isoSeqQC

Generated by Doxygen 1.16.1

Chapter 1

Overview

A C++17 library and software to assess the quality of iso-Seq read alignments.

1.1 Dependencies

The project requires a C++17 compiler and depends on htslib for handling BAM files. A cmake script is included in the repository for easy building and installation. It requires cmake version 3.21 or later and automatically downloads htslib. If one wants to optionally build tests, Catch2 is also required and is automatically downloaded using the cmake script.

1.2 Download and install

Clone the repository:

```
git clone https://github.com/tonymugen/isoSeqQC
```

Next, create a build directory

```
cd isoSeqQC
mkdir build
```

Finally, run cmake to build and install the software

```
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
cmake --build .
cmake --install .
```

Installation may require root privileges.

1.3 Use exoncoverage

`exoncoverage` is a command line tool that takes a BAM alignment and a GFF annotation file and reports exon coverage statistics for each BAM read record, taking into account secondary alignments. Running it without any command line flags prints usage information. The flags are

```
--input-bam   bam_file_name (input BAM file name; required).
--input-gff   gff_file_name (input GFF file name; required).
--out         out_file_name (output file name; required).
--threads     number_of_threads (maximal number of threads to use; defaults to maximal available).
```

and can be specified in any order.

The output is a tab-delimited text file with largely self-explanatory column names. Columns that require special explanation are:

```
best_alignment_start          -- start position of best alignment, incorporating secondary alignments.
best_alignment_end            -- the same for the end position of the alignment.
n_good_secondary_alignments   -- number of secondary alignments that are cover the same region as the primary and
n_local_reversed_strand       -- number of secondary alignments that are on the opposite strand but still in vici
alignment_quality_string      -- a string of alignment match fractions for each exon, by position on the referenc
best_alignment_quality_string -- the same, but taking into account secondary alignments.
```

1.4 Use partialmaps

`partialmaps` is a command line tool that takes a BAM alignment file and a GFF annotation file, identifies poorly mapped regions, exports data on these alignment gaps, and optionally saves the unmapped sequences to a FATSQ file for subsequent realignment. Alignment gap identification relies on a change point detection algorithm that uses sliding windows. Sliding window size is a user-controlled parameter. Running `paritalmaps` without any command line flags prints usage information. The flags are

```
--input-bam    bam_file_name (input BAM file name; required).
--input-gff    gff_file_name (input GFF file name; required).
--out          out_file_name (output file name; required).
--out-fastq    out_fastq_file_name (output FASTQ file name; optional, no FASTQ file created if omitted).
--window-size  sliding window size for poorly aligned region identification (defaults to 75).
--threads      number_of_threads (maximal number of threads to use; defaults to maximal available).
```

and can be specified in any order.

The output is a tab-delimited text file with the following columns:

```
read_name      -- read name.
read_length    -- read length.
unmapped_start -- 0-base index of the unmapped region start in the read.
unmapped_end   -- 0-base index of the unmapped region end in the read.
window_size    -- sliding window size.
```

FASTQ sequence identifier fields are read names, with read portion base-0 start and stop positions added, separated by underscores.

1.5 addremaps

addremaps is a command line tool that takes the initial alignment and a file with re-mapped read portions and adds successful re-maps to the original BAM records as a secondary alignment. CIGAR strings are modified to mark re-mapped regions as unaligned. Primary alignments from records without re-maps are transferred to the output unchanged. The output file is sorted by default, but this can be turned off. The CLI flags are

```
--input-bam        bam_file_name (input BAM file name; required).
--remapped-bam     remapped_bam_file_name (BAM file with remapped read portions; required).
--out              out_file_name (output BAM file name; required).
--remap-cutoff     identity cutoff for read portion remapping (defaults to 0.99).
--unsorted-output  if set (with no value), the output BAM file is unsorted (otherwise, sorted).
```

As with the other tools, running addremaps without flags prints the above list.

1.6 Tests and documentation

Unit tests can be optionally built, without installing the software on the system:

```
mkdir buildTest
cd buildTest
cmake -DCMAKE_BUILD_TYPE=Test ..
cmake --build .
./tests
```

Library API documentation is in the header files, and can be optionally built using Doxygen:

```
cmake -DCMAKE_BUILD_TYPE=Release -DBUILD_DOCS=ON
cmake --build .
```

in the build directory. This requires that doxygen and pdflatex are installed in the execution path.

1.7 Funding

This project is funded in part by an NIH NIGMS MIRA R35 GM133376 grant to Rebekah L. Rogers.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

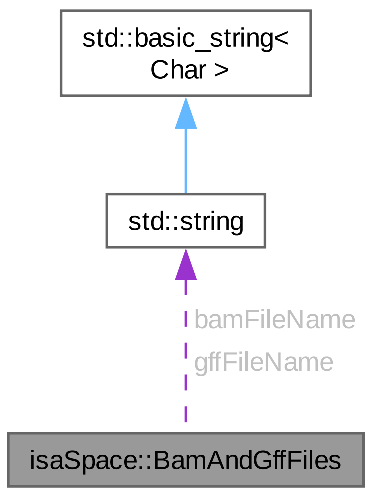
Class Documentation

4.1 isaSpace::BamAndGffFiles Struct Reference

BAM and GFF file name pair.

```
#include <isoseqAlgn.hpp>
```

Collaboration diagram for isaSpace::BamAndGffFiles:



Public Attributes

- std::string **bamFileName**

  BAM file name.
- std::string **gffFileName**

  GFF file name.

4.1.1 Detailed Description

BAM and GFF file name pair.

The documentation for this struct was generated from the following file:

- isoseqAlgn.hpp

4.2 isaSpace::BAMfile Class Reference

Records from a BAM file.

```
#include <isoseqAlgn.hpp>
```

Public Member Functions

- **BAMfile** ()=default

  Default constructor.
- BAMfile (const std::string &BAMfileName)

  Constructor with BAM file name.
- BAMfile (const BAMfile &toCopy)=delete

  Copy constructor.
- BAMfile & operator= (const BAMfile &toCopy)=delete

  Copy assignment operator.
- BAMfile (BAMfile &&toMove) noexcept=default

  Move constructor.
- BAMfile & operator= (BAMfile &&toMove) noexcept=default

  Move assignment operator.
- ∼**BAMfile** ()=default

  Destructor.
- size_t getPrimaryAlignmentCount () const noexcept

  Get the number of primary alignments.
- void addRemaps (const std::string &remapBAMfileName, const float &remapIdentityCutoff)

  Add re-mapped read regions.
- std::vector< std::string > saveRemappedBAM (const std::string &outputBAMfileName) const

  Save the reads with re-alignments to a BAM file.
- std::vector< std::string > saveSortedRemappedBAM (const std::string &outputBAMfileName) const

  Sort and save the reads with re-alignments to a BAM file.

4.2.1 Detailed Description

Records from a BAM file.

Primary records from a BAM file, including the header. Used to add realignments as secondary records.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BAMfile() [1/3]

```
isaSpace::BAMfile::BAMfile (
             const std::string & BAMfileName)
```

Constructor with BAM file name.

Parameters

in	*BAMfileName*	name of the input BAM file

4.2.2.2 BAMfile() [2/3]

```
isaSpace::BAMfile::BAMfile (
             const BAMfile & toCopy)  [delete]
```

Copy constructor.

Parameters

in	*toCopy*	object to copy

4.2.2.3 BAMfile() [3/3]

```
isaSpace::BAMfile::BAMfile (
             BAMfile && toMove)  [default], [noexcept]
```

Move constructor.

Parameters

in	*toMove*	object to move

4.2.3 Member Function Documentation

4.2.3.1 addRemaps()

```
void isaSpace::BAMfile::addRemaps (
             const std::string & remapBAMfileName,
             const float & remapIdentityCutoff)
```

Add re-mapped read regions.

Add re-mapped read regions as secondary alignments, fixing CIGAR strings for the primaries.

Parameters

in	*remapBAMfileName*	name of the BAM file with re-mapped read portions

in	*remapIdentityCutoff*	fraction of sites in the remapped read that are identical to the reference

4.2.3.2 getPrimaryAlignmentCount()

```
size_t isaSpace::BAMfile::getPrimaryAlignmentCount () const  [nodiscard], [noexcept]
```

Get the number of primary alignments.

Returns

number of primary alignments

4.2.3.3 operator=() [1/2]

```
BAMfile & isaSpace::BAMfile::operator= (
            BAMfile && toMove)  [default], [noexcept]
```

Move assignment operator.

Parameters

in	*toMove*	object to move

Returns

BAMfile object

4.2.3.4 operator=() [2/2]

```
BAMfile & isaSpace::BAMfile::operator= (
            const BAMfile & toCopy)  [delete]
```

Copy assignment operator.

Parameters

in	*toCopy*	object to copy

Returns

BAMfile object

4.2.3.5 saveRemappedBAM()

```
std::vector< std::string > isaSpace::BAMfile::saveRemappedBAM (
             const std::string & outputBAMfileName) const  [nodiscard]
```

Save the reads with re-alignments to a BAM file.

Parameters

in	*outputBAMfileName*	output BAM file name

Returns

names of reads that failed to be written

4.2.3.6 saveSortedRemappedBAM()

```
std::vector< std::string > isaSpace::BAMfile::saveSortedRemappedBAM (
             const std::string & outputBAMfileName) const  [nodiscard]
```

Sort and save the reads with re-alignments to a BAM file.

References saved in the order of the original BAM file. Reads sorted by start position (which is the gene end if mapped to the negative strand).

Parameters

in	*outputBAMfileName*	output BAM file name

Returns

names of reads that failed to be written

The documentation for this class was generated from the following file:

- isoseqAlgn.hpp

4.3 isaSpace::BAMheaderDeleter Struct Reference

BAM header pointer deleter.

```
#include <isoseqAlgn.hpp>
```

Public Member Functions

- void operator() (sam_hdr_t ∗header) const

  Functor operator.

4.3.1 Detailed Description

BAM header pointer deleter.

4.3.2 Member Function Documentation

4.3.2.1 operator()()

```
void isaSpace::BAMheaderDeleter::operator() (
            sam_hdr_t * header) const  [inline]
```

Functor operator.

Parameters

in	*header*	pointer to the header

The documentation for this struct was generated from the following file:

- isoseqAlgn.hpp

4.4 isaSpace::BAMrecord Class Reference

Summary of a BAM record set.

```
#include <isoseqAlgn.hpp>
```

Public Member Functions

- **BAMrecord** ()=default

  Default constructor.
- BAMrecord (const bam1_t ∗alignmentRecord, const sam_hdr_t ∗samHeader)

  Constructor with data.
- BAMrecord (const BAMrecord &toCopy)=default

  Copy constructor.
- BAMrecord & operator= (const BAMrecord &toCopy)=default

  Copy assignment operator.
- BAMrecord (BAMrecord &&toMove) noexcept=default

  Move constructor.
- BAMrecord & operator= (BAMrecord &&toMove) noexcept=default

  Move assignment operator.
- ∼**BAMrecord** ()=default

  Destructor.
- void addSecondaryAlignment (const bam1_t ∗alignmentRecord, const sam_hdr_t ∗samHeader, const hts_pos_t localWindow=500 '000)

  Add a secondary alignment record.
- std::string getReadName () const

  Output read name.
- hts_pos_t getMapStart () const noexcept

  Map start position.
- hts_pos_t getMapEnd () const noexcept

  Map end position.
- hts_pos_t getmRNAstart () const noexcept

  mRNA start position
- bool isRevComp () const noexcept

  Is the read reverse-complemented?
- bool isMapped () const noexcept

  Is the read mapped?
- bool hasSecondaryAlignments () const noexcept

  Are there any secondary alignments?
- bool hasLocalSecondaryAlignments () const noexcept

  Are there any local secondary alignments?
- uint16_t secondaryAlignmentCount () const noexcept

  Count of all secondary alignments regardless of position.
- uint16_t localSecondaryAlignmentCount () const noexcept

  Count of secondary alignments overlapping the primary.
- uint16_t localReversedSecondaryAlignmentCount () const noexcept

  Count of reversed secondary alignments overlapping the primary.
- hts_pos_t getReadLength () const noexcept

  Read length.
- std::vector< uint32_t > getCIGARvector () const

  CIGAR vector.
- std::string getCIGARstring () const

  CIGAR string.

