# Substitution-Cost Function

**Overview of function**

I have developed a substitution-cost function is designed to work with the nucleic acid notation: ACGT, which represent the four nucleotides commonly found in deoxyribonucleic acids (DNA). The function only operates on pairs of sequences (as opposed to pairs of letters) with length 2, however, t

All the reasoning below is fictious, by which I mean adheres to some ‘made-up’ biological concept of DNA sequences and their generation.

**Rules**:

**Matching**:

* The sequence **AT** will only be matched to **AT** – AT represents a special ‘marker’ in the sequences and cannot be found anywhere else.
* **AG** and **CT** will have double matching score as opposed to generic matches of pairs of sequences of length 2 – these sequences are extremely common in DNA and hence matching them would more likely provide a better overall alignment
* **All other adjacent pairs** of letters will be given equal score to there matching, as they are all equally as likely to occur

**Mismatching**:

* In the pair of sequences being mismatched, if both letters are being mismatched with their opposites (i.e. A with G, C with T) then score a lot harsher than if only one letter in the sequence is being mismatched with its opposite.
* **All other mismatches** of pairs of sequences are scored equally, as all have equal impact on the likelihood of potential alignment.

**Gaps**: A logarithmic gap penalty function will be incorporated. Such that a gap is scored through the function G(L) = A+BlnL, where A is the cost to open the gap, B is the gap extension penalty and L the length of the gap.

* The cost of opening a gap (A) will be large – we want gaps to be at a minimal
* Extensions to the gap will **decrease** **logarithmically**, so the longer the gap the less impact it has on the overall score of the alignment

This is implemented as long sequences of similar letters typically break up the more ‘**important**’ information found in DNA.