The Delve Manual

by Timo Lassmann

February 28, 2011

Contents

Overview

1.1 Background

There are many tools available to align or map sequenced reads to the genomes. The advent of next generation sequencing made it necessary to develop such tools since more traditional aligners such as Blat[2] were either too slow or too inaccurate when aligning millions of short reads. It is therefore understandable that the initial focus was on developing fast methods. However, it is now obvious that many reads cannot be placed accurately back to the genome. Often reads can map to multiple locations or reads originating from one loci are mapped incorrectly to another.

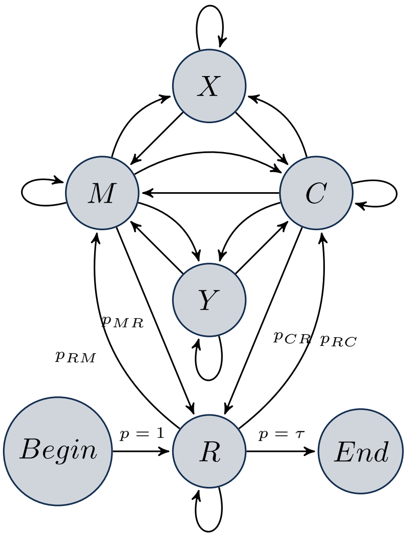
Delve is an aligner that strives to give the most accurate alignments possible. The high accuracy is achieved by modeling alignments using a pair-HMM model. All model parameters are directly estimated from the input sequences making it possible to detect biases towards individual types of mismatches, insertions and deletions. By taking these biases into account while mapping, Delve places reads to their most likely location.

1.2 Motivation

When mapping reads of length $\leq$ 30nt many reads are placed in wrong locations or are ambiguously mapped to the genome. By simulation it is possible to show this is a problem even at low to moderate error rates. The underlying cause is the repetitive nature of the genome which causes distant segments to be very similar to each other. Due to sequencing errors reads can map with fewer mismatches to a secondary location than their true location.

While the nature of the genome cannot be changed it is possible to improve the discrimination between several genomic hits of the same reads. Our method Delve models error frequencies accurately and thus can give a higher probability to a location with a frequently occurring errors than a location with a very rare errors.

This manual describes all the commands implemented in Delve.



pair-HMM model used in Delve

1.3 Available Commands

Delve includes several command which are useful for aligning reads to the genome. All commands use standard formats as input and output (fastq, SAM and BEDgraph).

index	Builds simple index of genome. Required for 'align'.
seed	Finds putative hit locations for all reads. Output is SAM with 'XA:Z:' tags indicating additional mapping locations. Cigar lines and mapping qualities are absent.
align	Estimates substitution, insertion and deletion probabilities for the pair-HMM model. Then it uses the model to assign mapping probabilities to all putative mapping locations. The output is the pHMM in binary format and the top 'X' mapping locations in SAM format.
show	Prints model parameters.
post	Calculates the posterior probabilities for each base of the genome being part of an alignment. Requires a SAM file with accurately calculated mapping probabilities as input. Output are two BEDgraph files for plus and minus strand respectively.
realign	Takes a SAM file and realigns all reads to locations indicated by the sam line including 'XA:Z:' tags using normal dynamic programming.
sim	Mutates sequences in a fastq file generated by wgsim (part of the SAM package) for testing.
bedsim	Simulates sequences within boundaries defined in a BED file.
eval	Evaluates the mapping of simulated reads.

1.4 Delve Workflow

The workflow on the right indicates which Delve commands have to be executed to obtain high quality read alignments.

Two important stages include generating putative hit locations with the **seed** command and the subsequent fine alignment using the **align** command.

In total Delve can produce three different types of SAM files:

1. Generated by **seed**: includes putative hits in the 'XA:Z:' tag. This is used as a temporary file and does not contain cigar lines or mapping qualities.
2. Generated by **realign**: removes 'XA:Z:' tags and prints one line per hit including the CIGAR line.
3. Generated by **align**: includes mapping quality and CIGAR lines for the top 'X' hits as determined by the forward algorithm. This file is the main output of Delve and should be used for analysis (shown in bold on the right).

The final SAM file is used as input by **post** to generate a single nucleotide resolution map indicating which genomic residues are part of an alignment - or are expressed.

The Delve distribution contains shell scripts to automate these steps.

