k-SLAM User Manual

David Ainsworth

March 2015

1 Introduction

k-SLAM is a program for alignment based metagenomic analysis of large sets of high-throughput sequence data. k-SLAM uses a k-mer based technique to rapidly find alignments between reads and genomes which are then validated using the Smith-Waterman algorithm [1]. Alignments are chained together into a pesudo-assembly to increase specificity. Taxonomy is inferred using a Lowest Common Ancestor technique. Genes and variants can also be found from the alignments and output in SAM format [2].
k-SLAM is fast and highly parallelisable, with speeds of $\sim$ 5 million reads per minute on 150bp paired end data. todo cite paper.

2 Installation

2.1 Compilation

k-SLAM requires a modern version of gcc that can compile C++11.
`tar xf SLAM.tgz`
`cd SLAM/build`
`make`
optionally add k-SLAM to path: `export PATH=$PATH:.`

2.2 Database build

k-SLAM is packaged with an installation script which will download the NCBI taxonomy and bacterial/viral genomes and create k-SLAM's index.
To download taxonomy and bacterial/viral genomes and create k-SLAM database in "database_dir":
`install_slam.sh database_dir bacteria viruses` (can be bacteria or viruses or both)

3 Usage

3.1 Metagenomics

k-SLAM takes a FASTQ file as its input (plus an R2 FASTQ file for paired data).
k-SLAM is called from the command line with the following command:
`SLAM --db=DATABASE_DIR --output-file=OUTFILE R1FILE.fastq (optional R2FILE.fastq)`
For optional SAM output use flag `--sam-file=SAM_FILE`
Note: Paired reads must be split into two FASTQ files with reads in the correct order.
Additional options:

Command	Arguments	Description	Defaults
Required			
--db	string	k-SLAM database directory which reads will be aligned against	N/A
Optional			
--output-file	string	write to this file instead of stdout	stdout
--sam-file	string	write SAM output to this file	none
--min-alignment-score	string	alignment score cutoff	0
--score-fraction-threshold	float	For each read, screen alignments with scores less than this fraction of the score of the best alignment	0.95
--num-reads	+ve int	Number of reads from R1/R2 File to align	all
--num-reads-at-once	+ve int	Reduce RAM usage by only analysing "arg" reads at once, this will increase execution time	10^7
--match-score	+ve int	Smith-Waterman match score	2
--mismatch-penalty	+ve int	Smith-Waterman mismatch penalty	3
--gap-open	+ve int	Smith-Waterman gap opening penalty	5
--gap-extend	+ve int	Smith-Waterman gap extend penalty	2
--num-alignments	+ve int	Number of alignments (or alignment pairs) to report per read in SAM file	10
Flags			
--sam-xa	N/A	only output primary alignment lines, use XA field for secondary alignments	N/A
--no-pseudo-assembly	N/A	do not link alignments together	N/A

Table 1: Command line arguments

3.2 Alignment

If metagenomic analysis is not required and only alignment is needed, then the flag "`--just-align`" can be used. The option "`--sam-file`" must be used. If a database was built from FASTA files, then the "`--just-align`" flag must be used.

3.3 Custom database build

k-SLAM can parse any genomes in the GenBank flat file format [3]:

- Create database directory
  `mkdir custom_db && cd custom_db`
- Build k-SLAM's taxonomy databases using the NCBI taxonomy nodes.dmp and names.dmp
  `SLAM --parse-taxonomy names.dmp nodes.dmp --output-file taxonomy`
- Build k-SLAM's index from any number of Genbank files
  `SLAM --output-file database --parse-genbank file1.gbk file2.gbk file3.gbk ... etc`

k-SLAM can also align reads to genomes in FASTA format (only alignment and not metagenomics):

- Create database directory
  `mkdir custom_db && cd custom_db`
- Build k-SLAM's index from any number of FASTA files
  `SLAM --output-file database --parse-fasta file1.fa file2.fa file3.fa ... etc`

Note: databases produced from FASTA files must be analysed using the "`--just-align`" flag.

4 System requirements

4.1 CPU

k-SLAM is designed to be run on a parallel platform, ideally with 8 or more cores. It will however run (albeit slower) on a system with fewer cores.

4.2 Memory

k-SLAM's k-mer list sort algorithm is fast but very memory intensive. For a dataset of 10 million paired reads aligning against the NCBI bacterial genomes, k-SLAM requires around 50GB RAM. For much larger datasets, k-SLAM can split them into smaller subsets (see table 1 `--num-reads-at-once`) which are analysed sequentially (still producing only one set of output files).

5 Output

k-SLAM outputs in several formats:

- Summary XML
- Per read taxonomy ID
- Abbreviated taxonomy
- SAM (optional, will use extra CPU time)

5.1 Summary XML

Each identified taxon is listed in descending order of the number of reads assigned to it. Because of the use of a Lowest Common Ancestor method, taxons may be at any rank. Each read is assigned to a maximum of one taxon (a genus entry will not contain reads that have been mapped to individual species within that genus but only reads that mapped to several species whose LCA was that genus).
Each taxon has the following tags: abundance (number of reads and percentage of total reads), taxonomyID, lineage (from NCBI taxonomy [4]), name, genes and reads.
Note: Output has to be in XML format therefore any annotations of genes etc will have the characters <, >, & ,' and " replaced with the relevant entity reference.

5.1.1 Genes

For each taxon, the genes found are listed using the gene tag. A maxinum of one gene is inferred for each aligned read, based on its position on the genome. The "count" field describes the number of reads that overlapped with that particular gene. The protein, locus, product, GeneID, reference sequence are listed (using NCBI data) along with the cds range. The same gene may appear in multiple taxons.

5.2 Per read taxonomy ID

Mapped reads are listed with their LCA taxonomy. Unmapped reads are not listed. Output is written to file with suffix "_per_read"

5.3 Abbreviated taxonomy

Each identified taxon is listed along with the percentage of reads that mapped to it. Output is written to file with suffix "_abbreviated"

5.4 SAM

Output in the Sequence Alignment/Map format [2]. A Bowtie [5] style output is used (for each read there is a primary line for each reference that it aligned to). A BWA [6] style XA tag can be printed (using the –sam-xa parameter) instead of listing all hits. Unmapped reads are not printed except when they belong to a pair where the other read was mapped. The following k-SLAM specific tags are used:

- XS: alignment score assigned by k-SLAM, using pseudo assembly.
- XO: number of hits for this segment.
- XT: taxonomy ID of this reference.
- XG: gene at this position in the reference.
- XP: protein ID of this gene.
- XR: product of this gene.
- XA: BWA style alternate hits in format (chr,pos,CIGAR,NM;)* (this tag is optional).

References

[1] Mengyao Zhao, Wan-Ping Lee, Erik P Garrison, and Gabor T Marth. Ssw library: An simd smith-waterman c/c++ library for use in genomic applications. *PloS one*, 8(12):e82138, 2013.

[2] Sam format specification. `https://samtools.github.io/hts-specs/SAMv1.pdf`. [Online; accessed 30-July-2014].

[3] Sample genbank record. `http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html`. [Online; accessed 30-July-2014].

[4] Ncbi taxonomy. `http://www.ncbi.nlm.nih.gov/taxonomy`. [Online; accessed 30-July-2014].

[5] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357–359, 2012.

[6] Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.