- uint32_t getFirstCIGAR () const noexcept

    Get the first cigar element with strand reversal.
- std::string getReferenceName () const

    Get reference name.
- MappedReadMatchStatus getBestReferenceMatchStatus () const

    Best reference-centric match status.
- std::vector< std::pair< float, hts_pos_t > > getReadCentricMatchStatus () const

    Reference match status along the read.
- std::vector< MappedReadInterval > getPoorlyMappedRegions (const BinomialWindowParameters &window↩ Parameters) const

    Identify unmapped portions of the read.
- std::string getSequenceAndQuality (const MappedReadInterval &segmentBoundaries) const

    Get a segment of the sequence and the ASCII quality score.

4.4.1 Detailed Description

Summary of a BAM record set.

Stores relevant information from BAM format alignment records.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 BAMrecord() [1/3]

```
isaSpace::BAMrecord::BAMrecord (
            const bam1_t * alignmentRecord,
            const sam_hdr_t * samHeader)
```

Constructor with data.

Constructs an object from an HTSLIB alignment record and the corresponding header.

Parameters

in	*alignmentRecord*	pointer to a read alignment record
in	*samHeader*	pointer to the corresponding BAM/SAM header

4.4.2.2 BAMrecord() [2/3]

```
isaSpace::BAMrecord::BAMrecord (
            const BAMrecord & toCopy)  [default]
```

Copy constructor.

Parameters

in	*toCopy*	object to copy

4.4.2.3 BAMrecord() `[3/3]`

```
isaSpace::BAMrecord::BAMrecord (
            BAMrecord && toMove)  [default], [noexcept]
```

Move constructor.

Parameters

in	*toMove*	object to move

4.4.3 Member Function Documentation

4.4.3.1 addSecondaryAlignment()

```
void isaSpace::BAMrecord::addSecondaryAlignment (
            const bam1_t * alignmentRecord,
            const sam_hdr_t * samHeader,
            const hts_pos_t localWindow = 500 '000)
```

Add a secondary alignment record.

Adds information from a secondary alignment record if it is a local secondary alignment. Otherwise, only increments the total number of secondary alignments.

Parameters

in	*alignmentRecord*	pointer to a read alignment record
in	*samHeader*	pointer to the corresponding BAM/SAM header
in	*localWindow*	window size for a secondary alignment to be considered local

4.4.3.2 getBestReferenceMatchStatus()

```
MappedReadMatchStatus isaSpace::BAMrecord::getBestReferenceMatchStatus () const  [nodiscard]
```

Best reference-centric match status.

For each position, best match among all primary and secondary alignments.

Returns

match status with map start

4.4.3.3 getCIGARstring()

```
std::string isaSpace::BAMrecord::getCIGARstring () const  [nodiscard]
```

CIGAR string.

Reversed if the read is reverse-complemented.

Returns

CIGAR string

4.4.3.4 getCIGARvector()

```
std::vector< uint32_t > isaSpace::BAMrecord::getCIGARvector () const  [inline], [nodiscard]
```

CIGAR vector.

Orientation independent of strand

Returns

CIGAR vector

4.4.3.5 getFirstCIGAR()

```
uint32_t isaSpace::BAMrecord::getFirstCIGAR () const  [nodiscard], [noexcept]
```

Get the first cigar element with strand reversal.

Returns

first CIGAR vector element or 0 if none

4.4.3.6 getMapEnd()

```
hts_pos_t isaSpace::BAMrecord::getMapEnd () const  [inline], [nodiscard], [noexcept]
```

Map end position.

Position of the first past the mapped region of the reference.

Returns

1-based read map end position

4.4.3.7 getMapStart()

```
hts_pos_t isaSpace::BAMrecord::getMapStart () const  [inline], [nodiscard], [noexcept]
```

Map start position.

Position of the first mapped nucleotide.

Returns

1-based read map start position

4.4.3.8 getmRNAstart()

```
hts_pos_t isaSpace::BAMrecord::getmRNAstart () const  [inline], [nodiscard], [noexcept]
```

mRNA start position

Position of the first mRNA read nucleotide, taking into account possible reverse-complement.

Returns

1-based read mRNA start position

4.4.3.9 getPoorlyMappedRegions()

```
std::vector< MappedReadInterval > isaSpace::BAMrecord::getPoorlyMappedRegions (
             const BinomialWindowParameters & windowParameters) const  [nodiscard]
```

Identify unmapped portions of the read.

Returns a vector of poorly mapped portions of the read. Vector is empty if the read is mapped.

Parameters

in	*windowParameters*	window parameters: size and the mapped/unmapped region binomial probabilities

Returns

vector of poorly mapped region coordinates

4.4.3.10 getReadCentricMatchStatus()

```
std::vector< std::pair< float, hts_pos_t > > isaSpace::BAMrecord::getReadCentricMatchStatus ()
const  [nodiscard]
```

Reference match status along the read.

Parses CIGAR to track reference match/mismatch (1.0 for match, 0.0 for mismatch) status along the read, relating each read position to the corresponding reference base-1 nucleotide position. The vector start begins at the position closest to the first exon start regardless of strand.

Returns

vector of match status/reference position pairs

4.4.3.11 getReadLength()

```
hts_pos_t isaSpace::BAMrecord::getReadLength () const  [inline], [nodiscard], [noexcept]
```

Read length.

Returns

read length in bases

4.4.3.12 getReadName()

```
std::string isaSpace::BAMrecord::getReadName () const  [inline], [nodiscard]
```

Output read name.

Returns

read name

4.4.3.13 getReferenceName()

```
std::string isaSpace::BAMrecord::getReferenceName () const  [inline], [nodiscard]
```

Get reference name.

Reference sequence (e.g., chromosome) name. If absent, returns $*$ like samtools.

Returns

reference name

4.4.3.14 getSequenceAndQuality()

```
std::string isaSpace::BAMrecord::getSequenceAndQuality (
            const MappedReadInterval & segmentBoundaries) const  [nodiscard]
```

Get a segment of the sequence and the ASCII quality score.

Returns sequence and printable ASCII quality scores separated by a newline, with a newline at the end. Negative strand alignments are reverse-complemented to export the original strand.

Parameters

in	*segmentBoundaries*	read interval to retrieve

Returns

sequence and quality

4.4.3.15 hasLocalSecondaryAlignments()

```
bool isaSpace::BAMrecord::hasLocalSecondaryAlignments () const  [inline], [nodiscard], [noexcept]
```

Are there any local secondary alignments?

Returns

`true` if there are any secondary alignments overlapping the primary

4.4.3.16 hasSecondaryAlignments()

```
bool isaSpace::BAMrecord::hasSecondaryAlignments () const  [inline], [nodiscard], [noexcept]
```

Are there any secondary alignments?

Returns

`true` if there are any secondary alignments

4.4.3.17 isMapped()

```
bool isaSpace::BAMrecord::isMapped () const  [inline], [nodiscard], [noexcept]
```

Is the read mapped?

Returns

`true` if the read is mapped

4.4.3.18 isRevComp()

```
bool isaSpace::BAMrecord::isRevComp () const  [inline], [nodiscard], [noexcept]
```

Is the read reverse-complemented?

Returns

`true` if the read is reverse-complemented

4.4.3.19 localReversedSecondaryAlignmentCount()

```
uint16_t isaSpace::BAMrecord::localReversedSecondaryAlignmentCount () const  [nodiscard], [noexcept]
```

Count of reversed secondary alignments overlapping the primary.

Returns

Local reversed secondary alignment count

4.4.3.20 localSecondaryAlignmentCount()

```
uint16_t isaSpace::BAMrecord::localSecondaryAlignmentCount () const  [inline], [nodiscard], [noexcept]
```

Count of secondary alignments overlapping the primary.

Returns

Local secondary alignment count

4.4.3.21 operator=() [1/2]

```
BAMrecord & isaSpace::BAMrecord::operator= (
             BAMrecord && toMove)  [default], [noexcept]
```

Move assignment operator.

Parameters

in	*toMove*	object to move

Returns

`BAMrecord` object

4.4.3.22 operator=() [2/2]

```
BAMrecord & isaSpace::BAMrecord::operator= (
             const BAMrecord & toCopy)  [default]
```

Copy assignment operator.

Parameters

in	*toCopy*	object to copy

Returns

BAMrecord object

4.4.3.23 secondaryAlignmentCount()

```
uint16_t isaSpace::BAMrecord::secondaryAlignmentCount () const  [inline], [nodiscard], [noexcept]
```

Count of all secondary alignments regardless of position.

Returns

Secondary alignment count

The documentation for this class was generated from the following file:

- isoseqAlgn.hpp

4.5 isaSpace::BAMrecordDeleter Struct Reference

BAM record pointer deleter.

`#include <isoseqAlgn.hpp>`

Public Member Functions

- void operator() (bam1_t ∗bamRecord) const

  Functor operator.

4.5.1 Detailed Description

BAM record pointer deleter.

4.5.2 Member Function Documentation

4.5.2.1 operator()()

```
void isaSpace::BAMrecordDeleter::operator() (
            bam1_t * bamRecord) const  [inline]
```

Functor operator.

Parameters

in	*bamRecord*	pointer to a BAM record

The documentation for this struct was generated from the following file:

- isoseqAlgn.hpp

4.6 isaSpace::BAMsafeReader Class Reference

BAM file safe reader.

```
#include <isoseqAlgn.hpp>
```

Public Member Functions

- **BAMsafeReader** ()=default

  Default constructor.
- BAMsafeReader (const std::string &bamFileName)

  Constructor with file name.
- BAMsafeReader (const BAMsafeReader &toCopy)=delete

  Copy constructor.
- BAMsafeReader & operator= (const BAMsafeReader &toCopy)=delete

  Copy assignment operator.
- BAMsafeReader (BAMsafeReader &&toMove) noexcept

  Move constructor.
- BAMsafeReader & operator= (BAMsafeReader &&toMove) noexcept

  Move assignment operator.
- ∼**BAMsafeReader** ()

  Destructor.
- std::unique_ptr< sam_hdr_t, BAMheaderDeleter > getHeaderCopy () const

  Get BAM header.
- std::pair< std::unique_ptr< bam1_t, BAMrecordDeleter >, int32_t > getNextRecord ()

  Get the next BAM record.

4.6.1 Detailed Description

BAM file safe reader.

Temporarily sets read-only on the BAM file while reading from it. This is because I observed that calls to some (unclear which) HTSL BAM file read function very rarely deletes the file instead of reading it. Restores permissions after closing.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 BAMsafeReader() `[1/3]`

```
isaSpace::BAMsafeReader::BAMsafeReader (
            const std::string & bamFileName)
```

Constructor with file name.

Parameters

in	*bamFileName*	BAM file name

4.6.2.2 BAMsafeReader() `[2/3]`

```
isaSpace::BAMsafeReader::BAMsafeReader (
            const BAMsafeReader & toCopy)  [delete]
```

Copy constructor.

Parameters

in	*toCopy*	object to copy

4.6.2.3 BAMsafeReader() `[3/3]`

```
isaSpace::BAMsafeReader::BAMsafeReader (
            BAMsafeReader && toMove)  [inline], [noexcept]
```

Move constructor.

Parameters

in	*toMove*	object to move

4.6.3 Member Function Documentation

4.6.3.1 getHeaderCopy()

```
std::unique_ptr< sam_hdr_t, BAMheaderDeleter > isaSpace::BAMsafeReader::getHeaderCopy () const
[nodiscard]
```

Get BAM header.

Returns a pointer to a copy of the stored BAM header.

Returns

pointer to a copy of the header

4.6.3.2 getNextRecord()

```
std::pair< std::unique_ptr< bam1_t, BAMrecordDeleter >, int32_t > isaSpace::BAMsafeReader::get←
NextRecord () [nodiscard]
```

Get the next BAM record.

Returns a pointer to the next BAM record in the file and number of bytes read. The byte count can also be used to test for read success.