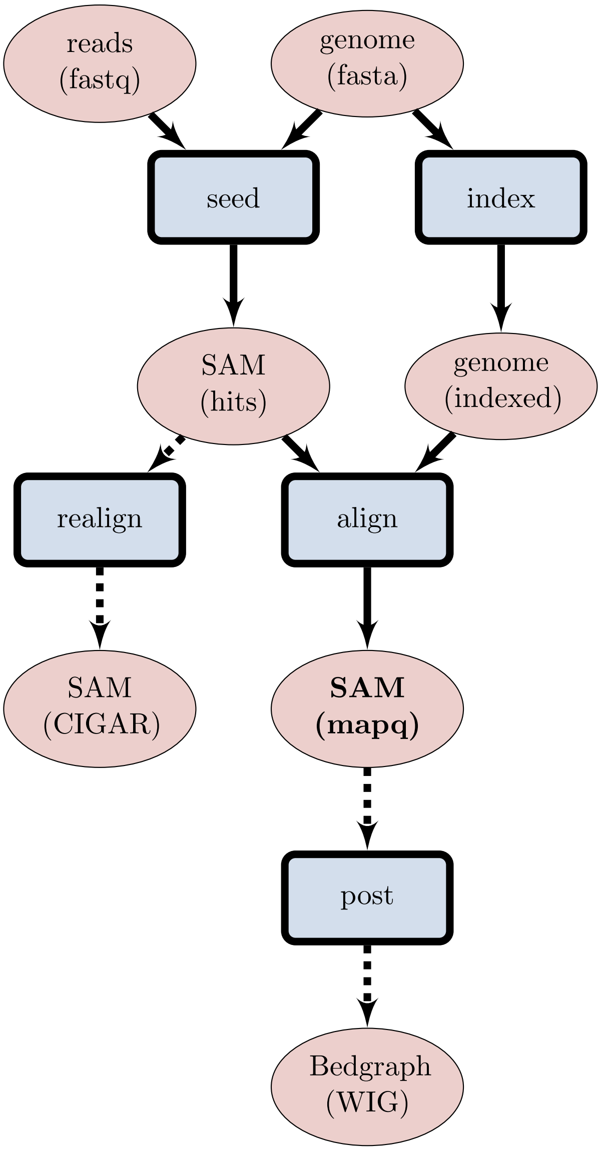


Figure 1.1: Delve workflow. Required steps are shown using solid arrows.

General Information

2.1 Usage

All commands are accessible by calling Delve first. For example:

```
bash-3.1$ delve index genome.fa
```

- builds an index of the genome.

```
bash-3.1$ delve realign alignments.sam
```

- realigns reads of the genome.

Concise help is given by calling Delve with a command but without arguments.

2.2 Formats

Delve employs widely used file formats to make integration into existing pipelines easier.

2.2.1 FASTA

Delve expects the genome to be in a single file standard fasta format. Reads can also be supplied in fasta format but fastq is preferred.

2.2.2 FASTQ

When available the reads should be supplied in fastq[1] format. The actual quality values are not used by Delve at this point but are parsed for downstream analysis to the SAM output files.

2.2.3 SAM / BAM

Alignments are given in SAM[4] format. For an intermediate SAM file we adopted the 'XA:Z:' tag for alternative hits used by BWA[3] to indicate additional alignments for each read in one line. In brief, the XA tag lists for each hit the chromosome, position, CIGAR line and number of mismatches separated by ';'. Specific to Delve the 'XP:Z:' tags lists the posterior probability of each nucleotide in the read to be aligned to the genome.

2.2.4 BEDgraph / WIG

A description of these formats can be found at the UCSC genome browser site:

- BEDgraph
- WIG

Commands

This section gives a detailed explanation of each of Delve's commands.

3.1 index

Delve occasionally needs to extract sequences from the genome. The index command creates the necessary files in the same directory as the input genome.

Usage: delve index <genome.fa>

3.2 seed

Seed finds short seed matches for input reads and verifies each hit with Myers bit-parallel dynamic programming algorithm[5]. The output is a SAM file with the top 18 hits for each sequence. At this stage hits are ranked based on the edit distance. The best hit is reported in the default SAM format while additional hits are given in the 'XA:Z:' tag.

Usage: delve seed [**options**] <reads.fq> <genome.fa>

OPTION	TYPE	DESCRIPTION
-l	INT	seed length [12]
-s	INT	step size [8]
-t	INT	number of threads [4]
-o	STR	output SAM file [STDOUT]
-as	STR	adds assembly infomation to SAM header [NA]

Longer seed lengths decreases the running time of Delve drastically risks missing hits.

Example of output format(one SAM line):

```
chr1_31332794_31333336_0:0:0_0:0:0_1 0 chr1 31332794 20 * * 0 0
   ACATAGACTTCAAAACTCAAACCAAATATTTC !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! NM:i:0
   XA:Z:chr10,-109598328,32M,5;chr5,-156754108,32M,5;chr4,+27656642,32M,5;chrX,+146319453,32M;
```

3.3 align

The align option both estimates parameters for the pair-HMM and then uses the parameterized model to rank alignments based on their fill probability calculated by the forward algorithm. Additionally the posterior probability for each nucleotide in the read coming from the genome is calculated. These are appended to the SAM file using an optional 'XP:Z' tag. Probabilities are phred scaled and assigned to letters in exactly the same way as Sanger base qualities. For testing purposes it is possible to use viterbi decoding for scoring but this is not recommended.

