



eXpress

0.9.5 BETA

<http://bio.math.berkeley.edu/eXpress>

Generated by Doxygen 1.7.3

Sat Feb 18 2012 07:21:49



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	BAMParser Class Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Constructor & Destructor Documentation . . . . .	6
3.1.2.1	BAMParser . . . . .	6
3.1.2.2	~BAMParser . . . . .	6
3.1.3	Member Function Documentation . . . . .	6
3.1.3.1	header . . . . .	6
3.1.3.2	next_fragment . . . . .	6
3.1.3.3	reset . . . . .	7
3.1.3.4	trans_index . . . . .	7
3.1.3.5	trans_lengths . . . . .	7
3.2	BAMWriter Class Reference . . . . .	7
3.2.1	Detailed Description . . . . .	8
3.2.2	Constructor & Destructor Documentation . . . . .	8
3.2.2.1	BAMWriter . . . . .	8
3.2.2.2	~BAMWriter . . . . .	8
3.2.3	Member Function Documentation . . . . .	8
3.2.3.1	write_fragment . . . . .	8
3.3	BiasBoss Class Reference . . . . .	9
3.3.1	Detailed Description . . . . .	9
3.3.2	Constructor & Destructor Documentation . . . . .	10
3.3.2.1	BiasBoss . . . . .	10
3.3.3	Member Function Documentation . . . . .	10
3.3.3.1	append_output . . . . .	10
3.3.3.2	get_transcript_bias . . . . .	10
3.3.3.3	normalize_expectations . . . . .	10
3.3.3.4	to_string . . . . .	11
3.3.3.5	update_expectations . . . . .	11
3.3.3.6	update_observed . . . . .	11
3.4	Bundle Class Reference . . . . .	11
3.4.1	Detailed Description . . . . .	12
3.4.2	Constructor & Destructor Documentation . . . . .	12

3.4.2.1	Bundle	12
3.4.3	Member Function Documentation	12
3.4.3.1	counts	12
3.4.3.2	incr_counts	13
3.4.3.3	size	13
3.4.3.4	transcripts	13
3.5	BundleTable Class Reference	13
3.5.1	Detailed Description	14
3.5.2	Constructor & Destructor Documentation	14
3.5.2.1	BundleTable	14
3.5.2.2	~BundleTable	14
3.5.3	Member Function Documentation	14
3.5.3.1	bundles	14
3.5.3.2	create_bundle	14
3.5.3.3	merge	15
3.5.3.4	size	15
3.6	CovarTable Class Reference	15
3.6.1	Detailed Description	16
3.6.2	Member Function Documentation	16
3.6.2.1	get	16
3.6.2.2	increment	16
3.6.2.3	size	17
3.7	FLD Class Reference	17
3.7.1	Detailed Description	17
3.7.2	Constructor & Destructor Documentation	17
3.7.2.1	FLD	17
3.7.3	Member Function Documentation	18
3.7.3.1	add_val	18
3.7.3.2	append_output	18
3.7.3.3	max_val	18
3.7.3.4	mean	18
3.7.3.5	pdf	19
3.7.3.6	to_string	19
3.7.3.7	tot_mass	19
3.8	FragHit Struct Reference	19
3.8.1	Detailed Description	20
3.8.2	Member Function Documentation	20
3.8.2.1	length	20
3.8.2.2	pair_status	21
3.8.3	Member Data Documentation	21
3.8.3.1	deletes_l	21
3.8.3.2	deletes_r	21
3.8.3.3	inserts_l	21
3.8.3.4	inserts_r	21
3.8.3.5	left	21
3.8.3.6	left_first	22
3.8.3.7	mapped_trans	22
3.8.3.8	mate_l	22
3.8.3.9	name	22
3.8.3.10	right	22

3.8.3.11	seq_l	22
3.8.3.12	seq_r	22
3.8.3.13	trans_id	23
3.9	Fragment Class Reference	23
3.9.1	Detailed Description	23
3.9.2	Constructor & Destructor Documentation	23
3.9.2.1	~Fragment	23
3.9.3	Member Function Documentation	24
3.9.3.1	add_map_end	24
3.9.3.2	hits	24
3.9.3.3	name	24
3.9.3.4	num_hits	24
3.9.3.5	sample_hit	24
3.10	FrequencyMatrix Class Reference	25
3.10.1	Detailed Description	25
3.10.2	Constructor & Destructor Documentation	26
3.10.2.1	FrequencyMatrix	26
3.10.2.2	FrequencyMatrix	26
3.10.3	Member Function Documentation	26
3.10.3.1	arr	26
3.10.3.2	increment	26
3.10.3.3	increment	27
3.10.3.4	operator()	27
3.10.3.5	operator()	27
3.10.3.6	row	28
3.10.3.7	set_logged	28
3.11	Globals Struct Reference	28
3.11.1	Detailed Description	28
3.12	Indel Struct Reference	29
3.12.1	Detailed Description	29
3.12.2	Constructor & Destructor Documentation	29
3.12.2.1	Indel	29
3.12.3	Member Data Documentation	29
3.12.3.1	len	29
3.12.3.2	pos	30
3.13	MarkovModel Class Reference	30
3.13.1	Detailed Description	30
3.14	MismatchTable Class Reference	30
3.14.1	Detailed Description	31
3.14.2	Constructor & Destructor Documentation	31
3.14.2.1	MismatchTable	31
3.14.3	Member Function Documentation	31
3.14.3.1	activate	31
3.14.3.2	append_output	31
3.14.3.3	log_likelihood	32
3.14.3.4	to_string	32
3.14.3.5	update	32
3.15	Parser Class Reference	33
3.15.1	Detailed Description	33
3.15.2	Member Function Documentation	33

3.15.2.1	header	33
3.15.2.2	next_fragment	34
3.15.2.3	reset	34
3.15.2.4	trans_index	34
3.15.2.5	trans_lengths	34
3.16	ParseThreadSafety Struct Reference	34
3.16.1	Detailed Description	35
3.16.2	Member Data Documentation	35
3.16.2.1	cond	35
3.16.2.2	frag_clean	35
3.16.2.3	mut	35
3.16.2.4	next_frag	35
3.17	PosWeightTable Class Reference	36
3.17.1	Detailed Description	36
3.17.2	Constructor & Destructor Documentation	36
3.17.2.1	PosWeightTable	36
3.17.3	Member Function Documentation	37
3.17.3.1	append_output	37
3.17.3.2	get_weight	37
3.17.3.3	get_weight	37
3.17.3.4	increment_expected	38
3.17.3.5	increment_expected	38
3.17.3.6	increment_observed	38
3.17.3.7	increment_observed	38
3.17.3.8	normalize_expected	39
3.18	RoundParams Struct Reference	39
3.18.1	Detailed Description	39
3.18.2	Constructor & Destructor Documentation	40
3.18.2.1	RoundParams	40
3.18.3	Member Data Documentation	40
3.18.3.1	ambig_mass	40
3.18.3.2	binom_var	40
3.18.3.3	mass	40
3.18.3.4	samp_var	40
3.18.3.5	tot_ambig_mass	40
3.18.3.6	tot_unc	41
3.19	SAMParser Class Reference	41
3.19.1	Detailed Description	41
3.19.2	Constructor & Destructor Documentation	42
3.19.2.1	SAMParser	42
3.19.3	Member Function Documentation	42
3.19.3.1	header	42
3.19.3.2	next_fragment	42
3.19.3.3	trans_index	42
3.19.3.4	trans_lengths	43
3.20	SAMWriter Class Reference	43
3.20.1	Detailed Description	43
3.20.2	Constructor & Destructor Documentation	44
3.20.2.1	SAMWriter	44
3.20.2.2	~SAMWriter	44

3.20.3	Member Function Documentation	44
3.20.3.1	write_fragment	44
3.21	Sequence Class Reference	44
3.21.1	Detailed Description	45
3.21.2	Constructor & Destructor Documentation	45
3.21.2.1	Sequence	45
3.21.2.2	Sequence	45
3.21.2.3	Sequence	45
3.21.2.4	~Sequence	46
3.21.3	Member Function Documentation	46
3.21.3.1	empty	46
3.21.3.2	length	46
3.21.3.3	operator=	46
3.21.3.4	operator[]	46
3.21.3.5	set	47
3.22	SeqWeightTable Class Reference	47
3.22.1	Detailed Description	47
3.22.2	Constructor & Destructor Documentation	48
3.22.2.1	SeqWeightTable	48
3.22.3	Member Function Documentation	48
3.22.3.1	append_output	48
3.22.3.2	copy_expected	48
3.22.3.3	copy_observed	48
3.22.3.4	get_weight	49
3.22.3.5	increment_expected	49
3.22.3.6	increment_observed	49
3.22.3.7	normalize_expected	50
3.22.3.8	to_string	50
3.23	ThreadedMapParser Class Reference	50
3.23.1	Detailed Description	50
3.23.2	Constructor & Destructor Documentation	51
3.23.2.1	ThreadedMapParser	51
3.23.2.2	~ThreadedMapParser	51
3.23.3	Member Function Documentation	51
3.23.3.1	reset_reader	51
3.23.3.2	threaded_parse	51
3.23.3.3	trans_index	52
3.23.3.4	trans_lengths	52
3.23.3.5	write_active	52
3.24	Transcript Class Reference	52
3.24.1	Detailed Description	53
3.24.2	Constructor & Destructor Documentation	54
3.24.2.1	Transcript	54
3.24.3	Member Function Documentation	54
3.24.3.1	add_mass	54
3.24.3.2	ambig_mass	54
3.24.3.3	binom_var	54
3.24.3.4	bundle	55
3.24.3.5	bundle	55
3.24.3.6	cached_effective_length	55

3.24.3.7	<a href="#">est_effective_length</a>	55
3.24.3.8	<a href="#">id</a>	56
3.24.3.9	<a href="#">incr_counts</a>	56
3.24.3.10	<a href="#">length</a>	56
3.24.3.11	<a href="#">log_likelihood</a>	56
3.24.3.12	<a href="#">mass</a>	57
3.24.3.13	<a href="#">name</a>	57
3.24.3.14	<a href="#">rho</a>	57
3.24.3.15	<a href="#">round_reset</a>	57
3.24.3.16	<a href="#">samp_var</a>	58
3.24.3.17	<a href="#">seq</a>	58
3.24.3.18	<a href="#">tot_ambig_mass</a>	58
3.24.3.19	<a href="#">tot_counts</a>	58
3.24.3.20	<a href="#">tot_uncertainty</a>	58
3.24.3.21	<a href="#">uniq_counts</a>	59
3.24.3.22	<a href="#">update_transcript_bias</a>	59
3.25	<a href="#">TranscriptTable Class Reference</a>	59
3.25.1	<a href="#">Detailed Description</a>	60
3.25.2	<a href="#">Constructor &amp; Destructor Documentation</a>	60
3.25.2.1	<a href="#">TranscriptTable</a>	60
3.25.2.2	<a href="#">~TranscriptTable</a>	61
3.25.3	<a href="#">Member Function Documentation</a>	61
3.25.3.1	<a href="#">covar_size</a>	61
3.25.3.2	<a href="#">get_covar</a>	61
3.25.3.3	<a href="#">get_trans</a>	61
3.25.3.4	<a href="#">merge_bundles</a>	62
3.25.3.5	<a href="#">num_bundles</a>	62
3.25.3.6	<a href="#">output_results</a>	62
3.25.3.7	<a href="#">round_reset</a>	62
3.25.3.8	<a href="#">size</a>	63
3.25.3.9	<a href="#">threaded_bias_update</a>	63
3.25.3.10	<a href="#">total_fpb</a>	63
3.25.3.11	<a href="#">update_covar</a>	63
3.25.3.12	<a href="#">update_total_fpb</a>	63
3.26	<a href="#">Writer Class Reference</a>	64
3.26.1	<a href="#">Detailed Description</a>	64
3.26.2	<a href="#">Member Function Documentation</a>	64
3.26.2.1	<a href="#">write_fragment</a>	64



# Chapter 1

## Class Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

BiasBoss . . . . .	9
Bundle . . . . .	11
BundleTable . . . . .	13
CovarTable . . . . .	15
FLD . . . . .	17
FragHit . . . . .	19
Fragment . . . . .	23
FrequencyMatrix . . . . .	25
Globals . . . . .	28
Indel . . . . .	29
MarkovModel . . . . .	30
MismatchTable . . . . .	30
Parser . . . . .	33
BAMParser . . . . .	5
SAMPParser . . . . .	41
ParseThreadSafety . . . . .	34
PosWeightTable . . . . .	36
RoundParams . . . . .	39
Sequence . . . . .	44
SeqWeightTable . . . . .	47
ThreadedMapParser . . . . .	50
Transcript . . . . .	52
TranscriptTable . . . . .	59
Writer . . . . .	64
BAMWriter . . . . .	7
SAMWriter . . . . .	43



## Chapter 2

# Class Index

### 2.1 Class List

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

BAMParser	5
BAMWriter	7
BiasBoss	9
Bundle	11
BundleTable	13
CovarTable	15
FLD	17
FragHit	19
Fragment	23
FrequencyMatrix	25
Globals	28
Indel	29
MarkovModel	30
MismatchTable	30
Parser	33
ParseThreadSafety	34
PosWeightTable	36
RoundParams	39
SAMParser	41
SAMWriter	43
Sequence	44
SeqWeightTable	47
ThreadedMapParser	50
Transcript	52
TranscriptTable	59
Writer	64



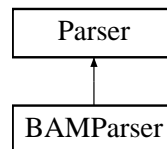
## Chapter 3

# Class Documentation

### 3.1 BAMParser Class Reference

```
#include <mapparser.h>
```

Inheritance diagram for BAMParser:



#### Public Member Functions

- [BAMParser](#) (BamTools::BamReader \*reader)
- [~BAMParser](#) ()
- const std::string [header](#) () const
- const TransIndex & [trans\\_index](#) () const
- const TransIndex & [trans\\_lengths](#) () const
- bool [next\\_fragment](#) ([Fragment](#) &f)
- void [reset](#) ()

#### 3.1.1 Detailed Description

The [BAMParser](#) class fills [Fragment](#) objects by parsing an input file in BAM format.