Returns

BAM header pointer and byte count

4.6.3.3 operator=() [1/2]

```
BAMsafeReader & isaSpace::BAMsafeReader::operator= (
            BAMsafeReader && toMove) [noexcept]
```

Move assignment operator.

Parameters

in	*toMove*	object to move

Returns

`BAMsafeReader` object

4.6.3.4 operator=() `[2/2]`

```
BAMsafeReader & isaSpace::BAMsafeReader::operator= (
              const BAMsafeReader & toCopy)   [delete]
```

Copy assignment operator.

Parameters

in	*toCopy*	object to copy

Returns

`BAMsafeReader` object

The documentation for this class was generated from the following file:

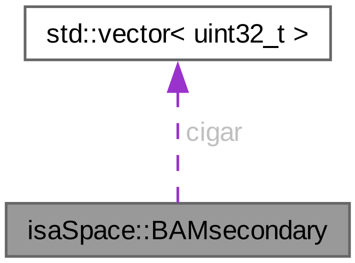
- isoseqAlgn.hpp

4.7 isaSpace::BAMsecondary Struct Reference

BAM secondary alignment.

```
#include <isoseqAlgn.hpp>
```

Collaboration diagram for isaSpace::BAMsecondary:



Public Attributes

- hts_pos_t **mapStart** {-1}

  Base-1 map start position on the reference.
- hts_pos_t **mapEnd** {-1}

  Base-1 map end position on the reference.
- bool **sameStrandAsPrimary** {false}

  Is it the same strand as the primary?
- std::vector< uint32_t > **cigar**

  CIGAR string.

4.7.1 Detailed Description

BAM secondary alignment.

Only secondary alignments on the same reference.

The documentation for this struct was generated from the following file:

- isoseqAlgn.hpp

4.8 isaSpace::BAMtoGenome Class Reference

Relate BAM alignments to exons.

```
#include <isoseqAlgn.hpp>
```

Public Member Functions

- **BAMtoGenome** ()=default

  Default constructor.
- BAMtoGenome (const BamAndGffFiles &bamGFFfilePairNames)

  Constructor intersecting iso-Seq alignments and exons.
- BAMtoGenome (const BAMtoGenome &toCopy)=default

  Copy constructor.
- BAMtoGenome & operator= (const BAMtoGenome &toCopy)=default

  Copy assignment operator.
- BAMtoGenome (BAMtoGenome &&toMove) noexcept=default

  Move constructor.
- BAMtoGenome & operator= (BAMtoGenome &&toMove) noexcept=default

  Move assignment operator.
- ∼**BAMtoGenome** ()=default

  Destructor.
- size_t nChromosomes () const noexcept

Number of chromosomes/scaffolds/linkage groups.
- size_t nExonSets () const noexcept

    Number of exon sets (genes with exons).
- void saveReadCoverageStats (const std::string &outFileName, const size_t &nThreads) const

    Save read coverage to file.
- void saveUnmappedRegions (const std::string &outFileName, const BinomialWindowParameters &window←↩ Parameters, const size_t &nThreads) const

    Save unmapped read statistics to file.
- void saveUnmappedRegions (const StatsAndFastqFiles &outFilePair, const BinomialWindowParameters &windowParameters, const size_t &nThreads) const

    Save unmapped statistics and read portions to files.

4.8.1 Detailed Description

Relate BAM alignments to exons.

Relates each BAM alignment to a gene and check which exons are covered by the read.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 BAMtoGenome() [1/3]

```
isaSpace::BAMtoGenome::BAMtoGenome (
            const BamAndGffFiles & bamGFFfilePairNames)
```

Constructor intersecting iso-Seq alignments and exons.

Uses exon positions from the provided GFF file to find iso-Seq alignments from the provided BAM file that may have mis-mapped first exons.

Parameters

in	*bamGFFfilePairNames*	BAM and GFF file name pair

4.8.2.2 BAMtoGenome() [2/3]

```
isaSpace::BAMtoGenome::BAMtoGenome (
            const BAMtoGenome & toCopy)  [default]
```

Copy constructor.

Parameters

in	*toCopy*	object to copy

4.8.2.3 BAMtoGenome() [3/3]

```
isaSpace::BAMtoGenome::BAMtoGenome (
            BAMtoGenome && toMove)  [default], [noexcept]
```

Move constructor.

Parameters

in	*toMove*	object to move

4.8.3 Member Function Documentation

4.8.3.1 nChromosomes()

```
size_t isaSpace::BAMtoGenome::nChromosomes () const  [nodiscard], [noexcept]
```

Number of chromosomes/scaffolds/linkage groups.

Counts the number of unique GFF `seqid` elements. These are typically chromosomes, scaffolds, or linkage groups.

Returns

number of chromosomes

4.8.3.2 nExonSets()

```
size_t isaSpace::BAMtoGenome::nExonSets () const  [nodiscard], [noexcept]
```

Number of exon sets (genes with exons).

Returns

number of exon sets

4.8.3.3 operator=() [1/2]

```
BAMtoGenome & isaSpace::BAMtoGenome::operator= (
            BAMtoGenome && toMove)  [default], [noexcept]
```

Move assignment operator.

Parameters

in	*toMove*	object to move

Returns

`BAMtoGenome` object

4.8.3.4 operator=() [2/2]

```
BAMtoGenome & isaSpace::BAMtoGenome::operator= (
             const BAMtoGenome & toCopy)   [default]
```

Copy assignment operator.

Parameters

in	*toCopy*	object to copy

Returns

BAMtoGenome object

4.8.3.5 saveReadCoverageStats()

```
void isaSpace::BAMtoGenome::saveReadCoverageStats (
             const std::string & outFileName,
             const size_t & nThreads) const
```

Save read coverage to file.

Saves the read coverage statistics to a file. If a file with the same name exists it is overwritten.

Parameters

in	*outFileName*	output file name
in	*nThreads*	number of concurrent threads

4.8.3.6 saveUnmappedRegions() [1/2]

```
void isaSpace::BAMtoGenome::saveUnmappedRegions (
             const StatsAndFastqFiles & outFilePair,
             const BinomialWindowParameters & windowParameters,
             const size_t & nThreads) const
```

Save unmapped statistics and read portions to files.

Only considers reads that are at least partially well mapped, but have some interval(s) with low alignment quality. Save unmapped portions to a FASTQ file. If files with the same names exist it is overwritten.

Parameters

in	*outFilePair*	output file name

in	*windowParameters*	sliding window parameters
in	*nThreads*	number of concurrent threads

4.8.3.7 saveUnmappedRegions() [2/2]

```
void isaSpace::BAMtoGenome::saveUnmappedRegions (
            const std::string & outFileName,
            const BinomialWindowParameters & windowParameters,
            const size_t & nThreads) const
```

Save unmapped read statistics to file.

Only considers reads that are at least partially well mapped, but have some interval(s) with low alignment quality. If a file with the same name exists it is overwritten.

Parameters

in	*outFileName*	output file name
in	*windowParameters*	sliding window parameters
in	*nThreads*	number of concurrent threads

The documentation for this class was generated from the following file:

- isoseqAlgn.hpp

4.9 isaSpace::BGZFhandleDeleter Struct Reference

BGZF handle pointer deleter.

`#include <isoseqAlgn.hpp>`

Public Member Functions

- void operator() (BGZF ∗bgzfHandle) const

  Functor operator.

4.9.1 Detailed Description

BGZF handle pointer deleter.

4.9.2 Member Function Documentation

4.9.2.1 operator()()

```
void isaSpace::BGZFhandleDeleter::operator() (
            BGZF * bgzfHandle) const  [inline]
```

Functor operator.

Parameters

in	*bgzfHandle*	pointer to a BGZF handle

The documentation for this struct was generated from the following file:

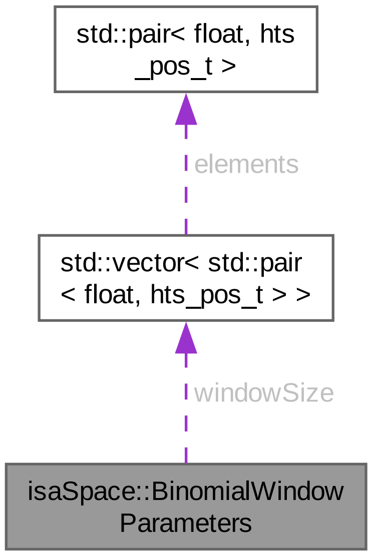
- isoseqAlgn.hpp

4.10 isaSpace::BinomialWindowParameters Struct Reference

Binomial window parameters.

```
#include <isoseqAlgn.hpp>
```

Collaboration diagram for isaSpace::BinomialWindowParameters:


std::pair< float, hts_pos_t >
elements
std::vector< std::pair < float, hts_pos_t > >
windowSize
isaSpace::BinomialWindowParameters

Public Attributes

- float **currentProbability** {0.0F}

    Probability of success up to this window.
- float alternativeProbability {0.0F}

    Probability of success to consider as an alternative.
- float **bicDifferenceThreshold** {0.0F}

    Minimum difference in BIC between windows with current and alternative probabilities.
- std::vector< std::pair< float, hts_pos_t > >::difference_type **windowSize** {0}

    Window size.

4.10.1 Detailed Description

Binomial window parameters.

4.10.2 Member Data Documentation

4.10.2.1 alternativeProbability

```
float isaSpace::BinomialWindowParameters::alternativeProbability {0.0F}
```

Probability of success to consider as an alternative.

Test for a switch to this probability when assessing evidence for a change point between mapped and unmapped regions of a read.

The documentation for this struct was generated from the following file:

- isoseqAlgn.hpp

4.11 isaSpace::ExonGroup Class Reference

Group of exons from the same gene.

```
#include <isoseqAlgn.hpp>
```

Public Member Functions

- **ExonGroup** ()=default

  Default constructor.
- ExonGroup (std::string geneName, const char strand, std::set< std::pair< hts_pos_t, hts_pos_t > > &exonSet)

  Constructor with lines from a GFF file.
- ExonGroup (std::string geneName, const char strand, const std::vector< std::string > &exonLinesFomGFF)

  Constructor with exon lines from a GFF file.
- ExonGroup (const ExonGroup &toCopy)=default

  Copy constructor.
- ExonGroup & operator= (const ExonGroup &toCopy)=default

  Copy assignment operator.
- ExonGroup (ExonGroup &&toMove) noexcept=default

  Move constructor.
- ExonGroup & operator= (ExonGroup &&toMove) noexcept=default

  Move assignment operator.
- ∼**ExonGroup** ()=default

  Destructor.
- bool empty () const noexcept

  Is the object empty?
- std::string geneName () const

  Report the gene name.
- size_t nExons () const noexcept

  Number of exons in the gene.
- char strand () const noexcept

  Strand ID.
- std::pair< hts_pos_t, hts_pos_t > at (const size_t &idx) const

  Range covered by a given exon.
- std::pair< hts_pos_t, hts_pos_t > geneSpan () const noexcept

  Gene span.
- std::pair< hts_pos_t, hts_pos_t > firstExonSpan () const noexcept

  First exon span.
- hts_pos_t firstExonLength () const noexcept

  First exon length.
- uint32_t firstExonAfter (const hts_pos_t &position) const noexcept

  Index of the first exon after a given position.
- uint32_t firstOverlappingExon (const hts_pos_t &position) const noexcept

  Index of the first exon overlapping a given position.
- uint32_t lastExonBefore (const hts_pos_t &position) const noexcept

  Index of the last exon before a given position.
- uint32_t lastOverlappingExon (const hts_pos_t &position) const noexcept

  Index of the last exon overlapping a given position.
- std::pair< hts_pos_t, hts_pos_t > getFirstIntronSpan () const

  First intron span.
- std::vector< float > getExonCoverageQuality (const BAMrecord &alignment) const

  Get read coverage quality per exon.
- std::vector< float > getBestExonCoverageQuality (const BAMrecord &alignment) const

  Get best read coverage quality per exon.

4.11.1 Detailed Description

Group of exons from the same gene.

