that the average would get closer and closer to the expected number $E = 1.01686$ for the absolute skip 27083 skip ELS of מנהטרור.

We can then ask, what is the probability that if were to randomly sample one text from the monkey text population we would find at least one placement that the ELS would match among all the N possible placements. This probability is

$$P(\text{At least one placement matches}) = 1 - (1 - p)^N$$

When $pN < 1$, this probability is approximately $pN = E$ Often the Poisson probability distribution is used as an approximation because the

Poisson approximation is valid regardless of the value of pN.

$$P_{poisson}(\text{k placements match}) = \frac{E^k e^{-E}}{k!}$$

And from the Poisson approximation we can compute the probability that there will be at least one match. The probability of at least one match is 1 minus the probability of no matches. Hence,

$$P(\text{At least one placement matches}) = 1 - \frac{E^0 e^{-E}}{0!} = 1 - e^{-E}$$

Using the Poisson approximation we can then develop a table that tells us for each ELS, the probability that there will be at least one placement of the ELS in which characters of the ELS match the corresponding positions in a randomly sampled text.

Key Word	**Skip**	**E**	**P**
מגהטרור	-27083	1.01686	.63827
תפילה	-6	2.28794	.89852
פגועמנא	5	6.210666×10^{-5}	6.2105×10^{-5}

Figure 3: **Shows for each ELS the expected number E of times a placement of the ELS will match a randomly sampled monkey text and the probability P that in a randomly sampled monkey text at least one placement of the ELS will match the text.**

Now we assume that the text has a large number of characters and the key words of the ELSs have a very small number of characters. In this case there is no interference of the random placements of the ELSs, meaning that if we were to randomly place each ELS, the probability that one text character position would be covered by some letter position from one ELS and some other letter position from another ELS is 0. In this case, we can compute the probability that in a randomly sampled monkey text, we will obtain at least one placement of each ELS that will match the text characters.

Here we must carefully state that an ELS of **מגהטרור** means one whose absolute skip is no more than 27083, an ELS of **תפילה** means one whose absolute skip is no more than 6 and an ELS of **פגועמנא** means one whose

absolute skip is no more than 5. Because of independence, this probability is the product of the individual ELS probabilities. In our example case we have,

P(Each ELS has at least one placement that matches)

$$\begin{aligned} &= .63827 \times .89852 \times 6.210666 \times 10^{-5} \\ &= 3.5618 \times 10^{-5} \end{aligned}$$

So it seems that this event of each ELS having at least one placement that matches is a rare event. However, this rare event does not correspond to the experiment done. The experiment done was searching for ELSs of the three key words in skips ranging from -30,000 to 30,000. It was this search that produced the ELSs among which were the ones we found in a configuration of what appears to be a compact table. Clearly, the probability associated with this experiment is larger than the one Code Finder computed: P(Each ELS has at least one placement that matches) $= 3.5618 \times 10^{-5}$. Let us see how much larger. Examine the calculations of Figure 4.

Key Word	Max Abs. Skip	p	N	E	P
מגהטרור	30000	1.098312×10^{-10}	1.28812×10^{10}	1.41557	.75721
תפילה	30000	6.255486×10^{-7}	1.46882×10^{10}	9188.17	1.00000
פגועמנא	30000	2.037707×10^{-11}	1.28812×10^{10}	.26248	.23086

Figure 4: Shows the intermediate calculations for the Poisson probabilities under the condition of the actual search done. This was a search for ELSs having absolute skip of $30,000$ or less. p is the probability that the characters of an ELS will match the characters of a text in a random placement. N is the number of possible placements of an ELSs whose absolute skip is $30,000$ or less. $E = pN$ is the expected number of times that in a random letter permuted text a random placement of an ELS of a skip $30,000$ or less will match the characters of the text in the placement.

Now we can compute the probability that the search of the experiment performed would produce at least one ELS for each key word in a randomly sampled monkey text. P(At least one ELS for each key word) $= .75721 \times 1.000 \times .23086 = .17481$, a little less than one out of five, hardly something that could be called rare.

In other words, it is expected that in a search for ELSs whose absolute skips is no more than 30000, that we expect to find one ELS for each of our three key words in a randomly sampled monkey text in a little less than one out five times. The difference between this probability and the one Code Finder calculates is about four order of magnitude. If there were more key words, this would tend to make the orders of magnitude difference even larger.

Of course the question is not that there are ELSs of each of the key words in a randomly sampled monkey text. The question is how unusual is it in the monkey text population that there are ELSs for each of the key words that can be found in as compact formation as we found them in the Torah text. To answer this question here is what the Code Finder program does.