#### Author

Adam Roberts

**Date**

2011 Artistic License 2.0

Definition at line 102 of file mapparser.h.

**3.1.2 Constructor & Destructor Documentation****3.1.2.1 BAMParser::BAMParser ( BamTools::BamReader \* *reader* )**

[BAMParser](#) constructor sets the reader

Definition at line 230 of file mapparser.cpp.

**3.1.2.2 BAMParser::~BAMParser ( ) [inline]**

[BAMParser](#) destructor deletes the reader

Definition at line 140 of file mapparser.h.

**3.1.3 Member Function Documentation****3.1.3.1 const std::string BAMParser::header ( ) const [inline, virtual]**

a member function that returns a string version of the header

**Returns**

string version of the header

Implements [Parser](#).

Definition at line 146 of file mapparser.h.

**3.1.3.2 bool BAMParser::next\_fragment ( Fragment & *f* ) [virtual]**

a member function that loads all mappings of the next fragment

**Parameters**

<i>f</i>	the empty <a href="#">Fragment</a> to add mappings to
----------	---

**Returns**

true if more reads remain in the BAM file, false otherwise

Implements [Parser](#).

Definition at line 253 of file mapparser.cpp.

### 3.1.3.3 void BAMParser::reset ( ) [virtual]

a member function that resets the parser and rewinds to the beginning of the BAM file

Implements [Parser](#).

Definition at line 310 of file mapparser.cpp.

### 3.1.3.4 const TransIndex& BAMParser::trans\_index ( ) const [inline, virtual]

a member function that returns the transcript-to-index map

#### Returns

the transcript-to-index map

Implements [Parser](#).

Definition at line 152 of file mapparser.h.

### 3.1.3.5 const TransIndex& BAMParser::trans\_lengths ( ) const [inline, virtual]

a member function that returns the transcript-to-length map

#### Returns

the transcript-to-length map

Implements [Parser](#).

Definition at line 158 of file mapparser.h.

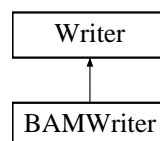
The documentation for this class was generated from the following files:

- src/mapparser.h
- src/mapparser.cpp

## 3.2 BAMWriter Class Reference

```
#include <mapparser.h>
```

Inheritance diagram for BAMWriter:



## Public Member Functions

- [BAMWriter](#) (BamTools::BamWriter \*writer, bool sample)
- [~BAMWriter](#) ()
- void [write\\_fragment](#) ([Fragment](#) &f)

### 3.2.1 Detailed Description

The [BAMWriter](#) class writes [Fragment](#) objects back to file (in BAM format) with per-mapping probabilistic assignments.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 179 of file mapparser.h.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 [BAMWriter::BAMWriter](#) ( [BamTools::BamWriter](#) \* *writer*, bool *sample* )

[BAMWriter](#) constructor stores a pointer to the BAM file

#### Parameters

<i>writer</i>	pointer to the BAM file object
<i>sample</i>	specifies if a single alignment should be sampled (true) or all output with their respective probabilities (false)

Definition at line 320 of file mapparser.cpp.

#### 3.2.2.2 [BAMWriter::~~BAMWriter](#) ( )

[BAMWriter](#) destructor flushes and deletes the Bamtools::BamWriter

Definition at line 324 of file mapparser.cpp.

### 3.2.3 Member Function Documentation

#### 3.2.3.1 void [BAMWriter::write\\_fragment](#) ( [Fragment](#) & *f* ) [virtual]

a member function that writes all mappings of the fragment to the output file in BAM format along with their probabilities in the "XP" field



**Parameters**

<i>f</i>	the processed <a href="#">Fragment</a> to output
----------	--

Implements [Writer](#).

Definition at line 330 of file mapparser.cpp.

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

- src/mapparser.h
- src/mapparser.cpp

**3.3 BiasBoss Class Reference**

```
#include <biascorrection.h>
```

**Public Member Functions**

- [BiasBoss](#) (double alpha)
- void **copy\_observations** (const [BiasBoss](#) &other)
- void **copy\_expectations** (const [BiasBoss](#) &other)
- void **update\_expectations** (const [Transcript](#) &trans, double mass=0)
- void **normalize\_expectations** ()
- void **update\_observed** (const [FragHit](#) &hit, double mass)
- double **get\_transcript\_bias** (std::vector< float > &start\_bias, std::vector< float > &end\_bias, const [Transcript](#) &trans) const
- std::string **to\_string** () const
- void **append\_output** (std::ofstream &outfile) const

**3.3.1 Detailed Description**

The [BiasBoss](#) class keeps track of sequence-specific and positional bias. It allows for the bias associated with a given fragment end to be calculated, and for the bias parameters to be updated based on additional observations. All stored and returned values are in log space.

**Author**

Adam Roberts

**Date**

2011 Artistic License 2.0

Definition at line 224 of file biascorrection.h.

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 BiasBoss::BiasBoss ( double *alpha* )

a private [SeqWeightTable](#) that stores the 5' sequence-specific bias parameters (logged)  
 a private [SeqWeightTable](#) that stores the 3' sequence-specific bias parameters (logged)  
[BiasBoss](#) Constructor

##### Parameters

<i>alpha</i>	a double specifying the strength of the uniform prior (logged pseudo-counts for each parameter)
--------------	---

Definition at line 241 of file biascorrection.cpp.

### 3.3.3 Member Function Documentation

#### 3.3.3.1 void BiasBoss::append\_output ( std::ofstream & *outfile* ) const

a member function that outputs the positional and sequence-specific bias parameter matrices

##### Parameters

<i>outfile</i>	the file to append to
----------------	-----------------------

#### 3.3.3.2 double BiasBoss::get\_transcript\_bias ( std::vector< float > & *start\_bias*, std::vector< float > & *end\_bias*, const Transcript & *trans* ) const

a member function that returns the 5' and 3' bias values at each position in a given transcript based on the current bias parameters

##### Parameters

<i>start_bias</i>	a vector containing the logged bias for each 5' start site in the transcript
<i>end_bias</i>	a vector containing the logged bias for each 3' end site in the transcript
<i>trans</i>	the transcript for which to calculate the logged bias

##### Returns

the product of the average 5' and 3' bias (logged)

Definition at line 312 of file biascorrection.cpp.

#### 3.3.3.3 void BiasBoss::normalize\_expectations ( )

a member function that normalizes the expected counts and converts them to the log scale

Definition at line 284 of file biascorrection.cpp.

**3.3.3.4 string BiasBoss::to\_string ( ) const**

a member function that returns a string containing the observed positional nucleotide probabilities (non-logged) in column-major order (A,C,G,T)

**Returns**

the string representation of the observed probabilities

Definition at line 344 of file biascorrection.cpp.

**3.3.3.5 void BiasBoss::update\_expectations ( const Transcript & trans, double mass = 0 )**

a member function that updates the expectation parameters (sequence-specific and positional) assuming uniform expression of and accross the transcript's sequence

**Parameters**

<i>trans</i>	the transcript to measure expected counts from
--------------	--

Definition at line 260 of file biascorrection.cpp.

**3.3.3.6 void BiasBoss::update\_observed ( const FragHit & hit, double mass )**

a member function that updates the observed parameters (sequence-specific and positional) given a fragment mapping to a transcript and its logged probabilistic assignment

**Parameters**

<i>hit</i>	the fragment hit (alignment)
<i>mass</i>	the logged probability of the mapping, which is the amount to update the observed counts by

Definition at line 292 of file biascorrection.cpp.

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

- src/biascorrection.h
- src/biascorrection.cpp

**3.4 Bundle Class Reference**

```
#include <bundles.h>
```

**Public Member Functions**

- [Bundle](#) (Transcript \*trans)

- void [incr\\_counts](#) (size\_t incr\_amt=1)
- size\_t [size](#) () const
- std::vector< [Transcript](#) \* > & [transcripts](#) ()
- size\_t [counts](#) () const

### 3.4.1 Detailed Description

The [Bundle](#) class keeps track of a group of transcripts that have shared ambiguous (multi-mapped) reads. Besides storing the transcript, it keeps track of the number of observed fragments, the total fragment mass, and the next fragment mass (which it also updates).

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 71 of file bundles.h.

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 Bundle::Bundle ( [Transcript](#) \* *trans* )

[Bundle](#) Constructor.

#### Parameters

<i>trans</i>	a pointer to the initial <a href="#">Transcript</a> object in the bundle
<i>fmt</i>	a pointer to the (global) FragMassTable

Definition at line 42 of file bundles.cpp.

### 3.4.3 Member Function Documentation

#### 3.4.3.1 size\_t Bundle::counts ( ) const [inline]

a member function that returns the total number of observed fragments mapped to transcripts in the bundle

#### Returns

the total number of fragments mapped to transcripts in the bundle

Definition at line 115 of file bundles.h.

**3.4.3.2 void Bundle::incr\_counts ( size\_t *incr\_amt* = 1 )**

a member function that increases the total bundle observed fragment counts by a given amount

**Parameters**

<i>incr_amt</i>	the amount to increase the counts by
-----------------	--------------------------------------

Definition at line 46 of file bundles.cpp.

**3.4.3.3 size\_t Bundle::size ( ) const [inline]**

a member function that returns the number of transcripts in the bundle

**Returns**

the number of transcripts in the bundle

Definition at line 103 of file bundles.h.

**3.4.3.4 std::vector<Transcript\*>& Bundle::transcripts ( ) [inline]**

a member function that returns a reference to the vector of pointers to transcripts in the bundle

**Returns**

reference to the vector pointing to bundle transcripts

Definition at line 109 of file bundles.h.

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

- src/bundles.h
- src/bundles.cpp

**3.5 BundleTable Class Reference**

```
#include <bundles.h>
```

**Public Member Functions**

- [BundleTable \(\)](#)
- [~BundleTable \(\)](#)
- const BundleSet & [bundles](#) () const
- size\_t [size](#) () const
- [Bundle \\* create\\_bundle](#) (Transcript \*trans)
- [Bundle \\* merge](#) (Bundle \*b1, Bundle \*b2)

### 3.5.1 Detailed Description

The [BundleTable](#) class keeps track of the [Bundle](#) objects for a given run. It has the ability to create, delete, and merge bundles. It also keeps track of the transcript covariances, since these are related to bundles in that all covariances outside of a bundle are nonzero.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 129 of file bundles.h.

### 3.5.2 Constructor & Destructor Documentation

#### 3.5.2.1 [BundleTable::BundleTable](#) ( ) [inline]

[BundleTable](#) constructor.

Definition at line 141 of file bundles.h.

#### 3.5.2.2 [BundleTable::~~BundleTable](#) ( )

[BundleTable](#) destructor. Deletes all [Bundle](#) objects.

Definition at line 51 of file bundles.cpp.

### 3.5.3 Member Function Documentation

#### 3.5.3.1 `const BundleSet& BundleTable::bundles ( ) const` [inline]

a member function that returns the set of current [Bundle](#) objects

#### Returns

a reference to the unordered\_set containing all current [Bundle](#) objects

Definition at line 152 of file bundles.h.

#### 3.5.3.2 `Bundle * BundleTable::create_bundle ( Transcript * trans )`

a member function that creates a new bundle, initially with only the single given [Transcript](#)

#### Parameters

<i>trans</i>	a pointer to the only <a href="#">Transcript</a> initially contained in the <a href="#">Bundle</a>
--------------	--

**Returns**

a pointer to the new [Bundle](#) object

Definition at line 59 of file bundles.cpp.