Gathers exons belonging to all transcripts of a gene.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 ExonGroup() [1/4]

```
isaSpace::ExonGroup::ExonGroup (
            std::string geneName,
            const char strand,
            std::set< std::pair< hts_pos_t, hts_pos_t > > & exonSet)
```

Constructor with lines from a GFF file.

The exon set is ordered by the exon starts. The first exon can be the first or the last, depending on the strand. The strand is assumed positive, unless explicitly specified as negative by passing the − character.

Parameters

in	*geneName*	gene name
in	*strand*	mRNA strand
in	*exonSet*	set of exons from the same gene

4.11.2.2 ExonGroup() [2/4]

```
isaSpace::ExonGroup::ExonGroup (
            std::string geneName,
            const char strand,
            const std::vector< std::string > & exonLinesFomGFF)
```

Constructor with exon lines from a GFF file.

Uses lines exon from a GFF file that belong to the same gene.

Parameters

in	*geneName*	gene name
in	*strand*	mRNA strand
in	*exonLinesFomGFF*	GFF file exon lines

4.11.2.3 ExonGroup() [3/4]

```
isaSpace::ExonGroup::ExonGroup (
            const ExonGroup & toCopy)  [default]
```

Copy constructor.

Parameters

in	*toCopy*	object to copy

4.11.2.4 ExonGroup() [4/4]

```
isaSpace::ExonGroup::ExonGroup (
            ExonGroup && toMove)  [default], [noexcept]
```

Move constructor.

Parameters

in	*toMove*	object to move

4.11.3 Member Function Documentation

4.11.3.1 at()

```
std::pair< hts_pos_t, hts_pos_t > isaSpace::ExonGroup::at (
            const size_t & idx) const  [inline], [nodiscard]
```

Range covered by a given exon.

Parameters

in	*idx*	exon index

Returns

the exon start and end nucleotide position pair

4.11.3.2 empty()

```
bool isaSpace::ExonGroup::empty () const  [inline], [nodiscard], [noexcept]
```

Is the object empty?

Returns

true is the object has no exons

4.11.3.3 firstExonAfter()

```
uint32_t isaSpace::ExonGroup::firstExonAfter (
            const hts_pos_t & position) const  [nodiscard], [noexcept]
```

Index of the first exon after a given position.

0-based index of the first exon found entirely after the given position. Equal to the index of the last exon if the position is after the last exon.

Parameters

in	*position*	genome position to test

Returns

index of the first exon after the given position

4.11.3.4 firstExonLength()

```
hts_pos_t isaSpace::ExonGroup::firstExonLength () const  [nodiscard], [noexcept]
```

First exon length.

Returns

first exon length

4.11.3.5 firstExonSpan()

```
std::pair< hts_pos_t, hts_pos_t > isaSpace::ExonGroup::firstExonSpan () const  [nodiscard], [noexcept]
```

First exon span.

Returns the position of the first exon, depends on the strand. First position is always smaller than the second regardless of strand, in keeping with the GFF3 specification.

Returns

first exon nucleotide position pair (1-based)

4.11.3.6 firstOverlappingExon()

```
uint32_t isaSpace::ExonGroup::firstOverlappingExon (
            const hts_pos_t & position) const  [nodiscard], [noexcept]
```

Index of the first exon overlapping a given position.

0-based index of the first exon found at least partially after the given position. Equal to the index of the last exon if the position is after the last exon.

Parameters

`in`	*position*	genome position to test

Returns

index of the first exon overlapping the given position

4.11.3.7 geneName()

```
std::string isaSpace::ExonGroup::geneName () const  [inline], [nodiscard]
```

Report the gene name.

Returns

gene name

4.11.3.8 geneSpan()

```
std::pair< hts_pos_t, hts_pos_t > isaSpace::ExonGroup::geneSpan () const  [nodiscard], [noexcept]
```

Gene span.

Returns the position span of the gene. First element of the pair is always smaller than the last regardless of the strand.

Returns

first and last nucleotide position (1-based) of the gene

4.11.3.9 getBestExonCoverageQuality()

```
std::vector< float > isaSpace::ExonGroup::getBestExonCoverageQuality (
             const BAMrecord & alignment) const  [nodiscard]
```

Get best read coverage quality per exon.

Use CIGAR information, adding local secondary alignments, to extract alignment quality for each exon. Quality is the fraction of reference nucleotides covered by matching read bases.

Parameters

in	*alignment*	BAM alignment record

Returns

vector best alignment qualities, one per exon

4.11.3.10 getExonCoverageQuality()

```
std::vector< float > isaSpace::ExonGroup::getExonCoverageQuality (
             const BAMrecord & alignment) const  [nodiscard]
```

Get read coverage quality per exon.

Use CIGAR information to extract alignment quality for each exon. Quality is the fraction of reference nucleotides covered by matching read bases.

Parameters

in	*alignment*	BAM alignment record

Returns

vector of alignment qualities, one per exon

4.11.3.11 getFirstIntronSpan()

```
std::pair< hts_pos_t, hts_pos_t > isaSpace::ExonGroup::getFirstIntronSpan () const  [nodiscard]
```

First intron span.

Smaller value first regardless of strand, but the intron is always the first in the gene, last in the sequence if the strand is negative. If there is only one exon, the start and end are reported as `-1`.

Returns

first intron start and end positions

4.11.3.12 lastExonBefore()

```
uint32_t isaSpace::ExonGroup::lastExonBefore (
            const hts_pos_t & position) const  [nodiscard], [noexcept]
```

Index of the last exon before a given position.

0-based index of the last exon found entirely before the given position. Equal to the index of the last exon if the position is after the last exon.

Parameters

in	*position*	genome position to test

Returns

index of the last exon before the given position

4.11.3.13 lastOverlappingExon()

```
uint32_t isaSpace::ExonGroup::lastOverlappingExon (
            const hts_pos_t & position) const  [nodiscard], [noexcept]
```

Index of the last exon overlapping a given position.

0-based index of the last exon found at least before after the given position. Equal to the index of the last exon if the position is after the last exon.

Parameters

in	*position*	genome position to test

Returns

index of the last exon overlapping the given position

4.11.3.14 nExons()

```
size_t isaSpace::ExonGroup::nExons () const  [inline], [nodiscard], [noexcept]
```

Number of exons in the gene.

Returns

number of exons

4.11.3.15 operator=() [1/2]

```
ExonGroup & isaSpace::ExonGroup::operator= (
                const ExonGroup & toCopy)  [default]
```

Copy assignment operator.

Parameters

in	*toCopy*	object to copy

Returns

ExonGroup object

4.11.3.16 operator=() [2/2]

```
ExonGroup & isaSpace::ExonGroup::operator= (
                ExonGroup && toMove)  [default], [noexcept]
```

Move assignment operator.

Parameters

in	*toMove*	object to move

Returns

ExonGroup object

4.11.3.17 strand()

```
char isaSpace::ExonGroup::strand () const  [inline], [nodiscard], [noexcept]
```

Strand ID.

Returns

strand ID (+ or −)

The documentation for this class was generated from the following file:

- isoseqAlgn.hpp

4.12 isaSpace::MappedReadInterval Struct Reference

Delineates an interval in a mapped read.

```
#include <isoseqAlgn.hpp>
```

Public Attributes

- hts_pos_t **readStart** {-1}

  Base-0 start position on the read.
- hts_pos_t **readEnd** {-1}

  Base-0 end position on the read.
- hts_pos_t **referenceStart** {-1}

  Base-1 start position on the reference.
- hts_pos_t **referenceEnd** {-1}

  Base-1 end position on the reference.

4.12.1 Detailed Description

Delineates an interval in a mapped read.

Provides a start and end of a read interval and corresponding positions on the reference.

The documentation for this struct was generated from the following file:

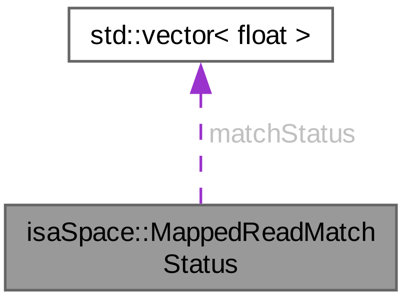
- isoseqAlgn.hpp

4.13 isaSpace::MappedReadMatchStatus Struct Reference

Read match and map start.

```
#include <isoseqAlgn.hpp>
```

Collaboration diagram for isaSpace::MappedReadMatchStatus:



Public Attributes

- hts_pos_t **mapStart** {-1}

  Map start.
- std::vector< float > **matchStatus**

  Match status vector.

4.13.1 Detailed Description

Read match and map start.

The documentation for this struct was generated from the following file:

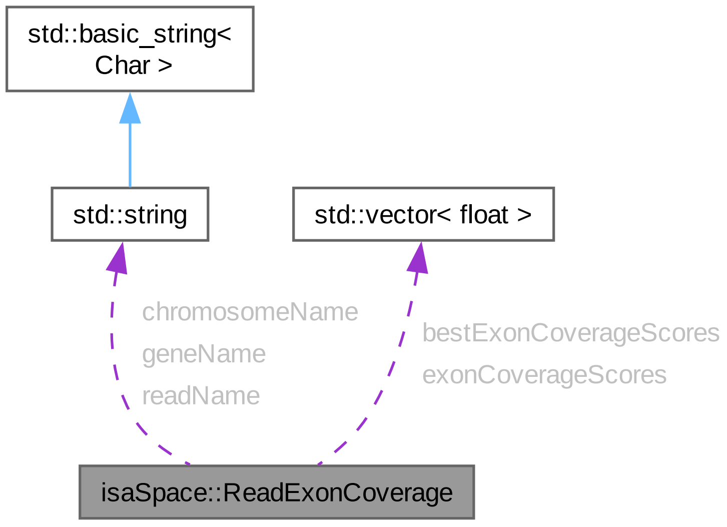
- isoseqAlgn.hpp

4.14 isaSpace::ReadExonCoverage Struct Reference

Exons covered by a read.

```
#include <isoseqAlgn.hpp>
```

Collaboration diagram for isaSpace::ReadExonCoverage:



Public Attributes

- std::string chromosomeName

  Chromosome name.
- std::string **readName**

  Read name.
- hts_pos_t alignmentStart {0}

  Alignment start.
- hts_pos_t alignmentEnd {0}

  Alignment end.
- hts_pos_t bestAlignmentStart {0}

  Best alignment start.
- hts_pos_t bestAlignmentEnd {0}

  Best alignment end.
- uint32_t firstSoftClipLength {0}

  Length of the soft clip at read start.
- uint16_t **nSecondaryAlignments** {0}

  Number of secondary alignments.
- uint16_t nGoodSecondaryAlignments {0}

  Number of good secondary alignments.
- uint16_t nLocalReversedAlignments {0}

  Number of locally mapping reverse-complemented reads.
- uint16_t **nExons** {0}

  Number of exons.
- std::string geneName

  Gene name.
- char strand {'+'}

  Strand ID.
- hts_pos_t firstExonLength {-1}

  First exon length.
- hts_pos_t firstExonStart {-1}

  First exon start.
- hts_pos_t lastExonEnd {-1}

  Last exon end.
- std::vector< float > exonCoverageScores

  Exon coverage scores.
- std::vector< float > bestExonCoverageScores

  Best exon coverage scores.

4.14.1 Detailed Description

Exons covered by a read.

For a given alignment, stores information on exons covered by the read. All positions are 1-based, indexes are 0-based.

4.14.2 Member Data Documentation

4.14.2.1 alignmentEnd

```
hts_pos_t isaSpace::ReadExonCoverage::alignmentEnd {0}
```

Alignment end.

Base-1 position of the read end from the primary alignment. Never smaller than `alignmentStart` regardless of strand.

4.14.2.2 alignmentStart

```
hts_pos_t isaSpace::ReadExonCoverage::alignmentStart {0}
```

Alignment start.

Base-1 position of the read start from the primary alignment. Never larger than `alignmentEnd` regardless of strand.

4.14.2.3 bestAlignmentEnd

```
hts_pos_t isaSpace::ReadExonCoverage::bestAlignmentEnd {0}
```

Best alignment end.

Base-1 position of the latest read end from the primary and all good secondary alignments. Never smaller than `bestAlignmentStart` regardless of strand.

