YAHA User Guide

Last updated: 14-Nov-2013
Current as of YAHA version 0.1.82

Gregory G. Faust and Ira M. Hall
University of Virginia

Contact: gf4ea@virginia.edu

Please Cite: Faust GG and Hall IM. *YAHA: fast and flexible long-read alignment with optimal breakpoint detection.* Bioinformatics (2012) **28**(19): 2417-2424. Epub 2012 Jul 24. doi:10.1093/bioinformatics.

Contents

1. Introduction

YAHA is a flexible, sensitive and accurate hash-based DNA aligner designed for relatively long queries. It supports three major modes of operation. In its default "Optimal Query Coverage" (**-OQC**) mode, it will report the best set of alignments that cover the length of each query. These alignments are called the "Optimal Coverage Set" (OCS) or "primary" alignments. In the second mode, called "Filter By Similarity" (**-FBS**), along with the primary alignments for each query, YAHA will also output any "secondary" alignment that is highly similar to one of the primary alignments in both the region of the query it covers and its alignment score. Finally, it can output all the alignments found for each query. Affine Gap Scoring is used to determine the degree to which the query matches a given reference region. All these modes share a number of parameters that control the sensitivity of the aligner, allowing the user to trade-off CPU runtimes for higher accuracy. These are described in detail below.

The **-OQC** and **-FBS** modes are specifically tuned to form split read mappings that can be used to accurately identify structural variation events (deletions, duplications, insertions or inversions) between the subject query and the reference genome. The algorithms used in these modes are unique to YAHA. Parameters control the size and genomic distance involved in reporting split alignments. These in turn allow the user to tune the aligner for the types of SV event of interest.

2. Running YAHA

In order to use YAHA to align queries, one must first generate an index of the reference genome. Depending on the size of the reference genome and index parameters used, the YAHA index can be quite large. For example, using the default indexing parameters for a human genome will result in an index size of about 16GB. Therefore, we recommend you run YAHA on a server class machine with enough physical memory to avoid excessive paging. However, if the same index is used by more than one YAHA process running on a machine, the memory for that index will be shared across those processes. Smaller, but possibly less sensitive, indexes can be created by using different values for the index creation parameters as described below.

The index is composed of two files, the names of which start with the root name of the input FASTA file for the reference genome. The first file is a compressed version of the genome and has a ".nib2" extension. The second file is the index itself. The name for this file will be of the form "*GenomeRoot*.X*LL_SS_HHHHH*S" where the "*LL*" represents the *word-length* used, "*SS*" represents the *skip-distance* used, and "*HHHHH*" represents the *max-hits* used when creating the index (see below for a description of these parameters). The compressed genome file will be shared by many indices made from different parameter settings if all the files are kept in the same directory, with the same root name. The index file has the *word-length* and *max-hits* parameters used during its creation stored in its header.

When aligning queries, one must specify the index file to use with the **–x** option. This will implicitly select the reference file from which the index was made. In addition, the **–q** option must be used to

specify the sequence reads to align. This must be either a FASTA or FASTQ file. See the Examples section at the end of this document for sample parameter settings for index creation and usage.

YAHA currently is supported on 64 bit Linux derived UNIX platforms. If you have any problems running YAHA in such an environment, do not hesitate to contact us at the email address given above.

3. YAHA Parameters

Most YAHA parameters are key-value pairs. In what follows, we will use **bold face** to indicate parameter keys, and *italics* to indicate parameter values. Also, parameters that control the operation of YAHA as a whole have lower case key names, while those that act as input to the various algorithms in YAHA have upper case key names. YAHA uses no positional parameters. Parameter order is completely meaningless except that if a parameter is specified twice on the command line, the right-most value will be used. Parameters of type "bool" will accept any of "Y", "y", "T", or "t" to mean yes or true, and "N", "n", "F", or "f" for no or false. Parameters of type "int" must have a positive integer value. Parameters of type "float" must take a value in the range $0<value\leq 1.0$;

3.1. Index Creation Parameters.

These parameters specify characteristics of the hash table index used by YAHA. To create an index, simply supply YAHA with a genome file (**–g** option), but do not specify a query file (**-q** option) or index file (**-x** option). For example, to create an index using default parameters, simply type:

```
yaha –g hg18_full.fasta
```

Key	Value	Type	Default	Usage
-g	*genome-file-name*		required	The reference genome to be indexed. It can be a FASTA file, or a .nib2 file. If it is a FASTA file, the .nib2 is automatically created.
-H	*max-Hits*	int≤65525	65525	During index creation, k-mers that appear more than *max-Hits* number of times in the reference genome will be randomly sampled down to *max-Hits* reference locations.
-L	*seed-Length*	int≤15	15	The length of k-mer seeds in the index.
-S	*Skip-distance*	1≤int≤*seed-length*	1	The distance to skip ahead on the reference before forming the next k-mer to index.

