BIOHDF – USERS GUIDE

VERSION 0.4 ALPHA

DECEMBER 2011

INTRODUCTION

BioHDF is a file format which stores reads and alignment hits from NGS instruments. At this time, BioHDF is conceptually similar to BAM though it has the potential to store more complicated data.

THE BIOHDF DATA MODEL

This section describes the abstract building blocks of BioHDF. Figure 1 shows the parts of the abstract data model and how they can be instantiated as BioHDF objects.

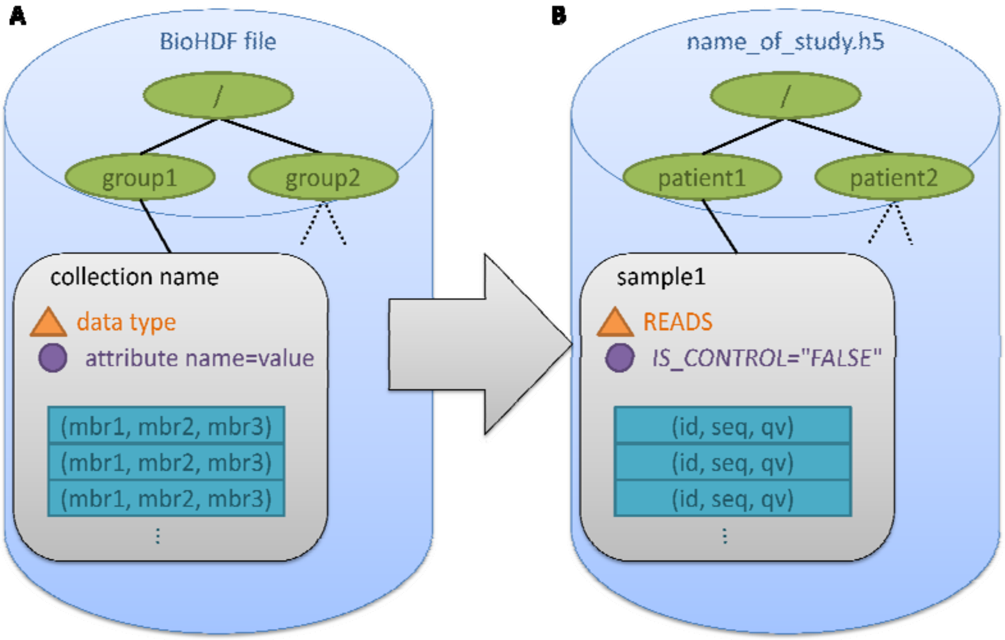

Figure 1- The parts of the BioHDF data model. (A) The abstract data model. (B) A more concrete instantiation of the model, showing a BioHDF read collection. The elements of the collection (blue rectangles) are the reads. Each read has three data members: an identifier, a sequence and a quality value.

DATA TYPE

The BioHDF data type refers to high-level, bioinformatics data types such as "read" or alignment".

In this document, data type never refers to low-level types like "integer" or "string".

COLLECTIONS, ELEMENTS AND MEMBERS

The fundamental unit of data storage in BioHDF is known as a collection. This is a named set of BioHDF data of a single BioHDF data type (*e.g.* a read collection named "sample_1_data"). The data items in a collection are known as elements. For example, a collection of reads has the individual reads as its elements. Each element is composed of one or more data members.

Collections can be linked. This is especially useful when a collection contains data which were derived from another collection. An example would be linking an alignment hit collection to the read collection from which it was derived.

A collection has a name, which is assigned by the user. BioHDF collection names can only contain alphanumeric ASCII characters and the underscore character (decimal 95).

GROUPS AND PATHS

BioHDF data can be organized using groups, which are analogous to directories in a file system. These groups can be nested hierarchically into paths which are represented by a forward-slash delimited string (*e.g.* /this/is/where/my/data/goes/). All paths are absolute and begin in the root group "/". BioHDF has no concept of a working group or relative paths.

A group layout in BioHDF has no established conventions, and data organization is left to the discretion of the user, application or community[1].

The final element of a path string can be the name of a collection (*e.g.* /path/to/collection) where a path to a collection is required. The API and tool documentation will very clear when this is required.

BioHDF group names can only contain alphanumeric ASCII characters and the underscore character (decimal 95). If the final element in a path is a collection, the path cannot end in a slash.