4.14.2.4 bestAlignmentStart

```
hts_pos_t isaSpace::ReadExonCoverage::bestAlignmentStart {0}
```

Best alignment start.

Base-1 position of the earliest read start from the primary and all good secondary alignments. Never larger than `bestAlignmentEnd` regardless of strand.

4.14.2.5 bestExonCoverageScores

```
std::vector<float> isaSpace::ReadExonCoverage::bestExonCoverageScores
```

Best exon coverage scores.

Fraction of reference bases in each exon covered by a matching base in the read from the best alignment for that exon among all alignments for the read.

4.14.2.6 chromosomeName

```
std::string isaSpace::ReadExonCoverage::chromosomeName
```

Chromosome name.

This can also be a linkage group or scaffold name, as listed in the GFF file in the `seqid` column.

4.14.2.7 exonCoverageScores

```
std::vector<float> isaSpace::ReadExonCoverage::exonCoverageScores
```

Exon coverage scores.

Fraction of reference bases in each exon covered by a matching base in the read from the primary alignment.

4.14.2.8 firstExonLength

```
hts_pos_t isaSpace::ReadExonCoverage::firstExonLength {-1}
```

First exon length.

Actual first exon, last in the sequence if the strand is negative.

4.14.2.9 firstExonStart

```
hts_pos_t isaSpace::ReadExonCoverage::firstExonStart {-1}
```

First exon start.

Base-1 position of the first exon start from the GFF file. End of the last exon if the strand negative.

4.14.2.10 firstSoftClipLength

```
uint32_t isaSpace::ReadExonCoverage::firstSoftClipLength {0}
```

Length of the soft clip at read start.

End of the CIGAR string if the read is reverse-complemented. Set to 0 if there is no soft clip.

4.14.2.11 geneName

```
std::string isaSpace::ReadExonCoverage::geneName
```

Gene name.

Set to `no_overlap` or `past_last_mRNA` if there is no gene in the GFF file that overlaps the alignment.

4.14.2.12 lastExonEnd

```
hts_pos_t isaSpace::ReadExonCoverage::lastExonEnd {-1}
```

Last exon end.

Base-1 position of the last exon end from the GFF file. Start of the first exon if the strand negative.

4.14.2.13 nGoodSecondaryAlignments

```
uint16_t isaSpace::ReadExonCoverage::nGoodSecondaryAlignments {0}
```

Number of good secondary alignments.

Secondary alignments that are on the same strand as the primary and overlap the same gene.

4.14.2.14 nLocalReversedAlignments

```
uint16_t isaSpace::ReadExonCoverage::nLocalReversedAlignments {0}
```

Number of locally mapping reverse-complemented reads.

Number of alignments on the opposite strand from the primary that overlap the same gene as the primary.

4.14.2.15 strand

```
char isaSpace::ReadExonCoverage::strand {'+'}
```

Strand ID.

Must be $+$ or $-$.

The documentation for this struct was generated from the following file:

- isoseqAlgn.hpp

4.15 isaSpace::ReadMatchWindowBIC Class Reference

Binomial BIC for a read window.

```
#include <isoseqAlgn.hpp>
```

Public Member Functions

- **ReadMatchWindowBIC** ()=default

  Default constructor.
- ReadMatchWindowBIC (const std::vector< std::pair< float, hts_pos_t > >::const_iterator &windowBegin, const BinomialWindowParameters &windowParameters)

  Constructor with a read match vector window.
- ReadMatchWindowBIC (const ReadMatchWindowBIC &toCopy)=default

  Copy constructor.
- ReadMatchWindowBIC & operator= (const ReadMatchWindowBIC &toCopy)=default

  Copy assignment operator.
- ReadMatchWindowBIC (ReadMatchWindowBIC &&toMove) noexcept=default

  Move constructor.
- ReadMatchWindowBIC & operator= (ReadMatchWindowBIC &&toMove) noexcept=default

  Move assignment operator.
- ∼**ReadMatchWindowBIC** ()=default

  Destructor.
- float getBICdifference () const noexcept

  Get BIC difference.

4.15.1 Detailed Description

Binomial BIC for a read window.

Estimates of the Bayesian information criterion for a window of the read-centric match status vector. The BIC compares the model with the same success probability as the previous window to a change. The success probabilities reflect identity between unrelated or matching sequences.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 ReadMatchWindowBIC() [1/3]

```
isaSpace::ReadMatchWindowBIC::ReadMatchWindowBIC (
            const std::vector< std::pair< float, hts_pos_t > >::const_iterator & windowBegin,
            const BinomialWindowParameters & windowParameters)  [inline]
```

Constructor with a read match vector window.

Parameters

in	*windowBegin*	start of a read match status vector
in	*windowParameters*	binomial probability parameters of the window

4.15.2.2 ReadMatchWindowBIC() [2/3]

```
isaSpace::ReadMatchWindowBIC::ReadMatchWindowBIC (
            const ReadMatchWindowBIC & toCopy)  [default]
```

Copy constructor.

Parameters

in	*toCopy*	object to copy

4.15.2.3 ReadMatchWindowBIC() [3/3]

```
isaSpace::ReadMatchWindowBIC::ReadMatchWindowBIC (
            ReadMatchWindowBIC && toMove)  [default], [noexcept]
```

Move constructor.

Parameters

in	*toMove*	object to move

4.15.3 Member Function Documentation

4.15.3.1 getBICdifference()

```
float isaSpace::ReadMatchWindowBIC::getBICdifference () const  [nodiscard], [noexcept]
```

Get BIC difference.

Reports the BIC difference between the model with the same mapping quality as the previous window and a model with a switch to a different quality.

Returns

evidence for a switch in mapping quality

4.15.3.2 operator=() [1/2]

```
ReadMatchWindowBIC & isaSpace::ReadMatchWindowBIC::operator= (
            const ReadMatchWindowBIC & toCopy)  [default]
```

Copy assignment operator.

Parameters

in	*toCopy*	object to copy

Returns

`ReadMatchWindowBIC` object

4.15.3.3 operator=() [2/2]

```
ReadMatchWindowBIC & isaSpace::ReadMatchWindowBIC::operator= (
             ReadMatchWindowBIC && toMove)  [default], [noexcept]
```

Move assignment operator.

Parameters

in	*toMove*	object to move

Returns

ReadMatchWindowBIC object

The documentation for this class was generated from the following file:

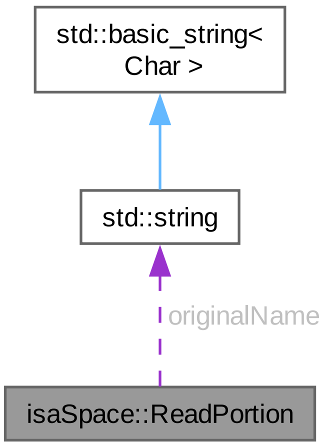
- isoseqAlgn.hpp

4.16 isaSpace::ReadPortion Struct Reference

Read portion for re-mapping.

```
#include <isoseqAlgn.hpp>
```

Collaboration diagram for isaSpace::ReadPortion:



Public Attributes

- size_t **start** {0}

  Base-0 read start.
- size_t **end** {0}

  Base-0 one past the read end.
- std::string **originalName**

  Original read name.

4.16.1 Detailed Description

Read portion for re-mapping.

The documentation for this struct was generated from the following file:

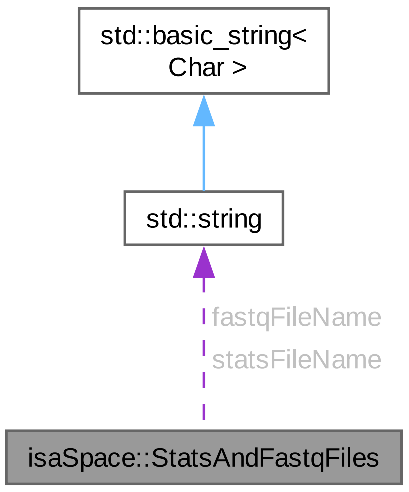
- isoseqAlgn.hpp

4.17 isaSpace::StatsAndFastqFiles Struct Reference

Mismatch statistics and FASTQ file name pair.

```
#include <isoseqAlgn.hpp>
```

Collaboration diagram for isaSpace::StatsAndFastqFiles:



Public Attributes

- std::string **statsFileName**

  Statistics file name.
- std::string **fastqFileName**

  FASTQ file name.

4.17.1 Detailed Description

Mismatch statistics and FASTQ file name pair.

The documentation for this struct was generated from the following file:

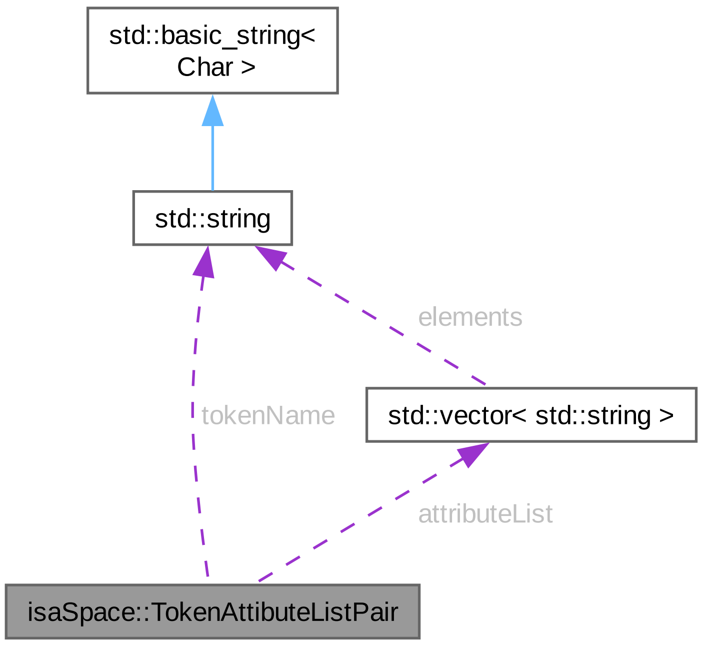
- isoseqAlgn.hpp

4.18 isaSpace::TokenAttibuteListPair Struct Reference

Token name and GFF attribute list pair.

`#include <isoseqAlgn.hpp>`

Collaboration diagram for isaSpace::TokenAttibuteListPair:


std::basic_string< Char >
std::string
elements
tokenName
std::vector< std::string >
attributeList
isaSpace::TokenAttibuteListPair

Public Attributes

- std::string **tokenName**

    GFF token name.
- std::vector< std::string > **attributeList**

    Attribute list.

4.18.1 Detailed Description

Token name and GFF attribute list pair.

The documentation for this struct was generated from the following file:

- isoseqAlgn.hpp

Chapter 5

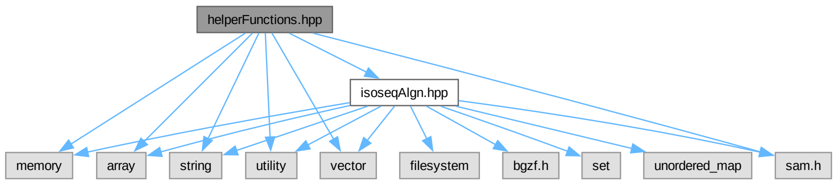
File Documentation

5.1 helperFunctions.hpp File Reference

Genomic analyses helper functions.

```
#include <memory>
#include <array>
#include <string>
#include <utility>
#include <vector>
#include "sam.h"
#include "isoseqAlgn.hpp"
```

Include dependency graph for helperFunctions.hpp:



Typedefs

- using **isaSpace::bamGFFvector** = std::vector< std::pair<BAMrecord, ExonGroup> >

  Alias for BAM record and exon group pair vector.

