Frequently in science, two seemingly unrelated fields find common ground in a research problem of interest. For example, the fields of biology and coding theory share the same challenge, which is to answer the question of whether or not there is an error-correcting mechanism in DNA sequences similar to the one employed in digital transmission systems. There are several facts about DNA sequences which motivate this line of questioning. One is that DNA sequences may be viewed as “words” written using four letters or nucleotide bases. Another is that some DNA patches code for protein sequences. Furthermore, several DNA sites have been well annotated in terms of pattern and information content [1]. The evolution of these biologically significant sequences is usually evolutionarily conserved, and it is important to avoid sequence errors in order to maintain their function. Another interesting point is that the number of genes an organism has does not correlate with its complexity. In fact, the number of non-coding DNA (ncDNA) regions, including repetitive sequences, seems to have been increasing since the beginning of the evolution of the higher eukaryotes, which suggests that organism complexity is related to gene regulation through ncDNA [2]. It is well established that non-coding sequences are biologically important; e.g. regulatory regions (promoters, TFBS, enhancer elements, ncRNA, introns, splicing sites etc). Finally, and most importantly, the DNA replication process is far from being the only source of sequence errors. DNA integrity is frequently jeopardized by physical and chemical agents, which means that DNA damage repair mechanisms are indispensable in preventing collateral effects [3]. Interestingly, more than one of these mechanisms is described in the literature [4]. Is it reasonable to infer that some DNA repair mechanisms are a biological implementation of error-correcting codes?

The coding theory community has proposed several methodologies to verify whether or not a particular DNA sequence, usually a protein coding sequence, has an underlying error-correcting code (ECC) [5] and [6]. In spite of their relevance, the results of earlier works do not provide the definitive answer. For instance, based on the procedure for determining whether or not the lac operon and cytochrome c gene can be identified as codewords of linear block codes, the answer is no [7]. Actually, we cannot even conclude that there is no linear block code in other DNA sequences.

Of course, as is often the case, there is at least one alternative approach to solving this problem, which is to demonstrate that an ECC underlies DNA sequences. This task is far from easy to accomplish, because a complex error-correcting scheme might consist of many distinct concatenated codes, rather than a single global one, although, to the best of our knowledge, there is no evidence that such an ECC exists. In [8], we attempted to answer a recurring question: Are there DNA sequences that can be identified as codewords for ECCs? If so, we will have taken the first step in a long research journey. The majority of candidate DNA sequences have been positively identified as codewords for a class of cyclic block codes. Such codewords are consistently different from actual DNA sequences by one single nucleotide. Is this
difference biologically significant? Are these codewords actually ancient DNA sequences? Up to now, researchers in the fields of biology and coding theory have been working almost independently of one another, and the two groups need to work together to address the new challenges. In this paper, we ask whether or not a whole intron-exon gene structure can be identified as a codeword, and, furthermore, can a whole genome be identified as a codeword? In the following sections, we describe our experiments and results.

Methods

BCH Code

ECCs are always used when transmitting or storing information. The main objective of an ECC is, as the name suggests, to correct errors that might occur during information transmission through noisy channels. BCH codes form a subset of parameterized ECCs, which were first proposed in 1959 by Hocquenghem [9] and independently rediscovered by Bose and Chaudhuri [10] in 1960. The acronym BCH is made up of the initials of Bose, Chaudhuri, and Hocquenghem, in that order. Usually BCH codes are employed in the transmission of information in computer networks and in sequence generation. Due to the simplicity of their encoding and decoding processes, these codes are good candidates for use in the identification and reproduction of DNA sequences, [8], [11]–[19]. By “identification”, we mean that the DNA sequence may be either a codeword for an ECC or one of the code sequences. These code sequences may differ from the codeword up to the error correction capability of the code. In the latter case, we say that such code sequences belong to a codeword set. The BCH codes constitute an important generalization of the Hamming codes by allowing multiple error corrections. The parameters associated with a BCH code are denoted by \((n,k,d)\), where \(n\) is the codeword length (number of base pairs in DNA sequences); \(k\) is the code dimension (length of the input information sequence responsible for generating the DNA sequence); and \(d\) is the minimum code distance (the smallest number of positions by which any two codewords may differ).