Suppose that the interactively constructed table from the Torah text has an area A, the product of the number of rows and columns of the table. Further suppose that there are M key words and and the expected number of their ELSs are $E_1, \ldots, E_M$, expected number meaning expected number of ELSs whose absolute skip is no more than the absolute skip of the ELSs found in the interactively constructed table. The corresponding R-values are $R_1, \ldots, R_M$ Let us also suppose that each key word has at least one ELS present.[1] Code Finder then multiples each expected number by the fraction A/Z to obtain what might be called the area adjusted expected number E' of ELSs within the table area.[2] And the corresponding matrix R-values, here denoted by R', are computed from these expectations.

$$\begin{aligned} E' &= \frac{A}{Z}E \\ R' &= log(1/E') \\ &= log(\frac{1}{(A/Z)E} \\ &= log(1/(A/Z)) + log(1/E) \\ &= R + log(Z/A) \end{aligned}$$

Each matrix R' value can be seen to be the R-value plus the log of the length

[1]This supposition itself is problematic because what typically happens is that a key word with no ELSs in a table will just be thrown away, making the key word set not *a priori*. But this problem is a problem with the *a priori* specification of the key words and not a problem with probability calculation itself.

[2]It can be seen from (2) that if the length of the text is reduced to half, the expected number E' of ELSs is in fact not reduced to half so this calculation is not quite right itself.

of the text divided by the area of the table. In the case of our example, $Z = 304805$ and $A = 7$ rows $\times$ 52 columns $= 364$. This makes

$$log(Z/A) = log(304805/364) = log(837.38) = 2.9229$$

Figure 5 shows the resulting calculation for the R' matrix R-values.

Keyword	Skip	p	N	E	R	R′
מגהטרור	−27083	1.098312×10^{-10}	1.00778×10^{10}	1.01686	−.04409	2.8788
תפילה	−6	6.255486×10^{-7}	3.04787×10^{6}	2.28794	−.35944	2.5635
פגועמגא	5	2.037707×10^{-11}	4.266934×10^{6}	6.210666×10^{-5}	4.20686	7.1298

Figure 5: **Shows the intermediate calculations for the R-value and what Code Finder calls the Matrix R-value, here denoted by R'. p is the probability that the characters of a randomly placed ELS will match the characters of a text. N is the number of possible placements of an ELS with absolute skip no more than that observed of the absolute skips of the corresponding Torah ELSs in the Torah code table. $E = pN$ is the expected number of times that in a random letter permuted text a random placement of an ELS of with absolute skip less than the absolute skip of the ELS found in the Torah text, will match the characters of the text in the placement.**

To understand what Code Finder does with the R-matrix values, we must first understand that Code Finder differentiates between the key words. The first key word in the key word list is special and is called the axis key word. The axis key word is typically the main topic key word and it is the one the researcher thinks should be governing the cylinder size of the cylinder on which the code table resides. That is, the code table cylinder size will be close to a small multiple of the the absolute skip of the selected ELS of the first key word. For example, if the selected ELS of the first key word has an absolute skip of $27,083$, the cylinder size of the code table can be $27,083$, $13,541$ or $13,542$, $9,027$ or $9,028$, $6,770$ or $6,771$ allowing the selected ELS of the first key word to appear either vertically, one letter successively below the other in the code table, or, in this case, diagonally every other row, every third row, or every fourth row and so on.

When a table is constructed it has three attributes. The first is its location, meaning for example, the text position coinciding with the upper left hand corner of the table. The second is the number of row and columns of the table, often combined as the area of the table and the third is a score

incorporating in some way the closeness of the ELSs in the table to the axis ELS and the degree to which the ELSs are small skip ELSs.

The protocol that Professor Haralick uses on the torahcode.us website only incorporates the third criteria by controlling the maximum absolute skip used in the search for the ELSs. This maximum skip is set so that the number of expected ELSs for a key word is a user defined parameter set to say, 10 or 50 or 100. This automatically insures that all the ELSs in the table will be relatively small skip ELSs. Professor Haralick's protocol does not distinguish between two tables of the same area, one of which has ELSs whose absolute skips are smaller than in the other table. This is must certainly be considered a deficiency in the Haralick protocol.

