Genetic Information Entropy Spectrum

(GENIES)

User manual

Melvin M. Vopson

University of Portsmouth, Contact e-mail: melvin.vopson@port.ac.uk

GENIES is a computer program developed to facilitate the study of genome sequences in a comparative way using the information entropy. The program can analyse genomes of any size by converting the genetic information contained in a given genome into a numerical Information Entropy Spectrum. This procedure is done for two genomes of the same size, one being a reference genome and the other being its mutated version. The program allows fast detection of base point mutations from the Information Entropy Spectra. Most importantly, the program could be used to research and identify predictive algorithms of future genetic mutations.
The program is free to use and it can be freely downloaded from:
https://sourceforge.net/projects/information-entropy-spectrum/

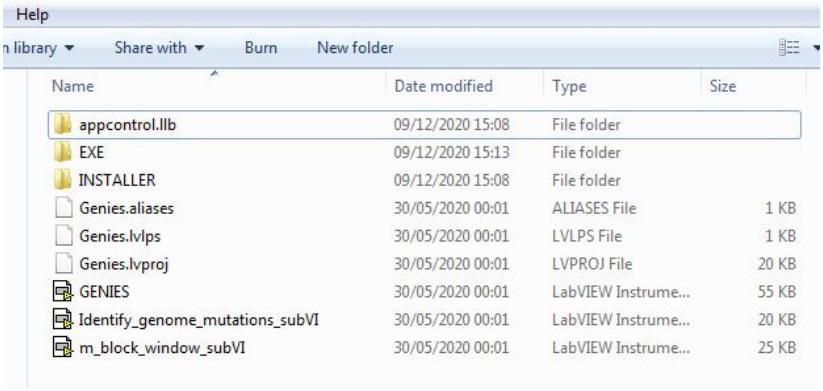
1. Requirements

To use this software you must have:

a) At least 1Gb free space on your HDD.
b) A Windows 64 bit operating system.
c) Admin rights on your PC.

2. Installation

Download the zipped folder “Genies.zip” and unzip the folder. The content of the folder is shown in the image below.



Click on the “INSTALLER”, which contains the folder “VOLUME”. Open “VOLUME” and click on the “setup” application file. Follow the on screen instructions to complete the installation. When installation finishes, a pop up message will say:

“The Installation has finished updating the system”

then click “NEXT”.

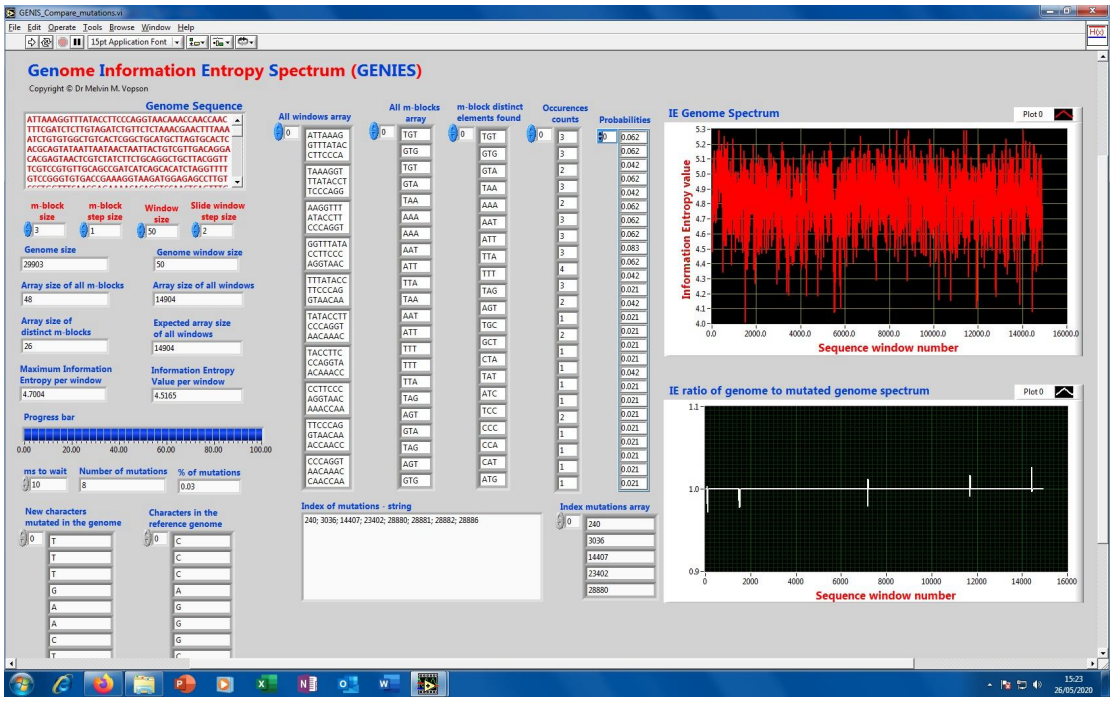
Another pop up message will ask you to restart the computer in order to complete the installation. After restarting, the Genies program will appear in your list of Windows programs. Please note that the installer does not add a desktop icon. You must do it yourself, if you wish one.