Converting Nucleotides into Numbers

It is desirable that the alphabet of an ECC have an associated algebraic structure. Although the genetic code has an associated alphabet, the identification of a related algebraic structure remains an open problem. We have considered the ring of integers modulo 4, denoted by \(Z_4\), owing to the easy of code construction of using this algebraic structure. Since the alphabet of the genetic code differs from the algorithm in [20] in that it considers the Galois ring extension, the larger the number of polynomials \(p(x)\) with degree \(r\) to be used in the Galois ring extension, \(GR(4, r) = Z_4[x]/\langle p(x) \rangle\) (Step 2 and Step 3) and all labeling A, B and C (Step 4), owing to the as yet unknown interdependence of the geometric and algebraic structures in the code construction, where \(Z_4[x]\) denotes the ring of all the polynomials with coefficients in \(Z_4\), and \(\langle p(x) \rangle\) denotes the ideal generated by \(p(x)\).

In the code construction process, the DNA sequence generation algorithm takes into consideration three important facts. The first is to consider every possible value taken by the minimum distance \(d\) of the code, that is, \(d = 2t + 1\), where \(t\) denotes the number of errors the code is able to correct. The second is to consider all polynomials \((4, r)\) with degree \(r\) to be used in the Galois ring extension, \(GR(4, r) = Z_4[x]/\langle p(x) \rangle\) (Step 2 and Step 3) and all labeling A, B and C (Step 4), owing to the as yet unknown interdependence of the geometric and algebraic structures in the code construction, where \(Z_4[x]\) denotes the ring of all the polynomials with coefficients in \(Z_4\), and \(\langle p(x) \rangle\) denotes the ideal generated by \(p(x)\). The third is to consider determining the group of units \(G_4\) in \(GR(4, r)\), where \(n = 2^r - 1\) denotes the cardinality of \(G_4\) and \(GR(4, r)\) denotes the set of all non-zero elements in \(GR(4, r)\). The additional computational complexity in the solution of this problem comes from the fact that the greater the degree of the Galois ring extension, the larger the number of \(p(x)\) to be considered in the code construction.

Knowing that the number of codewords generated by these codes grows exponentially with the code dimension, instead of generating all the codewords and comparing them with the given DNA sequence, the twenty-four permutations are applied to that DNA sequence, and these sequences are considered as “possible codewords”. Then, to determine which of the twenty-four sequences are, in fact, codewords, the relation \(vH^T = 0\) is employed, where \(v\) is each of the possible codewords and \(H^T\) denotes the transpos of the parity-check matrix. The analysis to be performed with the DNA sequence, as a result of the one nucleotide difference from the codeword, is to consider the other three possible nucleotides at each position in the sequence for each permutation, and again to use the relation \(vH^T = 0\), in order to verify whether or not \(v\) is a possible codeword.

Single stranded DNA sequences, such as single stranded chromosomes, genes, introns, exons, repetitive DNA, and mRNA sequences, may be either a codeword for an ECC or belong to the codeword set of an ECC. In order to verify whether or not a DNA sequence may actually be identified as a codeword, we can use an ad hoc strategy, i.e. generate all the codewords and compare the DNA sequence with each codeword. However, this is not a practical strategy, because the computational effort to do this would be prohibitive, as explained below. In order to address this identification problem, we have developed an algorithm called the DNA Sequence Generation Algorithm, which verifies whether or not a given DNA sequence can be identified as a codeword of an ECC. This algorithm is the same as the one in [8], however it differs from the algorithm in [20] in that it considers the Galois ring extension as the algebraic structure, instead of the Galois field extension. There are also some conceptual differences, which are discussed in [13] and [17].

