



eXpress

0.90 BETA

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

Generated by Doxygen 1.7.3

Thu Sep 15 2011 09:40:24



# 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
3.1.3	Member Function Documentation . . . . .	6
3.1.3.1	next_fragment . . . . .	6
3.2	BiasBoss Class Reference . . . . .	6
3.2.1	Detailed Description . . . . .	7
3.2.2	Constructor & Destructor Documentation . . . . .	7
3.2.2.1	BiasBoss . . . . .	7
3.2.3	Member Function Documentation . . . . .	7
3.2.3.1	append_output . . . . .	7
3.2.3.2	get_transcript_bias . . . . .	7
3.2.3.3	to_string . . . . .	8
3.2.3.4	update_expectations . . . . .	8
3.2.3.5	update_observed . . . . .	8
3.3	FLD Class Reference . . . . .	9
3.3.1	Detailed Description . . . . .	9
3.3.2	Constructor & Destructor Documentation . . . . .	9
3.3.2.1	FLD . . . . .	9
3.3.3	Member Function Documentation . . . . .	10
3.3.3.1	add_val . . . . .	10
3.3.3.2	append_output . . . . .	10
3.3.3.3	max_val . . . . .	10
3.3.3.4	mean . . . . .	10
3.3.3.5	pdf . . . . .	10
3.3.3.6	to_string . . . . .	11
3.3.3.7	tot_mass . . . . .	11
3.4	FragMap Struct Reference . . . . .	11
3.4.1	Detailed Description . . . . .	12
3.4.2	Member Function Documentation . . . . .	12
3.4.2.1	length . . . . .	12

3.4.2.2	pair_status	12
3.4.3	Member Data Documentation	13
3.4.3.1	left	13
3.4.3.2	left_first	13
3.4.3.3	mapped_trans	13
3.4.3.4	mate_l	13
3.4.3.5	name	13
3.4.3.6	right	13
3.4.3.7	seq_l	13
3.4.3.8	seq_r	14
3.4.3.9	trans_id	14
3.5	Fragment Class Reference	14
3.5.1	Detailed Description	14
3.5.2	Constructor & Destructor Documentation	15
3.5.2.1	~Fragment	15
3.5.3	Member Function Documentation	15
3.5.3.1	add_map_end	15
3.5.3.2	maps	15
3.5.3.3	name	15
3.5.3.4	num_maps	15
3.6	FrequencyMatrix Class Reference	16
3.6.1	Detailed Description	16
3.6.2	Constructor & Destructor Documentation	16
3.6.2.1	FrequencyMatrix	16
3.6.2.2	FrequencyMatrix	17
3.6.3	Member Function Documentation	17
3.6.3.1	arr	17
3.6.3.2	increment	17
3.6.3.3	increment	17
3.6.3.4	operator()	18
3.6.3.5	operator()	18
3.6.3.6	row	18
3.7	MismatchTable Class Reference	19
3.7.1	Detailed Description	19
3.7.2	Constructor & Destructor Documentation	19
3.7.2.1	MismatchTable	19
3.7.3	Member Function Documentation	20
3.7.3.1	activate	20
3.7.3.2	append_output	20
3.7.3.3	log_likelihood	20
3.7.3.4	to_string	20
3.7.3.5	update	21
3.8	Parser Class Reference	21
3.8.1	Detailed Description	21
3.8.2	Member Function Documentation	22
3.8.2.1	next_fragment	22
3.9	ParseThreadSafety Struct Reference	22
3.9.1	Detailed Description	22
3.9.2	Member Data Documentation	23
3.9.2.1	next_frag	23