3. Running the program

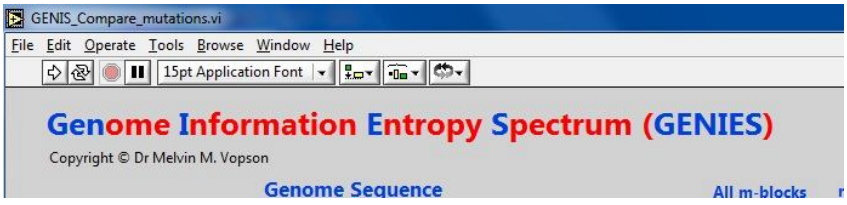
When running the program, it will open its own folder in Program Files asking for the input genome sequences. The first input file must be the reference genome and the second is the mutated sequence. The program comes with two examples of RNA sequences of COVID-19. These could be used to run / test the program. One file is the reference Covid-19 from Wuhan, Dec. 2019: MN908947.3_Reference_China and the second file is a mutated version collected in Japan, March 2020: LC542809_Japan. The program will ask then for a file name and location to save the data after completing the computation.

Attention: each input file must contain genome sequence of the same size. They must be text files containing no other characters and no text formatting. The input files must be prepared in the correct format prior to running the program. If they are not the same or contain other characters than A,C,T,G, then program returns an error when running.

The image below shows the front panel of the program.



To run the program click on the arrow:



To stop or to pause the program click on the red button, or the pause button respectively:

4. Program input variables

The key variables of the program are:

- **m-block size** – this must be set to 3 (size of codons)
- **m-block step size** – this must be always 1 to make sure it captures any possible correlations within a window
- **window size** – this is a very important variable. Too large results in mutations being picked up more than one. Too small, it results in segments that could miss some mutations as the IE values are unchanged. Depending on the genome size, a typical window of 50 characters is reasonable. For very large genomes, this could be increased to WS = 100 or even more. We recommend using this WS as a research parameter to investigate the optimal value for a specific project.
- **Sliding window step size** – this is also a very important variable. To fully capture everything the smallest step size SS = 1 must be used. The smaller the step, the longer is the computation process. The SS must always be smaller or equal than the WS. If the wrong value is inputted, the program generates an error. A reasonable value is SS = 2, but we recommend to use this a research variable in order to identify the optimal scanning conditions for a given project.
- **ms to wait** - this is the time to wait for each iteration. It can be set to zero, meaning the program runs at the fastest rate. When a non zero value is set, the program runs slowly, which is useful when the investigator wishes to observe the codons array in real time or to probe in real time other aspects of the program.