Code Finder, on the other hand, creates a score that explicitly tries to take into account both the area of the table and the degree to which the ELSs in the table are small skip ELSs. It does this by identifying the place of the table by the place of the axis ELS. In this way, the Code Finder analytic calculation does not have to take into account all the possible places that the table might have formed. Therefore the matrix R-value for the ELS of the axis key word is not used in the table score formula.

Code Finder then takes into account the degree to which the ELSs in the table are small skip ELSs by summing up only the positive matrix R-values, ignoring the R-value of the ELS of the first key word which is the axis key word. The sum is what Code Finder calls the cumulative matrix R-value and Code Finder considers it as the log of the odds ratio that the table would have been produced by a text in the letter perturbed monkey text population. However, it must only be considered as a score for the table. Indeed it does not have an interpretation of being the odds ratio for any Torah code experiment involving the given key words and skip search protocol.

$$R_{cumulative\_matrix} = \sum_{\substack{m=2 \\ R'_m > 0}}^{M} R'_m$$

First let us understand how $R_{cumulative\_matrix}$ can be understood as the log of an odds ratio. Recall that the matrix R-values used in the sum defining $R_{cumulative\_matrix}$ are each defined as the $log(1/E)$ where E is the expected number of ELSs in the monkey text population having absolute skip no more than the absolute skip of the corresponding ELSs in the table. In the case that E is small, we have seen that it has the interpretation of

being a probability. Summing the matrix R-values produces a result that is the log of the product of these probabilities. So under the non-interference and independence assumptions for the placements of the ELSs, this product would have the meaning of the joint probability of observing these kinds of ELSs in a table from the monkey text population. From this perspective $10^{-R_{cumulative\_matrix}}$ would be this joint probability.

In more detail, assume that there are M key words and each one has one ELS in the table. Let E'_m be the matrix expected number associated with the m^{th} ELS. Then the summation of the matrix R-values is like taking the product of these matrix expected numbers.

$$\begin{aligned} Q &= \prod_{m=1}^{M} \frac{1}{E'_m} \\ &= 10^{-R_{cumulative\_matrix}} \end{aligned}$$

The case of interest is when each E'_m is very small, less than one. When E'_m is small, $E'_m << 1$, by the Poisson approximation to the binomial distribution, E'_m is the probability that at least one ELS of the absolute skip of the m^{th} ELS or smaller will appear in the table. Hence, the product of the expectations is the product of the probabilities that at least one ELS of the absolute skip of the ELS or less will occur in the table. Probabilities are multiplied when events are independent. So under the assumption that the occurrence of one key word having an ELS in the table is independent of the occurrence of another key word having an ELS in the table, the product $\prod_{m=1}^{M} E'_m$ is an approximation for the probability that each key word has at least one ELS in the table. From this we understand the Code Finder's Q is the odds ratio $1 : Q$ that each key word would have at least one ELS in the table which is given at a fixed place of the ELS of the first key word, where the absolute skips of the ELSs are less than or equal to the absolute skips of the ELSs actually found in the table.

3 Code Finder Bias

Now let us see what the problems are with the Code Finder probability interpretation of its score calculation and try to understand why $Q = 10^{-R_{cumulative\_matrix}}$ is not the probability that a table as good as the one constructed from the

Torah text would have been found in a text sampled from the monkey text population.

First we have already mentioned that the Code Finder calculation is biased in favor of the Torah text since the skips of the ELSs in the monkey texts are, by the analytic computation, limited to be less than or equal to the corresponding absolute skips of the ELSs found in the Torah text and occurring in the table constructed from the Torah text ELSs. This means, for example, that there could be monkey texts which have ELSs in a much smaller area table, but some with absolute skips higher than the corresponding ELSs in the Torah text and some with absolute skips lower than the corresponding ELSs in the Torah text. And these monkey text tables, which are better than that found in the Torah text are not counted as better.

Next, let us for the moment assume that the user has constructed the smallest area table containing at least one ELS of each of the key words. Some of the key words in the table may have more than one ELS. The Code Finder summing method gives extra reward when there is more than one ELS of a key word in the table. Some of the Torah code researchers have argued that this is important.

This creates a problem with the probability calculation itself. Suppose that a key word has only one ELS in the Torah code table and that its R-value is not greater than zero. Then, in effect, the assumed *a priori* word list has been changed based on information obtained from the search of the Torah text. And the effect of this change in the score calculation is to bias the score in favor of Torah text. The reason that the bias is toward the Torah text is that in Code Finder's analytic calculation, the very same procedure is, in effect, not calculated for each text of the text population as required by the symmetry or uniformity condition of the experiment that defines what the probability is.