**3.5.3.3 Bundle \* BundleTable::merge ( Bundle \* b1, Bundle \* b2 )**

a member function that merges two [Bundle](#) objects into one the Transcripts are all move to the larger bundles and the other is deleted

**Parameters**

<i>b1</i>	a pointer to one of the <a href="#">Bundle</a> objects to merge
<i>b2</i>	a pointer to the other <a href="#">Bundle</a> object to merge

**Returns**

a pointer to the merged [Bundle](#) object

Definition at line 66 of file bundles.cpp.

**3.5.3.4 size\_t BundleTable::size ( ) const [inline]**

a member function that returns the size of the set of current [Bundle](#) objects, which is the current number of bundles

**Returns**

the current number of bundles

Definition at line 159 of file bundles.h.

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

- src/bundles.h
- src/bundles.cpp

**3.6 CovarTable Class Reference**

```
#include <bundles.h>
```

**Public Member Functions**

- void [increment](#) (TransID trans1, TransID trans2, double covar)

- double [get](#) (TransID trans1, TransID trans2)
- size\_t [size](#) () const

### 3.6.1 Detailed Description

The [CovarTable](#) is a sparse matrix for storing and updating pairwise covariances between targets.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 26 of file bundles.h.

### 3.6.2 Member Function Documentation

#### 3.6.2.1 double CovarTable::get ( TransID *trans1*, TransID *trans2* )

a member function that returns the covariance between two transcripts these returned value will be the the negative of the true value (logged)

##### Parameters

<i>trans1</i>	one of the transcripts in the pair
<i>trans2</i>	the other transcript in the pair

##### Returns

the negative of the pair's covariance (logged)

Definition at line 29 of file bundles.cpp.

#### 3.6.2.2 void CovarTable::increment ( TransID *trans1*, TransID *trans2*, double *covar* )

a member function that increases the covariance between two transcripts by the specified amount (logged) these values are stored positive even though they are negative

##### Parameters

<i>trans1</i>	one of the transcripts in the pair
<i>trans2</i>	the other transcript in the pair
<i>covar</i>	a double specifying the amount to increase the pair's covariance by (logged)

Definition at line 15 of file bundles.cpp.



**3.6.2.3** `size_t CovarTable::size ( ) const` `[inline]`

a member function that returns the number of pairs of transcripts with non-zero covariance

**Returns**

the number of transcript pairs with non-zero covariance

Definition at line 60 of file bundles.h.

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

- src/bundles.h
- src/bundles.cpp

**3.7 FLD Class Reference**

```
#include <fld.h>
```

**Public Member Functions**

- [FLD](#) (double *alpha*, size\_t *max\_val*, size\_t *mean*, size\_t *std\_dev*)
- size\_t [max\\_val](#) () const
- size\_t [min\\_val](#) () const
- double [mean](#) () const
- void [add\\_val](#) (size\_t *len*, double *mass*)
- double [pdf](#) (size\_t *len*) const
- double [tot\\_mass](#) () const
- std::string [to\\_string](#) () const
- void [append\\_output](#) (std::ofstream &*outfile*) const

**3.7.1 Detailed Description**

The [FLD](#) class keeps track of the observed fragment length distribution. It starts with a Gaussian prior with parameters specified by the arguments. A small "Gaussian" kernel is added for each observation. All mass values and probabilities are stored and returned in log space (except in [to\\_string](#)).

Definition at line 21 of file fld.h.

**3.7.2 Constructor & Destructor Documentation****3.7.2.1** `FLD::FLD ( double alpha, size_t max_val, size_t mean, size_t std_dev )`

[FLD](#) Constructor

**Parameters**

<i>alpha</i>	double that sets the average pseudo-counts (logged)
<i>max_val</i>	an integer that sets the maximum allowable <a href="#">FragHit</a> length
<i>mean</i>	a <code>size_t</code> for the mean of the prior gaussian dist
<i>std_dev</i>	a <code>size_t</code> for the std dev of the prior gaussian dist

Definition at line 21 of file fld.cpp.

**3.7.3 Member Function Documentation****3.7.3.1 void FLD::add\_val ( `size_t` *len*, double *mass* )**

a member function that updates the distribution based on a new [FragHit](#) observation

**Parameters**

<i>len</i>	an integer for the observed <a href="#">FragHit</a> length
<i>mass</i>	a double for the mass (logged) of the observed <a href="#">FragHit</a>

Definition at line 53 of file fld.cpp.

**3.7.3.2 void FLD::append\_output ( `std::ofstream` & *outfile* ) const**

a member function that appends the [FLD](#) parameters to the end of the given file

**Parameters**

<i>outfile</i>	the file to append to
----------------	-----------------------

**3.7.3.3 `size_t` FLD::max\_val ( ) const**

a member function that returns the maximum allowed [FragHit](#) length

**Returns**

max allowed [FragHit](#) length

Definition at line 41 of file fld.cpp.

**3.7.3.4 double FLD::mean ( ) const**

a member function that returns the mean [FragHit](#) length

**Returns**

mean observed [FragHit](#) length

Definition at line 84 of file fld.cpp.

**3.7.3.5 double FLD::pdf ( size\_t len ) const**

a member function that returns the (logged) probability of a given [FragHit](#) length

**Parameters**

<i>len</i>	an integer for the <a href="#">FragHit</a> length to return the probability of
------------	--

**Returns**

(logged) probability of observing the given [FragHit](#) length

Definition at line 73 of file fld.cpp.

**3.7.3.6 string FLD::to\_string ( ) const**

a member function that returns a string containing the current distribution

**Returns**

space-separated string of probabilities ordered from length 0 to max\_val (non-logged)

Definition at line 89 of file fld.cpp.

**3.7.3.7 double FLD::tot\_mass ( ) const**

a member function that returns the (logged) number of observed FragHits (including pseudo-counts)

**Returns**

number of observed fragments

Definition at line 79 of file fld.cpp.

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

- src/fld.h
- src/fld.cpp

**3.8 FragHit Struct Reference**

```
#include <fragments.h>
```

**Public Member Functions**

- size\_t [length](#) () const
- PairStatus [pair\\_status](#) () const

## Public Attributes

- std::string [name](#)
- TransID [trans\\_id](#)
- Transcript \* [mapped\\_trans](#)
- Sequence [seq\\_l](#)
- Sequence [seq\\_r](#)
- size\_t [left](#)
- size\_t [right](#)
- int [mate\\_l](#)
- bool [left\\_first](#)
- std::vector< [Indel](#) > [inserts\\_l](#)
- std::vector< [Indel](#) > [deletes\\_l](#)
- std::vector< [Indel](#) > [inserts\\_r](#)
- std::vector< [Indel](#) > [deletes\\_r](#)
- double [probability](#)
- BamTools::BamAlignment [bam\\_l](#)
- BamTools::BamAlignment [bam\\_r](#)
- std::string [sam\\_l](#)
- std::string [sam\\_r](#)

### 3.8.1 Detailed Description

The [FragHit](#) struct stores the information for a single (multi-)mapping of a fragment.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 62 of file fragments.h.

### 3.8.2 Member Function Documentation

#### 3.8.2.1 `size_t FragHit::length ( ) const` `[inline]`

a member function returning the length of the fragment according to this mapping note, that this result will be invalid if the fragment is single-end

#### Returns

int length of fragment mapping

Definition at line 140 of file fragments.h.

### 3.8.2.2 PairStatus FragHit::pair\_status ( ) const [inline]

a member function returning whether the mapping is PAIRED, LEFT\_ONLY, or RIGHT\_ONLY LEFT\_ONLY denotes that the single read is not reverse complemented => its left end is the left fragment end RIGHT\_ONLY denotes that the single read is reverse complemented => its right end is the right fragment end

#### Returns

PairStatus the pair status of the mapping

Definition at line 152 of file fragments.h.

## 3.8.3 Member Data Documentation

### 3.8.3.1 std::vector<Indel> FragHit::deletes\_l

a public vector of [Indel](#) objects storing all deletions from the reference in the left read (in order from left to right)

Definition at line 123 of file fragments.h.

### 3.8.3.2 std::vector<Indel> FragHit::deletes\_r

a public vector of [Indel](#) objects storing all deletions from the reference in the left read (in order from right to left)

Definition at line 133 of file fragments.h.

### 3.8.3.3 std::vector<Indel> FragHit::inserts\_l

a public vector of [Indel](#) objects storing all insertions to the reference in the left read (in order from left to right)

Definition at line 118 of file fragments.h.

### 3.8.3.4 std::vector<Indel> FragHit::inserts\_r

a public vector of [Indel](#) objects storing all insertions to the reference in the right read (in order from right to left)

Definition at line 128 of file fragments.h.

### 3.8.3.5 size\_t FragHit::left

a public size\_t containing the 0-based leftmost coordinate mapped to in the transcript valid only if PairStatus is PAIRED or LEFT\_ONLY

Definition at line 93 of file fragments.h.

### 3.8.3.6 `bool FragHit::left_first`

a public bool specifying that the "right" (second according to SAM flag) is reverse complemented when true and the "left" (first according to SAM flag) is reverse complemented when false in other words, the "left" read is truly left of the "right" read in transcript coordinate space when true

Definition at line 113 of file fragments.h.

### 3.8.3.7 `Transcript* FragHit::mapped_trans`

a public pointer to the transcript mapped to

Definition at line 77 of file fragments.h.

### 3.8.3.8 `int FragHit::mate_l`

a public int containing the left position for the mate of the first read read in from the SAM file 0 if single-end fragment this is temporarily used to help find the mate, but is not important later on

Definition at line 106 of file fragments.h.

### 3.8.3.9 `std::string FragHit::name`

a public string for the SAM "Query Template Name" (fragment name)

Definition at line 67 of file fragments.h.

### 3.8.3.10 `size_t FragHit::right`

a public size\_t containing the position following the 0-based rightmost coordinate mapped to in the transcript valid only if PairStatus is PAIRED or RIGHT\_ONLY

Definition at line 99 of file fragments.h.

### 3.8.3.11 `Sequence FragHit::seq_l`

a public string containing the "left" read sequence (first according to SAM flag)

Definition at line 82 of file fragments.h.

### 3.8.3.12 `Sequence FragHit::seq_r`

a public string containing the "right" read sequence (second according to SAM flag)

Definition at line 87 of file fragments.h.

### 3.8.3.13 TransID FragHit::trans\_id

a public TransID for the transcript mapped to

Definition at line 72 of file fragments.h.

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

- src/fragments.h

## 3.9 Fragment Class Reference

```
#include <fragments.h>
```

### Public Member Functions

- [~Fragment](#) ()
- bool [add\\_map\\_end](#) (FragHit \*f)
- const std::string & [name](#) () const
- const size\_t [num\\_hits](#) () const
- const std::vector< [FragHit](#) \* > & [hits](#) () const
- const [FragHit](#) \* [sample\\_hit](#) () const

### 3.9.1 Detailed Description

The [Fragment](#) class stores information for all multi-mappings of a single fragment. By design, only paired-end mappings of paired-end reads will be accepted. All mappings of single-end reads will be accepted.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 177 of file fragments.h.

### 3.9.2 Constructor & Destructor Documentation

#### 3.9.2.1 Fragment::~~Fragment ( )

[Fragment](#) destructor deletes all [FragHit](#) objects pointed to by the [Fragment](#)

Definition at line 17 of file fragments.cpp.

### 3.9.3 Member Function Documentation

#### 3.9.3.1 `bool Fragment::add_map_end ( FragHit * f )`

a member function that adds a new [FragHit](#) (single read at this point) to the [Fragment](#) if it is the first [FragHit](#), it sets the [Fragment](#) name and is added to `_open_mates`, if the fragment is not paired, it is added to `_frag_hits`, otherwise, `add_open_mate` is called

##### Parameters

<i>f</i>	the <a href="#">FragHit</a> to be added
----------	---

Definition at line 30 of file `fragments.cpp`.

#### 3.9.3.2 `const std::vector<FragHit*>& Fragment::hits ( ) const` `[inline]`

a member function that returns [FragHit](#) multi-mappings of the fragment

##### Returns

a vector containing pointers to the [FragHit](#) multi-mappings

Definition at line 234 of file `fragments.h`.

#### 3.9.3.3 `const std::string& Fragment::name ( ) const` `[inline]`

a member function that returns the SAM "Query Template Name" (fragment name)

##### Returns

the string SAM "Query Template Name" (fragment name)

Definition at line 222 of file `fragments.h`.