5. Main program outputs

The key outputs of the program are:

- **Genome size** – total number of characters in the genome
- **Array size of all windows** – the size of the Information Entropy Spectrum
- **Number of mutations** – the number of point bases detected different in the two genomes
- **% of mutations** – the % value of the genome that suffered mutations
- **IE Genome Spectrum** - top image shows the IE spectrum of the reference genome and then of the mutated genome
- **IE ratio spectrum** – bottom image shows the ratio of the two IE spectra

The ASCII data saved by the program contains four columns arranged in the following sequence:

Column 1 = IE ratio spectrum
Column 2 = IE mutated genome spectrum
Column 3 = IE reference genome
Column 4 = Window index

6. Theory

6.1 Information theory

Shannon gave the mathematical formulation of the amount of information extracted from observing the occurrence of an event in his 1948 seminal paper [*C.E. Shannon, A mathematical theory of communication, The Bell System Technical Journal, Vol. 27, pp. 379–423 (1948)*]. Ignoring any particular features of the event, the observer or the observation method, Shannon developed his theory using an axiomatic approach in which he defined information (I) extracted from observing an event as a function of the probability (p) of the event to occur or not, I(p). The second axiomatic property is that the information measure is a continuous positive function of the probability $I(p) \geq 0$. An event that is certain, i.e. p = 1, gives therefore no information from its occurrence, so I(1) = 0. Assuming that for n independent events of individual probabilities p_i the joint probability p is the product of their individual probabilities, then the information we get from observing the set of n events is the sum of the individual event's information, $I(p) = I(p_1 \cdot p_2 \cdot \ldots \cdot p_n) = I(p_1) + I(p_2) + \ldots + I(p_n)$. Shannon identified that the only function satisfying these axiomatic properties is a logarithmic function and, for an event whose probability of occurring is p, the information extracted from observing the event is:

$$I(p) = -\log_b p = \log_b(1/p) \qquad (1)$$

where b is an arbitrary base, which gives the units of information, i.e. for binary bits of information, b = 2.

Let us assume a set of n independent and distinctive events $X = \{x_1, x_2, \ldots, x_n\}$ having a probability distribution $P = \{p_1, p_2, \ldots, p_n\}$ on X, so that each event x_j has a probability of occurring $p_j = p(x_j)$, where $p_j \geq 0$ and $\sum_{j=1}^{n} p_j = 1$.

According to Shannon, the average information per event, or the number of bits of information per event, one can extract when observing the set of events X once is:

$$H(X) = -\sum_{j=1}^{n} p_j \cdot \log_b p_j \qquad (2)$$

The function *H(X)* resembles an information entropy function and it is maximum when the events x_j have equal probabilities of occurring, $p_j = 1/n$, so $H(x) = \log_b n$. When observing N sets of events X, or equivalently observing N times the set of events X, the number of bits of information extracted from the observation is $N \cdot H(X)$.

6.2 Application to genome studies

This approach could be used to study other forms of information systems, such as the biologically encoded DNA / RNA information. Thinking of the genome as a coding system, it is highly probable that certain stretches of the DNA / RNA will have an information entropy value different from other stretches. Genetic sequences are

examined here from an external point of view, as information storage systems, without taking into account the detailed physical – chemical mechanisms for information processing.
Let us examine the specific case of biological DNA information. A typical DNA genome sequence contains a very large string of A, C, G, and T letters. In the actual genome, these letters stand for the nucleotides: adenine (A), cytosine (C), guanine (G), and thymine (T). Viruses have typically RNA genomes, where thymine (T) is replaced by uracil (U), but here we will use for simplicity A, C, G, T. Our set of n independent and distinctive events becomes $X = \{A, C, G, T\}$, so n = 4 and a probability distribution $P = \{p_A, p_C, p_G, p_T\}$. Using digital information units, b = 2, for four possible distinctive events / states, n = 4, then we need 2 bits per letter, $H(x) = \log_b n = \log_2 4 = 2$ to encode the message: A = 00, C = 01, G = 10, T = 11.
Let us consider a random DNA subset, consisting of N = 34 letters:

CACTTATCATTCTGACTGCTACGGGCAATATGTG

The above subset is randomly generated and the grouping rule of the bases A and T (or U in RNA) and C and G is not respected. If the letters within this DNA subset would have equal probabilities to occur (1/4), then the subset would have H(X) = 2, and a total entropy of N·H(X) = 68 bits of information. However, the above subset has the following probability distribution:

$$P = \left\{ p_A = \frac{8}{34}, p_C = \frac{8}{34}, p_G = \frac{11}{34}, p_T = \frac{7}{34} \right\} \tag{3}$$

resulting in:

$$H(X) = -\left(\frac{8}{34}\log_2\left(\frac{8}{34}\right) + \frac{8}{34}\log_2\left(\frac{8}{34}\right) + \frac{11}{34}\log_2\left(\frac{11}{34}\right) + \frac{7}{34}\log_2\left(\frac{7}{34}\right) \right) = 1.978 \tag{4}$$

Therefore, instead of 68 bits to encode the sequence we can get by with only 67.25 bits, or the entropy of the subset is 67.25.
Equation (2) and the above approach give information only about the distribution of the individual symbols within the DNA subset and their entropy values, but it doesn't reflect the correlations between the symbols. In order to incorporate these possible correlations, equation (2) has to be generalized to the so-called block information entropies:

$$H_m(X_m) = -\sum_{j=1}^{m} p_j^{(m)} \cdot \log_b p_j^{(m)} \tag{5}$$

where $p^{(m)}$ are the probabilities of the combinations of m symbols, where $1 < m \leq n$. The index now extends over all possible combinations of m symbols, which are called m-blocks. In our case m can be any positive integer value larger than 1 and smaller than 5. The choice of the number of symbols or m-blocks can be based on the observation that coding sequences for peptides and proteins are encoded via codons, i.e. sequences of blocks of three nucleotides. Hence, choosing m = 3, we will consider

each three-nucleotide codon to be a distinct symbol in Shannon's information theory framework. We can then take a subset of the genome and estimate the probability of occurrence of each codon by simply counting and dividing by the length. Let us consider again the case of our DNA subset, consisting of N = 34 letters:

CACTTATCATTCTGACTGCTACGGGCAATATGTG

Assuming the readout could start at any nucleotide, counting from left to right each three adjacent nucleotides, a set of 32 three letters combinations with 29 distinct combinations is obtained:

$$X_{original} = \left\{ \begin{matrix} \text{CAC, ACT, CTT, TTA, TAT, ATC, TCA, CAT, ATT, TTC, TCT, CTG, TGA, GAC, TGC, GCT, CTA,} \\ \text{TAC, ACG, CGG, GGG, GGC, GCA, CAA, AAT, ATA, ATG, TGT, GTG} \end{matrix} \right\} \quad (6)$$

and their probability distribution:

$$P_{original} = \left\{ \frac{1}{32}, \frac{2}{32}, \frac{1}{32}, \frac{1}{32}, \frac{2}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{2}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32} \right\} \quad (7)$$

The maximum possible entropy of the DNA subset is $\log_2 29 = 4.858$ bits. Using the equation (5) and the probability distribution (7), the actual entropy of the original subset is $H_{original} = 4.813$ bits.
To demonstrate the principle proposed here for studying genome mutations, let us now assume that the same 34 letters genome subset suffers a random mutation at base point 11, as represented in the figure below:

CACTTATCATTCTGACTGCTACGGGCAATATGTG - Original subset

↓

CACTTATCAT<u>A</u>CTGACTGCTACGGGCAATATGTG - Mutated subset

Figure 1. Example of 1 base genome mutation at index 11 (T mutates into A).

