**Velvet *de novo* assembler for short-read sequencing technologies**

**Introduction**

*Velvet de novo assembler was designed to build contigs and eventually scaffold from short read sequencing data. The follow report describe how to use Velvet, read its output, and tune its parameters for optimal results.*

**Usage**

*The Velvet de novo assembler (*[*Zerbino et al., 2008*](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2952100/#R9)*) can be used to quickly build long continuous sequences, or contigs, as well as gapped assemblies of contigs, or scaffolds, out of short-read datasets as produced by next-generation sequencing technologies. This function is mainly useful when studying data from a new organism for which a reference genome has not been assembled yet, or when trying to determine the origin of unmapped reads. Velvet can also be used to analyze colorspace data, however, this requires specific settings and the use of other programs (Will be specified later in this report).*

**Algorithm & Data structure**

*Velvet builds a de Bruijn graph from the reads and removes errors from the graph (*[*Zerbino et al., 2008*](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2952100/#R9)*). It then tries to resolve repeats, based on the available information, whether long reads or paired-end reads (Zerbino et al., 2009). It finishes by outputting an assembly of the reads, along with various statistics.*

**Available programs**

*Velvet is centered around two programs, velveth, velvetg. These two programs are always used together. They require parameter tuning in order to reach needed results.*

**Protocols to run Velvet**

**A. Assembling a set of reads**

*The basic Velvet assembly process is to take short read sequences (in this case plasmid reads) then it produces an assembly of these reads.*

*Steps:*

*- Hashing & building a graph using velveth and velvetg respectively.*

*- Velveth reads the sequence file and builds a dictionary of all words of length k, k is a parameter that the user can define manually.*

*- Velvetg reads the alignment produces by velveth and build a de Bruijn graph.*

velveth

Compilation settings:

CATEGORIES = 2

MAXKMERLENGTH = 31

Synopsis:

- Short single end reads:

velveth Assem 29 -short -fastq s\_1\_sequence.txt

- Paired-end short reads (remember to interleave paired reads):

velveth Assem 31 -shortPaired -fasta interleaved.fna

- Paired-end short reads (using separate files for the paired reads)

velveth Assem 31 -shortPaired -fasta -separate left.fa right.fa

- Two channels and some long reads:

velveth Assem 43 -short -fastq unmapped.fna -longPaired -fasta SangerReads.fasta

- Three channels:

velveth Assem 35 -shortPaired -fasta pe\_lib1.fasta -shortPaired2 pe\_lib2.fasta -short3 se\_lib1.fa

Commands used:

./velveth directory 21 -fastq ColE9-J.fastq

#above command was used to create a directory that includes #sequences, roadmaps and logs

./velvetg directory/ -exp\_cov auto -cov\_cutoff auto

# Final graph has 2 nodes and n50 of 7463, max 7463, total 7465, #using 2500/2500 reads

#creates a huerisitc graph of the reads in the roadmap file #created from plasmid\_reads