File 7: The Argos algorithm: Exhaustive motif count

Argos is designed to evaluate if a given sequence S (typically a few hundred bases long) has repeated motifs which are very unlikely in the light of genome wide statistics. Sequence motifs are modelled by consensus sequences of length l with m mutations. Typically, we work with motif lengths between 6 and 11 nucleotides and up to 4 mutations. The first step is to establish background counts for all motifs:

Background counts: The task is to count all motifs in the genome. This is done by (a) counting all n-mers in the genome (b) computing all possible mutations of each n-mer. Thus, the copy number of any given motif in the genome is computed. Some thought about efficient data structures using the STL library in C++ allows convenient runtime performance, for example the Drosophila genome can be evaluated in a few hours. Argos has options to adjust the overlap between motifs in (a) and to treat the reverse complement. In this case, there is a problem with motifs which have some degree of palyndromicity. These motifs would be counted twice. The problem is solved by computing the volume of the overlap (in hamming space) of any two motifs related by reverse complimency and subtracting it from their total volume.

The second step then is to score a putative module S against the background counts:

Scoring modules: Similar to the computation for the background procedure explained above, all copy numbers of all motifs are computed for S. Then, Argos computes from the background counts (which are read from flat files) the expected number E_h of copies of any motif h in S and sorts all motifs in S by their Poisson score $-\log(\text{prob}(x \geq E_h))$ in S. When also allowing for reverse complement motifs, the code maximizes (for each motif) probabilities, i.e. it will decide whether the single strand model or the double strand model is more significant.

The problem is that many of the significant motifs will be related to each other, for example by mutations or by shifts. We employ a greedy algorithm to resolve these dependencies. The top motif eliminates all motifs which are related to it (up to an optional number of mutations or shifts) from the list; then the second highest motif eliminates all other related motifs, and so on. The algorithms stops when more than M top motifs are produced. Typically it turned out to be useful to work with a fixed motif lengths of 8, to allow 2 mutations in the consensus, to eliminate all motifs which are related by less than 4 mutations and/or shifts, and to set $M = 5$. The overall score for S is then the sum of all Poisson scores of the top motifs. Since Argos needs to be run over potentially hundreds of millions of sequences, care was taken to implement it efficiently. Efficiency was achieved by two technical points. First, we work with a fixed minimum significance threshold. This makes it possible to precompute the minimum copy number needed for each motif to be significant. It is then easy to compute the list of significant motifs. Second, we use a sliding window (typically sliding by 1 nucleotide) to score each sequence S in a genomic region (or the whole genome). Our data data structures allow to efficiently compute the differences in all counts between sucessive windows, thus greatly speeding up run time.