3.9.2.2	<a href="#">parse_lk</a>	23
3.9.2.3	<a href="#">proc_lk</a>	23
3.10	<a href="#">PosWeightTable Class Reference</a>	23
3.10.1	<a href="#">Detailed Description</a>	23
3.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	24
3.10.2.1	<a href="#">PosWeightTable</a>	24
3.10.3	<a href="#">Member Function Documentation</a>	24
3.10.3.1	<a href="#">append_output</a>	24
3.10.3.2	<a href="#">get_weight</a>	24
3.10.3.3	<a href="#">get_weight</a>	25
3.10.3.4	<a href="#">increment_expected</a>	25
3.10.3.5	<a href="#">increment_expected</a>	25
3.10.3.6	<a href="#">increment_observed</a>	25
3.10.3.7	<a href="#">increment_observed</a>	26
3.11	<a href="#">SAMParse Class Reference</a>	26
3.11.1	<a href="#">Detailed Description</a>	27
3.11.2	<a href="#">Constructor &amp; Destructor Documentation</a>	27
3.11.2.1	<a href="#">SAMParse</a>	27
3.11.3	<a href="#">Member Function Documentation</a>	27
3.11.3.1	<a href="#">next_fragment</a>	27
3.12	<a href="#">SeqWeightTable Class Reference</a>	27
3.12.1	<a href="#">Detailed Description</a>	28
3.12.2	<a href="#">Constructor &amp; Destructor Documentation</a>	28
3.12.2.1	<a href="#">SeqWeightTable</a>	28
3.12.3	<a href="#">Member Function Documentation</a>	28
3.12.3.1	<a href="#">append_output</a>	28
3.12.3.2	<a href="#">get_weight</a>	29
3.12.3.3	<a href="#">increment_expected</a>	29
3.12.3.4	<a href="#">increment_observed</a>	29
3.12.3.5	<a href="#">to_string</a>	29
3.13	<a href="#">ThreadedMapParser Class Reference</a>	30
3.13.1	<a href="#">Detailed Description</a>	30
3.13.2	<a href="#">Constructor &amp; Destructor Documentation</a>	30
3.13.2.1	<a href="#">ThreadedMapParser</a>	30
3.13.3	<a href="#">Member Function Documentation</a>	31
3.13.3.1	<a href="#">threaded_parse</a>	31
3.14	<a href="#">Transcript Class Reference</a>	31
3.14.1	<a href="#">Detailed Description</a>	32
3.14.2	<a href="#">Constructor &amp; Destructor Documentation</a>	32
3.14.2.1	<a href="#">Transcript</a>	32
3.14.3	<a href="#">Member Function Documentation</a>	32
3.14.3.1	<a href="#">add_mass</a>	32
3.14.3.2	<a href="#">bundle_counts</a>	33
3.14.3.3	<a href="#">effective_length</a>	33
3.14.3.4	<a href="#">est_effective_length</a>	33
3.14.3.5	<a href="#">id</a>	33
3.14.3.6	<a href="#">incr_bundle_counts</a>	34
3.14.3.7	<a href="#">length</a>	34
3.14.3.8	<a href="#">log_likelihood</a>	34
3.14.3.9	<a href="#">mass</a>	34

3.14.3.10	name	34
3.14.3.11	seq	35
3.14.3.12	tot_counts	35
3.14.3.13	unbiased_effective_length	35
3.14.3.14	uniq_counts	35
3.14.3.15	update_transcript_bias	36
3.14.3.16	var	36
3.15	TranscriptTable Class Reference	36
3.15.1	Detailed Description	37
3.15.2	Constructor & Destructor Documentation	37
3.15.2.1	TranscriptTable	37
3.15.2.2	~TranscriptTable	37
3.15.3	Member Function Documentation	37
3.15.3.1	covar_size	37
3.15.3.2	get_covar	38
3.15.3.3	get_trans	38
3.15.3.4	get_trans_rep	38
3.15.3.5	merge_bundles	39
3.15.3.6	num_bundles	39
3.15.3.7	output_bundles	39
3.15.3.8	output_results	39
3.15.3.9	size	40
3.15.3.10	threaded_bias_update	40
3.15.3.11	update_covar	40