Functions

- std::string isaSpace::extractAttributeName (const TokenAttibuteListPair &tokenAndAttrList)

  Extract a name.
- std::string isaSpace::extractParentName (const std::string &attributeString)

  Extract parent name.
- bool isaSpace::rangesOverlap (const std::pair< hts_pos_t, hts_pos_t > &range1, const std::pair< hts_pos_t, hts_pos_t > &range2) noexcept

  Test for range overlap.
- std::array< std::string, nGFFfields > isaSpace::parseGFFline (const std::string &gffLine)

  Parse a GFF line into fields.
- std::unordered_map< std::string, std::vector< ExonGroup > > isaSpace::parseGFF (const std::string &gffFile←Name)

  Parse a GFF file.
- std::vector< std::vector< float >::const_iterator > isaSpace::getPeaks (const std::vector< float > &values, const float &threshold)

  Identify peaks in numerical data.
- std::vector< std::vector< float >::const_iterator > isaSpace::getValleys (const std::vector< float > &values, const float &threshold)

  Identify valleys in numerical data.
- std::vector< float > isaSpace::getReferenceMatchStatus (const std::vector< uint32_t > &cigar)

  Read match status along the reference.
- ReadExonCoverage isaSpace::getExonCoverageStats (const std::pair< BAMrecord, ExonGroup > &readAnd←Exons)

  Extract exon coverage statistics for a read.
- std::string isaSpace::stringifyExonCoverage (const ReadExonCoverage &readRecord, char separator='\t')

  Convert `ReadExonCoverage` to string.
- std::string isaSpace::stringifyAlignmentRange (const bamGFFvector::const_iterator &begin, const bam←GFFvector::const_iterator &end)

  Produce a string from a range of read alignments.
- std::string isaSpace::stringifyUnmappedRegions (const bamGFFvector::const_iterator &begin, const bam←GFFvector::const_iterator &end, const BinomialWindowParameters &windowParameters)

  Produce a string of poorly aligned region statistics from an alignment range.
- std::pair< std::string, std::string > isaSpace::getUnmappedRegionsAndFASTQ (const bamGFFvector::const←_iterator &begin, const bamGFFvector::const_iterator &end, const BinomialWindowParameters &window←Parameters)

  Produce a string of poorly aligned region statistics and corresponding FASTQ records from an alignment range.
- ReadPortion isaSpace::parseRemappedReadName (const std::string &remappedReadName)

  Extract original read name and coordinates.
- std::unique_ptr< bam1_t, BAMrecordDeleter > isaSpace::modifyCIGAR (const ReadPortion &modRange, const std::unique_ptr< bam1_t, BAMrecordDeleter > &bamRecord)

  Modify the BAM CIGAR string to erase alignment in a range.
- void isaSpace::addRemappedSecondaryAlignment (const std::unique_ptr< sam_hdr_t, BAMheaderDeleter > &newRecordHeader, const std::unique_ptr< bam1_t, BAMrecordDeleter > &newRecord, const ReadPortion &remapInfo, const std::unique_ptr< sam_hdr_t, BAMheaderDeleter > &originalHeader, const float &remap←IdentityCutoff, std::vector< std::unique_ptr< bam1_t, BAMrecordDeleter > > &readMapVector)

  Add a re-mapped secondary alignment.
- std::unique_ptr< BGZF, BGZFhandleDeleter > isaSpace::openBGZFtoAppend (const std::string &bamFileName)

  Open a BGZF file handle for appending.

- std::vector< std::pair< bamGFFvector::const_iterator, bamGFFvector::const_iterator > > isaSpace::makeThreadRanges (const bamGFFvector &targetVector, const size_t &threadCount)

  Make per-thread alignment record/annotation vector ranges.
- std::unordered_map< std::string, std::string > isaSpace::parseCL (int &argc, char **argv)

  Command line parser.
- void isaSpace::extractCLinfo (const std::unordered_map< std::string, std::string > &parsedCLI, std::unordered↩ _map< std::string, int > &intVariables, std::unordered_map< std::string, float > &floatVariables, std::↩ unordered_map< std::string, std::string > &stringVariables)

  Extract parameters from parsed command line interface flags.

Variables

- constexpr size_t **isaSpace::nGFFfields** {9UL}

  Number of GFF file fields.

5.1.1 Detailed Description

Genomic analyses helper functions.

Author

Anthony J. Greenberg and Rebekah Rogers

Copyright

Copyright (c) 2024 Anthony J. Greenberg and Rebekah Rogers

Version

0.2

Definitions of class-external functions needed by genomic analyses.

5.1.2 Function Documentation

5.1.2.1 addRemappedSecondaryAlignment()

```
void isaSpace::addRemappedSecondaryAlignment (
            const std::unique_ptr< sam_hdr_t, BAMheaderDeleter > & newRecordHeader,
            const std::unique_ptr< bam1_t, BAMrecordDeleter > & newRecord,
            const ReadPortion & remapInfo,
            const std::unique_ptr< sam_hdr_t, BAMheaderDeleter > & originalHeader,
            const float & remapIdentityCutoff,
            std::vector< std::unique_ptr< bam1_t, BAMrecordDeleter > > & readMapVector)
```

Add a re-mapped secondary alignment.

Add a read portion remap as a secondary alignment to a vector of BAM records. Only reads that pass the identity threshold are added.

Parameters

in	*newRecordHeader*	header corresponding to the re-mapped record

in	*newRecord*	remapped BAM record
in	*remapInfo*	original name and read segment range
in	*originalHeader*	header corresponding to the original record
in	*remapIdentityCutoff*	fraction of sites in the remapped read that are identical to the reference
in,out	*readMapVector*	vector of alignments of a read, first element is the primary alignment

5.1.2.2 extractAttributeName()

```
std::string isaSpace::extractAttributeName (
            const TokenAttibuteListPair & tokenAndAttrList)  [nodiscard]
```

Extract a name.

Extract a token name from GFF attributes.

Parameters

in	*tokenAndAttrList*	field token and the list of attributes

5.1.2.3 extractCLinfo()

```
void isaSpace::extractCLinfo (
            const std::unordered_map< std::string, std::string > & parsedCLI,
            std::unordered_map< std::string, int > & intVariables,
            std::unordered_map< std::string, float > & floatVariables,
            std::unordered_map< std::string, std::string > & stringVariables)
```

Extract parameters from parsed command line interface flags.

Extracts needed variable values, indexed by `std::string` encoded variable names.

Parameters

in	*parsedCLI*	flag values parsed from the command line
out	*intVariables*	indexed `int` variables for use by `main()`
out	*floatVariables*	indexed `float` variables for use by `main()`
out	*stringVariables*	indexed `std::string` variables for use by `main()`

5.1.2.4 extractParentName()

```
std::string isaSpace::extractParentName (
              const std::string & attributeString)  [nodiscard]
```

Extract parent name.

Extract the value of the `Parent=` attribute from the provided GFF attribute string.

Parameters

in	*attributeString*	GFF attribute string

Returns

parent name

5.1.2.5 getExonCoverageStats()

```
ReadExonCoverage isaSpace::getExonCoverageStats (
              const std::pair< BAMrecord, ExonGroup > & readAndExons)  [nodiscard]
```

Extract exon coverage statistics for a read.

Parameters

in	*readAndExons*	read alignment with the corresponding exon group

Returns

exon coverage object

5.1.2.6 getPeaks()

```
std::vector< std::vector< float >::const_iterator > isaSpace::getPeaks (
              const std::vector< float > & values,
              const float & threshold)  [nodiscard]
```

Identify peaks in numerical data.

Returns iterators to the elements in a vector that correspond to peaks above the provided threshold. Last element is the `const_iterator` to the end of the vector unless it is empty, and is the only element if there are no peaks.

Parameters

in	*values*	vector of values

in	*threshold*	value that must be exceeded for a peak call

Returns

vector of iterators to peak elements

5.1.2.7 getReferenceMatchStatus()

```
std::vector< float > isaSpace::getReferenceMatchStatus (
            const std::vector< uint32_t > & cigar)  [nodiscard]
```

Read match status along the reference.

Parses CIGAR to track read (query) match/mismatch (1.0 for match, 0.0 for mismatch) status along the reference. This means that insertions in the read are ignored. The vector start begins at the position closest to the first exon start regardless of strand.

Parameters

in	*cigar*	CIGAR vector

Returns

vector of match status

5.1.2.8 getUnmappedRegionsAndFASTQ()

```
std::pair< std::string, std::string > isaSpace::getUnmappedRegionsAndFASTQ (
            const bamGFFvector::const_iterator & begin,
            const bamGFFvector::const_iterator & end,
            const BinomialWindowParameters & windowParameters)  [nodiscard]
```

Produce a string of poorly aligned region statistics and corresponding FASTQ records from an alignment range.

Only saves information from reads that have poorly mapped regions, potentially multiple per read.

Parameters

in	*begin*	start iterator
in	*end*	end iterator
in	*windowParameters*	sliding window parameters

Returns

strings with coverage information (`.first`) and FASTQ (`.second`)

5.1.2.9 getValleys()

```
std::vector< std::vector< float >::const_iterator > isaSpace::getValleys (
            const std::vector< float > & values,
            const float & threshold)  [nodiscard]
```

Identify valleys in numerical data.

Returns iterators to the elements in a vector that correspond to valleys below the provided threshold. Last element is the `const_iterator` to the end of the vector unless it is empty, and is the only element if there are no valleys.

Parameters

`in`	*values*	vector of values
`in`	*threshold*	value that must exceed the valley values

Returns

vector of iterators to valley elements

5.1.2.10 makeThreadRanges()

```
std::vector< std::pair< bamGFFvector::const_iterator, bamGFFvector::const_iterator > > isa↩
Space::makeThreadRanges (
            const bamGFFvector & targetVector,
            const size_t & threadCount)  [nodiscard]
```

Make per-thread alignment record/annotation vector ranges.

Constructs a vector of iterator pairs bracketing chunks of a vector to be processed in parallel.

Parameters

`in`	*targetVector*	the vector to be processed
`in`	*threadCount*	number of threads

Returns

vector of iterator pairs for each thread

5.1.2.11 modifyCIGAR()

```
std::unique_ptr< bam1_t, BAMrecordDeleter > isaSpace::modifyCIGAR (
            const ReadPortion & modRange,
            const std::unique_ptr< bam1_t, BAMrecordDeleter > & bamRecord)  [nodiscard]
```

Modify the BAM CIGAR string to erase alignment in a range.

Substitute operations in a BAM record within the given range with non-matching operations.

Parameters

`in`	*modRange*	modification range

in	*bamRecord*	BAM record to be modified

Returns

BAM record with the CIGAR vector replaced

5.1.2.12 openBGZFtoAppend()

```
std::unique_ptr< BGZF, BGZFhandleDeleter > isaSpace::openBGZFtoAppend (
            const std::string & bamFileName)
```

Open a BGZF file handle for appending.

Opens a handle to the BAM file for appending, deleting the original if it exists.

Parameters

in	*bamFileName*	BAM file name

5.1.2.13 parseCL()

```
std::unordered_map< std::string, std::string > isaSpace::parseCL (
            int & argc,
            char ** argv)  [nodiscard]
```

Command line parser.

Maps flags to values. Flags assumed to be of the form `--flag-name value`.

Parameters

in	*argc*	size of the `argv` array
in	*argv*	command line input array

Returns

map of tags to values

5.1.2.14 parseGFF()

```
std::unordered_map< std::string, std::vector< ExonGroup > > isaSpace::parseGFF (
            const std::string & gffFileName) [nodiscard]
```

Parse a GFF file.

Extract exons from a GFF file and group them by gene and chromosome/linkage group. The map keys are linkage groups, scaffolds, or chromosomes plus strand ID.

Parameters

in	*gffFileName*	GFF file name

Returns

collection of exon group vectors by chromosome and strand

5.1.2.15 parseGFFline()

```
std::array< std::string, nGFFfields > isaSpace::parseGFFline (
            const std::string & gffLine) [nodiscard]
```

Parse a GFF line into fields.

Places `FAIL` in the first element if the number of fields is not `nGFFfields` as required by the GFF specification.

Parameters

in	*gffLine*	one line of a GFF file

Returns

each GFF field in a separate element

5.1.2.16 parseRemappedReadName()

```
ReadPortion isaSpace::parseRemappedReadName (
            const std::string & remappedReadName) [nodiscard]
```

Extract original read name and coordinates.

The remapped read names are original names, with _-separated start and end coordinates. Underscores in the original read name are allowed. If the coordinates are absent, returns an empty object.

