http://research.cs.wisc.edu/wham/manual/ Go
16 captures
10 Sep 2012 - 30 Jan 2019
NOV JAN FEB
30
2016 2019 2020
About this capture

WHAM

WISCONSIN'S HIGH-THROUGHPUT ALIGNMENT METHOD

MAIN TUTORIAL MANUAL DOWNLOAD
Search

WHAM Index Builder

WHAM index builder is responsible for building indices on reference sequences. A built index is stored on diskreferenced by a index name. An index contains multiple files: `<idxname>.head.whm`, `<idxname>.interval.whm`, `<idxname>.i0.whm`, `<idxname>.i1.whm`, `...`, and`<idxname>.sequence.whm`. These files together constitute the index: they are all that is needed to align reads to that reference. The original sequence files are no longer used by WHAM once the index is built.

WHAM uses 32-bit pointer internally, and thus can handle a maximum of 2^32-1 (around 4 billions) characters in reference sequence(s). If your sequence is longer than 2^32-1 characters, you can split the reference sequence(s) into smaller pieces, and build a separate index on each.

A WHAM index is specified to the length of reads and the number of maximum errors. If you are working on the data sets with a variety of read lengths (and error numbers), you can build multiple indices and store them on disk. When you align a particular query file, specific the index built with the same read length and error number.

The index size primarily depends on many parameters: the length of reads, the number of errors, and so on. For example, an index built on the whole non masked human genome for aligning 60bps short reads with up to 2 mismatches (options: `-v 2 -l 60 --mask`) is around 9GB. Typically, an index built for shorter reads and more errors requires more space. If your machine has no sufficient memory to hold the index, you can break up the reference sequence(s) into smaller pieces, and build a separate index for each.

Usage:

```
wham-build [OPTIONS]* -l <int> <FILE...> <IDXNAME>
```

Parameters:

`-l <int>` Specifies the length of short reads for the subsequent query file. New Indices must be built if the length changes.

`<FILE...>` A comma-separated (no spaces) list of files containing reference sequences. Reference sequences must be in the FASTA file format.

`<IDXNAME>` Specifies the index path and name (`path/name`). The WHAM builder writes new files: `path/name.head.whm`,`path/name.interval.whm`, `path/name.i0.whm`, `path/name.i1.whm`, …, and `path/name.sequence.whm`.

Options:

`-v <int>` Report hits with less than or equal to <int> errors. In the alignment phase, WHAM can find ALL valid alignments with up to <int> errors, and can also report SOME valid alignments with more than <int> errors. See -v option in the alignment phase for more details. The default value is 2, and possible values are 0-5.

`-p <int>` Specifies the number of fragments for alignments. <int> must be greater than the number of errors (specified by -v). Using a greater value of <int> requires more space for the index, but may improve the performance by reducing the hashing collisions if the reads are relatively short. By default, the WHAM cost model picks an <int> with near-optimal performance.

`-b <int>` Specifies the number of buckets. WHAM will then pick a prime number that is greater than or equal to <int>. Using a greater value of <int> improve the performance, but require more space to store the index. By default, the WHAM chooses a value same to the number of characters in the reference sequence(s).

`-a` Create indices that can find ALL valid matches. In this mode, WHAM is slower, and needs much more memory (default: off).

`--mask` Keep masked (lowercase) characters in the sequences (default: on).

`--unmask` Discard masked (lowercase) characters in the sequences. Masks are treated as Ns, and thus, omitted.

`-h/--help` Print the usage message.

WHAM Aligner

WHAM aligner is responsible for computing the alignments between query file(s) and a WHAM index. The index is pre-built on data using the WHAM index builder command. The used index must match the requirements of alignments including the length of shot reads, the number of allowed mismatches. Alignments are outputted into a file(s). By default, WHAM outputs alignments in a format similar to Bowtie. Each alignment is a separate line in the output file, and contains five fields:

- Reference strand aligned to, `+` for forward strand, `-` for reverse
- Name of reference sequence where alignment occurs
- 0-based offset into the forward reference strand where leftmost character of the alignment occurs
- Read sequence (reverse-complemented if orientation is -).
- Comma-separated list of mismatch descriptors. If there are no mismatches in the alignment, this field is empty. A single descriptor has the format offset:reference-base>read-base. The offset is expressed as a 0-based offset from the high-quality end of the read.