DNA Sequence Generation Algorithm

Input data: 1) \(seq\) = original DNA sequence in nucleotides \(\langle NCBi \rangle\); 2) \(n = 2^r - 1\); and 3) \(d_{HT} = 2r + 1\).

- **Step 1** - Generate all primitive polynomials \(p(x)\) with degree \(r\) to be used in the Galois ring extensions;
- **Step 2** - Select one \(p(x)\) from **Step 1**, and find the set in which the elements have the inverse, the group of units of \(GR(4, r)\), denoted by \(GR(4, r)\);
- **Step 3** - Find the generator and parity-check polynomials of the BCH code by knowing the minimum distance and the primitive polynomial derived in **Step 2**. In this way, the
generator, as well as the parity-check matrices and its transposes, are determined;

- **Step 4** - From the mapping $N \rightarrow \mathbb{Z}_4$, convert the seq with elements in $N$ into the corresponding sequence with elements in $\mathbb{Z}_4$;

- **Step 5** - Verify by use of the syndrome $s = (v \cdot H^T)$, whether or not each of the converted DNA sequences is a codeword:
  - If $s = 0$, then store the sequence;
  - If $s \neq 0$ implies that up to $t$ nucleotide differences may exist. If so, then the $n$ combinations $t$ to $t$ must be considered by taking into account the other three nucleotide possibilities in each of the combinations of the DNA sequence. Verify that every combination is a codeword: if so, store it; otherwise disregard it;

- **Step 6** - From the mapping $\mathbb{Z}_4 \rightarrow N$ convert each stored sequence in **Step 5** with elements in $\mathbb{Z}_4$ into the corresponding sequence with elements in $N$. Compare each of these sequences with the seq and show the position at which the nucleotides differ;

- **Step 7** - Go to **Step 1.** Select another $p(x)$ and verify whether or not all the $p(x)$ have already been used: if not, repeat **Steps 2 to 6** for each $p(x)$ from **Step 1**; otherwise, go to **Step 8**.

- **Step 8** - End.

### Results and Discussion

We have successfully applied this algorithm to the TRAV7 gene sequence and the plasmid *Lactococcus lactis* genome sequence. These sequences are represented in Table 1 and Table 2 using the following abbreviations: Ont = original nucleotide; Olb = original labeling; Glb = generated labeling and Gnt = generated nucleotide. Although we have used all the $p(x)$, all the corresponding $g(x)$, and all the possible minimum code distances in the construction of the BCH code over $GR(4, r)$, the results show that only codes with the minimum distance $d = 3$ associated with a specific $g(x)$, which in turn is associated with its $p(x)$ and labeling, are able to identify the TRAV7 gene and the plasmid genome sequences. Consequently, the algebraic structure, alphabet, labeling, $p(x)$, and $g(x)$ have to be considered in the construction of BCH codes over rings.

The fact that a DNA sequence is identified as a sequence belonging to a codeword set of a BCH code with the minimum distance $d = 3$ (and no other minimum distance) implies that this $(n, k, 3)$ BCH code is equivalent to the Hamming code with parameters $(2^{r-1}, 2^r-1-r, 3)$, independently of the algebraic structure associated with the alphabet of the code. Therefore, the Hamming codes constructed by considering the group of units $G_n$ in $GR(4, r)$ are able to identify and reproduce the DNA sequences that differ by one nucleotide from the posted NCBI sequences. We have also noted that the labeling, which is the set consisting of the twenty-four permutations, is split into three subsets, each of which contains eight permutations and defines a labeling denoted by $A$, $B$, and $C$ - Figure 1.