3.2. Alignment Input/Output Parameters.

These parameters control the input/output behavior of the alignment run. At most one of **–osh**, **-oss**, or **–o8** can be specified. The value for any of these three output parameters can be 'stdout' in which case the output will be sent to `stdout`. By default, the output will sent to `stdout` in hard clipped SAM format. For example, to use all default parameters, type the following:

```
yaha –q myqueries.fasta –x hg18_full.X15_01_65525S
```

Key	Value	Default	Usage
-q	*query-file-name*	required	This is the name of the file containing sequence data to be aligned. It can be a FASTA or FASTQ file.
-osh	*sam-file-name*		Output the alignments in SAM format with hard clipping. *sam-file-name* can be `stdout`.
-oss	*sam-file-name*		Output the alignments in SAM format with soft clipping. *sam-file-name* can be `stdout`.
-o8	*blast8-file-name*		Output the alignments in BLAST8 format. *blast8-file-name* can be `stdout`.
-t	*numThreads*	1	Number of threads to use.
-v			Output more verbose information to `stderr`.
-x	*index-file-name*	required	Specifies the name of an index file previously created by YAHA.

3.3. General Alignment Parameters.

In what follows, we will use the term "gap" to refer to a small insertion or deletion (indel). We use the term "desert" to refer to stretches of an alignment between seed matches.

Key	Value	Type	Default	Usage
-BW	*BandWidth*	int	5	The width **on either side** of the diagonal of the banded Smith-Waterman calculations during alignment. YAHA only uses SW to align between seed hits and for extensions. When aligning between seeds, the band will be the size of the gap between seeds (if any) + 2 **BandWidth*. During alignment extensions, the band will be 1 + 4 * *BandWidth*.
-G	*max-Gap*	int	50	The maximum size of an insertion/deletion allowed within an alignment. Gaps larger than this will result in a split alignment.
-H	*max-Hits*	int	650	During query alignment, k-mers that appear more than *max-Hits* number of times will be ignored. For an alignment run, the *max-Hits* value used will be the minimum of the value specified in the alignment run and the value specified at index creation. To use YAHA's reference sampling capability, use the same value for *max-Hits* during index creation and alignment.
-M	*min-Match*	int	25	The minimum total number of bases in seed matches to initiate an alignment.
-MD	*Max-Desert*	int	50	The maximum stretch along a potential alignment without seed matches before the alignment will be broken into two. This filter is applied before SW is run in the "desert" to see what matching bases may be there. It helps to eliminate "bar-bell" alignments that have a high score on both ends, a low score in the middle, but the middle negative score is not sufficient to make an invalid local alignment. This can occur, for example, with a local inversion embedded in a long read.
-P	*%-min-identity*	0<float≤1	0.9	The minimum ratio of matches/alignment-length. Alignments that fall below this threshold are not output.
-X	*X-dropoff*	int	25	When the score for an alignment extension falls more than *X-dropoff* below the best extension, the search for a longer extension will cease.

3.4. Affine Gap Scoring Parameters.

Note that the cost parameters (**-GEC**, **-GOC**, and **-RC**) are all specified as positive integers.

Key	Value	Type	Default	Usage
-AGS	*use-AGS*	bool	Y	This parameter controls use of Affine Gap Scoring (AGS). If AGS is turned off, all cost/score parameters below are ignored, and a simple edit distance (Levenschtein) score calculation will be used.
-GEC	*Gap-Extension-Cost*	int	2	The cost of opening a new insertion or deletion.
-GOC	*Gap-Open-Cost*	int	5	The cost for each base in an insertion or deletion.
-MS	*Match-Score*	int	1	The positive score added for each matching base.
-RC	*Replacement-Cost*	int	3	The cost of a mis-matched base.

3.5. Optimal Query Coverage and Filter by Similarity Parameters.

Some comment about why this is useful for SV detection.

Some comment about how the breakpoint penalty is calculated.

