Additional script functionality

**chromo\_gc.py**

Script identifies the GC content of a given set of chromosomes, and identifies GC content in windows around given ALVE insertion sites. Window sizes can be changed within the script file.

**cluster\_counter.py**

Script takes a LTR retrotransposon positions file and identifies clusters, given a minimum element number for a cluster and a minimum cluster distance. Script reports the numbers within each cluster and can give the genera and identity of the sites if provided in the original positions file.

**dendro\_model.py**

Used to shuffle observed ALVE occurrences across a total number of ALVEs to identify whether observed dendrogram had shared site structure, or whether it was simply based on number of observations. (Chapter 7).

**DiscoverAlleleInPoolSequencing.R**

Script written to identify likelihood of missing inserts in pooled sequencing data given parameters on insert frequency, pool size, population size, coverage and other parameters such as sequencing error rates. Dr Gregor Gorjanc (Roslin Institute) was a great help in the construction of this script before I adapted it for ALVE detection.

**fastq\_sorter.py**

Takes paired FASTQ files and sorts them, including identifying unmatched pairs. Similar sorting can be undertaken by other programs such as BEDTools, GATK *etc.* Be careful with available memory.

**gc\_calculator.py**

Calculates genome-wide GC content and other features such as total Ns, but this does not account for chromosome-specific values, just overall content.

**gc\_with\_age.py**

Calculates the GC content of annotated LTR retrotransposons with respect to their insertion age.

**hexamer\_model.py**

Randomly generates hexamers from a string of the entire reference genome, and classifies each on its GC content. Used to identify insertion site skews (chapter 7).

**ltr\_identity\_extractor.py**

Calculates the identity of paired LTRs from a positions file of the LTR pairs. It extracts the sequence from the reference genome, aligns them using muscle and calculates the identity.

**model\_sampling.py**

Effectively a script for calculating binomial probabilities that an insert will be missed from individual sequencing given the insert frequency, sample size used for sequencing and the overall population size. This includes randomised population genotypes and multiple iterations. Included within the DiscoverAlleleInPoolSequencing.R as the first output.

**pos\_merger.py**

Takes a concatenated file of positions and merges them together. This script was specific for the LocaTR positions file format, but BEDTools merge is more efficient.

**pseudo\_fastq.py**

Script takes a concatenated FASTQ file (such as read1 concatenated with read2) and renames all reads to generate a single FASTQ file where reads are no longer related. This was used to test the ALVE ID pipeline sensitivity with single read data (Chapter 7).

**random\_age\_generator.py**

Script takes an existing positions file with element ages and redistributes the ages at random a given number of replicate times. This can be used to compare whether elements have been selected against over time.

**random\_insert\_generator.py**

Used to generate a given number of random integration sites in a reference genome, and replicated a given number of times. This can be used to generate random distributions to compare to observed variant sites. BEDTools shuffle is more efficient.

**random\_insert\_generator\_per\_chromosome.py**

From a given reference genome, this script generates random positions of a given number, and does this multiple times if necessary. This can be used to generate random position distributions for individual chromosomes to compare to observed distributions. However, the BEDTools shuffle program can do this more efficiently within a BASH loop.

**seq\_extract.py**

Extracts sequences from a given reference genome given a list of positions.

**seq\_formatter.py**

Splits multi-sequence FASTA files into individual files, and can perform reverse complementation.

**static\_functions.py**

A group of functions used by other script files, including fasta processing, ref genome dictionary creation, sequence extraction from a positions file, list formatting of positions files, merging of positions files, and some other accessory tasks