The TRAV7 predicted gene has 511 nucleotides, and therefore the codeword length is $n = 511$ - Table 1. Using the equality $n = 2^r - 1$, it is easy to calculate the degree $r$ of the Galois ring extension, which is 9. The number of $p(x)$ for this extension is $48 \ [11], \ [12]$. Among these, just one $p(x)$ is associated with a $g(x)$ of the Hamming code $(511, 502, 3)$, that is,

---

**Figure 1.** Permutations associated with labelings $A$, $B$ and $C$.
doi:10.1371/journal.pone.0036644.g001
\[ p(x) = x^9 + x^8 + x^5 + x^4 + 1 \]

and

\[ g(x) = x^9 + 3x^8 + 2x^7 + 2x^6 + x^5 + x^4 + 2x^2 + 3. \]

Furthermore, this identification was made using the C labeling.

A statistical analysis related to the \textit{TRAV7} gene sequence chromosome 14 of the human genome is as follows: with each primitive polynomial there is a corresponding generator polynomial of a code. For the given DNA sequence we use the 24 labeling and the resulting 24 sequences are multiplied by the generator matrix. From this operation results 24 codewords. Each one of these codewords is multiplied by the parity-check matrix. If the result is zero then the given DNA sequence is a codeword. Otherwise, we have to verify what happens if in each position we have different nucleotides. To do that, we have to realize three substitutions in each position of the original DNA sequence and verify again if this modified sequence is or is not a codeword. Since the \textit{TRAV7} gene genomic sequence has \( n \approx 2^9 - 1 = 511 \), it follows that \( r = 9 \). From this, the degree of the primitive polynomial is 9 and as a result we have 48 different primitive polynomials. Since for each one of them we have to use the 24 labeling, this leads to 1152 codewords to verify for a given error-correcting capability. Since in this case we have 256 possibilities, an upperbound is 294,912 codewords to be tested. Now, since there is always one nucleotide difference, we have to realize three times 63 tests for each one of the 294,912 codewords. Therefore, yielding a total of

