#### Python Program – ‘Random Genome Generator’:

A Python 2.5 program was developed to randomly shuffle the genome of choice, preserving the length and base composition of the original sequence. To provide an appropriate level of confidence in statistical analysis, as defined by the central limit theorem, 30 random shuffles were generated for each organism.

#### **Python Program – ‘Palindrome Searcher’:**

Python 2.5 program using, biopython-1.49 (Cock *et al.,* 2009), numpy-1.2.1 and scipy-0.7.0 (Jones *et al.,* 2001) was developed to search for palindromes of between user define lengths bp in length (L) across the whole genome sequence. 150 could be chosen as the upper limit due to the instability of longer palindromes within genomes (Leach *et al.,* 1994), although longer palindromes of length 165-170bp have been identified (De Gregorio *et al.,* 2009).

The base composition of the genome is calculated for the purpose of the probability calculation. The exact base composition will have an effect on the probability of a specific palindrome being considered to occur at random. If a palindrome was ‘ATTTTAAAT’ and the entire genome was composed of AT bases then this could be considered to occur through chance, hence having a relatively high probability value. If there was a CG within this example (the CG ratio of the genome being extremely low) the probability would be much lower, hence altering the chance of this palindrome randomly occurring within the genome. The probability function uses the base composition for the genome in question to calculate the probability for a match (*Pmatch*) and mismatch (*Pmismatch*). A match is defined when a base from the 3’ end is equal to its reverse complement in its equivalent position from the 5’ end (AT, TA, CG and GC), hence the relative ratio of the base composition will affect the chance of what the corresponding reverse complement base would be, therefore altering the probability of the palindrome sequence occurring at random. A mismatch is the number of bases that do not match the definition of a match. The probability function could be defined by:



Where  is the number of ways of getting M mismatches in the half length of palindrome (L/2). The probability of getting a match (*Pmatch*) is: Σ(A\*T, T\*A, C\*G, G\*C)(Number of matches), the values for A, T, C and C are returned from the base composition function. The probability of getting a mismatch (*Pmismatch*) is: (1- *Pmatch*)(Number of Mismatches).

A probability threshold (P) was set to 1/ (length of the genome) to return palindromes that would be expected to occur randomly only once within a sequence of that specific length. A mismatch threshold was set to 50% of the half-length (L/2) of the palindrome length being searched for. The purpose of these thresholds was to return palindromes that had a low probability of occurring at random but ranged from perfect to 50% mismatches (half-length of palindrome) to allow imperfect palindromes to be returned.

The base composition of an organism changes across the genome. This was not taken into consideration in this study and would be an interesting investigation to allow for changing frequencies. Such changing base composition will change the probability values of the palindromes resulting in new palindromes being identified and some being rejected by the threshold barriers.

#### Python Program – ‘Palindrome Reducer’ (Appendix 3):

Identified palindromes from the Palindrome Searcher program were then reduced in order to remove palindromes contained within larger palindromes. The largest palindromes could not by definition be contained within another palindrome. Therefore the identified palindromes were ordered by length largest first and condensed based on the logic:

* putative palindrome = P
* Identified palindrome = I
* If: I(start position)≤ P(start position), and P(end position) ≤ I(end position)

Then P is not a new unique palindrome, but P is contained within the palindrome I. If this equation evaluates as false the new, unique palindrome P is returned (the process continues until there are no further putative palindromes P to be tested).

#### Python 2.5 Program – ‘Clustering Analysis’ (Appendix 4):

A Python program was developed to return the distance in base pairs from a palindrome’s start position to the next palindrome start in an ordered list of palindrome starts, starting with the highest value first. This will provide the distance from one palindrome to the next throughout the entire genome. The distribution of a clustered data set distances would be expected to skew to the shorter distances compared to the randomly generated population.

#### Python 2.5 Program – ‘Palindrome: Intergenic or Intragenic?’

A Python program was developed to determine the location of the palindrome based on the genes stop and start co-ordinates defined in the .ptt file (this also work with a GFF).

If: gene start ≤ palindrome start, and palindrome end ≤ gene end, then this was defined as intragenic, otherwise the palindrome was defined as intergenic.

#### **GenomeDiagram**:

GenomeDiagram (Pritchard *et al.,* 2006) was used to illustrate the genes and palindromes using Genbank files (also works with EMBL files) and the results returned in the study.