Parameters

in	*remappedReadName*	re-mapped read portion name

Returns

original read name and segment coordinates

5.1.2.17 rangesOverlap()

```
bool isaSpace::rangesOverlap (
            const std::pair< hts_pos_t, hts_pos_t > & range1,
            const std::pair< hts_pos_t, hts_pos_t > & range2)  [nodiscard], [noexcept]
```

Test for range overlap.

The first position in each range need not be before the second in the same range.

Parameters

in	*range1*	first range
in	*range2*	second range

Returns

`true` if there is overlap

5.1.2.18 stringifyAlignmentRange()

```
std::string isaSpace::stringifyAlignmentRange (
            const bamGFFvector::const_iterator & begin,
            const bamGFFvector::const_iterator & end)  [nodiscard]
```

Produce a string from a range of read alignments.

Parameters

in	*begin*	start iterator
in	*end*	end iterator

Returns

string with coverage information

5.1.2.19 stringifyExonCoverage()

```
std::string isaSpace::stringifyExonCoverage (
            const ReadExonCoverage & readRecord,
            char separator = '\t')  [nodiscard]
```

Convert `ReadExonCoverage` to string.

Parameters

in	*readRecord*	individual read record

in	*separator*	field separator

Returns

`std::string` with the read record elements, without a new line at the end

5.1.2.20 stringifyUnmappedRegions()

```
std::string isaSpace::stringifyUnmappedRegions (
            const bamGFFvector::const_iterator & begin,
            const bamGFFvector::const_iterator & end,
            const BinomialWindowParameters & windowParameters)  [nodiscard]
```

Produce a string of poorly aligned region statistics from an alignment range.

Only saves information from reads that have poorly mapped regions, potentially multiple per read.

Parameters

in	*begin*	start iterator
in	*end*	end iterator
in	*windowParameters*	sliding window parameters

Returns

string with coverage information

5.2 helperFunctions.hpp

Go to the documentation of this file.

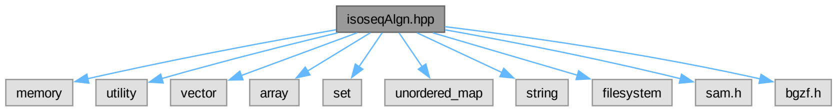
```
/*
 * Copyright (c) 2024-2025 Anthony J. Greenberg and Rebekah Rogers
 *
 * Redistribution and use in source and binary forms, with or without modification, are permitted provided
 that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice, this list of conditions and
 the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions
 and the following disclaimer in the documentation and/or other materials provided with the distribution.
 *
 * 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or
 promote products derived from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
 WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
 EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
 * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
 * IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
 * THE POSSIBILITY OF SUCH DAMAGE.
 */



 #pragma once

 #include <memory>
 #include <array>
 #include <string>
 #include <utility> // for std::pair
 #include <vector>

 #include "sam.h"

 #include "isoseqAlgn.hpp"

 namespace isaSpace {
     constexpr size_t nGFFfields{9UL};
     using bamGFFvector = std::vector< std::pair<BAMrecord, ExonGroup> >;

     [[nodiscard]] std::string extractAttributeName(const TokenAttibuteListPair &tokenAndAttrList);

     [[nodiscard]] std::string extractParentName(const std::string &attributeString);

     [[nodiscard]] bool rangesOverlap(const std::pair<hts_pos_t, hts_pos_t> &range1, const std::pair<hts_pos_t, hts_pos_t> &range2) noexcept;

     [[nodiscard]] std::array<std::string, nGFFfields> parseGFFline(const std::string &gffLine);

     [[nodiscard]] std::unordered_map< std::string, std::vector<ExonGroup> > parseGFF(const std::string &gffFileName);

     [[nodiscard]] std::vector<std::vector<float>::const_iterator> getPeaks(const std::vector<float> &values, const float &threshold);

     [[nodiscard]] std::vector<std::vector<float>::const_iterator> getValleys(const std::vector<float> &values, const float &threshold);

     [[nodiscard]] std::vector<float> getReferenceMatchStatus(const std::vector<uint32_t> &cigar);

     [[nodiscard]] ReadExonCoverage getExonCoverageStats(const std::pair<BAMrecord, ExonGroup> &readAndExons);

     [[nodiscard]] std::string stringifyExonCoverage(const ReadExonCoverage &readRecord, char separator = '\t');

     [[nodiscard]] std::string stringifyAlignmentRange(const bamGFFvector::const_iterator &begin, const bamGFFvector::const_iterator &end);
     [[nodiscard]] std::string stringifyUnmappedRegions(const bamGFFvector::const_iterator &begin, const bamGFFvector::const_iterator &end, const BinomialWindowParameters &windowParameters);
     [[nodiscard]] std::pair<std::string, std::string> getUnmappedRegionsAndFASTQ(const bamGFFvector::const_iterator &begin, const bamGFFvector::const_iterator &end, const BinomialWindowParameters &windowParameters);

     [[nodiscard]] ReadPortion parseRemappedReadName(const std::string &remappedReadName);
     [[nodiscard]] std::unique_ptr<bam1_t, BAMrecordDeleter> modifyCIGAR(const ReadPortion &modRange, const std::unique_ptr<bam1_t, BAMrecordDeleter> &bamRecord);
     void addRemappedSecondaryAlignment(
             const std::unique_ptr<sam_hdr_t, BAMheaderDeleter> &newRecordHeader, const std::unique_ptr<bam1_t, BAMrecordDeleter> &newRecord, const ReadPortion &remapInfo,
             const std::unique_ptr<sam_hdr_t, BAMheaderDeleter> &originalHeader, const float &remapIdentityCutoff, std::vector< std::unique_ptr<bam1_t, BAMrecordDeleter> > &readMapVector);

     std::unique_ptr<BGZF, BGZFhandleDeleter> openBGZFtoAppend(const std::string &bamFileName);

     [[nodiscard]] std::vector< std::pair<bamGFFvector::const_iterator, bamGFFvector::const_iterator> >
         makeThreadRanges(const bamGFFvector &targetVector, const size_t &threadCount);

     [[nodiscard]] std::unordered_map<std::string, std::string> parseCL(int &argc, char **argv);

     void extractCLinfo(const std::unordered_map<std::string, std::string> &parsedCLI,
             std::unordered_map<std::string, int> &intVariables,
             std::unordered_map<std::string, float> &floatVariables,
             std::unordered_map<std::string, std::string> &stringVariables);
 }
```

5.3 isoseqAlgn.hpp File Reference

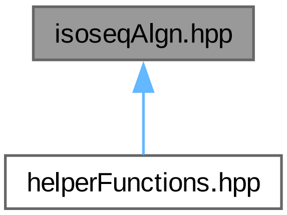
Read isoSeq alignments and compare to genome annotations.

```
#include <memory>
#include <utility>
#include <vector>
#include <array>
#include <set>
#include <unordered_map>
#include <string>
#include <filesystem>
#include "sam.h"
#include "bgzf.h"
```

Include dependency graph for isoseqAlgn.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct isaSpace::BamAndGffFiles

  BAM and GFF file name pair.
- struct isaSpace::StatsAndFastqFiles

  Mismatch statistics and FASTQ file name pair.
- struct isaSpace::TokenAttibuteListPair

  Token name and GFF attribute list pair.
- struct isaSpace::MappedReadInterval

Delineates an interval in a mapped read.
- struct isaSpace::MappedReadMatchStatus

  Read match and map start.
- struct isaSpace::BinomialWindowParameters

  Binomial window parameters.
- struct isaSpace::ReadExonCoverage

  Exons covered by a read.
- struct isaSpace::ReadPortion

  Read portion for re-mapping.
- struct isaSpace::BAMsecondary

  BAM secondary alignment.
- struct isaSpace::BAMheaderDeleter

  BAM header pointer deleter.
- struct isaSpace::BAMrecordDeleter

  BAM record pointer deleter.
- struct isaSpace::BGZFhandleDeleter

  BGZF handle pointer deleter.
- class isaSpace::BAMsafeReader

  BAM file safe reader.
- class isaSpace::ExonGroup

  Group of exons from the same gene.
- class isaSpace::ReadMatchWindowBIC

  Binomial BIC for a read window.
- class isaSpace::BAMrecord

  Summary of a BAM record set.
- class isaSpace::BAMtoGenome

  Relate BAM alignments to exons.
- class isaSpace::BAMfile

  Records from a BAM file.

5.3.1 Detailed Description

Read isoSeq alignments and compare to genome annotations.

Author

Anthony J. Greenberg and Rebekah Rogers

Copyright

Copyright (c) 2024 Anthony J. Greenberg and Rebekah Rogers

Version

0.2

Interface definitions of classes that take `.bam` files with isoSeq alignments and identify potential fused transcripts.