\begin{table}
\centering
\caption{\textit{TRAV7} gene sequence chromosome 14.}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
1 & Ont: & atggagaaga & tgcggagacc & tgtcttaatt & atattttgtc & tattttcgg & ctgaagttg \\
& Obb: & 0311010010 & 3121101022 & 3132230033 & 0303333132 & 3031323311 & 2313001331 \\
& Gbb: & 0311010010 & 3121101022 & 3132230033 & 0303333132 & 3031323311 & 2313001331 \\
& Gnt: & atggagaaga & tgcggagacc & tgtcttaatt & atattttgtc & tattttcgg & ctgaagttg \\
\hline
61 & Ont: & aggttctaa & gaactgggga & cccaggaga & catttacca & agttcttttg & gggagatggg \\
& Obb: & 0111332300 & 1002311110 & 2220011010 & 2033303230 & 0132233331 & 1110103111 \\
& Gbb: & 0111332300 & 1002311110 & 2220011010 & 2033303230 & 0132233331 & 1110103111 \\
& Gnt: & aggttctaa & gaactgggga & cccaggaga & catttacca & agttcttttg & gggagatggg \\
\hline
121 & Ont: & gatgtagctt & ggactacttt & gtctattctt & gttgtggatt & aagaaattaa & atttataagg \\
& Obb: & 1031301323 & 1102320333 & 1320331233 & 1333101033 & 0010003000 & 0330310001 \\
& Gbb: & 1131301323 & 1102320333 & 1320331233 & 1333101033 & 0010003000 & 0330310001 \\
& Gnt: & gatgtagctt & ggactacttt & gtctattctt & gttgtggatt & aagaaattaa & atttataagg \\
\hline
181 & Ont: & gtctaaatta & aatgtcatct & atgttgactt & atgttcttct & gtagaagggc & aaatgagaa \\
& Obb: & 1323003033 & 0003132033 & 0331322331 & 0313233232 & 1003110112 & 0003110100 \\
& Gbb: & 1323003033 & 0003132033 & 0331322331 & 0313233232 & 1003110112 & 0003110100 \\
& Gnt: & gtctaaatta & aatgtcatct & atgttgactt & atgttcttct & gtagaagggc & aaatgagaa \\
\hline
241 & Ont: & aaccagttgg & agcagccgcc & tcattttcctg & gggaccccacc & agggacagct & tggctccagt \\
& Obb: & 0022011311 & 0120201222 & 3203332331 & 1102220212 & 0111010213 & 3122323231 \\
& Gbb: & 0022011311 & 0120201222 & 3203332331 & 1102220212 & 0111010213 & 3122323231 \\
& Gnt: & aaccagttgg & agcagccgcc & tcattttcctg & gggaccccacc & agggacagct & tggctccagt \\
\hline
301 & Ont: & agtgcagct & acctctctcag & tctgtttaac & aatgttgact & gttgtcaggct & aatcaggg \\
& Obb: & 0123210213 & 0232312031 & 3213333202 & 0033312013 & 1132021112 & 0030201111 \\
& Gbb: & 0123210213 & 0232312031 & 3213333202 & 0033312013 & 1132021112 & 0030201111 \\
& Gnt: & agtgcagct & acctctctcag & tctgtttaac & aatgttgact & gttgtcaggct & aatcaggg \\
\hline
361 & Ont: & atgggtcca & aaccactatct & atgttgatcag & tctctgtgatc & atggagaaca & gaagagaa \\
& Obb: & 0311132220 & 0022023033 & 0322031303 & 3201231103 & 0310100120 & 0001001100 \\
& Gbb: & 0311132220 & 0022023033 & 0322031303 & 3201231103 & 0310100120 & 0001001100 \\
& Gnt: & atgggtcca & aaccactatct & atgttgatcag & tctctgtgatc & atggagaaca & gaagagaa \\
\hline
421 & Ont: & ctaaatgcta & cattactga & gaaatggaagc & agctctgata & ttacagcgct & gacgcctgaa \\
& Obb: & 2300031320 & 2033023233 & 1003110112 & 0123310320 & 3302012211 & 1201223100 \\
& Gbb: & 2300031320 & 2033023233 & 1003110112 & 0123310320 & 3302012211 & 1201223100 \\
& Gnt: & ctaaatgcta & cattactga & gaaatggaagc & agctctgata & ttacagcgct & gacgcctgaa \\
\hline
481 & Ont: & gattcagcca & ccttacctctg & tgtctgtgatc & g \\
& Obb: & 1033201220 & 2230332331 & 3123130103 & 1 \\
& Gbb: & 1033201220 & 2230332331 & 3123130103 & 1 \\
& Gnt: & gattcagcca & ccttacctctg & tgtctgtgatc & g \\
\hline
\end{tabular}
\end{table}
Table 2. *Lactococcus lactis* plasmid genomic sequence.

