Accelerating SNP genotyping from whole genome sequencing data for bedside diagnostics

Chen Sun and Paul Medvedev

Supplementary Information

In this section, we present a bitwise comparisons routine to determine whether two k-mers k_1 and k_2 are within a Hamming distance of one, and, if so, where the differing position is. The two k-mers are encoded as unsigned 64bit integers *a* and *b*, respectively. Recall that our bit encoding of k-mers is the natural one, representing the nucleotides with 2 bits each, in the same order as they appear in the k-mer. First, Algorithm 1 can check if k_1 and k_2 are within one Hamming distance. It uses the C language.

Algorithm 1.

Input: unsigned 64 bit integers *a* and *b* encoding k_1 and k_2, respectively.

Output: True if k_1 and k_2 differ in at most one nucleotide.

```
1.   x = a ^ b
2.   if(x == 0) return true              // a is equal to b
3.   if((x & (x-1)) == 0) return true    // a and b have only one mismatch
4.   y <- x & odd_mask                   // take all odd bits of x
5.   if((y & (y-1)) != 0) return false   // check if x has only one bit in odd position
6.   z <- x & even_mask                  // take all even bits of x
7.   if((z & (z-1)) != 0) return false   // check if x has only one bit in even position
8.   if(y == (z << 1)) return true       // check if odd bit and even bit are consecutive
9.   return false
```

The next step is to find the differing position, given that Algorithm 1 returns true and that the k-mers are not identical (i.e $x \neq 0$ in line 1). Consider x. It will have at most two non-zero bits, corresponding to the differing nucleotides. There are 32 possibilities for the locations of those bits, and there are three possibilities for their values (10, 01, 11). Thus, x can take on only 96 values. We have a simple lookup hash table T, such that $\mathrm{T}[x]$ corresponds to the differing position that results in the value of x. Note that T needs to be constructed just once and holds only 96 values.