5.4 isoseqAlgn.hpp

Go to the documentation of this file.

```
/*
 * Copyright (c) 2024-2025 Anthony J. Greenberg and Rebekah Rogers
 *
 * Redistribution and use in source and binary forms, with or without modification, are permitted provided
that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice, this list of conditions and
the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions
and the following disclaimer in the documentation and/or other materials provided with the distribution.
 *
 * 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or
promote products derived from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
 * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER
 * IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
 * THE POSSIBILITY OF SUCH DAMAGE.
 */



#pragma once

#include <memory>
#include <utility>
#include <vector>
#include <array>
#include <set>
#include <unordered_map>
#include <string>
#include <filesystem>

#include "sam.h"
#include "bgzf.h"

namespace isaSpace {
    struct BamAndGffFiles;
    struct StatsAndFastqFiles;
    struct TokenAttibuteListPair;
    struct MappedReadInterval;
    struct MappedReadMatchStatus;
    struct BinomialWindowParameters;
    struct ReadExonCoverage;
    struct ReadPortion;
    struct BAMsecondary;
    struct BAMheaderDeleter;
    struct BAMrecordDeleter;
    struct BGZFhandleDeleter;
    class  BAMsafeReader;
    class  ExonGroup;
    class  ReadMatchWindowBIC;
    class  BAMrecord;
    class  BAMtoGenome;
    class  BAMfile;

    struct BamAndGffFiles {
        std::string bamFileName;
        std::string gffFileName;
    };
    struct StatsAndFastqFiles {
        std::string statsFileName;
        std::string fastqFileName;
    };
    struct TokenAttibuteListPair {
        std::string tokenName;
        std::vector<std::string> attributeList;
```

```
};
struct MappedReadInterval {
    hts_pos_t readStart{-1};
    hts_pos_t readEnd{-1};
    hts_pos_t referenceStart{-1};
    hts_pos_t referenceEnd{-1};
};
struct MappedReadMatchStatus {
    hts_pos_t mapStart{-1};
    std::vector<float> matchStatus;
};
struct BinomialWindowParameters {
    float currentProbability{0.0F};
    float alternativeProbability{0.0F};
    float bicDifferenceThreshold{0.0F};
    std::vector< std::pair<float, hts_pos_t> >::difference_type windowSize{0};
};
struct ReadExonCoverage {
    std::string chromosomeName;
    std::string readName;
    hts_pos_t alignmentStart{0};
    hts_pos_t alignmentEnd{0};
    hts_pos_t bestAlignmentStart{0};
    hts_pos_t bestAlignmentEnd{0};
    uint32_t firstSoftClipLength{0};
    uint16_t nSecondaryAlignments{0};
    uint16_t nGoodSecondaryAlignments{0};
    uint16_t nLocalReversedAlignments{0};
    uint16_t nExons{0};
    std::string geneName;
    char strand{'+'};
    hts_pos_t firstExonLength{-1};
    hts_pos_t firstExonStart{-1};
    hts_pos_t lastExonEnd{-1};
    std::vector<float> exonCoverageScores;
    std::vector<float> bestExonCoverageScores;
};
struct ReadPortion {
    size_t start{0};
    size_t end{0};
    std::string originalName;
};
struct BAMsecondary {
    hts_pos_t mapStart{-1};
    hts_pos_t mapEnd{-1};
    bool sameStrandAsPrimary{false};
    std::vector<uint32_t> cigar;
};
struct BAMheaderDeleter {
    void operator()(sam_hdr_t *header) const { sam_hdr_destroy(header); }
};
struct BAMrecordDeleter {
    void operator()(bam1_t *bamRecord) const { bam_destroy1(bamRecord); }
};
struct BGZFhandleDeleter {
    void operator()(BGZF *bgzfHandle) const { bgzf_close(bgzfHandle); }
};

class BAMsafeReader {
public:
    BAMsafeReader() = default;
    BAMsafeReader(const std::string &bamFileName);
    BAMsafeReader(const BAMsafeReader &toCopy) = delete;
    BAMsafeReader& operator=(const BAMsafeReader &toCopy) = delete;
    BAMsafeReader(BAMsafeReader &&toMove) noexcept {
        *this = std::move(toMove);
    };
    BAMsafeReader& operator=(BAMsafeReader &&toMove) noexcept;
    ~BAMsafeReader();

    [[nodiscard]] std::unique_ptr<sam_hdr_t, BAMheaderDeleter> getHeaderCopy() const;
    [[nodiscard]] std::pair<std::unique_ptr<bam1_t, BAMrecordDeleter>, int32_t> getNextRecord();
private:
    static const uint16_t nRetries_;
    BGZF *fileHandle_{nullptr};
    std::unique_ptr<sam_hdr_t, BAMheaderDeleter> headerUPointer_{nullptr};
    std::string fileName_;
    std::filesystem::perms initialPermissions_{std::filesystem::perms::none};
};
class ExonGroup {
public:
    ExonGroup() = default;
    ExonGroup(std::string geneName, const char strand, std::set< std::pair<hts_pos_t, hts_pos_t> > &exonSet);
    ExonGroup(std::string geneName, const char strand, const std::vector<std::string> &exonLinesFomGFF);
    ExonGroup(const ExonGroup &toCopy) = default;
    ExonGroup& operator=(const ExonGroup &toCopy) = default;
    ExonGroup(ExonGroup &&toMove) noexcept = default;
    ExonGroup& operator=(ExonGroup &&toMove) noexcept = default;
    ~ExonGroup() = default;

    [[nodiscard]] bool empty() const noexcept { return exonRanges_.empty(); };
    [[nodiscard]] std::string geneName() const { return geneName_; };
    [[nodiscard]] size_t nExons() const noexcept { return exonRanges_.size(); };
    [[nodiscard]] char strand() const noexcept { return isNegativeStrand_ ? '-' : '+'; };
    [[nodiscard]] std::pair<hts_pos_t, hts_pos_t> at(const size_t &idx) const {return exonRanges_.at(idx);};
    [[nodiscard]] std::pair<hts_pos_t, hts_pos_t> geneSpan() const noexcept;
    [[nodiscard]] std::pair<hts_pos_t, hts_pos_t> firstExonSpan() const noexcept;
    [[nodiscard]] hts_pos_t firstExonLength() const noexcept;
    [[nodiscard]] uint32_t firstExonAfter(const hts_pos_t &position) const noexcept;
    [[nodiscard]] uint32_t firstOverlappingExon(const hts_pos_t &position) const noexcept;
    [[nodiscard]] uint32_t lastExonBefore(const hts_pos_t &position) const noexcept;
    [[nodiscard]] uint32_t lastOverlappingExon(const hts_pos_t &position) const noexcept;
    [[nodiscard]] std::pair<hts_pos_t, hts_pos_t> getFirstIntronSpan() const;
    [[nodiscard]] std::vector<float> getExonCoverageQuality(const BAMrecord &alignment) const;
    [[nodiscard]] std::vector<float> getBestExonCoverageQuality(const BAMrecord &alignment) const;
private:
    static const size_t strandIDidx_;
    static const char   gffDelimiter_;
    static const size_t spanStart_;
    static const size_t spanEnd_;
    std::string geneName_;
    bool isNegativeStrand_{false};
    std::vector< std::pair<hts_pos_t, hts_pos_t> > exonRanges_;
    // TODO: to implement tracking transcripts, additional vector of iterators pointing to the next exon
    // Keep the number the same and equal to the number of transcripts listed in the GFF
};

class ReadMatchWindowBIC {
public:
    ReadMatchWindowBIC() = default;
    ReadMatchWindowBIC(const std::vector< std::pair<float, hts_pos_t> >::const_iterator &windowBegin, const BinomialWindowParameters &windowParameters) :
        leftProbability_{windowParameters.currentProbability},
        rightProbability_{windowParameters.alternativeProbability},
        kSuccesses_{calculateKsuccesses_(windowBegin, windowParameters)},
        nTrials_{static_cast<float>(windowParameters.windowSize)} {};
    ReadMatchWindowBIC(const ReadMatchWindowBIC &toCopy) = default;
    ReadMatchWindowBIC& operator=(const ReadMatchWindowBIC &toCopy) = default;
    ReadMatchWindowBIC(ReadMatchWindowBIC &&toMove) noexcept = default;
    ReadMatchWindowBIC& operator=(ReadMatchWindowBIC &&toMove) noexcept = default;
    ~ReadMatchWindowBIC() = default;

    [[nodiscard]] float getBICdifference() const noexcept;
private:
    float leftProbability_{0.0F};
    float rightProbability_{0.0F};
    float kSuccesses_{0.0F};
    float nTrials_{0.0F};

    [[nodiscard]] static float calculateKsuccesses_(const std::vector< std::pair<float, hts_pos_t> >::const_iterator &windowBegin, const BinomialWindowParameters &windowParameters);
};

class BAMrecord {
public:
    BAMrecord() = default;
    BAMrecord(const bam1_t *alignmentRecord, const sam_hdr_t *samHeader);
    BAMrecord(const BAMrecord &toCopy) = default;
    BAMrecord& operator=(const BAMrecord &toCopy) = default;
    BAMrecord(BAMrecord &&toMove) noexcept = default;
    BAMrecord& operator=(BAMrecord &&toMove) noexcept = default;
    ~BAMrecord() = default;

    void addSecondaryAlignment(const bam1_t *alignmentRecord, const sam_hdr_t *samHeader, const hts_pos_t localWindow = 500'000);
    [[nodiscard]] std::string getReadName() const {return readName_; };
    [[nodiscard]] hts_pos_t getMapStart() const noexcept { return mapStart_; };
    [[nodiscard]] hts_pos_t getMapEnd() const noexcept { return mapEnd_; };
    [[nodiscard]] hts_pos_t getmRNAstart() const noexcept {
        return isRev_ ? mapEnd_ : mapStart_;
    };
    [[nodiscard]] bool isRevComp() const noexcept { return isRev_; };
    [[nodiscard]] bool isMapped() const noexcept { return isMapped_; };
    [[nodiscard]] bool hasSecondaryAlignments() const noexcept { return secondaryAlignmentCount_ > 0; };
    [[nodiscard]] bool hasLocalSecondaryAlignments() const noexcept { return !localSecondaryAlignments_.empty(); };
    [[nodiscard]] uint16_t secondaryAlignmentCount() const noexcept { return secondaryAlignmentCount_; };
    [[nodiscard]] uint16_t localSecondaryAlignmentCount() const noexcept { return static_cast<uint16_t>( localSecondaryAlignments_.size() ); };
    [[nodiscard]] uint16_t localReversedSecondaryAlignmentCount() const noexcept;
    [[nodiscard]] hts_pos_t getReadLength() const noexcept { return static_cast<hts_pos_t>( sequenceAndQuality_.size() ); };
    [[nodiscard]] std::vector<uint32_t> getCIGARvector() const { return cigar_; };
    [[nodiscard]] std::string getCIGARstring() const;
    [[nodiscard]] uint32_t getFirstCIGAR() const noexcept;
    [[nodiscard]] std::string getReferenceName() const {return referenceName_; };
    [[nodiscard]] MappedReadMatchStatus getBestReferenceMatchStatus() const;
    [[nodiscard]] std::vector< std::pair<float, hts_pos_t> > getReadCentricMatchStatus() const;
    [[nodiscard]] std::vector<MappedReadInterval> getPoorlyMappedRegions(const BinomialWindowParameters &windowParameters) const;
    [[nodiscard]] std::string getSequenceAndQuality(const MappedReadInterval &segmentBoundaries) const;
private:
    static const uint16_t sequenceMask_;
    static const uint16_t qualityShift_;
    static const uint16_t suppSecondaryAlgn_;
    static const std::array<hts_pos_t, 10> queryConsumption_;
    static const std::array<hts_pos_t, 10> referenceConsumption_;
    static const std::array<char, 16> seqNT16str_;
    static const std::array<char, 16> complSeqNT16str_;
    static const std::array<float, 10> sequenceMatch_;
    bool isMapped_{false};
    bool isRev_{false};
    uint16_t secondaryAlignmentCount_{0};
    hts_pos_t mapStart_{0};
    hts_pos_t mapEnd_{0};
    std::string readName_;
    std::string referenceName_;
    std::vector<uint32_t> cigar_;
    std::vector<uint16_t> sequenceAndQuality_;
    std::vector<BAMsecondary> localSecondaryAlignments_;
};

class BAMtoGenome {
public:
    BAMtoGenome() = default;
    BAMtoGenome(const BamAndGffFiles &bamGFFfilePairNames);
    BAMtoGenome(const BAMtoGenome &toCopy) = default;
    BAMtoGenome& operator=(const BAMtoGenome &toCopy) = default;
    BAMtoGenome(BAMtoGenome &&toMove) noexcept = default;
    BAMtoGenome& operator=(BAMtoGenome &&toMove) noexcept = default;
    ~BAMtoGenome() = default;

    [[nodiscard]] size_t nChromosomes() const noexcept;
    [[nodiscard]] size_t nExonSets() const noexcept;
    void saveReadCoverageStats(const std::string &outFileName, const size_t &nThreads) const;
    void saveUnmappedRegions(const std::string &outFileName, const BinomialWindowParameters &windowParameters, const size_t &nThreads) const;
    void saveUnmappedRegions(const StatsAndFastqFiles &outFilePair, const BinomialWindowParameters &windowParameters, const size_t &nThreads) const;
private:
    static const uint16_t suppSecondaryAlgn_;

    std::vector< std::pair<BAMrecord, ExonGroup> > readsAndExons_;

    [[nodiscard]] std::vector<ExonGroup>::const_iterator findOverlappingGene_(const std::vector<ExonGroup> &chromosomeExonGroups,
            const std::vector<ExonGroup>::const_iterator &exonGroupSearchStart, BAMrecord &alignedRead);
    void processPrimaryAlignment_(const std::string &referenceName, const BAMrecord &alignmentRecord, std::unordered_map<std::string, std::vector<ExonGroup>::const_iterator> &latestExonGroupIts);
    void processSecondaryAlignment_(const std::string &referenceName, const BAMrecord &alignmentRecord, const std::unordered_map<std::string, std::vector<ExonGroup>::const_iterator> &latestExonGroupIts);
};

class BAMfile {
```

```
    public:
        BAMfile() = default;
        BAMfile(const std::string &BAMfileName);
        BAMfile(const BAMfile &toCopy) = delete;
        BAMfile& operator=(const BAMfile &toCopy) = delete;
        BAMfile(BAMfile &&toMove) noexcept = default;
        BAMfile& operator=(BAMfile &&toMove) noexcept = default;
        ~BAMfile() = default;

        [[nodiscard]] size_t getPrimaryAlignmentCount() const noexcept;
        void addRemaps(const std::string &remapBAMfileName, const float &remapIdentityCutoff);
        [[nodiscard]] std::vector<std::string> saveRemappedBAM(const std::string &outputBAMfileName) const;
        [[nodiscard]] std::vector<std::string> saveSortedRemappedBAM(const std::string &outputBAMfileName) const;
    private:
        static const uint16_t secondaryOrUnmappedAlgn_;
        std::unordered_map<
            std::string,
            std::unordered_map<
                std::string,
                std::vector< std::unique_ptr<bam1_t, BAMrecordDeleter> >
            >
        > bamRecords_;
        std::unique_ptr<sam_hdr_t, BAMheaderDeleter> bamFileHeader_;
    };
}
```

Index