# Chapter 1

## Class Index

### 1.1 Class Hierarchy

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

BiasBoss . . . . .	6
FLD . . . . .	9
FragMap . . . . .	11
Fragment . . . . .	14
FrequencyMatrix . . . . .	16
MismatchTable . . . . .	19
Parser . . . . .	21
BAMParser . . . . .	5
SAMParse . . . . .	26
ParseThreadSafety . . . . .	22
PosWeightTable . . . . .	23
SeqWeightTable . . . . .	27
ThreadedMapParser . . . . .	30
Transcript . . . . .	31
TranscriptTable . . . . .	36





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">BAMParser</a>	5
<a href="#">BiasBoss</a>	6
<a href="#">FLD</a>	9
<a href="#">FragMap</a>	11
<a href="#">Fragment</a>	14
<a href="#">FrequencyMatrix</a>	16
<a href="#">MismatchTable</a>	19
<a href="#">Parser</a>	21
<a href="#">ParseThreadSafety</a>	22
<a href="#">PosWeightTable</a>	23
<a href="#">SAMParser</a>	26
<a href="#">SeqWeightTable</a>	27
<a href="#">ThreadedMapParser</a>	30
<a href="#">Transcript</a>	31
<a href="#">TranscriptTable</a>	36



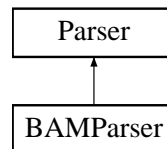
## Chapter 3

# Class Documentation

### 3.1 BAMParser Class Reference

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

Inheritance diagram for BAMParser:



#### Public Member Functions

- [BAMParser](#) (BamTools::BamReader \*reader)
- bool [next\\_fragment](#) ([Fragment](#) &f)

#### 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 46 of file mapparser.h.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 BAMParser::BAMParser ( BamTools::BamReader \* *reader* )

[BAMParser](#) constructor opens the file

Definition at line 117 of file mapparser.cpp.

### 3.1.3 Member Function Documentation

#### 3.1.3.1 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 133 of file mapparser.cpp.

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

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

## 3.2 BiasBoss Class Reference

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

### Public Member Functions

- [BiasBoss](#) (double alpha)
- void [update\\_expectations](#) (const [Transcript](#) &trans)
- void [update\\_observed](#) (const [FragMap](#) &frag, double mass)
- double [get\\_transcript\\_bias](#) (std::vector< double > &start\_bias, std::vector< double > &end\_bias, const [Transcript](#) &trans) const
- std::string [to\\_string](#) () const
- void [append\\_output](#) (std::ofstream &outfile) const

### 3.2.1 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 208 of file biascorrection.h.

### 3.2.2 Constructor & Destructor Documentation

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

[BiasBoss](#) Constructor

##### Parameters

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

Definition at line 218 of file biascorrection.cpp.

### 3.2.3 Member Function Documentation

#### 3.2.3.1 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
----------------	-----------------------

#### 3.2.3.2 double BiasBoss::get\_transcript\_bias ( std::vector< double > & *start\_bias*, std::vector< double > & *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 284 of file biascorrection.cpp.

**3.2.3.3 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 313 of file biascorrection.cpp.

**3.2.3.4 void BiasBoss::update\_expectations ( const Transcript & trans )**

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 225 of file biascorrection.cpp.

**3.2.3.5 void BiasBoss::update\_observed ( const FragMap & frag, 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>frag</i>	the fragment mapping
<i>mass</i>	the logged probability of the mapping, which is the amount to update the observed counts by

Definition at line 245 of file biascorrection.cpp.

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

- src/biascorrection.h

- `src/biascorrection.cpp`

### 3.3 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
- 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

#### 3.3.1 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`.

#### 3.3.2 Constructor & Destructor Documentation

##### 3.3.2.1 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="#">FragMap</a> length
<i>mean</i>	a size_t for the mean of the prior gaussian dist
<i>std_dev</i>	a size_t for the std dev of the prior gaussian dist

Definition at line 21 of file `fld.cpp`.