#### 3.9.3.4 `const size_t Fragment::num_hits ( ) const` `[inline]`

a member function that returns the number of multi-mappings for the fragment

##### Returns

number of multi-mappings for fragment

Definition at line 228 of file `fragments.h`.

#### 3.9.3.5 `const FragHit * Fragment::sample_hit ( ) const`

a member function that returns a single [FragHit](#) of the fragment sampled at random based on the probabilistic assignment



**Returns**

a randomly sampled [FragHit](#)

Definition at line 98 of file fragments.cpp.

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

- src/fragments.h
- src/fragments.cpp

## 3.10 FrequencyMatrix Class Reference

```
#include <frequencymatrix.h>
```

**Public Member Functions**

- [FrequencyMatrix](#) ()
- [FrequencyMatrix](#) (size\_t m, size\_t n, double alpha, bool logged=true)
- double [operator\(\)](#) (size\_t i, size\_t j) const
- double [operator\(\)](#) (size\_t k) const
- void [increment](#) (size\_t i, size\_t j, double incr\_amt)
- void [increment](#) (size\_t k, double incr\_amt)
- double [arr](#) (size\_t k) const
- double [row](#) (size\_t i) const
- void [set\\_logged](#) (bool logged)

### 3.10.1 Detailed Description

The [FrequencyMatrix](#) class keeps track of the frequency parameters in order to allow for constant-time probability look-ups and updates. The table is rectangular to allow for multiple distributions to be stored in one [FrequencyMatrix](#). Rows are distributions. Values are in log space by default.

**Author**

Adam Roberts

**Date**

2011 Artistic License 2.0

Definition at line 24 of file frequencymatrix.h.

### 3.10.2 Constructor & Destructor Documentation

#### 3.10.2.1 `FrequencyMatrix::FrequencyMatrix ( )` `[inline]`

dummy constructor

Definition at line 56 of file `frequencymatrix.h`.

#### 3.10.2.2 `FrequencyMatrix::FrequencyMatrix ( size_t m, size_t n, double alpha, bool logged = true )`

[FrequencyMatrix](#) constructor initializes the matrix based on the log of the given pseudo-counts

##### Parameters

<i>m</i>	a size_t specifying the number of distributions (rows)
<i>n</i>	a size_t specifying the number of values in each distribution (columns)
<i>alpha</i>	a double specifying the initial pseudo-counts (un-logged)
<i>logged</i>	bool that specifies if the table is in log space

Definition at line 16 of file `frequencymatrix.cpp`.

### 3.10.3 Member Function Documentation

#### 3.10.3.1 `double FrequencyMatrix::arr ( size_t k ) const` `[inline]`

a member function that returns the raw value stored at a given position of the flattened matrix

##### Parameters

<i>k</i>	the array position
----------	--------------------

##### Returns

a double specifying the raw value stored at the given position of the flattened matrix

Definition at line 102 of file `frequencymatrix.h`.

#### 3.10.3.2 `void FrequencyMatrix::increment ( size_t i, size_t j, double incr_amt )`

a member function to increase the mass of a given position in the matrix

##### Parameters

<i>i</i>	the distribution (row)
<i>j</i>	the value (column)
<i>incr_amt</i>	the amount to increase the mass by (logged if table is logged)

Definition at line 39 of file frequencymatrix.cpp.

#### 3.10.3.3 void FrequencyMatrix::increment ( size\_t *k*, double *incr\_amt* )

a member function to increase the mass of a given position in the flattened matrix (logged if table is logged)

##### Parameters

<i>k</i>	the array position
<i>incr_amt</i>	the amount to increase the mass by (logged if table is logged)

Definition at line 56 of file frequencymatrix.cpp.

#### 3.10.3.4 double FrequencyMatrix::operator() ( size\_t *k* ) const

a member function to extract the probability of a given position in the flattened matrix (logged if table is logged)

##### Parameters

<i>k</i>	the array position
----------	--------------------

##### Returns

a double specifying the probability of the given position in the flattened matrix (logged if table is logged)

Definition at line 33 of file frequencymatrix.cpp.

#### 3.10.3.5 double FrequencyMatrix::operator() ( size\_t *i*, size\_t *j* ) const

a member function to extract the probability of a given position in the matrix (logged if table is logged)

##### Parameters

<i>i</i>	the distribution (row)
<i>j</i>	the value (column)

##### Returns

a double specifying the probability of the given value in the given distribution (logged if table is logged)

Definition at line 24 of file frequencymatrix.cpp.

### 3.10.3.6 double FrequencyMatrix::row ( size\_t i ) const [inline]

a member function that returns the raw row sum

#### Parameters

<i>i</i>	the distribution (row)
----------	------------------------

#### Returns

a double specifying the raw row sum for the given distribution

Definition at line 109 of file frequencymatrix.h.

### 3.10.3.7 void FrequencyMatrix::set\_logged ( bool logged )

a member function that converts the table between log-space and non-log space

#### Parameters

<i>logged</i>	bool specifying if the table should be converted to logged or non-logged space
---------------	--

Definition at line 61 of file frequencymatrix.cpp.

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

- src/frequencymatrix.h
- src/frequencymatrix.cpp

## 3.11 Globals Struct Reference

```
#include <main.h>
```

### Public Attributes

- [FLD](#) \* fld
- [MismatchTable](#) \* mismatch\_table
- [BiasBoss](#) \* bias\_table
- [TranscriptTable](#) \* trans\_table

#### 3.11.1 Detailed Description

a struct for holding pointers to the global parameter tables (bias\_table, mismatch\_table, fld)

Definition at line 29 of file main.h.

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

- `src/main.h`

## 3.12 Indel Struct Reference

```
#include <fragments.h>
```

### Public Member Functions

- [Indel](#) (`size_t p`, `size_t l`)

### Public Attributes

- `size_t pos`
- `size_t len`

#### 3.12.1 Detailed Description

The [Indel](#) struct stores the information for a single insertion or deletion.

#### Author

Adam Roberts

#### Date

2012 Artistic License 2.0

Definition at line 38 of file `fragments.h`.

#### 3.12.2 Constructor & Destructor Documentation

**3.12.2.1** `Indel::Indel ( size_t p, size_t l )` [`inline`]

[Indel](#) constructor

Definition at line 53 of file `fragments.h`.

#### 3.12.3 Member Data Documentation

**3.12.3.1** `size_t Indel::len`

a public `size_t` for the length of the [Indel](#) in the read

Definition at line 48 of file `fragments.h`.

### 3.12.3.2 `size_t Indel::pos`

a public `size_t` for the position of the [Indel](#) in the read

Definition at line 43 of file `fragments.h`.

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

- `src/fragments.h`

## 3.13 MarkovModel Class Reference

### Public Member Functions

- **MarkovModel** (`size_t` order, `size_t` window\_size, `size_t` num\_pos, double alpha, bool logged=true)
- double **transition\_prob** (`size_t` p, `size_t` cond, `size_t` curr) const
- double **seq\_prob** (const [Sequence](#) &seq, int left) const
- double **marginal\_prob** (`size_t` w, `size_t` nuc) const
- void **update** (const [Sequence](#) &seq, int left, double mass)
- void **fast\_learn** (const [Sequence](#) &seq, double mass)
- void **calc\_marginals** ()
- void **set\_logged** (bool logged)

### 3.13.1 Detailed Description

Definition at line 18 of file `markovmodel.h`.

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

- `src/markovmodel.h`
- `src/markovmodel.cpp`

## 3.14 MismatchTable Class Reference

```
#include <mismatchmodel.h>
```

### Public Member Functions

- [MismatchTable](#) (double alpha)
- void **activate** (bool active=true)
- double **log\_likelihood** (const [FragHit](#) &f) const
- void **update** (const [FragHit](#) &, double mass)
- std::string **to\_string** () const
- void **append\_output** (std::ofstream &outfile) const

### 3.14.1 Detailed Description

The [MismatchTable](#) class is used to store and update mismatch (error) parameters using a first-order Markov model based on nucleotide and position in a ride and to return likelihoods of mismatches in given reads. All values are stored and returned in log space.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 24 of file mismatchmodel.h.

### 3.14.2 Constructor & Destructor Documentation

#### 3.14.2.1 MismatchTable::MismatchTable ( double *alpha* )

[MismatchTable](#) constructor initializes the model parameters using the specified (non-logged) pseudo-counts.

#### Parameters

<i>alpha</i>	a double containing the non-logged pseudo-counts for parameter initialization
--------------	---

Definition at line 19 of file mismatchmodel.cpp.

### 3.14.3 Member Function Documentation

#### 3.14.3.1 void MismatchTable::activate ( bool *active* = true ) [inline]

member function that 'activates' the table to allow its values to be used in calculating log-likelihoods when it is sufficiently burned-in

#### Parameters

<i>active</i>	a boolean specifying whether to activate (true) or deactivate (false)
---------------	---

Definition at line 69 of file mismatchmodel.h.

#### 3.14.3.2 void MismatchTable::append\_output ( std::ofstream & *outfile* ) const

a member function that outputs the final model parameters in a tab-separated file the file has 1 row for each read position and the parameters are in columns indexed as (ref, prev, obs) in base 4 with A,C,G,T encoded as 0,1,2,3.

**Parameters**

<i>file</i>	stream to append to
-------------	---------------------

Definition at line 70 of file biascorrection.cpp.

**3.14.3.3 double MismatchTable::log\_likelihood ( const FragHit & f ) const**

member function returns the log likelihood of mismatches in the mapping given the current error model parameters

**Parameters**

<i>f</i>	the fragment mapping to calculate the log likelihood for
----------	--

**Returns**

the log likelihood of the mapping based on mismatches

Definition at line 28 of file mismatchmodel.cpp.

**3.14.3.4 string MismatchTable::to\_string ( ) const**

member function that returns a string containing a collapsed confusion matrix based on the model parameters for the first read

**Returns**

a space-separated string for the flattened, collapsed confusion matrix in row-major format (observed value as rows)

Definition at line 251 of file mismatchmodel.cpp.

**3.14.3.5 void MismatchTable::update ( const FragHit & f, double mass )**

member function that updates the error model parameters based on a mapping and its (logged) mass

**Parameters**

<i>f</i>	the fragment mapping
<i>mass</i>	the logged mass to increase the parameters by

Definition at line 144 of file mismatchmodel.cpp.

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

- src/mismatchmodel.h
- src/biascorrection.cpp
- src/fld.cpp

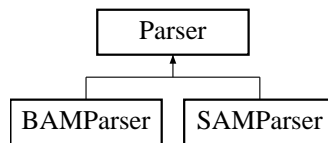


- src/mismatchmodel.cpp

## 3.15 Parser Class Reference

```
#include <mapparser.h>
```

Inheritance diagram for Parser:



### Public Member Functions

- virtual const std::string `header` () const =0
- virtual const TransIndex & `trans_index` () const =0
- virtual const TransIndex & `trans_lengths` () const =0
- virtual bool `next_fragment` (Fragment &f)=0
- virtual void `reset` ()=0

### 3.15.1 Detailed Description

The `Parser` class is an abstract class that can be a `SAMParser` or `BAMParser`. It fills `Fragment` objects by parsing an input file in SAM/BAM format.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 33 of file mapparser.h.

### 3.15.2 Member Function Documentation

#### 3.15.2.1 virtual const std::string Parser::header ( ) const [pure virtual]

a member function that returns a string version of the header

#### Returns

string version of the header

Implemented in `BAMParser`, and `SAMParser`.

**3.15.2.2** `virtual bool Parser::next_fragment ( Fragment & f )` `[pure virtual]`

a member function that loads all mappings of the next fragment

**Parameters**

<i>f</i>	the empty <a href="#">Fragment</a> to add mappings to
----------	---

**Returns**

true if more reads remain in the SAM/BAM file/stream, false otherwise

Implemented in [BAMParser](#), and [SAMParser](#).

**3.15.2.3** `virtual void Parser::reset ( )` `[pure virtual]`

a member function that resets the parser and rewinds to the beginning of the input

Implemented in [BAMParser](#).

**3.15.2.4** `virtual const TransIndex& Parser::trans_index ( ) const` `[pure virtual]`

a member function that returns the transcript-to-index map

**Returns**

the transcript-to-index map

Implemented in [BAMParser](#), and [SAMParser](#).

**3.15.2.5** `virtual const TransIndex& Parser::trans_lengths ( ) const` `[pure virtual]`

a member function that returns the transcript-to-length map

**Returns**

the transcript-to-length map

Implemented in [BAMParser](#), and [SAMParser](#).

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

- `src/mapparser.h`

**3.16 ParseThreadSafety Struct Reference**

```
#include <threadsafety.h>
```

## Public Attributes

- [Fragment](#) \* [next\\_frag](#)
- boost::mutex [mut](#)
- boost::condition\_variable [cond](#)
- bool [frag\\_clean](#)

### 3.16.1 Detailed Description

The [ParseThreadSafety](#) struct stores objects to allow for parsing to safely occur on a separate thread from processing.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 23 of file threadsafety.h.