Next suppose that there is some text that could have been sampled from the monkey text population and in the smallest area table that could be constructed from this monkey text, the table has 2 or even 3 relatively small skip ELSs having large R-values. The probability calculation specifically does not take into account this possibility, thereby biasing the score calculation to be smaller for the Torah text than for such a monkey text.

4 Reinhold's Protocol

Roy Reinhold, the originator of the Torah code website
http://ad2004.com/Biblecodes/index.html
is a Code Finder promotor. He has a protocol that modifies the Code Finder calculation, in effect, reducing the odds ratio Code Finder produces by applying what is known as the Bonferroni tax.

Mr. Reinhold reasons the following way. The user had to interactively construct the table. In this interaction the user had to go through and select from a potentially enormous number of combinations and try them out. Among these combinations, the user tries out different cylinder sizes, cylinder sizes related to the skip of the axis ELS divided by 2 or 3 etc.

Generally Mr. Reinhold's resonance specification is that the row skip of the axis ELS on the cylinder must be at least 1 and no more than 6. So in effect the user is examining one position and six cylinder sizes for each ELS of the first key word. Reinhold therefore penalizes the user with a Bonferroni tax for each ELS of the first key word and for each cylinder size tried. This Bonferroni tax on the odds ratio is the number of ELSs that the first key word had times 6, the number of resonant cylinder sizes that Reinhold would search. The adjusted odds ratio is then $1 : Q/B$ where B is the Bonferroni penalty.

This Bonferroni penalty is based on the number of ELSs of the first key word found in the Torah text. But the number of ELSs of the first key word found in the letter permuted monkey texts are not the same as that found in the Torah text. And so the calculation uses a quantity from the Torah text that is not applicable to each monkey text.

Had the user been required to make the window extend around the first ELS the same number of columns to the left and right and the same number of rows to the left and right, there would be no question that the Reinhold's Bonferroni tax is sufficient. But because this was not a requirement, the user had additional extension possibilities not accounted for and the Bonferroni tax is too low. Thus, the Q/B of the odds ratio $1 : Q/B$ is too high and the probability calculated is therefore too small, even with Reinhold's protocol modification.

5 Probability and Score

We have argued that the computation made by the Code Finder program must be considered a score and not a number that has a probability interpretation for any kind of real Torah code experiment. In this section we further amplify that argument.

Recall our initial description of the probability of interest. We are interested in the probability $Prob(C \leq C_0)$, where C_0 is the compactness score of the Torah text and C is a random variable of the compactness score resulting from the same procedure as done on the Torah text done on a randomly sampled text from the monkey text population. Code Finder claims that the C_0 it computes in fact is the probability $Prob(C \leq C_0)$. That is Code Finder claims

$$Prob(C \leq C_0) = C_0$$

This is the case when the probability density function for the random variable C is the uniform $U(0,1)$ and indeed this is the requirement for all p-values. If C is indeed a p-value, its density function will be uniform (0,1).

It is not that hard to check whether the score function of Code Finder indeed satisfies this requirement by a Monte Carlo experiment. Let C_0 be the Code Finder score it interprets as a probability for the Torah text. Let $C_1, \ldots C_N$ be the scores resulting from the same operation done on randomly samples monkey texts. For large N we should find that

$$\left| C_0 - \frac{|\{n \in \{0, 1, \ldots, N\} \mid C_n \leq C_0\}|}{N+1} \right| \to 0$$

Since the Code Finder program does not do Monte Carlo experiments, this condition could never be checked by Code Finder. The interactive user would get bored by the time even 10 experiments would be done. But if we had a program that could do Monte Carlo experiments, we can easily check this condition. Indeed we do have such a program and made the Monte Carlo experiment.

We set the maximum skip for each ELS so that the expected number of ELSs was 50. For each text, we find the smallest area table. We score the table using the Code Finder scoring method for the probability associated with the Code Finder odds ratio. Our monkey text population is the ELS random placment population. For our 3 keyword example, the Torah text had a Code Finder score of 2.0265×10^{-10} which in cumulative R-value terms

is about 9.693. In a run of $1,000,000$ trials, the p-value estimate produced by the Monte Carlo experiment using the Code Finder scoring methodology was 5/1,000,000, more than 4 orders of magnitude larger than the score interpreted as a probability as Code Finder would suggest it be interpreted.