### 3.3.3 Member Function Documentation

#### 3.3.3.1 void FLD::add\_val ( size\_t *len*, double *mass* )

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

##### Parameters

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

Definition at line 46 of file fld.cpp.

#### 3.3.3.2 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
----------------	-----------------------

#### 3.3.3.3 size\_t FLD::max\_val ( ) const

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

##### Returns

max allowed [FragMap](#) length

Definition at line 41 of file fld.cpp.

#### 3.3.3.4 double FLD::mean ( ) const

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

##### Returns

mean observed [FragMap](#) length

Definition at line 74 of file fld.cpp.

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

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

##### Parameters

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



**Returns**

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

Definition at line 62 of file fld.cpp.

**3.3.3.6 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 79 of file fld.cpp.

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

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

**Returns**

number of observed fragments

Definition at line 69 of file fld.cpp.

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

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

## 3.4 FragMap Struct Reference

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

**Public Member Functions**

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

**Public Attributes**

- std::string [name](#)
- TransID [trans\\_id](#)
- Transcript \* [mapped\\_trans](#)

- `std::string seq_l`
- `std::string seq_r`
- `int left`
- `int right`
- `int mate_l`
- `bool left_first`

### 3.4.1 Detailed Description

The `FragMap` struct stores the information for a single (multi-)mapping of a fragment.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 36 of file `fragments.h`.

### 3.4.2 Member Function Documentation

#### 3.4.2.1 `int FragMap::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 94 of file `fragments.h`.

#### 3.4.2.2 `PairStatus FragMap::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 105 of file `fragments.h`.

### 3.4.3 Member Data Documentation

#### 3.4.3.1 `int FragMap::left`

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

Definition at line 67 of file fragments.h.

#### 3.4.3.2 `bool FragMap::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 87 of file fragments.h.

#### 3.4.3.3 `Transcript* FragMap::mapped_trans`

a public pointer to the transcript mapped to

Definition at line 51 of file fragments.h.

#### 3.4.3.4 `int FragMap::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 80 of file fragments.h.

#### 3.4.3.5 `std::string FragMap::name`

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

Definition at line 41 of file fragments.h.

#### 3.4.3.6 `int FragMap::right`

a public int 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 73 of file fragments.h.

#### 3.4.3.7 `std::string FragMap::seq_l`

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

Definition at line 56 of file fragments.h.

#### 3.4.3.8 `std::string FragMap::seq_r`

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

Definition at line 61 of file fragments.h.

#### 3.4.3.9 `TransID FragMap::trans_id`

a public TransID for the transcript mapped to

Definition at line 46 of file fragments.h.

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

- src/fragments.h

## 3.5 Fragment Class Reference

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

### Public Member Functions

- [~Fragment](#) ()
- bool [add\\_map\\_end](#) ([FragMap](#) \*f)
- const std::string [name](#) () const
- const size\_t [num\\_maps](#) () const
- const std::vector< [FragMap](#) \* > & [maps](#) () const

#### 3.5.1 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 123 of file fragments.h.

### 3.5.2 Constructor & Destructor Documentation

#### 3.5.2.1 `Fragment::~~Fragment ( )`

`Fragment` destructor deletes all `FragMap` objects pointed to by the `Fragment`

Definition at line 14 of file `fragments.cpp`.

### 3.5.3 Member Function Documentation

#### 3.5.3.1 `bool Fragment::add_map_end ( FragMap * f )`

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

##### Parameters

<code>f</code>	the <code>FragMap</code> to be added
----------------	--------------------------------------

Definition at line 27 of file `fragments.cpp`.

#### 3.5.3.2 `const std::vector<FragMap*>& Fragment::maps ( ) const [inline]`

a member function that returns `FragMap` multi-mappings of the fragment

##### Returns

a vector containing pointers to the `FragMap` multi-mappings

Definition at line 181 of file `fragments.h`.

#### 3.5.3.3 `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 169 of file `fragments.h`.

#### 3.5.3.4 `const size_t Fragment::num_maps ( ) 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 175 of file fragments.h.