The mutated genome subset has again 32 three letters combinations, but 26 distinct letters instead of 29, as the original subset:

$$X_{mutated} = \left\{ \begin{matrix} \text{CAC, ACT, CTT, TTA, TAT, ATC, TCA, CAT, ATA, TAC, CTG, TGA, GAC, TGC, GCT, CTA, ACG,} \\ \text{CGG, GGG, GGC, GCA, CAA, AAT, ATG, TGT, GTG} \end{matrix} \right\} \quad (8)$$

and their probability distribution:

$$P_{mutated} = \left\{ \frac{1}{32}, \frac{3}{32}, \frac{1}{32}, \frac{1}{32}, \frac{2}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{2}{32}, \frac{2}{32}, \frac{2}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32} \right\} \quad (9)$$

The maximum entropy of the mutated genome subset is $\log_2 26 = 4.7$ bits, and using equation (5) and the probability distribution (9), the actual entropy of the mutated subset is $H_{mutated} = 4.601$ bits. This is different to the entropy of the original subset, demonstrating the concept of using the information theory to detect and study genetic mutations. To quantify the difference between the original and the mutated subsets one could use either the difference between the two information entropies, $\Delta H =$

$H_{original}$ - $H_{mutated}$ = 0.212 bits, or the Information Entropies Ratio, IER = $H_{original}$ / $H_{mutated}$ = 1.046. Using the difference, any non-zero value would indicate the presence of a mutation within the subset. However, using the ratio, any value not equal to 1 would indicate the occurrence of at least one mutation. From a convenience point of view, the Information Entropy Ratio, IER, is a better representation due to the fact that the IER parameter has no units, while ΔH has units of bits.
Using the concept of information entropy to study genome mutations has been briefly demonstrated in the above approach. However, the example discussed was just for a small genome subset of 34 characters. The main objective is to implement this technique for studying full size genomes.
The method proposed here involves computing a so-called Information Entropy Spectrum (IES) of the genome. This is achieved by splitting a large genome into subsets, called "windows". A "window" has a given number of base points (characters) called "window size", WS, which is the equivalent of the random genome subset discussed in the previous section, where WS = 34. Starting from left to right, one slides the "window" across the whole genome, where each position of the window is obtained by sliding it for a fixed number of characters, called "step size", SS. The SS must be at least 1 and its maximum size is WS, so $1 < SS \leq WS$, in order to ensure that all sections of the genome are captured by this process.
By doing this, a given genome of N characters (number of base points), will result in N_w windows, given by the formula:

$$N_w = \frac{N}{SS} - WS + SS \qquad (10)$$

To study the whole genome, one needs to compute the information entropy spectrum of the genome, which is obtained by calculating the information entropy (IE) value of each window, and plotting the IE values obtained as a function of the window index location within the genome. The window index location, or the size of the information entropy spectrum takes values from 1 to N_w, depending on the values of N, WS and SS, according to (10). In fact the procedure of window sliding across the whole genome is exactly the same process used to generate the set of 32 three letters combinations of the random 34 bases genome subset in the above example (see relation (6)), where the SS =1 and WS = 3, so $N_w = 34/1 - 3 + 1 = 32$.
To clarify this procedure, we present a graphical example in figure 2, by reconsidering a random genome sequence example. In this example, the window size is WS = 12 and the step size is SS = 4.