[1] A research community may wish to specify a particular group and collection layout in order to standardize their data storage and processing.

ATTRIBUTES

Attributes are small data elements which can be attached to a BioHDF group or collection. They are intended to store small quantities of auxiliary data and are not optimized for the storage of large quantities of data. There are three attribute types in BioHDF: 64-bit signed integer, double-precision floating point number and character string.

BioHDF attribute names can only contain alphanumeric ASCII characters and the underscore character (decimal 95). String attributes can store any ASCII characters[2]. There is no upper limit to the size of a string attribute.

BIOHDF DATA TYPES

This section describes the various BioHDF data types and how they are represented in BioHDF.

READS

A BioHDF read collection contains the output of DNA sequencers and their associated quality values along with an identifier; *i.e.* the equivalent of the data contained in a FASTQ entry.

Each data element (read) consists of:

- **Identifier** (string[3]): The FASTA or FASTQ identifier.
- **Sequence** (string): The sequence string. Can be in either base or color space.
- **Quality Value** (string): The quality value string. Any quality encoding scheme can be used.

The base or color space nature of the reads is identified with an attribute named READ_SPACE with a value of BASE_SPACE or COLOR_SPACE.

ALIGNMENTS

A BioHDF alignment collection contains elements which map reads (stored in a separate collection) to a reference sequence or sequences; *i.e.* the data contained in a SAM entry.

Each data element consists of:

- **Read ID** (integer): A reference to the source read.
- **Reference Name** (string): The name of the reference sequence to which the read was aligned.
- **Position** (integer): The start position of the alignment.
- **Length** (integer): The length of the alignment.

[2] BioHDF does not support Unicode.
[3] The basic types of each data field/member are indicated for clarity.

- **MAPQ** (integer): The mapping quality.
- **CIGAR** (string): The CIGAR string representation of the alignment mismatch.
- **FLAG** (integer): The SAM bitwise flags.
- **TAGS** (string): The SAM tags (key/type/value triplets)

Input files which have a header (*e.g.* SAM) will have that header stored verbatim, with no effort made to parse it.

The alignments are typically indexed by reference and position after import.

The alignments group will also include an attribute containing the path to the associated reads group.

PAIRED-END AND STROBE DATA

We will also allow storing information concerning paired reads and strobe data. This data will be stored in an auxiliary data structure in the same location as the alignment elements. This data mimics the way in which SAM/BAM stores this type of data, with one record per alignment.

- **RNEXT** (string): The name of the reference to which the next alignment in the pair/strobe matches.
- **PNEXT** (integer): The mapped position of the next alignment in the pair/strobe. This will be zero for unmapped reads.
- **TLEN** (integer): The total length of the underlying pair or strobe.

TOOLS

BioHDF includes several command line tools which can be used to import and export data to/from a BioHDF file.

BIOH5G_IMPORT_READS

This tool writes reads from a file or stdin into a named collection in a BioHDF file.

Supported formats: FASTA (and CSFASTA), FASTQ

Options:

`--biohdf-file|-f <filename>` **REQUIRED**

This specifies the name of the destination BioHDF file which will receive the reads. It will be created if it does not exist. We recommend using the .h5 file extension when creating BioHDF files.

`--reads-collection|-R <collection>` **REQUIRED**

This is the location in the BioHDF file where the reads will be stored. It is composed of a BioHDF group path and collection name. *e.g.* "/experiment1/sample1/reads/"

Any groups in the path which do not exist will be created.

See the data model section earlier in this document for more information about BioHDF paths, groups and collections.

`--in-file|-i <filename>`

This specifies the source of the reads. If no file is specified, the input is read from stdin.

`--in-format|-F fasta|fastq`

This specifies the input format of the reads. FASTA and FASTQ are supported. FASTA entries can span multiple lines.

CSFASTA files can be imported with `–F fasta` as the file format is identical. BioHDF simply stores the raw sequences and does not convert between base- and color-space.

For FASTQ files, the quality values are stored and returned as characters. BioHDF does no conversion or normalization between quality scoring systems.

```
--seqs-length|-s <length>
```

```
--ids-length|-d <length>
```

These allow you to specify the maximum string lengths of the sequences or IDs, if they are known, when creating a new reads collection. If no lengths are specified, much more inefficient scheme that can hold any length string is used to store the data.