Usage: delve align [**options**] <in.sam> <genome.fa>

Option	Type	Description
-m	STR	model file [off]
-u	INT	number of reported alignments [1]
-t	INT	number of threads [4]
-o	STR	output SAM file [STDOUT]
-ts	INT	minimum number of reads to train model (in millions) [1]
-lw	N/A	use local density of mappings score alignments[off]
-as	STR	adds assembly infomation to SAM header [NA]
-viterbi	N/A	use viterbi algorithm to score hits [off]

The -m option has two functions: if no model exists the trained model parameters are written to file STR for future use otherwise model parameters are read from the file and the training is skipped.

The -u option allows users to print out sub-optimal alignments for the same read. This may be desired for downstream method implementing other strategies to resolve ambigiously mapping tags.

Example of output format(one SAM line):

```
chr1_31332794_31333336_0:0:0_0:0:0_1 0 chr1 31332794 50 32M * 0 0
   ACATAGACTTCAAAACTCAAACCAAATATTTC !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
   NM:i:0 XP:Z:C@ACBBCB?>A;;;;DEC=>=BB===BJ=<=M
```

3.4 show

Reads model file generated by **align** and prints emission and transition parameters to STDOUT.

Usage: delve show <in.model>

3.5 post

This option calculates the posterior probabilities of each nucleotide of the genome being part of an alignment. The calculations are analogous to those generating the 'XP:Z' tags for the **align** command.

Warning: This can take more than 16gb of memory for datasets which cover large proportions of the genome.

Usage: delve post [**options**] -m <model file> <in.sam> <genome.fa> -o <output file>

OPTION	TYPE	DESCRIPTION
-o	STR	BEDgraph output files [STDOUT]
-m	STR	model file [required]
-wig	NA	generates WIG formated files

The **post** command will generate one file for the plus and minus strand.

3.6 realign

Realigns sequences based on hit locations given in a SAM file. This is useful for creating a standard SAM file from a SAM file listing several hit locations using the 'XA:Z' tag. Delve uses standard dynamic programming with gap and mismatch penalties set to 1.

Usage: delve realign [**options**] <in.sam> <genome.fa>

OPTION	TYPE	DESCRIPTION
-u	INT	number of reported alignments [1]
-t	INT	number of threads [4]
-o	STR	output SAM file [STDOUT]
-as	STR	adds assembly infomation to SAM header [NA]

3.7 sim

Used for development purposes. The input are reads generated by the SAMtools program wgsim.

Usage: delve sim <reads.fq> FLT FLT FLT FLT FLT

Each FLT corresponds to:

NUMBER	DESCRIPTION
1	error rate at 5' end [0.01]
2	error rate at 3' end [0.03]
3	mismatch rate [0.8]
4	probability of 5' G addition [0.0]
5	probability of 3' A addition [0.0]

3.8 bedsim

Simulates reads from regions defined in an input BED file.

Usage: delve bedsim [**options**] <in.bed> <genome.fa>

OPTION	TYPE	DESCRIPTION
-read-len	INT	length of simulated reads [30]
-start-error	FLT	error rate at 5' end [0.02]
-stop-error	FLT	error rate at 3' end [0.03]
-mismatch-rate	FLT	fraction of simulated mismatches [0.8]
-sw	INT	simulate reads INT nucleotides up- and downstream of the start of BED entries [0]

3.9 eval

Evaluates the mapping of reads generated by **bedsim**.

Usage: delve eval <in.sam> <genome.fa>

The output is a table reporting the the number of misplaced tags and the total number of mapped tags within an alignment quality bin.

Scripts

Scripts to automate the workflow can be found in the 'scripts' directory.

4.1 run_delve.sh

This script runs the seed and align command on all fasta and fastq files within a directory. The following output files are generated in the same directory:

1. XXX.sam - seed alignments in SAM format.
2. XXX.hmm.sam - final alignments in SAM format.
3. XXX.mdl - the model file.

where 'XXX' is the name of the input fasta / fastq file.

On systems running the SUN Grid Engine the script will use qsub and submit multiple jobs in parallel.

Usage: run_delve.sh <directory>

Bibliography

[1] Peter J A Cock, Christopher J Fields, Naohisa Goto, Michael L Heuer, and Peter M Rice. The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic Acids Res*, Dec 2009.

[2] W Kent. Blat-the blast-like alignment tool. *Genome Res*, Jan 2002.

[3] H Li and R Durbin. Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, May 2009.

[4] H Li, B Handsaker, A Wysoker, T Fennell, J Ruan, N Homer, G Marth, G Abecasis, R Durbin, and 1000 Subgroup. The sequence alignment/map (sam) format and samtools. *Bioinformatics*, Jun 2009.

[5] Gene Myers. A fast bit-vector algorithm for approximate string matching based on dynamic programming. Mar 1998.