Key	Value	Type	Default	Usage
-OQC	*use-OQC*	bool	Y	This parameter controls use of the optional "Optimal Query Coverage" algorithm. If this is turned off, all alignments meeting other threshold parameters will be output, and all parameters below will be ignored. With this turned on, the aligner will search for the best collection of alignments that cover the length of each query; the Optimal Coverage Set (OCS).
-BP	*Breakpoint-Penalty*	int	5	This is one of the two factors that determine the total penalty for including an additional alignment in the OCS. See notes above.
-MGDP	*Max-Genomic-Distance-Penalty*	1≤int≤9	5	The second of the two factors that determines the total penalty for including an additional alignment in the OCS. This controls the maximum genomic distance penalty that can be assessed on a $\log_{10}$ scale. For example, **-MGDP** *5* indicates the genomic distance penalty will be maxed out at 100K base pairs.
-MNO	*Min-Non-Overlap*	int	*min-match*	Two adjoining alignments must each cover at least this amount of non-overlapping region on the query to be included in the OCS.
-FBS	*use-FBS*	bool	N	This parameter (ignored if **-OQC** is off) controls use of the "Filter By Similarity" algorithm. When on, alignments "similar" to the OCS are also included in the output. The selection of similar alignments is controlled by the following parameters, which are ignored when **-FBS** is off. A similar alignment must satisfy BOTH of the following thresholds.
-PRL	*%-Reciprocal-Length*	0<float≤1	0.9	An alignment will be considered similar to another only if their lengths are ≥ this percent reciprocally overlapping. That is, the overlapping region of the two alignments must cover at least this percentage of the length of BOTH of the alignments.
-PSS	*%-Similar-Score*	0<float≤1	0.9	An alignment will be considered similar to another only if the score of the secondary alignment is ≥ this percentage of the primary alignment score.

4. YAHA Output

The primary output format is SAM, although there is also an option to output Blast8 format. The standard SAM output is augmented with four YAHA specific tag fields described below. All tag ids start with the letter Y to indicate YAHA. The majority of the tags help to identify primary alignments, their position along the query, and their corresponding secondary alignments. As the SAM format standard evolves, there may be ways of expressing this information in a standardized way. If and when that happens, YAHA will comply with such standards.

Tag Id	Type	Mnemonic	Usage
YF	x	YAHA FLAGS	This tag is composed of a number of status bits (ala the SAM FLAG field). The only bit currently of interest to users is 0x20. When YAHA is run in **-OQC** mode, this bit will be set for primary alignments, but not for secondary alignments. When outputting all alignments, this bit is never set. THIS TAG FIELD HAD TAG ID "YS" IN YAHA RELEASES PRIOR TO 0.1.78.
YI	i	YAHA INDEX	Indicates which position a primary or secondary alignment falls along the query from left to right, starting with 1 for the left most. Note that this takes strand into account, so it is the part of the query that is covered that determines order. This can be used to determine which alignments are adjacent, and therefore could define a breakpoint. For secondary alignments, this will also indicate which primary it corresponds to.
YP	i	YAHA PRIMARIES	Indicates the total number of primary alignments for this query. Taken together with YI, one can tell if an alignment is in position, say, number 2 of 3. This is also very useful to pull split alignments out of a file (if YP > 1).
YS	i	YAHA SECONDARIES	As of now, this tag only appears on primary alignments and tells how many secondary alignments pass the FilterBySimilarity criteria. If FBS is on, then this will tell you how many secondaries to look for in the file that correspond to this primary. If FBS off, it still gives you the count of secondaries that WOULD have been output if FBS on. This can be useful to determine uniqueness (or lack thereof) for a primary alignment, using the tunable FBS parameters for % reciprocal overlap and minimum %score. THIS TAG FIELD HAD TAG ID "YC" IN YAHA RELEASES PRIOR TO 0.1.78.

5. Examples

1. To create an index with the default, maximally sensitive parameters:
   ```
   $ yaha -g hg18_full.fa
   ```
   This will create two files called `hg18_full.nib2` and `hg18_full.X15_01_65525S` in the same directory that the input genome file was located.
2. To use the above index to find only primary alignments for a queries in a FASTA file, and create hard clipped sam output to `stdout`:
   ```
   $ yaha -q myqueries.fa -x hg18_full.X15_01_65525S
   ```
3. To output the primary alignments as well any secondary alignments similar to them:
   ```
   $ yaha -q myqueries.fa -x hg18_full.X15_01_65525S -FBS Y
   ```
4. To do the same as 2 above, but use soft clipping and create a bam file for the alignments (Note: this requires SAMtools):
   ```
   $ yaha -q myqueries.fa -x hg18_full.X15_01_65525S -oss stdout
          | samtools view -Sb - > myqueries.bam
   ```
5. To do the same as 2 above, but input from a gzipped input file.
   ```
   $ zcat myqueries.fa.gz | yaha -q stdin -x hg18_full.X15_01_65525S
   ```
6. To create an index with a *seed-size* of 13 a *skip-distance* of 13, and a *max-hits* of 10,000, such as is used by default by SSAHA2:
   ```
   $ yaha -g hg18_full.fa -L 13 -S 13 -H 10000
   ```
7. To use the default index created above to output all alignments for queries in a FASTA file, and create a sam file with hard clipping:
   ```
   $ yaha -q myqueries.fa -x hg18_full.X15_01_65525S -osh myqueries.sam
          -OQC N
   ```
8. To do the same as in 7 above but increase sensitivity to repetitive regions:
   ```
   $ yaha -q myqueries.fa -x hg18_full.X15_01_65525S -osh myqueries.sam
          -H 2000 -OQC N
   ```
9. To maximize sensitivity to repetitive regions, use the power of the sampled index by using the same *max-hits* as is used while creating the index:
   ```
   $ yaha -q myqueries.fa -x hg18_full.X15_01_65525S -osh myqueries.sam
          -H 65525 -OQC N
   $ yaha -q myqueries.fa -x hg18_full.X13_13_10000S -osh myqueries.sam
          -H 10000 -OQC N
   ```
10. To do the same as in 7 above, but instead lower the *min-identity* to include in the output alignments with less similarity with the reference genome:
    ```
    $ yaha -q myqueries.fa -x hg18_full.X15_01_65525S -osh myqueries.sam
          -OQC N -P 0.8
    ```
11. To do the same as in 1 above using 16 threads.
    ```
    $ yaha -t 16 -q myqueries.fa -x hg18_full.X15_01_65525S
    ```