The seqs-length parameter also specifies the length of the quality value strings for FASTQ data.

These settings are ignored if the reads collection has already been created. The data container is NOT resized.

See the data model section earlier in this document for more information about field sizes and their affect on performance and data size.

WARNING: Specifying a length that is too short will result in data truncation. The tool will warn you about this if it occurs.

```
--chunk-size|-C <HDF5 chunk size in bytes>
```

This allows you to specify the HDF5 chunk size. The default is 1024. This is an advanced option – see the HDF5 documentation for more information.

```
--compression-level |-Z <integer compression level from 0 to 9>
```

This is the compression level used in the collection. BioHDF uses zlib/gzip compression with a default compression level of 5. Compression level zero will produce uncompressed output.

BIOH5G_EXPORT_READS

This tool writes reads from a named collection in a BioHDF file to a file or STDOUT.

Supported formats: FASTA, FASTQ, CSFASTA, reads only, quality values only

Options:

`--biohdf-file|-f <filename>` **REQUIRED**

This specifies the name of the BioHDF file which contains the reads.

`--reads-collection|-R <collection>` **REQUIRED**

This is the location in the BioHDF file where the reads have been stored. It is composed of a BioHDF group path and collection name. *e.g.* "/experiment1/sample1/reads/"

See the data model section for more information about BioHDF paths, groups and collections.

`--out-file|-o <filename>`

This specifies the destination of the reads. If no file is specified, the output is sent to stdout.

`--out-format|-F fasta|fastq|reads|qualities`

This specifies the output format of the reads. In addition to FASTA and FASTQ, just the reads or just the quality values can be emitted (one per line).

CSFASTA files can be exported with `–F fasta` as the file format is identical. BioHDF simply stores the raw sequences and does not convert between base- and color-space.

For FASTQ files, the quality values are stored and returned as characters. BioHDF does no conversion or normalization between quality scoring systems.

`--beginning|-b <number of entries>`

This returns a fixed number of entries from the beginning of the collection, similar to the unix `head` command.

`--end|-e <number of entries>`

This returns a fixed number of entries from the end of the collection, similar to the unix `tail` command.

BIOH5G_IMPORT_ALIGNMENTS

This tool writes alignment hits from a file or STDIN into a named collection in a BioHDF file. The SAM header (if any) is stored verbatim, with no parsing.

Supported formats: SAM

Options:

`--biohdf-file|-f <filename>` **REQUIRED**

This specifies the name of the destination BioHDF file which will receive the alignments. It will be created if it does not exist. We recommend using the .h5 file extension when creating BioHDF files.

`--reads-collection|-R <collection>` **REQUIRED**

`--alignments-collection|-A <collection>` **REQUIRED**

These are the locations in the BioHDF file where the reads and alignments will be stored. They are composed of a BioHDF group path and collection name. *e.g.* "/experiment1/sample1/reads/" A reads collection must be specified as SAM contains read data, which is stored separately from the alignment data in BioHDF.

Any groups which do not exist will be created.

See the data model section for more information about BioHDF paths, groups and collections.

`--in-file|-i <filename>`

This specifies the source of the alignments. If no file is specified, the input is read from stdin.

`--in-format|-F sam`

This specifies the input format of the alignments. Only the SAM format is supported at this time.

`--seqs-length|-s <length>`

`--ids-length|-d <length>`

`--refs-length|-r <length>`

`--cigar-length|-c <length>`

```
--tags-length|-t <length>
```

These allow you to specify the maximum string lengths of various BioHDF string containers, if they are known. If no lengths are specified, a much less efficient scheme that can hold any length string is used to store the data. You can use the bioh5g_preparse_sam tool to pre-parse SAM files to determine these lengths.

The –d parameter is for the read identifier (QNAME).

The –s parameter is for the read sequence and quality values (SEQ and QUAL).

The –r parameter is for the reference names (RNEXT and RNAME).

The –c parameter is for the CIGAR strings (CIGAR).

The –t parameter is for the SAM tags (the entire tag string).

These settings are ignored if the reads collection has already been created. The data container is NOT resized.

See the data model section for more information about field sizes and their affect on performance and data size.