### 3.16.2 Member Data Documentation

#### 3.16.2.1 boost::condition\_variable ParseThreadSafety::cond

a conditional variable where the processor waits for a new [Fragment](#) and the parser waits for the [Fragment](#) pointer to be copied by the processor

Definition at line 39 of file threadsafety.h.

#### 3.16.2.2 bool ParseThreadSafety::frag\_clean

a bool specifying the condition that the current next\_frag pointer is clean, meaning that it hasn't been copied by the processor

Definition at line 45 of file threadsafety.h.

#### 3.16.2.3 boost::mutex ParseThreadSafety::mut

a mutex for the conditional variable

Definition at line 33 of file threadsafety.h.

#### 3.16.2.4 Fragment\* ParseThreadSafety::next\_frag

a pointer to the next [Fragment](#) to be processed by the main thread

Definition at line 28 of file threadsafety.h.

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

- `src/threadsafety.h`

## 3.17 PosWeightTable Class Reference

```
#include <biascorrection.h>
```

### Public Member Functions

- [PosWeightTable](#) (const std::vector< size\_t > &len\_bins, const std::vector< double > &pos\_bins, double alpha)
- const std::vector< size\_t > & **len\_bins** () const
- const std::vector< double > & **pos\_bins** () const
- void [increment\\_expected](#) (size\_t len, double pos)
- void [increment\\_expected](#) (size\_t l, size\_t p)
- void [normalize\\_expected](#) ()
- void [increment\\_observed](#) (size\_t len, double pos, double normalized\_mass)
- void [increment\\_observed](#) (size\_t l, size\_t p, double normalized\_mass)
- double [get\\_weight](#) (size\_t len, double pos) const
- double [get\\_weight](#) (size\_t l, size\_t p) const
- void [append\\_output](#) (std::ofstream &outfile) const

### 3.17.1 Detailed Description

The [PosWeightTable](#) class keeps track of fractional position bias parameters in log space. It allows for the bias associated with a given fractional position to be calculated, and for the bias parameters to be updated based on additional fragment observations.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 118 of file `biascorrection.h`.

### 3.17.2 Constructor & Destructor Documentation

- 3.17.2.1 [PosWeightTable::PosWeightTable](#) ( const std::vector< size\_t > &len\_bins, const std::vector< double > &pos\_bins, double alpha )

[PosWeightTable](#) Constructor

#### Parameters

<i>len_bins</i>	a vector of unsigned integers specifying the bin ranges for transcript lengths
<i>pos_bins</i>	a vector of doubles specifying the bin ranges for fractional positions
<i>alpha</i>	a double specifying the strength of the uniform prior (logged pseudo-counts for each parameter)

Definition at line 146 of file biascorrection.cpp.

### 3.17.3 Member Function Documentation

#### 3.17.3.1 void PosWeightTable::append\_output ( std::ofstream & *outfile* ) const

a member function that outputs the fractional position probabilities in matrix format with length bins as rows and fractional position bins as columns

##### Parameters

<i>outfile</i>	the file to append to
----------------	-----------------------

#### 3.17.3.2 double PosWeightTable::get\_weight ( size\_t *len*, double *pos* ) const

a member function that return the bias weight (logged) of a fractional transcript position

##### Parameters

<i>len</i>	the transcript length
<i>pos</i>	the fractional transcript position

##### Returns

the logged bias weight for the fractional transcript position

Definition at line 182 of file biascorrection.cpp.

#### 3.17.3.3 double PosWeightTable::get\_weight ( size\_t *l*, size\_t *p* ) const

a member function that return the bias weight (logged) of a fractional transcript position bin

##### Parameters

<i>l</i>	the transcript length bin
<i>p</i>	the fractional transcript position bin

##### Returns

the logged bias weight for the fractional transcript position

Definition at line 189 of file biascorrection.cpp.

**3.17.3.4 void PosWeightTable::increment\_expected ( size\_t *l*, size\_t *p* )**

a member function that increments the expected counts for the given fractional position bin by 1 (logged)

**Parameters**

<i>l</i>	the transcript length bin
<i>p</i>	the fractional transcript position bin

Definition at line 160 of file biascorrection.cpp.

**3.17.3.5 void PosWeightTable::increment\_expected ( size\_t *len*, double *pos* )**

a member function that increments the expected counts for the given fractional position by 1 (logged)

**Parameters**

<i>len</i>	the transcript length
<i>pos</i>	the fractional transcript position

Definition at line 153 of file biascorrection.cpp.

**3.17.3.6 void PosWeightTable::increment\_observed ( size\_t *l*, size\_t *p*, double *normalized\_mass* )**

a member function that increments the observed counts for the given fragment position bin by some mass (logged)

**Parameters**

<i>l</i>	the transcript length bin
<i>p</i>	the fractional transcript position bin
<i>normalized_mass</i>	the mass (logged probabilistic assignment) of the fragment normalized by its estimated expression

Definition at line 177 of file biascorrection.cpp.

**3.17.3.7 void PosWeightTable::increment\_observed ( size\_t *len*, double *pos*, double *normalized\_mass* )**

a member function that increments the observed counts for the given fragment position by some mass (logged)

**Parameters**

<i>len</i>	the transcript length
------------	-----------------------

<i>pos</i>	the fractional transcript position
<i>normalized_- mass</i>	the mass (logged probabilistic assignment) of the fragment normalized by its estimated expression

Definition at line 170 of file biascorrection.cpp.

### 3.17.3.8 void PosWeightTable::normalize\_expected ( )

a member function that normalizes the expected counts and converts them to the log scale

Definition at line 165 of file biascorrection.cpp.

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

- src/biascorrection.h
- src/biascorrection.cpp

## 3.18 RoundParams Struct Reference

```
#include <transcripts.h>
```

### Public Member Functions

- [RoundParams](#) ()

### Public Attributes

- double [mass](#)
- double [ambig\\_mass](#)
- double [tot\\_ambig\\_mass](#)
- double [binom\\_var](#)
- double [samp\\_var](#)
- double [tot\\_unc](#)

### 3.18.1 Detailed Description

The [RoundParams](#) struct stores the transcript parameters unique to a given round (iteration) of EM

### Author

Adam Roberts

**Date**

2012 Artistic License 2.0

Definition at line 34 of file transcripts.h.

**3.18.2 Constructor & Destructor Documentation****3.18.2.1 RoundParams::RoundParams ( ) [inline]**

[RoundParams](#) constructor sets initial values for parameters

Definition at line 69 of file transcripts.h.

**3.18.3 Member Data Documentation****3.18.3.1 double RoundParams::ambig\_mass**

a private double that stores the (logged) assigned mass derived from ambiguous fragments

Definition at line 44 of file transcripts.h.

**3.18.3.2 double RoundParams::binom\_var**

a private double that stores the (logged) binomial variance of the mass

Definition at line 54 of file transcripts.h.

**3.18.3.3 double RoundParams::mass**

a private double that stores the (logged) assigned mass based on observed fragment mapping probabilities

Definition at line 39 of file transcripts.h.

**3.18.3.4 double RoundParams::samp\_var**

a private double that stores the (logged) sampling variance of the mass

Definition at line 59 of file transcripts.h.

**3.18.3.5 double RoundParams::tot\_ambig\_mass**

a private double that stores the (logged) total mass of ambiguous fragments mapping to the transcript

Definition at line 49 of file transcripts.h.



### 3.18.3.6 double RoundParams::tot\_unc

a private double that stores the (logged) variance due to uncertainty on p

Definition at line 64 of file transcripts.h.

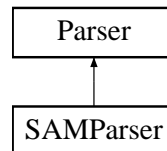
The documentation for this struct was generated from the following file:

- src/transcripts.h

## 3.19 SAMParser Class Reference

```
#include <mapparser.h>
```

Inheritance diagram for SAMParser:



### Public Member Functions

- [SAMParser](#) (std::istream \*in)
- const std::string [header](#) () const
- const TransIndex & [trans\\_index](#) () const
- const TransIndex & [trans\\_lengths](#) () const
- bool [next\\_fragment](#) ([Fragment](#) &f)

### 3.19.1 Detailed Description

The [SAMParser](#) class fills [Fragment](#) objects by parsing an input in SAM format. The input may come from a file or stdin.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 223 of file mapparser.h.

### 3.19.2 Constructor & Destructor Documentation

#### 3.19.2.1 SAMParser::SAMParser ( std::istream \* *in* )

[SAMParser](#) constructor removes the header, and parses the first line

Definition at line 350 of file mapparser.cpp.

### 3.19.3 Member Function Documentation

#### 3.19.3.1 const std::string SAMParser::header ( ) const [inline, virtual]

a member function that returns a string version of the header

##### Returns

string version of the header

Implements [Parser](#).

Definition at line 271 of file mapparser.h.

#### 3.19.3.2 bool SAMParser::next\_fragment ( Fragment & *f* ) [virtual]

a member function that loads all mappings of the next fragment when the next fragment is reached, the current alignment is left in the `_frag_buff` for the next call

##### Parameters

<i>f</i>	the empty <a href="#">Fragment</a> to add mappings to
----------	---

##### Returns

true if more reads remain in the SAM file, false otherwise

Implements [Parser](#).

Definition at line 406 of file mapparser.cpp.

#### 3.19.3.3 const TransIndex& SAMParser::trans\_index ( ) const [inline, virtual]

a member function that returns the transcript-to-index map

##### Returns

the transcript-to-index map

Implements [Parser](#).

Definition at line 277 of file mapparser.h.

### 3.19.3.4 `const TransIndex& SAMParser::trans_lengths ( ) const` `[inline, virtual]`

a member function that returns the transcript-to-length map

#### Returns

the transcript-to-length map

Implements [Parser](#).

Definition at line 283 of file `mapparser.h`.

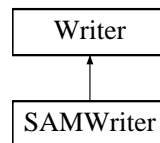
The documentation for this class was generated from the following files:

- `src/mapparser.h`
- `src/mapparser.cpp`

## 3.20 SAMWriter Class Reference

```
#include <mapparser.h>
```

Inheritance diagram for SAMWriter:



### Public Member Functions

- [SAMWriter](#) (`std::ostream *out`, `bool sample`)
- [~SAMWriter](#) ()
- void [write\\_fragment](#) ([Fragment](#) &`f`)

### 3.20.1 Detailed Description

The [SAMWriter](#) class writes [Fragment](#) objects back to file (in SAM format) with per-mapping probabilistic assignments.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 301 of file `mapparser.h`.

### 3.20.2 Constructor & Destructor Documentation

#### 3.20.2.1 SAMWriter::SAMWriter ( std::ostream \* *out*, bool *sample* )

[SAMWriter](#) constructor stores a pointer to the output stream

##### Parameters

<i>out</i>	SAM output stream
<i>sample</i>	specifies if a single alignment should be sampled (true) or all output with their respective probabilities (false)

Definition at line 531 of file mapparser.cpp.

#### 3.20.2.2 SAMWriter::~~SAMWriter ( )

[SAMWriter](#) destructor flushes and deletes output stream

Definition at line 535 of file mapparser.cpp.

### 3.20.3 Member Function Documentation

#### 3.20.3.1 void SAMWriter::write\_fragment ( **Fragment** & *f* ) [virtual]

a member function that writes all mappings of the fragment to the output file in SAM format along with their probabilities in the "XP" field

##### Parameters

<i>f</i>	the processed <a href="#">Fragment</a> to output
----------	--

Implements [Writer](#).

Definition at line 541 of file mapparser.cpp.

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

- src/mapparser.h
- src/mapparser.cpp

## 3.21 Sequence Class Reference

```
#include <sequence.h>
```

### Public Member Functions

- [Sequence](#) ()
- [Sequence](#) (const std::string &seq, bool rev)

- [Sequence](#) (const [Sequence](#) &other)
- [Sequence](#) & [operator=](#) (const [Sequence](#) &other)
- [~Sequence](#) ()
- void [set](#) (const std::string &seq, bool rev)
- bool [empty](#) () const
- size\_t [operator\[\]](#) (const size\_t index) const
- size\_t [length](#) () const

### 3.21.1 Detailed Description

The [Sequence](#) class is used to store and access encoded nucleotide sequences.

#### Author

Adam Roberts

#### Date

2012 Artistic License 2.0

Definition at line 20 of file sequence.h.

### 3.21.2 Constructor & Destructor Documentation

#### 3.21.2.1 [Sequence::Sequence](#) ( )

dummy constructor

Definition at line 58 of file sequence.cpp.