6. Release History

Version 0.1.82, November 14, 2013

Algorithmic Update:

Improved a heuristic that in rare cases produces better primary alignments in complex genomic regions.

Bug Fix:

1. Fixed a bug that caused YAHA to give incorrect YP and YI tag values is very rare cases.

Version 0.1.79, July 11, 2013

Bug Fix:

1. Fixed a bug that caused YAHA to create invalid compressed genome (.nib2) files and index files from reference fasta files that contain certain non-base characters such as carriage-return. This is particularly important for reference genomes from Windows that use carriage-return and line-feed for the end of line indicator. If you have such a reference genome, please delete the .nib2 file and all of your indexes for it, and recreate them using this YAHA release.

Version 0.1.78, June 6, 2013

New Features:

1. YAHA nows works with reference genomes containing an indefinite number of sequences. This can be useful, for example, when you wish to align to a set up contigs or scaffolds that have not yet been fully assembled into chromosomes. To provide this feature, it was necessary to update the format of the .nib2 file which stores the compressed reference genome. However, YAHA can use either the old or new format, so there is no need to recreate the .nib2 file or any indexes you already have.
2. The YAHA specific SAM tag fields have been upgraded in an **INCOMPATIBLE** way. The changes make it much easier to process primary alignments for reads that produce split-read mappings to find breakpoints, as well as processing their corresponding secondary alignments when using FBS mode. Please read Section 4 for details.

Bug Fixes:

1. Fixed a bug that caused incorrect calculation of alignment mapping quality in rare circumstances.
2. Fixed another bug that crashed YAHA in rare circumstances with alignments mapping near to offset 0 in the reference.

Version 0.1.72, Feb 14, 2013

New Feature:

Added the ability to use multiple threads during alignment. All of the alignments for a given query will be in contiguous lines of the resultant sam file, but not necessarily in the same order that they appear in the input fasta/fastq file.

Bug Fixes:

1. Changed index creation to truncate reference sequence names at the first whitespace character. For example, in the hg19 reference genome, chromosome 1 is labeled with "1 dna:chromosome chromosome:GRCh37:1:1:249250621:1" in the fasta file, but alignments to chromosome 1 will have the sequence name truncated to "1".
2. Fixed a few bugs that crashed YAHA in rare circumstances.

Version 0.1.64, Dec 3, 2012

Algorithmic Update:

Improved heuristics leading to increased sensitivity with approximately 1% more generated alignments for comparable parameter settings. These additional alignments are all reported when not using OQC. When using OQC, they are used to potentially generate better primary alignment sets. These changes marginally improved essentially all statistics reported in the original paper.

New Features:

1. Added ability to read fasta and fastq files with newlines in the query and/or quality strings.
2. Added ability to read query files from stdin. In particular, one can now pipe gzipped fasta or fastq files into YAHA using zcat.
3. Improved error messages throughout.

Bug Fixes:

1. Alignments at offset 0 in the reference genome should no longer crash YAHA. Such alignments are particularly common in bacterial genomes due to the break in the circular chromosome.
2. Fixed a bug that incorrectly reported starting reference offsets for soft clipped alignments.
3. Fixed a bug in which *seed-Length* was taken from the parameter settings during alignment. YAHA now properly loads *seed-Length* from the index file.
4. Fixed a bug that crashed YAHA if given an index filename with no extension.

Version 0.1.38, May 3, 2012

Initial Release

7. Current YAHA Limitations

1. YAHA is a long read DNA alignment tool. It does not take advantage of the insert distance between paired-end reads. That is, both ends of the pair-end read will be aligned separately. It also does not align protein sequences, but of course can be used to map cDNA reads.
2. YAHA is currently available in binary form only, for 64-bit LINUX based systems. We intend to make YAHA available as an open source tool in the near future.
3. YAHA cannot index reference genomes greater than 4 billion base pairs in length.
4. YAHA currently cannot align read of greater than 32 kilobases in length. We expect to remove this restriction in the near future.