WHAM can also output alignments in SAM format. See the SAM Format Specification for more details.

Validity of alignments is determined by the alignment policy specified by options `-v`, `-e`, `--nofw/--norc`, and so on. The number of mismatches is the basic parameter to determine the alignment policy. In addition to the number of mismatches, the quality values can be used to filter out low quality alignment and to sort alignments in an order of high-to-low quality. WHAM can also be configured to align to forward and/or reverse-complement of the reference strand.

The number of maximum mismatches is a key parameter to determine the alignment policy. To find alignments with `k` mismatches, the first way is to build an index with the option `-v k`. However, if the value of `k` is large or the length of reads is relatively short. The index associated with `k`mismatches may become so large that it cannot fit into memory, and thus reduce the performance. An alternative way is to build an index with the less mismatches, e.g. `-v k' (k' < k)`. Then, specifies `-v k` in the aligner. In this way, WHAM can find all valid alignments with up to `k'`mismatches, and find some valid alignments with more than `k'` mismatches. The performance of the second method is much higher than that of the first one, in expense of missing some valid alignments. In practice, if `k'` is close to `k`, WHAM can find most of valid alignments.

The performance of WHAM aligner primarily depends on two parameters: the read length and the number of maximum mismatches. WHAM runs faster for longer reads and less mismatches. Theoretically, WHAM achieves the best performance when `l/(k+1) > 15`, where `l` is the number of characters in each read, and `k` is the number of mismatches.

Usage:

```
wham [OPTIONS]∗ {<inFILE...> | −1 <m1FILE...> −2 <m2FILE...>} <IDXNAME> <outFILE>
```

Parameters:

`<inFILE...>` A comma-separated list of files containing unpaired reads (used only for single-end reads). Reads must be in the FASTQ file format. All reads in the files mush have the same length.

`<m1FILE...>` A comma-separated list of files containing upstream mates (used only for pair-end reads). Reads specified in `<m1FILE...>` must correspond file-for-file and read-for-read with those specified in `<m2FILE...>`. Reads must be in the FASTQ file format. All reads in`<m1FILE...>` and `<m2FILE...>` must have the same length.

`<m2FILE...>` A comma-separated list of files containing downstream mates (used only for pair-end reads). Reads specified in `<m2FILE...>` must correspond file-for-file and read-for-read with those specified in `<m1FILE...>`. Reads must be in the FASTQ file format. All reads in`<m1FILE...>` and `<m2FILE...>` must have the same length.

`<IDXNAME>` Specifies the index path and name. The read length of the index should either match the length of reads specified in `<inFILE...>`,`<m1FILE...>`, and `<m2FILE...>`, or match with `−l <int>` (See option `−l <int>` of the WHAM aligner for more details).

`<outFILE>` Specifies the path and name of the file to which valid alignments are written.

Input options:

`-l <int>` Aligns the first `<int>` characters in each read. All alignments that has less than k errors (specified by `-v <int>`) in the first `<int>`characters are treated as valid alignments, and are reported. If this option is specified, the value of `<int>` must match the read length of the specified index. By default, WHAM aligns the entire reads, e.g. `<int>` = length of reads.

Alignment options:

`-v <int>` Specifies the maximum number of errors in a valid alignment. If `<int>` is less than or equal to the number of errors specified in the WHAM index builder (denoted as `v` here), WHAM reports ALL valid alignments. Otherwise, if `<int>` is greater than `v`, WHAM reports ALL valid alignments with up to `v` mismatches, and reports SOME valid alignments with more than `v`mismatches. Practically, if `<int>` is close to `v`, WHAM can find most of valid alignments. By default, we use the number of errors specified in WHAM index builder.