#### 3.21.2.2 [Sequence::Sequence](#) ( const std::string & seq, bool rev )

[Sequence](#) constructor encodes and stores the given nucleotide sequence

#### Parameters

<i>seq</i>	the nucleotide sequence to encode and store
<i>rev</i>	a boolean if the sequence should be reverse complemented before encoding

Definition at line 60 of file sequence.cpp.

#### 3.21.2.3 [Sequence::Sequence](#) ( const [Sequence](#) & other )

[Sequence](#) copy constructor

#### Parameters

<i>other</i>	the <a href="#">Sequence</a> object to copy
--------------	---

Definition at line 65 of file sequence.cpp.

#### 3.21.2.4 Sequence::~~Sequence ( )

[Sequence](#) destructor. Deletes the char array.

Definition at line 88 of file sequence.cpp.

### 3.21.3 Member Function Documentation

#### 3.21.3.1 bool Sequence::empty ( ) const [inline]

a member function that returns true iff the encoded sequence has zero length

##### Returns

true iff the encoded sequence has zero length

Definition at line 76 of file sequence.h.

#### 3.21.3.2 size\_t Sequence::length ( ) const [inline]

a member function that returns the length of the encoded sequence

##### Returns

the length of the encoded sequence

Definition at line 90 of file sequence.h.

#### 3.21.3.3 Sequence & Sequence::operator= ( const Sequence & *other* )

[Sequence](#) assignment constructor

##### Parameters

<i>other</i>	the <a href="#">Sequence</a> object to copy
--------------	---

Definition at line 75 of file sequence.cpp.

#### 3.21.3.4 size\_t Sequence::operator[] ( const size\_t *index* ) const

a member function that returns the encoded character at the given index

##### Parameters

<i>index</i>	the index of the encoded character to return (assumed to be < _len)
--------------	---

**Returns**

the encoded character at the given index

Definition at line 119 of file sequence.cpp.

**3.21.3.5 void Sequence::set ( const std::string & seq, bool rev )**

a member function that encodes the given sequence and overwrites the current stored sequence with it

**Parameters**

<i>seq</i>	the nucleotide sequence to encode and store
<i>rev</i>	a boolean if the sequence should be reverse complemented before encoding

Definition at line 95 of file sequence.cpp.

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

- src/sequence.h
- src/sequence.cpp

**3.22 SeqWeightTable Class Reference**

```
#include <biascorrection.h>
```

**Public Member Functions**

- [SeqWeightTable](#) (size\_t window\_size, double alpha)
- void [copy\\_observed](#) (const [SeqWeightTable](#) &other)
- void [copy\\_expected](#) (const [SeqWeightTable](#) &other)
- void [increment\\_expected](#) (const [Sequence](#) &seq, double mass)
- void [normalize\\_expected](#) ()
- void [increment\\_observed](#) (const [Sequence](#) &seq, size\_t i, double mass)
- double [get\\_weight](#) (const [Sequence](#) &seq, size\_t i) const
- std::string [to\\_string](#) () const
- void [append\\_output](#) (std::ofstream &outfile) const

**3.22.1 Detailed Description**

The [SeqWeightTable](#) class keeps track of sequence-specific bias parameters. It allows for the bias associated with a given sequence to be calculated, and for the bias parameters to be updated based on additional observations. All values stored in log space.

**Author**

Adam Roberts

**Date**

2011 Artistic License 2.0

Definition at line 30 of file biascorrection.h.

**3.22.2 Constructor & Destructor Documentation****3.22.2.1 SeqWeightTable::SeqWeightTable ( size\_t *window\_size*, double *alpha* )**

[SeqWeightTable](#) Constructor

**Parameters**

<i>window_size</i>	an unsigned integer specifying the size of the bias window surrounding fragment ends
<i>alpha</i>	a double specifying the strength of the uniform prior (logged pseudo-counts for each parameter)

Definition at line 31 of file biascorrection.cpp.

**3.22.3 Member Function Documentation****3.22.3.1 void SeqWeightTable::append\_output ( std::ofstream & *outfile* ) const**

a member function that outputs the positional nucleotide probabilities in matrix format with nucleotides (A,C,G,T) as rows and window position as columns

**Parameters**

<i>outfile</i>	the file to append to
----------------	-----------------------

**3.22.3.2 void SeqWeightTable::copy\_expected ( const SeqWeightTable & *other* )**

a member function that overwrites the "expected" parameters with those from another [SeqWeightTable](#)

**Parameters**

<i>other</i>	another <a href="#">SeqWeightTable</a> from which to copy the parameters
--------------	--

Definition at line 41 of file biascorrection.cpp.

**3.22.3.3 void SeqWeightTable::copy\_observed ( const SeqWeightTable & *other* )**

a member function that overwrites the "observed" parameters with those from another [SeqWeightTable](#)



**Parameters**

<i>other</i>	another <a href="#">SeqWeightTable</a> from which to copy the parameters
--------------	--

Definition at line 36 of file biascorrection.cpp.

**3.22.3.4 double SeqWeightTable::get\_weight ( const Sequence & seq, size\_t i ) const**

a member function that calculates the bias weight (logged) of a bias window

**Parameters**

<i>seq</i>	the transcript sequence the fragment hits to
<i>i</i>	the fragment end point (the central point of the bias window)

**Returns**

the bias weight for the bias window which is the product of the individual nucleotide bias weights

Definition at line 63 of file biascorrection.cpp.

**3.22.3.5 void SeqWeightTable::increment\_expected ( const Sequence & seq, double mass )**

a member function that increments the expected counts for a sliding window through the given transcript sequence by some mass

**Parameters**

<i>seq</i>	the transcript sequence
<i>mass</i>	the amount of used to weight the transcript's sequence in the parameter table

Definition at line 46 of file biascorrection.cpp.

**3.22.3.6 void SeqWeightTable::increment\_observed ( const Sequence & seq, size\_t i, double mass )**

a member function that increments the observed counts for the given fragment sequence by some mass (logged)

**Parameters**

<i>seq</i>	the transcript sequence (possibly reverse complemented) to which the fragment end maps
<i>i</i>	the index into the sequence at which to center the bias window (where the fragment starts/ends)
<i>mass</i>	the fragment's mass

Definition at line 57 of file biascorrection.cpp.

### 3.22.3.7 void SeqWeightTable::normalize\_expected ( )

a member function that normalizes the expected counts and converts them to the log scale

Definition at line 51 of file biascorrection.cpp.

### 3.22.3.8 std::string SeqWeightTable::to\_string ( ) const

a member function that returns a string containing the positional nucleotide probabilities in column-major order (A,C,G,T)

#### Returns

the string representation of the positional nucleotide probabilities

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

- src/biascorrection.h
- src/biascorrection.cpp

## 3.23 ThreadedMapParser Class Reference

```
#include <mapparser.h>
```

### Public Member Functions

- [ThreadedMapParser](#) (std::string input\_file, std::string output\_file, bool write\_active)
- [~ThreadedMapParser](#) ()
- void [threaded\\_parse](#) (ParseThreadSafety \*thread\_safety, [TranscriptTable](#) \*trans\_table)
- const TransIndex & [trans\\_index](#) ()
- const TransIndex & [trans\\_lengths](#) ()
- void [write\\_active](#) (bool b)
- void [reset\\_reader](#) ()

### 3.23.1 Detailed Description

The [ThreadedMapParser](#) class is meant to be run on as a separate thread from the main processing. Once started, this thread will read input from a file or stream in SAM/BAM format, parse, and collect read alignments into fragment alignments, and fragment alignments into fragments, which are placed on a buffer for the processing thread. Once the processing thread copies the fragment address from the buffer, the parser is unlocked to load the next fragment. The process stops when EOF is reached

**Author**

Adam Roberts

**Date**

2011 Artistic License 2.0

Definition at line 348 of file mapparser.h.

**3.23.2 Constructor & Destructor Documentation****3.23.2.1 ThreadedMapParser::ThreadedMapParser ( *std::string input\_file*, *std::string output\_file*, *bool write\_active* )**

[ThreadedMapParser](#) constructor determines what format the input is in and initializes the correct parser.

**Parameters**

<i>input_file</i>	string containing the path to the input SAM/BAM file
<i>output_file</i>	string containing the path the output file less its extension (empty if writing is to be disabled)

Definition at line 97 of file mapparser.cpp.

**3.23.2.2 ThreadedMapParser::~~ThreadedMapParser ( )**

[ThreadedMapParser](#) destructor deletes the parser and writer (if it exists).

Definition at line 161 of file mapparser.cpp.

**3.23.3 Member Function Documentation****3.23.3.1 void ThreadedMapParser::reset\_reader ( ) [inline]**

a member function that resets the input parser

Definition at line 406 of file mapparser.h.

**3.23.3.2 void ThreadedMapParser::threaded\_parse ( *ParseThreadSafety \* thread\_safety*, *TranscriptTable \* trans\_table* )**

a member function that drives the parse thread when all valid mappings of a fragment have been parsed, its mapped transcripts are found and the information is passed in a [Fragment](#) object to the processing thread through the [ParseThreadSafety](#) struct

**Parameters**

<i>thread_safety</i>	a pointer to the struct containing shared locks and data with the processing thread
<i>trans_table</i>	a pointer to the table of <a href="#">Transcript</a> objects to lookup the mapped transcripts

Definition at line 168 of file mapparser.cpp.

### 3.23.3.3 `const TransIndex& ThreadedMapParser::trans_index ( )` `[inline]`

a member function that returns the transcript-to-index map

#### Returns

the transcript-to-index map

Definition at line 388 of file mapparser.h.

### 3.23.3.4 `const TransIndex& ThreadedMapParser::trans_lengths ( )` `[inline]`

a member function that returns the transcript-to-length map

#### Returns

the transcript-to-length map

Definition at line 394 of file mapparser.h.

### 3.23.3.5 `void ThreadedMapParser::write_active ( bool b )` `[inline]`

a member function that sets the write-active status of the parser this specifies whether or not the alignments (sampled or with probs) could be output

#### Parameters

<i>b</i>	updated write-active status
----------	-----------------------------

Definition at line 401 of file mapparser.h.

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

- src/mapparser.h
- src/mapparser.cpp

## 3.24 Transcript Class Reference

```
#include <transcripts.h>
```

## Public Member Functions

- [Transcript](#) (const size\_t id, const std::string &name, const std::string &seq, double alpha, const [Globals](#) \*globs)
- const std::string & [name](#) () const
- TransID [id](#) () const
- const [Sequence](#) & [seq](#) (bool rev) const
- size\_t [length](#) () const
- double [rho](#) () const
- double [mass](#) (bool with\_pseudo=true) const
- double [ambig\\_mass](#) () const
- double [tot\\_ambig\\_mass](#) () const
- double [binom\\_var](#) () const
- double [samp\\_var](#) () const
- double [tot\\_uncertainty](#) () const
- void [round\\_reset](#) ()
- size\_t [tot\\_counts](#) () const
- size\_t [uniq\\_counts](#) () const
- [Bundle](#) \* [bundle](#) ()
- void [bundle](#) ([Bundle](#) \*b)
- void [add\\_mass](#) (double p, double v, double mass)
- void [incr\\_counts](#) (bool uniq, size\_t incr\_amt=1)
- double [log\\_likelihood](#) (const [FragHit](#) &frag, bool with\_pseudo) const
- double [est\\_effective\\_length](#) ([FLD](#) \*fld=NULL, bool with\_bias=true) const
- double [cached\\_effective\\_length](#) (bool with\_bias=true) const
- void [update\\_transcript\\_bias](#) ([BiasBoss](#) \*bias\_table=NULL, [FLD](#) \*fld=NULL)

### 3.24.1 Detailed Description

The [Transcript](#) class is used to store objects for the transcripts being mapped to. Besides storing basic information about the object (id, length), it also stores a mass based on the number of fragments mapping to the object. To help with updating this number, it returns the likelihood that a given fragment originated from it. These values are stored and returned in log space.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 86 of file transcripts.h.

### 3.24.2 Constructor & Destructor Documentation

#### 3.24.2.1 Transcript::Transcript ( const size\_t *id*, const std::string & *name*, const std::string & *seq*, double *alpha*, const Globals \* *globs* )

[Transcript](#) Constructor

##### Parameters

<i>name</i>	a string that stores the transcript name
<i>seq</i>	a string that stores the transcript sequence
<i>alpha</i>	a double that specifies the initial pseudo-counts (non-logged)
<i>globs</i>	a pointer to the struct containing pointers to the global parameter tables (bias_table, mismatch_table, fld)

Definition at line 23 of file transcripts.cpp.

### 3.24.3 Member Function Documentation

#### 3.24.3.1 void Transcript::add\_mass ( double *p*, double *v*, double *mass* )

a member function that increases the expected fragment counts and variance by a given (logged) fragment mass

##### Parameters

<i>p</i>	a double for the (logged) probability that the fragment was generated by this transcript
<i>v</i>	a double for the (logged) approximate variance (uncertainty) on the probability
<i>mass</i>	a double specifying the (logged) mass of the fragment being mapped

Definition at line 48 of file transcripts.cpp.