WARNING: Specifying a length that is too short will result in data truncation. The tool will warn you about this.

```
--chunk-size|-C <HDF5 chunk size in bytes>
```

This allows you to specify the HDF5 chunk size for the collection. The default is 1024. This is an advanced option – see the HDF5 documentation for more information.

```
--compression-level |-Z <integer compression level from 0 to 9>
```

This is the compression level used in the collection. BioHDF uses zlib/gzip compression with a default compression level of 5. Compression level zero will produce uncompressed output.

```
--do-not-index
```

This will prevent indexing the alignments, which is useful when importing multiple files into a single collection. If this option is not set, the alignments will be sorted by reference (strcmp order) and position.

```
--temp-file-path |-T <path to location where temp files can be created>
```

Indexing large files may require more memory than the computer has available. BioHDF solves this problem by using an external sort algorithm which writes data to temporary files as it progresses.

This option lets you set where temp files can be written. For best speed, pick a different physical disk than where the BioHDF file is stored, if possible. The default is the current working directory. Temp files are removed after the indexing is complete unless there were errors (we keep them for debugging).

NOTE: The path MUST include a directory separator ('/' on unix systems) at this time.

`--max-index-memory-mb |-M <max memory to use for index creation in MB>`

Indexing large files may require more memory than the computer has available. BioHDF solves this problem by using an external sort algorithm which writes data to temporary files as it progresses.

This option lets you specify the max amount of memory that the index algorithm will use to hold the temp data before it is written out to disk. The default is 512 MB. More is better if you have a computer with a lot of memory.

Note that this does not include memory used by other parts of BioHDF so this is not the *total* memory used by the program – it's just so that we can cap the memory usage for very large files, preventing swaps to virtual memory.

`--do-not-store-header`

This will prevent storing the SAM header.

NOTE: Only one SAM header can be stored per BioHDF collection. If you want to store multiple SAM files with different headers in the same BioHDF file, it's probably better to use separate collections. Or use samtools merge to join them before importing them.

BIOH5G_EXPORT_ALIGNMENTS

This tool writes alignments from a named collection in a BioHDF file to a file or STDOUT. The stored SAM header (if any) is emitted verbatim, with no updating or parsing.

Supported formats: SAM

Options:

`--biohdf-file|-f <filename>` **REQUIRED**

This specifies the name of the BioHDF file which contains the alignments.

`--alignments-collection|-A <collection>` **REQUIRED**

This is the location in the BioHDF file where the alignments have been stored. It is composed of a BioHDF group path and collection name. *e.g.* "/experiment1/sample1/alignments/"

See the data model section for more information about BioHDF paths, groups and collections.

`--out-file|-o <filename>`

This specifies the destination of the alignment hits. If no file is specified, the output is sent to stdout.

`--out-format|-F sam`

This specifies the output format of the alignment hits. Only SAM output is supported at this time.

`--beginning|-b <number of entries>`

This returns a fixed number of entries from the beginning of the collection, similar to the unix `head` command.

`--end|-e <number of entries>`

This returns a fixed number of entries from the end of the collection, similar to the unix `tail` command.

`<reference>:<start>:<end>`

Specifies regions to query. The start and end are 1-based and inclusive. The start and end are optional and either or both may be omitted (so 'chr18::' is a valid region reference). Omitting the start or end implies that the region extends to the start or end of the reference, respectively. Multiple regions may be specified.

Unlike samtools, regions MUST include both colons.

The default is to return all alignments.

`--min-quality|-Q <min MAPQ value (0-255, inclusive)>`

Only alignments with a MAPQ score equal to or above the min MAPQ score will be returned.

The default is to return all alignments.

```
--sam-flag-mask|-M <unsigned integer SAM FLAG mask>
```

Only alignments will all the bits in the mask set will be returned. See the samtools documentation for the flag meanings.

The default is to return all alignments.

```
--no-header
```

The SAM header (if stored) will not be emitted. The default is to emit the SAM header.

BIOH5G_PREPARSE_SAM

This tool scans a SAM file and returns summary statistics which are useful when setting the string length parameters to bioh5g_import_alignment hits.

It takes the name of the SAM file as its only argument.

It returns:

- The number of SAM lines
- The header length in characters
- The minimum length and maximum length of the following strings (I've added the pertinent BioHDF flags to the SAM field): QNAME (-d), RNAME/RNEXT (-r), CIGAR (-c), SEQ (-s), QUAL (-s) and TAGS (-t).

It runs fairly quickly, even for large files.