| 1  | Ont: cctacatttt ccttattgctc tgcctatgatt gtttatcgat agttttttat acagataagc | Olb: 1130103333 333032131 3213032033 2333031203 0233333030 0102030021 |
|----|--------------------------------------------------------------------------|----------------------------------------------------------------------------|
|    | Glb: cctacatttt ccttattgctc tgcctatgatt gtttatcgat agttttttat acagataagc | Glb: 1130103333 333032131 3213032033 2333031203 0233333030 0102030021 |
| 61 | Ont: gtgcgacgct gtgcctttcc gaggaggaag tcatgctgac aagcacggca gagcctccgc | Olb: 2321201213 3213331311 2022022002 3103213201 0021012210 2021131121 |
|    | Glb: gtgcgacgct gtgcctttcc gaggaggaag tcatgctgac aagcacggca gagcctccgc | Glb: 2321201213 3213331311 2022022002 3103213201 0021012210 2021131121 |
| 121| Ont: atgaaatgct ctcaatgaaa ttgccggcgg agcttttttg agcttgtgcc acttgcgaaa | Olb: 0320003213 1300320000 3321122212 0213333320 0213323211 0133212000 |
|    | Glb: atgaaatgct ctcaatgaaa ttgccggcgg agcttttttg agcttgtgcc acttgcgaaa | Glb: 0320003213 1300320000 3321122212 0213333320 0213323211 0133212000 |
| 181| Ont: aaacaagaa caaaagagac aggaaactgt ctttttttgc ttgcttgggg attggggcaa | Olb: 0000100200 1000020200 022001323 1333333321 3233222122 0332221120 |
|    | Glb: aaacaagaa caaaagagac aggaaactgt ctttttttgc ttgcttgggg attggggcaa | Glb: 0000100200 1000020200 022001323 1333333321 3233222122 0332221120 |
| 241| Ont: cgccccaaaa ataaaaagaa tcgtctgaaa cgaggaacaa actaaaatgt aaattttagt | Olb: 2321001213 0300000200 3213331311 2022022002 0103000023 0033303323 |
|    | Glb: cgccccaaaa ataaaaagaa tcgtctgaaa cgaggaacaa actaaaatgt aaattttagt | Glb: 2321001213 0300000200 3213331311 2022022002 0103000023 0033303323 |
| 301| Ont: tgttaccgag tggaagatga atacttttta acctatgtgt atacacacat agtaagctcg | Olb: 3233011202 3220020320 0301333330 0113032303 0301010103 0230021312 |
|    | Glb: tgttaccgag tggaagatga atacttttta acctatgtgt atacacacat agtaagctcg | Glb: 3233011202 3220020320 0301333330 0113032303 0301010103 0230021312 |
| 361| Ont: ctataatact ttataacgtt tttatttaca tgagcaaagc gagtttttcc aacacgttta | Olb: 1303003310 330301233 3330333010 3201200021 2023333311 0010231330 |
|    | Glb: ctataatact ttataacgtt tttatttaca tgagcaaagc gagtttttcc aacacgttta | Glb: 1303003310 330301233 3330333010 3201200021 2023333311 0010231330 |
| 421| Ont: atctaaaata ttggcaattt ataccatgat tttcatggta tgtaagtgcg cccttaggaa | Olb: 0313000030 3322100333 0301103203 3333103230 3230023212 1133022000 |
|    | Glb: atctaaaata ttggcaattt ataccatgat tttcatggta tgtaagtgcg cccttaggaa | Glb: 0313000030 3322100333 0301103203 3333103230 3230023212 1133022000 |
| 481| Ont: aataatttga atatatttca gattttcaat ctgactgctc ctgtcatcga gcagaccgat | Olb: 0030003320 3030303310 2033313003 1201321321 1323103120 2102011203 |
|    | Glb: aataatttga atatatttca gattttcaat ctgactgctc ctgtcatcga gcagaccgat | Glb: 0030003320 3030303310 2033313003 1201321321 1323103120 2102011203 |
| 541| Ont: gaggaacaaca aaaaagggcct taacaaaaaa aagttagtct tcctttgttct tcgaagtgtt | Olb: 1302001231 3033332310 3333021000 3333021300 1302012222 0333303313 |
|    | Glb: gaggaacaaca aaaaagggcct taacaaaaaa aagttagtct tcctttgttct tcgaagtgtt | Glb: 1302001231 3033332310 3333021000 3333021300 1302012222 0333303313 |
| 601| Ont: ctgaacgcggct attttttgga ttttaagcga tttgttttc gattgggggg attttttattt | Olb: 0202000201 0002002201 3000100000 0233302311 3133332333 3230023023 |
|    | Glb: ctgaacgcggct attttttgga ttttaagcga tttgttttc gattgggggg attttttattt | Glb: 0202000201 0002002201 3000100000 0233302311 3133332333 3230023023 |
| 661| Ont: agaatattt ccaaaagctt gtaaagctgt ctaatctgct tcaagacct ctaagacctt | Olb: 0200003303 100012313 2300023213 3000301233 3130020213 3330233233 |
|    | Glb: agaatattt ccaaaagctt gtaaagctgt ctaatctgct tcaagacct ctaagacctt | Glb: 0200003303 100012313 2300023213 3000301233 3130020213 3330233233 |
| 721| Ont: attttcgtatta ttattgagcct ctaatctgct aacaggggtc atctagtggg ctaagacctt | Olb: 0333123330 2330122120 3001233300 0010221233 0312301222 0002111133 |
|    | Glb: attttcgtatta ttattgagcct ctaatctgct aacaggggtc atctagtggg ctaagacctt | Glb: 0333123330 2330122120 3001233300 0010221233 0312301222 0002111133 |
| Gnt: | atttcgttta | gtataggcga | taactgtttaa | aacaggcgtt | atctgtacgg | aaaagccctt |
|---|---|---|---|---|---|---|
| Ont: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 781 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 841 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 901 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 961 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 1021 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 1081 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 1141 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 1201 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 1261 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 1321 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 1381 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 1441 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |
| Gnt: | gagcgtagcg | tgctctgcca | gttaagcgtg | tgcgtctag | atctagcag | ccgataacg |
| 1501 | 0032003000 | 3002130321 | 3112210100 | 2202132022 | 2202013200 | 1120300132 |