#### 3.24.3.2 double Transcript::ambig\_mass ( ) const [inline]

a member function that returns the (logged) total mass derived from ambiguous fragments mapping to the transcript

##### Returns

the (logged) total mass derived from ambiguous fragments mapping to the transcript

Definition at line 233 of file transcripts.h.

#### 3.24.3.3 double Transcript::binom\_var ( ) const [inline]

a member function that returns the current (logged) binomial variance

**Returns**

logged binomial mass variance

Definition at line 245 of file transcripts.h.

**3.24.3.4 void Transcript::bundle ( Bundle \* *b* ) [inline]**

a member function that set the [Bundle](#) this [Transcript](#) is a member of

**Parameters**

<i>b</i>	a pointer to the <a href="#">Bundle</a> to set this <a href="#">Transcript</a> as a member of
----------	---

Definition at line 287 of file transcripts.h.

**3.24.3.5 Bundle\* Transcript::bundle ( ) [inline]**

a member function that returns the [Bundle](#) this [Transcript](#) is a member of

**Returns**

a pointer to the [Bundle](#) this transcript is a member of

Definition at line 281 of file transcripts.h.

**3.24.3.6 double Transcript::cached\_effective\_length ( bool *with\_bias* = true ) const**

a member function that returns the most recently estimated effective length (logged) as calculated by the bias updater thread

**Returns**

the cached effective length of the transcript calculated

Definition at line 146 of file transcripts.cpp.

**3.24.3.7 double Transcript::est\_effective\_length ( FLD \* *fld* = NULL, bool *with\_bias* = true ) const**

a member function that calculates and returns the estimated effective length of the transcript (logged) using the avg bias

**Parameters**

<i>fld</i>	an optional pointer to a different <a href="#">FLD</a> than the global one, for thread-safety
<i>with_bias</i>	a boolean specifying whether or not the average bias should be used in the calculation

**Returns**

the estimated effective length of the transcript calculated as  $\tilde{l} = \bar{bias} \sum_{l=1}^{L(t)} D(l)(L(t) - l + 1)$

Definition at line 123 of file transcripts.cpp.

**3.24.3.8 TransID Transcript::id ( ) const [inline]**

a member function that returns the transcript id

**Returns**

TransID transcript ID

Definition at line 203 of file transcripts.h.

**3.24.3.9 void Transcript::incr\_counts ( bool *uniq*, size\_t *incr\_amt* = 1 ) [inline]**

a member function that increases the count of fragments mapped to this transcript

**Parameters**

<i>uniq</i>	a bool specifying whether or not the fragment uniquely maps to this transcript
<i>incr_amt</i>	a size_t to increase the counts by

Definition at line 302 of file transcripts.h.

**3.24.3.10 size\_t Transcript::length ( ) const [inline]**

a member function that returns the transcript length

**Returns**

transcript length

Definition at line 214 of file transcripts.h.

**3.24.3.11 double Transcript::log\_likelihood ( const FragHit & *frag*, bool *with\_pseudo* ) const**

a member function that returns (a value proportional to) the log likelihood the given fragment originated from this transcript

**Parameters**

<i>frag</i>	a <a href="#">FragHit</a> to return the likelihood of being originated from this transcript
<i>with_pseudo</i>	a <a href="#">FragHit</a> specifying whether or not pseudo-counts should be included in the calculation



**Returns**

(a value proportional to) the log likelihood the given fragment originated from this transcript

Definition at line 90 of file transcripts.cpp.

**3.24.3.12 double Transcript::mass ( bool *with\_pseudo* = true ) const**

a member function that returns the current (logged) probabilistically assigned fragment mass

**Parameters**

<i>with_pseudo</i>	a boolean specifying whether pseudo-counts should be included in returned mass
--------------------	--

**Returns**

logged mass

Definition at line 82 of file transcripts.cpp.

**3.24.3.13 const std::string& Transcript::name ( ) const [inline]**

a member function that returns the transcript name

**Returns**

string containing transcript name

Definition at line 197 of file transcripts.h.

**3.24.3.14 double Transcript::rho ( ) const**

a member function that returns the current estimated rho (logged, w/ pseudo-counts) for the transcript

**Returns**

the current estimated rho

Definition at line 73 of file transcripts.cpp.

**3.24.3.15 void Transcript::round\_reset ( )**

a member function that readies the transcript object for the next round of batch EM

Definition at line 66 of file transcripts.cpp.

**3.24.3.16 double Transcript::smp\_var ( ) const [inline]**

a member function that returns the (logged) sampling variance

**Returns**

the (logged) sampling variance

Definition at line 251 of file transcripts.h.

**3.24.3.17 const Sequence& Transcript::seq ( bool rev ) const [inline]**

a member function that returns the transcript sequence

**Returns**

string containing transcript sequence

Definition at line 208 of file transcripts.h.

**3.24.3.18 double Transcript::tot\_ambig\_mass ( ) const [inline]**

a member function that returns the (logged) total mass of ambiguous fragments mapping to the transcript

**Returns**

the (logged) total mass of ambiguous fragments mapping to the transcript

Definition at line 239 of file transcripts.h.

**3.24.3.19 size\_t Transcript::tot\_counts ( ) const [inline]**

a member function that returns the current count of fragments mapped to this transcript (uniquely or ambiguously)

**Returns**

total fragment count

Definition at line 269 of file transcripts.h.

**3.24.3.20 double Transcript::tot\_uncertainty ( ) const [inline]**

a member function that returns the (logged) variance due to uncertainty on p

**Returns**

the (logged) variance due to uncertainty on p

Definition at line 258 of file transcripts.h.

**3.24.3.21** `size_t Transcript::uniq_counts ( ) const` `[inline]`

a member function that returns the current count of fragments uniquely mapped to this transcript

**Returns**

unique fragment count

Definition at line 275 of file transcripts.h.

**3.24.3.22** `void Transcript::update_transcript_bias ( BiasBoss * bias_table = NULL, FLD * fld = NULL )`

a member function that causes the transcript bias to be re-calculated by the `_bias_table` based on current parameters

**Parameters**

<i>bias_table</i>	an optional pointer to a different <a href="#">BiasBoss</a> than the global one, for thread-safety
<i>fld</i>	an optional pointer to a different <a href="#">FLD</a> than the global one, for thread-safety

Definition at line 155 of file transcripts.cpp.

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

- src/transcripts.h
- src/transcripts.cpp

**3.25 TranscriptTable Class Reference**

```
#include <transcripts.h>
```

**Public Member Functions**

- [TranscriptTable](#) (const std::string &trans\_fasta\_file, const TransIndex &trans\_index, const TransIndex &trans\_lengths, double alpha, const AlphaMap \*alpha\_map, [Globals](#) \*globs)
- [~TranscriptTable](#) ()
- [Transcript \\* get\\_trans](#) (TransID id)
- void [round\\_reset](#) ()
- `size_t` [size](#) () const
- double [total\\_fpb](#) () const
- void [update\\_total\\_fpb](#) (double incr\_amt)
- void [update\\_covar](#) (TransID trans1, TransID trans2, double covar)
- double [get\\_covar](#) (TransID trans1, TransID trans2)

- `size_t covar_size () const`
- `Bundle * merge_bundles (Bundle *b1, Bundle *b2)`
- `size_t num_bundles ()`
- `void output_results (std::string output_dir, size_t tot_counts, bool output_varcov)`
- `void threaded_bias_update (boost::mutex *mut)`

### 3.25.1 Detailed Description

The [TranscriptTable](#) class is used to keep track of the [Transcript](#) objects for a run. The constructor parses a fasta file to generate the [Transcript](#) objects and store them in a map that allows them to be looked up based on their string id.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 353 of file transcripts.h.

### 3.25.2 Constructor & Destructor Documentation

**3.25.2.1** `TranscriptTable::TranscriptTable ( const std::string & trans_fasta_file, const TransIndex & trans_index, const TransIndex & trans_lengths, double alpha, const AlphaMap * alpha_map, Globals * globs )`

[TranscriptTable](#) Constructor

#### Parameters

<i>trans_fasta_file</i>	a string storing the path to the fasta file from which to load transcripts
<i>trans_index</i>	the transcript-to-index map from the alignment file
<i>trans_lengths</i>	the transcript-to-length map from the alignment file
<i>alpha</i>	a double that specifies the initial pseudo-counts for each bp of the transcripts (non-logged)
<i>alpha_map</i>	an optional pointer to a map object that specifies proportional weights of pseudo-counts for each transcript
<i>globs</i>	a pointer to the struct containing pointers to the global parameter tables (bias_table, mismatch_table, fld)

Definition at line 176 of file transcripts.cpp.

**3.25.2.2 TranscriptTable::~~TranscriptTable ( )**

[TranscriptTable](#) Destructor deletes all of the transcript objects in the table

Definition at line 265 of file transcripts.cpp.

**3.25.3 Member Function Documentation****3.25.3.1 size\_t TranscriptTable::covar\_size ( ) const [inline]**

a member function that returns the number of pairs of transcripts with non-zero covariance

**Returns**

the number of transcript pairs with non-zero covariance

Definition at line 466 of file transcripts.h.

**3.25.3.2 double TranscriptTable::get\_covar ( TransID *trans1*, TransID *trans2* ) [inline]**

a member function that returns the covariance between two transcripts these returned value will be the log of the negative of the true value

**Parameters**

<i>trans1</i>	one of the transcripts in the pair
<i>trans2</i>	the other transcript in the pair

**Returns**

the negative of the pair's covariance (logged)

Definition at line 460 of file transcripts.h.

**3.25.3.3 Transcript \* TranscriptTable::get\_trans ( TransID *id* )**

a member function that returns a pointer to the transcript with the given id

**Parameters**

<i>id</i>	of the transcript queried
-----------	---------------------------

**Returns**

pointer to the transcript wit the given id

Definition at line 295 of file transcripts.cpp.

**3.25.3.4 Bundle \* TranscriptTable::merge\_bundles ( Bundle \* *b1*, Bundle \* *b2* )**

a member function that merges the given Bundles

**Parameters**

<i>b1</i>	a pointer to the first <a href="#">Bundle</a> to merge
<i>b2</i>	a pointer to the second <a href="#">Bundle</a> to merge

**Returns**

a pointer to the merged [Bundle](#)

Definition at line 300 of file transcripts.cpp.

**3.25.3.5 size\_t TranscriptTable::num\_bundles ( )**

a member function that returns the number of bundles in the partition

**Returns**

the number of bundles in the partition

Definition at line 309 of file transcripts.cpp.

**3.25.3.6 void TranscriptTable::output\_results ( std::string *output\_dir*, size\_t *tot\_counts*, bool *output\_varcov* )**

a member function that outputs the final expression data in a file called 'results.xprs' and (optionally) the variance-covariance matrix in 'varcov.xprs' in the given output directory

**Parameters**

<i>output_dir</i>	the directory to output the expression file to
<i>tot_counts</i>	the total number of observed mapped fragments
<i>output_varcov</i>	boolean specifying whether to also output the variance-covariance matrix

Definition at line 379 of file transcripts.cpp.

**3.25.3.7 void TranscriptTable::round\_reset ( )**

a member function that readies all [Transcript](#) objects in the table for the next round of batch EM

Definition at line 314 of file transcripts.cpp.

**3.25.3.8** `size_t TranscriptTable::size ( ) const` `[inline]`

a member function that returns the number of transcripts in the table

**Returns**

number of transcripts in the table

Definition at line 430 of file transcripts.h.

**3.25.3.9** `void TranscriptTable::threaded_bias_update ( boost::mutex * mut )`

a member function for driving a thread that continuously updates the transcript bias values

Definition at line 549 of file transcripts.cpp.

**3.25.3.10** `double TranscriptTable::total_fpb ( ) const`

a member function that returns the (logged) total mass per base (including pseudo-counts)

**Returns**

the (logged) total mass per base (including pseudo-counts)

Definition at line 537 of file transcripts.cpp.

**3.25.3.11** `void TranscriptTable::update_covar ( TransID trans1, TransID trans2, double covar )`  
`[inline]`

a member function that increases the covariance between two transcripts by the specified amount these values are stored positive even though they are negative (logged)

**Parameters**

<i>trans1</i>	one of the transcripts in the pair
<i>trans2</i>	the other transcript in the pair
<i>covar</i>	a double specifying the amount to increase the pair's covariance by (logged)

Definition at line 451 of file transcripts.h.