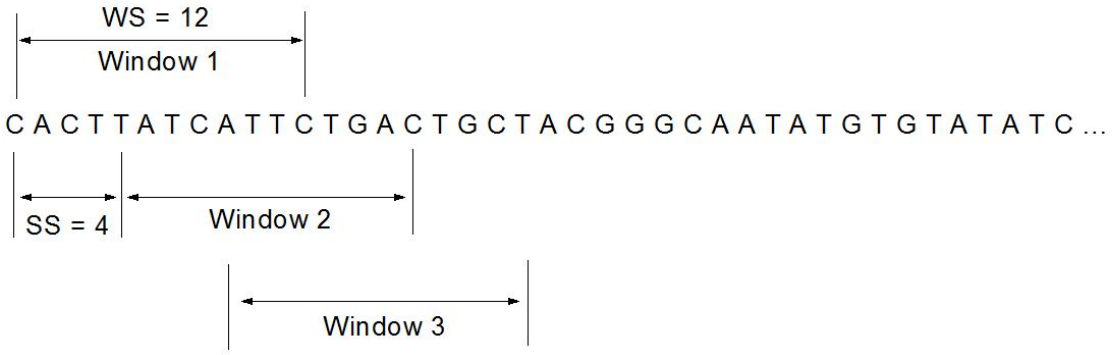


Figure 2. Example of computing the Information Entropy Spectrum of a genome by sliding a Window across it, with window size is WS = 12 and step size is SS = 4.

For these particular WS and SS parameters, assuming a fictitious genome size of N = 210000 bases, according to (10) we will generate a set of 52492 windows, i.e. N_w = 52492. The information entropy of each window is then computed by counting the distinct 3 letters m-blocks within each window and their occurrence probabilities. As already mentioned, this is done by setting WS = 3 and SS = 1. According to (10), for each window in this example we get $N_w = 12/1 – 3 + 1 = 10$ elements in each window. For Window 1, the distinct 3-letter occurrences are: CAC, ACT, CTT, TTA, TAT, ATC, TCA, CAT, ATT, TTC and using equation (5) we get the Information Entropy of Window 1, $IE_{W1} = 3.322$ bits, which is also the maximum possible value as all occurrences have the same probability, 1/10. Repeating this process for all 52492 windows, a numerical value for each window is obtained converting the biological information contained within a window to a single numerical value. Plotting these values against their index location within the genome, results in what we call The Information Entropy Spectrum.

The Information Entropy Spectrum is a numerical representation of the genetically encoded information within a given genome. This is a very convenient algorithm as it allows further processing of the information contained in the spectrum. One application of this technique would be the study of genetic mutations. In order to study genetic mutations, the Information Entropy Spectrum of two genomes of the same entity must be computed. One sequence could be seen as a reference genome and the second is a genome sequence collected at a different time, or at a different location. If mutations occurred, when the information entropy spectra of the two genomes are divided to each other, the information entropy ratio (IER) spectrum is obtained. Any value within the IER spectrum not equal to 1 would indicate a mutation. This technique is a rapid and time effective method of detecting genetic mutations without a full single base point nucleotide comparison between the two genomes. However, the true potential of this proposed methodology is achieved when using it in reverse, as a potential predictor of future genetic mutations. The idea is to analyze the inflection points where known mutations occurred, which could allow one to corroborate *special features* in the information entropy spectrum to the index location of the mutations. One could calibrate this process and use the *special features* in the information entropy as a predictor of future mutations. The key is to identify these correlations and *special features*, which could be done through careful data processing and analysis, or more effectively using machine learning plug in software. In any case, the proposed methodology can only work on fully automated computer software, and we developed LabView software called Genome Information Entropy Spectrum (GENIES) to do this.

The GENIES program is a stand-alone fully functional code that has an easy to use graphical interface and performs the following tasks:

a) it loads up two genome files (original and mutated);
b) it splits up each sequence into a series of windows with a given slide step (both window size and slide step are input variables in the code);
c) it splits up each window into a set of three characters, counting each occurrence and computing the probabilities within each window;
d) it computes the information entropy spectrum of each genome;
e) it computes the information entropy ratio spectrum;
f) it extracts the number of mutations identified, their index location and then it saves all the data including the spectra;

g) in addition, the program performs a base point to base point comparison of the two genomes to detect the mutations via direct comparison.

This final function is very important to cross check the proposed methodology and to facilitate its calibration for predictions of future mutations.