Table 2. Continued.
55.73 × 10^6 tests to be realized. Thus, the probability of finding a given sequence is 1.79 × 10^{-8}, that is, approximately 1 sequence out of 10^8.

The *Lactococcus lactis* plasmid genomic sequence has 2047 nucleotides. So, the codeword length is n = 2047 and the degree of the Galois ring extension r is 11. The number of p(x) is 176 [11], [12]. Again, among these, only one p(x) is associated with a g(x) of the Hamming code (2047,2036,3), that is,

\[ p(x) = x^{11} + x^{10} + x^7 + x^2 + 1 \]

and this identification was made using the B labeling, as shown in Table 2.

A statistical analysis related to the *Lactococcus lactis* plasmid genomic sequence is as follows: with each primitive polynomial there is a corresponding generator polynomial of a code. For the given DNA sequence we use the 24 labeling and the resulting 24 sequences are multiplied by the generator matrix. From this operation results 24 codewords. Each one of these codewords is multiplied by the parity-check matrix. If the result is zero then the

\[ g(x) = x^{11} + 3x^{10} + 2x^9 + x^7 + 2x^6 + 2x^5 + 3x^2 + 2x + 1 \]

and this identification was made using the B labeling, as shown in Table 2.
given DNA sequence is a codeword. Otherwise, we have to verify what happens if in each position we have different nucleotides. To do that, we have to realize three substitutions in each position of the original DNA sequence and verify again if this modified sequence is or is not a codeword. Since the Lactococcus lactis plasmid genomic sequence has \( n = 2^r - 1 = 2047 \), it follows that \( r = 11 \).
From this, the degree of the primitive polynomial is 11 and as a result we have 176 different primitive polynomials. Since for each one of them we have to use the 24 labeling, this leads to 4224 codewords to verify for a given error-correcting capability. Since in this case we have 1010 possibilities, an upperbound is 4,300,032 codewords to be tested. Now, since there is always one nucleotide difference, we have to realize three times 63 tests for each one of the 4,300,032 codewords. Therefore, yielding a total of 812,7×106 tests to be realized. Thus, the probability of finding a given sequence is 1.23×10−9, that is, approximately 1 sequence out of 109.

Note that g(x) is also a primitive polynomial, since by reducing modulo 2 its coefficients lead to p(x). Therefore, both polynomials are associated with the same algebraic and geometric properties. Contrary to our expectations, there is just one p(x), its corresponding g(x), and a labeling capable of identifying each sequence under consideration. This suggests the existence of an intrinsic geometric property that may be associated with each DNA sequence.

What has been observed is that, in all the DNA sequences previously identified, there is always a difference of a single nucleotide between the \( NCBI \) sequence and the codeword generated by a Hamming code. Although the code (owing to its error correction capability) allows a difference in any position in the sequence, this difference occurs at one specific position. In the biological context, this mismatch is known as a single nucleotide polymorphism (SNP).