**3.25.3.12** `void TranscriptTable::update_total_fpb ( double incr_amt )`

a member function that increments the (logged) total mass per base

**Parameters**

<i>incr_amt</i>	the (logged) amount to increment by
-----------------	-------------------------------------

Definition at line 543 of file transcripts.cpp.

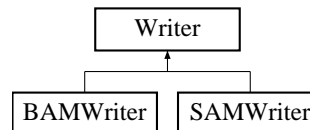
The documentation for this class was generated from the following files:

- src/transcripts.h
- src/transcripts.cpp

## 3.26 Writer Class Reference

```
#include <mapparser.h>
```

Inheritance diagram for Writer:



### Public Member Functions

- virtual void [write\\_fragment](#) ([Fragment](#) &*f*)=0

#### 3.26.1 Detailed Description

The [Writer](#) class is an abstract class that can be a [SAMWriter](#) or [BAMWriter](#). It writes [Fragment](#) objects back to file (in SAM/BAM format) with per-mapping probabilistic assignments.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 76 of file mapparser.h.

### 3.26.2 Member Function Documentation

#### 3.26.2.1 virtual void [Writer::write\\_fragment](#) ( [Fragment](#) & *f* ) [pure virtual]

a member function that writes all mappings of the fragment to the output file along with their probabilities in the "XP" field

#### Parameters



$f$ the processed <a href="#">Fragment</a> to output
--

Implemented in [BAMWriter](#), and [SAMWriter](#).

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

- `src/mapparser.h`

# Index

- ~BAMParser
  - BAMParser, [6](#)
- ~BAMWriter
  - BAMWriter, [8](#)
- ~BundleTable
  - BundleTable, [14](#)
- ~Fragment
  - Fragment, [23](#)
- ~SAMWriter
  - SAMWriter, [44](#)
- ~Sequence
  - Sequence, [46](#)
- ~ThreadedMapParser
  - ThreadedMapParser, [51](#)
- ~TranscriptTable
  - TranscriptTable, [60](#)
- activate
  - MismatchTable, [31](#)
- add\_map\_end
  - Fragment, [24](#)
- add\_mass
  - Transcript, [54](#)
- add\_val
  - FLD, [18](#)
- ambig\_mass
  - RoundParams, [40](#)
  - Transcript, [54](#)
- append\_output
  - BiasBoss, [10](#)
  - FLD, [18](#)
  - MismatchTable, [31](#)
  - PosWeightTable, [37](#)
  - SeqWeightTable, [48](#)
- arr
  - FrequencyMatrix, [26](#)
- BAMParser, [5](#)
  - ~BAMParser, [6](#)
  - BAMParser, [6](#)
  - header, [6](#)
  - next\_fragment, [6](#)
  - reset, [6](#)
  - trans\_index, [7](#)
  - trans\_lengths, [7](#)
- BAMWriter, [7](#)
  - ~BAMWriter, [8](#)
  - BAMWriter, [8](#)
  - write\_fragment, [8](#)
- BiasBoss, [9](#)
  - append\_output, [10](#)
  - BiasBoss, [10](#)
  - get\_transcript\_bias, [10](#)
  - normalize\_expectations, [10](#)
  - to\_string, [10](#)
  - update\_expectations, [11](#)
  - update\_observed, [11](#)
- binom\_var
  - RoundParams, [40](#)
  - Transcript, [54](#)
- Bundle, [11](#)
  - Bundle, [12](#)
  - counts, [12](#)
  - incr\_counts, [12](#)
  - size, [13](#)
  - transcripts, [13](#)
- bundle
  - Transcript, [55](#)
- bundles
  - BundleTable, [14](#)
- BundleTable, [13](#)
  - ~BundleTable, [14](#)
  - bundles, [14](#)
  - BundleTable, [14](#)
  - create\_bundle, [14](#)
  - merge, [15](#)
  - size, [15](#)
- cached\_effective\_length
  - Transcript, [55](#)
- cond
  - ParseThreadSafety, [35](#)

- copy\_expected
  - SeqWeightTable, 48
- copy\_observed
  - SeqWeightTable, 48
- counts
  - Bundle, 12
- covar\_size
  - TranscriptTable, 61
- CovarTable, 15
  - get, 16
  - increment, 16
  - size, 16
- create\_bundle
  - BundleTable, 14
- deletes\_l
  - FragHit, 21
- deletes\_r
  - FragHit, 21
- empty
  - Sequence, 46
- est\_effective\_length
  - Transcript, 55
- FLD, 17
  - add\_val, 18
  - append\_output, 18
  - FLD, 17
  - max\_val, 18
  - mean, 18
  - pdf, 18
  - to\_string, 19
  - tot\_mass, 19
- frag\_clean
  - ParseThreadSafety, 35
- FragHit, 19
  - deletes\_l, 21
  - deletes\_r, 21
  - inserts\_l, 21
  - inserts\_r, 21
  - left, 21
  - left\_first, 21
  - length, 20
  - mapped\_trans, 22
  - mate\_l, 22
  - name, 22
  - pair\_status, 20
  - right, 22
  - seq\_l, 22
  - seq\_r, 22
  - trans\_id, 22
- Fragment, 23
  - ~Fragment, 23
  - add\_map\_end, 24
  - hits, 24
  - name, 24
  - num\_hits, 24
  - sample\_hit, 24
- FrequencyMatrix, 25
  - arr, 26
  - FrequencyMatrix, 26
  - increment, 26, 27
  - operator(), 27
  - row, 27
  - set\_logged, 28
- get
  - CovarTable, 16
- get\_covar
  - TranscriptTable, 61
- get\_trans
  - TranscriptTable, 61
- get\_transcript\_bias
  - BiasBoss, 10
- get\_weight
  - PosWeightTable, 37
  - SeqWeightTable, 49
- Globals, 28
- header
  - BAMParser, 6
  - Parser, 33
  - SAMParser, 42
- hits
  - Fragment, 24
- id
  - Transcript, 56
- incr\_counts
  - Bundle, 12
  - Transcript, 56
- increment
  - CovarTable, 16
  - FrequencyMatrix, 26, 27
- increment\_expected
  - PosWeightTable, 37, 38
  - SeqWeightTable, 49
- increment\_observed
  - PosWeightTable, 38

- SeqWeightTable, 49
- Indel, 29
  - Indel, 29
  - len, 29
  - pos, 29
- inserts\_l
  - FragHit, 21
- inserts\_r
  - FragHit, 21
- left
  - FragHit, 21
- left\_first
  - FragHit, 21
- len
  - Indel, 29
- length
  - FragHit, 20
  - Sequence, 46
  - Transcript, 56
- log\_likelihood
  - MismatchTable, 32
  - Transcript, 56
- mapped\_trans
  - FragHit, 22
- MarkovModel, 30
- mass
  - RoundParams, 40
  - Transcript, 57
- mate\_l
  - FragHit, 22
- max\_val
  - FLD, 18
- mean
  - FLD, 18
- merge
  - BundleTable, 15
- merge\_bundles
  - TranscriptTable, 61
- MismatchTable, 30
  - activate, 31
  - append\_output, 31
  - log\_likelihood, 32
  - MismatchTable, 31
  - to\_string, 32
  - update, 32
- mut
  - ParseThreadSafety, 35
- name
  - FragHit, 22
  - Fragment, 24
  - Transcript, 57
- next\_frag
  - ParseThreadSafety, 35
- next\_fragment
  - BAMParser, 6
  - Parser, 33
  - SAMParser, 42
- normalize\_expectations
  - BiasBoss, 10
- normalize\_expected
  - PosWeightTable, 39
  - SeqWeightTable, 49
- num\_bundles
  - TranscriptTable, 62
- num\_hits
  - Fragment, 24
- operator()
  - FrequencyMatrix, 27
- operator=
  - Sequence, 46
- output\_results
  - TranscriptTable, 62
- pair\_status
  - FragHit, 20
- Parser, 33
  - header, 33
  - next\_fragment, 33
  - reset, 34
  - trans\_index, 34
  - trans\_lengths, 34
- ParseThreadSafety, 34
  - cond, 35
  - frag\_clean, 35
  - mut, 35
  - next\_frag, 35
- pdf
  - FLD, 18
- pos
  - Indel, 29
- PosWeightTable, 36
  - append\_output, 37
  - get\_weight, 37
  - increment\_expected, 37, 38
  - increment\_observed, 38
  - normalize\_expected, 39
  - PosWeightTable, 36

- reset
  - BAMParser, 6
  - Parser, 34
- reset\_reader
  - ThreadedMapParser, 51
- rho
  - Transcript, 57
- right
  - FragHit, 22
- round\_reset
  - Transcript, 57
  - TranscriptTable, 62
- RoundParams, 39
  - ambig\_mass, 40
  - binom\_var, 40
  - mass, 40
  - RoundParams, 40
  - samp\_var, 40
  - tot\_ambig\_mass, 40
  - tot\_unc, 40
- row
  - FrequencyMatrix, 27
- samp\_var
  - RoundParams, 40
  - Transcript, 57
- SAMParser, 41
  - header, 42
  - next\_fragment, 42
  - SAMParser, 42
  - trans\_index, 42
  - trans\_lengths, 42
- sample\_hit
  - Fragment, 24
- SAMWriter, 43
  - ~SAMWriter, 44
  - SAMWriter, 44
  - write\_fragment, 44
- seq
  - Transcript, 58
- seq\_l
  - FragHit, 22
- seq\_r
  - FragHit, 22
- Sequence, 44
  - ~Sequence, 46
  - empty, 46
  - length, 46
  - operator=, 46
  - Sequence, 45
- set, 47
- SeqWeightTable, 47
  - append\_output, 48
  - copy\_expected, 48
  - copy\_observed, 48
  - get\_weight, 49
  - increment\_expected, 49
  - increment\_observed, 49
  - normalize\_expected, 49
  - SeqWeightTable, 48
  - to\_string, 50
- set
  - Sequence, 47
- set\_logged
  - FrequencyMatrix, 28
- size
  - Bundle, 13
  - BundleTable, 15
  - CovarTable, 16
  - TranscriptTable, 62
- threaded\_bias\_update
  - TranscriptTable, 63
- threaded\_parse
  - ThreadedMapParser, 51
- ThreadedMapParser, 50
  - ~ThreadedMapParser, 51
  - reset\_reader, 51
  - threaded\_parse, 51
  - ThreadedMapParser, 51
  - trans\_index, 52
  - trans\_lengths, 52
  - write\_active, 52
- to\_string
  - BiasBoss, 10
  - FLD, 19
  - MismatchTable, 32
  - SeqWeightTable, 50
- tot\_ambig\_mass
  - RoundParams, 40
  - Transcript, 58
- tot\_counts
  - Transcript, 58
- tot\_mass
  - FLD, 19
- tot\_unc
  - RoundParams, 40
- tot\_uncertainty
  - Transcript, 58
- total\_fpb

- TranscriptTable, 63
- trans\_id
  - FragHit, 22
- trans\_index
  - BAMParser, 7
  - Parser, 34
  - SAMParser, 42
  - ThreadedMapParser, 52
- trans\_lengths
  - BAMParser, 7
  - Parser, 34
  - SAMParser, 42
  - ThreadedMapParser, 52
- Transcript, 52
  - add\_mass, 54
  - ambig\_mass, 54
  - binom\_var, 54
  - bundle, 55
  - cached\_effective\_length, 55
  - est\_effective\_length, 55
  - id, 56
  - incr\_counts, 56
  - length, 56
  - log\_likelihood, 56
  - mass, 57
  - name, 57
  - rho, 57
  - round\_reset, 57
  - samp\_var, 57
  - seq, 58
  - tot\_ambig\_mass, 58
  - tot\_counts, 58
  - tot\_uncertainty, 58
  - Transcript, 54
  - uniq\_counts, 58
  - update\_transcript\_bias, 59
- transcripts
  - Bundle, 13
- TranscriptTable, 59
  - ~TranscriptTable, 60
  - covar\_size, 61
  - get\_covar, 61
  - get\_trans, 61
  - merge\_bundles, 61
  - num\_bundles, 62
  - output\_results, 62
  - round\_reset, 62
  - size, 62
  - threaded\_bias\_update, 63
  - total\_fpb, 63
  - TranscriptTable, 60
  - update\_covar, 63
  - update\_total\_fpb, 63
- uniq\_counts
  - Transcript, 58
- update
  - MismatchTable, 32
- update\_covar
  - TranscriptTable, 63
- update\_expectations
  - BiasBoss, 11
- update\_observed
  - BiasBoss, 11
- update\_total\_fpb
  - TranscriptTable, 63
- update\_transcript\_bias
  - Transcript, 59
- write\_active
  - ThreadedMapParser, 52
- write\_fragment
  - BAMWriter, 8
  - SAMWriter, 44
  - Writer, 64
- Writer, 64
  - write\_fragment, 64