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

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

## 3.6 FrequencyMatrix Class Reference

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

### Public Member Functions

- [FrequencyMatrix](#) ()
- [FrequencyMatrix](#) (size\_t m, size\_t n, double alpha)
- 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

### 3.6.1 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. All values are stored and returned in log space.

#### Author

Adam Roberts

#### Date

2011 Artistic License 2.0

Definition at line 24 of file frequencymatrix.h.

### 3.6.2 Constructor & Destructor Documentation

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

dummy constructor

Definition at line 51 of file frequencymatrix.h.

### 3.6.2.2 FrequencyMatrix::FrequencyMatrix ( size\_t *m*, size\_t *n*, double *alpha* )

[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)

Definition at line 16 of file frequencymatrix.cpp.

### 3.6.3 Member Function Documentation

#### 3.6.3.1 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 96 of file frequencymatrix.h.

#### 3.6.3.2 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 logged amount to increase the mass by

Definition at line 37 of file frequencymatrix.cpp.

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

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

#### Parameters

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

<i>incr_amt</i>	the logged amount to increase the mass by
-----------------	---

Definition at line 45 of file frequencymatrix.cpp.

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

a member function to extract the logged probability of a given position in the flattened matrix

##### Parameters

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

##### Returns

a double specifying the logged probability of the given position in the flattened matrix

Definition at line 31 of file frequencymatrix.cpp.

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

a member function to extract the logged probability of a given position in the matrix

##### Parameters

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

##### Returns

a double specifying the logged probability of the given value in the given distribution

Definition at line 24 of file frequencymatrix.cpp.

#### 3.6.3.6 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 103 of file frequencymatrix.h.



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

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

## 3.7 MismatchTable Class Reference

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

### Public Member Functions

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

### 3.7.1 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 read 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.

### 3.7.2 Constructor & Destructor Documentation

#### 3.7.2.1 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 18 of file mismatchmodel.cpp.

### 3.7.3 Member Function Documentation

#### 3.7.3.1 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 59 of file mismatchmodel.h.

#### 3.7.3.2 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 85 of file biascorrection.cpp.

#### 3.7.3.3 double MismatchTable::log\_likelihood ( const FragMap & *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 25 of file mismatchmodel.cpp.

#### 3.7.3.4 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 113 of file mismatchmodel.cpp.

### 3.7.3.5 void MismatchTable::update ( const FragMap & *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 72 of file mismatchmodel.cpp.

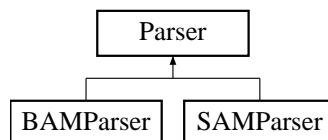
The documentation for this class was generated from the following files:

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

## 3.8 Parser Class Reference

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

Inheritance diagram for Parser:



### Public Member Functions

- virtual bool [next\\_fragment](#) ([Fragment](#) &*f*)=0

### 3.8.1 Detailed Description

The [Parser](#) class is an abstract class that can be a [SAMPParser](#) or [BAMParser](#).

#### Author

Adam Roberts

**Date**

2011 Artistic License 2.0

Definition at line 27 of file mapparser.h.

**3.8.2 Member Function Documentation****3.8.2.1** `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](#).

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

- src/mapparser.h

**3.9 ParseThreadSafety Struct Reference**

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

**Public Attributes**

- [Fragment](#) \* `next_frag`
- boost::mutex `proc_lk`
- boost::mutex `parse_lk`

**3.9.1 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 126 of file mapparser.h.

### 3.9.2 Member Data Documentation

#### 3.9.2.1 `Fragment* ParseThreadSafety::next_frag`

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

Definition at line 131 of file mapparser.h.

#### 3.9.2.2 `boost::mutex ParseThreadSafety::parse_lk`

a mutex to lock the parsing thread when the next\_frag pointer should be not modified

Definition at line 141 of file mapparser.h.