`-g/--gap <int>` Specifies the maximum number of gaps (insertions/deletions) in a valid alignment. The total number of errors (including mismatches and gaps) is specified by `-v`. (default: 0 gap).

`-e/--maqerr <int>` Specifies the minimum quality score. The quality scores are calculated based on statistics model. If specify -e 0, WHAM outputs quality scores for all alignments based on the error model, i.e. number of errors and gaps. By default, this is not used.

`--nofw/--norc` Disable to align to forward/reverse-complement of the reference strand. The
default is to use both the forward and reverse-complement strands.

`--fr/--rf/--ff` Indicates the upstream/downstream orientation (used only for pair-end reads): forward/reverse-complement, reverse-complement/forward, forward/forward (default: `--fr`).

`-I/--minins <int>` Specifies the minimum insert size for paired-end alignment (used only for pair-end reads). The insert size is the distance (in terms of characters) between upstream and downstream reads. This option specifies the minimum distance for a valid pair-end reads. Default: 0.

`-X/--maxins <int>` Specifies the maximum insert size for paired-end alignment (used only for pair-end reads). The insert size is the distance (in terms of characters) between upstream and downstream reads. This option specifies the minimum distance for a valid pair-end reads. Default: 250.

Reporting options:

`-k <int>` Report up to `<int>` valid alignments per read or per pair. If more than one valid alignment exists for the read and `-a/--al` is not specified, WHAM reports the ones with the least offsets in the reference sequences. Default: 1.

`-a/--all` Report all valid alignments per read or per pair. Since the frequent repeats are filtered out in the build phase (with option `-m <int>`of WHAM index builder), valid alignments on frequent repeats can be missed. In most cases, those alignments are viewed as junks. But if you are interested in those alignments, build a index with a greater value of `-m <int>`. Default: off.

`--best` Report valid alignments in a sorted order of quality values. If `-k <int>` is also specified, WHAM output the top `<int>` alignments. In particular, if `-k 1`, WHAM outputs the best alignment. If `-a/--all` is specified, WHAM output all valid alignments in best-to-worst order. Default: off.

`-m <int>` Discard reads with more than `<int>` valid alignments. By default, this is not used. Validity of alignments is determined by the alignment policy specified by options `-v`, `-e`, `--nofw/--norc`, and so on.

Output options:

`-S/--sam` Write alignments in SAM format. See the SAM Format Specification for more details.

`--al <FILE>` Wirte aligned reads/pairs to file(s) `<FILE>`. For pair-end reads, WHAM writes two files for upstream and downstream reads with suffix`"_1"` and `"_2"`. e.g. writes file `<FILE>_1` for upstream aligned reads, writes file `<FILE>_2` for downstream aligned reads.

`--un <FILE>` Write unaligned reads/pairs to file(s) `<FILE>`. For pair-end reads, WHAM writes two files for upstream and downstream reads with suffix `"_1"` and `"_2"`. e.g. writes file `<FILE>_1` for upstream unaligned reads, writes file `<FILE>_2` for downstream unaligned reads.

Multi-thread options:

`-t <int>` Specify the number of threads for aligning reads. On a multi-processor or multi-core system, the threads can run parallelly, with each processor or core running a particular thread. WHAM achieves a nearly linear speedup as the number of processors and cores increase. Each thread reports alignments into a separate file with prefix `<outFILE>`. If `--nocat` is not specified, all output files are concatenated into a single file named `<outFILE>`.

`--nocat` Do not concatenate output files generated by threads. The outputted alignments are in files: `<outFILE>_1`, `<outFILE>_2`,`<outFILE>_3`, …, `<outFILE>_<#threads>`. Default: off.

Other options:

`--version`	Print version information.
`-h/--help`	Print usage message.

WHAM

PAGES

Comparison using wgsim
Download
Main
Manual
Tutorial

THE LATEST

New release
A new version (v0.1.5) of WHAM with lots of improvements is available […]

MORE

Thanks for dropping by! Feel free to join the discussion by leaving comments, and stay updated by subscribing to the RSS feed.

© 2012 WHAM