We can observe that the SNP occurred at position 122 in the TRAV7 predicted gene, changing \( A \rightarrow G \), and so originating a transition mutation (change of one purine/purine or pyrimidine/pyrimidine) - Table 1. In contrast, in the \( Lactococcus lactis \) plasmid genomic sequence, the SNP occurred at position 1547, changing \( A \rightarrow C \), and so originating a transversion mutation (change of a purine for a pyrimidine, or vice-versa) - Table 2. Note that in the TRAV7 predicted gene the SNP occurred in the intronic region, whereas in the \( Lactococcus lactis \) plasmid genomic sequence the SNP occurred in the \( L \) region, where the repB gene is located - Figure 2. One possible interpretation is that either the codeword generated by a Hamming code is an ancestor of the corresponding \( NCBI \) sequence, or it is an SNP with respect to the corresponding \( NCBI \) sequence, or the other way around. However, since this mismatch is within the error correction capability of the code, it follows that the modified Berlekamp-Massey decoding algorithm [13] is capable of detecting and correcting such a mismatch.

Conclusion

In this paper, we have shown that not only are some protein coding sequences identified with the codewords of Hamming codes, but a gene, and even a whole genome, is identified with codewords as well. Although this is not a definitive answer to the question of whether or not there is an error-correcting code underlying actual DNA sequences, it is an encouraging result.

The majority of the DNA sequences were reproduced by the Hamming codes over rings. One possible explanation is provided by the arithmetic and computational flexibilities of this algebraic structure. As a consequence, sequences reproduced by the Hamming codes over fields exhibit less adaptability than those offered by the Hamming codes over rings. This observation suggests that it is possible to classify the proteins according to their stability in the mutation index.

As usually occurs when a new result appears, many new questions emerge. Do they, in fact, reveal the existence of a mathematical structure underlying DNA sequences? Why does the code point to a specific position for each reproduced sequence? Biologically, how important is the SNP in the position pointed out by the code?

Acknowledgments

The authors would like to thank the anonymous referees for the comments and suggestions which improved the presentation of the paper and also Peter Seelig for the advices and technical discussions.

Author Contributions

Conceived and designed the experiments: LCBF ASLR RP. Performed the experiments: LCBF ASLR. Analyzed the data: LCBF ASLR JHK MCSF EB RH MY RP. Wrote the paper: LCBF ASLR JHK MCSF EB RH MY RP. Developed the software: JHK. Conceived and designed the study: RP.

References

1. Schneider TD, Stormo GD, Gold L, Ehrenfeucht A (1986) Information content of binding sites on nucleotide sequences. Journal of Molecular Biology 188: 413–431.
2. Kumar RP, Senthikumar R, Singh V, Mishra RK (2010) Repeat performance: how do genome packaging and regulation depend on simple repeats? Bioessays 32: 65–174.
3. Hoeijmakers JHJ (2001) Genome maintenance mechanism for preventing cancer. Nature 411: 366374.
4. Ozturks S, Demir D (2011) DNA repair mechanisms in mammalian germ cells. Histology and Histopathology 26: 505–517.
5. DR F (1981) Are introns in-series error-detecting sequences? J Theoretical Biol 93: 861–866.
6. Rosen GL (2006) Examining Coding Structure and Redundancy in DNA. IEEE Eng In Medicine and Biology Magazine 25: 62–68.
7. Liebowitch LS, Tsao Y, Todorov AT, Levine L (1996) Is There an Error Correcting Code in the Base Sequence in DNA? Biophysical Journal 71: 1539–1544.
8. Faria LCB, Rocha ASL, Kleinschmidt JH, Palazzo R, Silva-Filho MC (2010) DNA sequences generated by BCH codes over GF(4). Electronics Letters 46: 292–293.
9. Hoogcrwelm A (1959) Codes correcteurs d'erreurs. Chiffres 2: 147–156.
10. Bose RC, Chaudhuri DK (1966) On a class of error-correcting binary group codes. Inf Control 3: 60–79.