#### 3.9.2.3 `boost::mutex ParseThreadSafety::proc_lk`

a mutex to lock the main (processing) thread when next\_frag has not yet been updated

Definition at line 136 of file mapparser.h.

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

- src/mapparser.h

## 3.10 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 [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

#### 3.10.1 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 101 of file biascorrection.h.

**3.10.2 Constructor & Destructor Documentation****3.10.2.1 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 paramater)

Definition at line 123 of file biascorrection.cpp.

**3.10.3 Member Function Documentation****3.10.3.1 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
----------------	-----------------------

**3.10.3.2 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 156 of file biascorrection.cpp.

### 3.10.3.3 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 55 of file biascorrection.cpp.

### 3.10.3.4 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 137 of file biascorrection.cpp.

### 3.10.3.5 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 130 of file biascorrection.cpp.

### 3.10.3.6 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 143 of file biascorrection.cpp.

**3.10.3.7** `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 44 of file biascorrection.cpp.

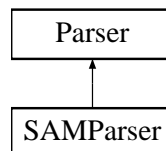
The documentation for this class was generated from the following files:

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

**3.11 SAMParser Class Reference**

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

Inheritance diagram for SAMParser:

**Public Member Functions**

- [SAMParser](#) (std::istream \*in)
- bool [next\\_fragment](#) ([Fragment](#) &f)



### 3.11.1 Detailed Description

The [SAMParse](#) 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 85 of file mapparser.h.

### 3.11.2 Constructor & Destructor Documentation

#### 3.11.2.1 SAMParse::SAMParse ( std::istream \* in )

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

Definition at line 183 of file mapparser.cpp.

### 3.11.3 Member Function Documentation

#### 3.11.3.1 bool SAMParse::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

<code>f</code>	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 211 of file mapparser.cpp.

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

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

## 3.12 SeqWeightTable Class Reference

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

## Public Member Functions

- [SeqWeightTable](#) (size\_t window\_size, double alpha)
- void [increment\\_expected](#) (char c)
- void [increment\\_observed](#) (std::string &seq, double normalized\_mass)
- double [get\\_weight](#) (const std::string &seq, int i) const
- std::string [to\\_string](#) () const
- void [append\\_output](#) (std::ofstream &outfile) const

### 3.12.1 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 29 of file biascorrection.h.

### 3.12.2 Constructor & Destructor Documentation

#### 3.12.2.1 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 paramater)

Definition at line 29 of file biascorrection.cpp.

### 3.12.3 Member Function Documentation

#### 3.12.3.1 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
----------------	-----------------------

**3.12.3.2 double SeqWeightTable::get\_weight ( const std::string & seq, int i ) const**

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

**Parameters**

<i>seq</i>	the transcript sequence the fragment maps 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

**3.12.3.3 void SeqWeightTable::increment\_expected ( char c )**

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

**Parameters**

<i>c</i>	a char representing a nucleotide that has been observed in the transcriptome
----------	--

Definition at line 34 of file biascorrection.cpp.

**3.12.3.4 void SeqWeightTable::increment\_observed ( std::string & seq, double normalized\_mass )**

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

**Parameters**

<i>seq</i>	a string of nucleotides in the bias window for the sequenced fragment end
<i>normalized_mass</i>	the mass (logged probabilistic assignment) of the fragment normalized by its estimated expression

**3.12.3.5 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

Definition at line 68 of file biascorrection.cpp.

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

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

### 3.13 ThreadedMapParser Class Reference

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

**Public Member Functions**

- [ThreadedMapParser](#) (std::string input\_file)
- void [threaded\\_parse](#) ([ParseThreadSafety](#) \*thread\_safety, [TranscriptTable](#) \*trans\_table)

#### 3.13.1 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 154 of file mapparser.h.

#### 3.13.2 Constructor & Destructor Documentation

##### 3.13.2.1 ThreadedMapParser::ThreadedMapParser ( std::string input\_file )

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

Definition at line 51 of file mapparser.cpp.