Does applying Reinhold's Bonferroni tax fix this problem? Since there are 6 ELSs of מגהטרור and since Reinhold would allow an axis ELS to have a maximum row skip of 6, Reinhold would use a Bonferroni tax of $6 \times 6 = 36$. Thus the Bonferroni corrected score would be $2.0265 \times 10^{-10} \times 36 = 7.295 \times 10^{-9}$. The ratio of the Monte Carlo probability to the Bonferroni corrected score is now about 685, making the Bonferroni corrected Code Finder score between two and three orders of magnitude too small.

6 Concluding Discussion

We have illustrated an example table, indeed the first one we examined, for which the Code Finder score calculation is orders of magnitude too small to be interpreted as a probability. We have also explained the reasons why this is so. The main reason being that the score calculation uses the R-value associated with actual skip of the ELS in the table rather than the maximum skip searched for. It is interesting that the insight for wanting to use the actual skip of the ELS in the R-value measure is correct. The smaller the skip of the ELS, the better the ELS is in the sense being closer to the minimal skip ELS. Hence the smaller the skip of the ELS, the more significance it ought to have. But that argument is more an argument for what should be in a table scoring function than an argument for its use in a p-value probability calculation.

In summary, Code Finder produces a score that cannot be interpreted as the p-value probability that the constructed table could have arisen by chance. Indeed, Code Finder's score is, in general, orders of magnitude smaller than the probability that the table could have arisen by chance. Or saying this the other way, Code Finder's odds ratio is orders of magnitude larger than the odds ratio that the table indeed could have arisen by chance.

7 Appendix: Number of Placements

Let L be the length of the key word, Z be the length of the text, The span of a skip s, $s > 0$, ELS of a key word with L characters is $1 + (L-1)s$. The number N of placements such an ELS has in a text of length Z is

$$\begin{aligned} N &= Z - (1 + (L-1)s) + 1 \\ &= Z - (L-1)s \end{aligned}$$

Let S_{min} be the minimum absolute skip and S_{max} be the maximum absolute skip for the ELSs of this key word of length L. Then the total number N_T of placements the ELSs have of the key word is

$$\begin{aligned} N_T &= \sum_{s=S_{min}}^{S_{max}} Z - (L-1)s \\ &= (S_{max} - S_{min} + 1)Z - (L-1) \sum_{s=S_{min}}^{S_{max}} s \\ &= (S_{max} - S_{min} + 1)Z - (L-1) \left(\frac{S_{max}(S_{max}+1)}{2} - \frac{(S_{min}-1)S_{min}}{2} \right) \\ &= (S_{max} - S_{min} + 1)Z - \frac{L-1}{2} \left(S_{max}^2 + S_{max} - S_{min}^2 + S_{min} \right) \\ &= (S_{max} - S_{min} + 1)Z - \frac{L-1}{2} \left(S_{max}^2 - S_{min}^2 + S_{max} + S_{min}) \right) \\ &= (S_{max} - S_{min} + 1)Z - \frac{L-1}{2} ((S_{max} + S_{min})(S_{max} - S_{min}) + S_{max} + S_{min}) \\ &= (S_{max} - S_{min} + 1)Z - \frac{L-1}{2} ((S_{max} - S_{min} + 1)(S_{max} + S_{min})) \\ &= (S_{max} - S_{min} + 1) \left(Z - \frac{L-1}{2} (S_{max} + S_{min}) \right) \end{aligned}$$

If the key word is not symmetric, meaning spelled the same way forward and backward, and if the skips allowed are from $-S_{max}$ through $-S_{min}$ and from S_{min} through S_{max} then the total number of placements must be doubled. In this case,

$$N_T = (S_{max} - S_{min} + 1)[2Z - (L-1)(S_{max} + S_{min})]$$

Example
If $L = 7$, $S_{max} = 20783$, $S_{min} = 1$, and $Z = 304805$, then

$$\begin{aligned} N_T &= (20783 - 1 + 1)[2 * 304805 - (7 - 1)(20783 + 1)] \\ &= 20783[609610 - 6 * 20784] \\ &= 20783[609610 - 124704] \\ &= 20783[484906] \\ &= 1.00778 \times 10^{10} \end{aligned}$$

If the product of the character probabilities is $p = 1.098312 \times 10^{-10}$, then the expected number E of ELSs is

$$\begin{aligned} E &= pN_T \\ &= 1.098312 \times 10^{-10} \times 1.00778 \times 10^{10} \\ &= 1.10685687 \end{aligned}$$

The definition of the R-value is

$$R = log_{10}\frac{1}{E}$$

In our example

$$R = log_{10}\frac{1}{1.10685687} = -.04409$$