### 3.13.3 Member Function Documentation

**3.13.3.1** `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 80 of file mapparser.cpp.

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

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

## 3.14 Transcript Class Reference

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

### Public Member Functions

- [Transcript](#) (const std::string &name, const std::string &seq, double alpha, const FLD \*fld, const [BiasBoss](#) \*bias\_table, const [MismatchTable](#) \*mismatch\_table)
- const std::string & [name](#) () const
- TransID [id](#) () const
- const std::string & [seq](#) () const
- size\_t [length](#) () const
- double [mass](#) () const
- double [var](#) () const
- size\_t [tot\\_counts](#) () const
- size\_t [uniq\\_counts](#) () const
- void [add\\_mass](#) (double p, double mass)
- void [incr\\_bundle\\_counts](#) (size\_t incr\_amt=1)
- size\_t [bundle\\_counts](#) ()
- double [log\\_likelihood](#) (const [FragMap](#) &frag) const
- double [effective\\_length](#) () const
- double [est\\_effective\\_length](#) () const
- double [unbiased\\_effective\\_length](#) () const
- void [update\\_transcript\\_bias](#) ()

### 3.14.1 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 57 of file transcripts.h.

### 3.14.2 Constructor & Destructor Documentation

**3.14.2.1** `Transcript::Transcript ( const std::string & name, const std::string & seq, double alpha, const FLD * fld, const BiasBoss * bias_table, const MismatchTable * mismatch_table )`

[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>fld</i>	a pointer to the global <a href="#">Fragment Length Distribution (FLD)</a> object
<i>bias_table</i>	a pointer to the global <a href="#">BiasBoss</a> object
<i>mismatch_table</i>	a pointer to the global <a href="#">MismatchTable</a> object

Definition at line 24 of file transcripts.cpp.

### 3.14.3 Member Function Documentation

**3.14.3.1** `void Transcript::add_mass ( double p, 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>mass</i>	a double specifying the (logged) mass of the fragment being mapped

Definition at line 41 of file transcripts.cpp.

#### 3.14.3.2 `size_t Transcript::bundle_counts ( ) [inline]`

a member function that returns the counts mapping to the bundle this transcript is in the total bundle counts is the sum of this value for all transcripts in the bundle

##### Returns

a portion of the counts mapping to the bundle this transcript is in

Definition at line 231 of file transcripts.h.

#### 3.14.3.3 `double Transcript::effective_length ( ) const`

a member function that calculates and returns the effective length of the transcript (non-logged)

##### Returns

the effective length of the transcript calculated as  $\tilde{l} = \sum_{l=1}^{L(t)} \sum_{i=1}^{L(t)} D(l) b_5[i] * b_3[i+l]$

Definition at line 110 of file transcripts.cpp.

#### 3.14.3.4 `double Transcript::est_effective_length ( ) const`

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

##### 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 91 of file transcripts.cpp.

#### 3.14.3.5 `TransID Transcript::id ( ) const [inline]`

a member function that returns the transcript id

##### Returns

TransID transcript ID

Definition at line 172 of file transcripts.h.

**3.14.3.6 void Transcript::incr\_bundle\_counts ( size\_t *incr\_amt* = 1 ) [inline]**

a member function that increases the counts mapping to the bundle this transcript is in the total bundle counts is the sum of this value for all transcripts in the bundle

**Parameters**

<i>incr_amt</i>	a size_t to increase the counts by
-----------------	------------------------------------

Definition at line 221 of file transcripts.h.

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

a member function that returns the transcript length

**Returns**

transcript length

Definition at line 183 of file transcripts.h.

**3.14.3.8 double Transcript::log\_likelihood ( const FragMap & *frag* ) 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="#">FragMap</a> to return the likelihood of being originated from this transcript
-------------	---

**Returns**

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

Definition at line 57 of file transcripts.cpp.

**3.14.3.9 double Transcript::mass ( ) const [inline]**

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

**Returns**

logged mass

Definition at line 189 of file transcripts.h.

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

a member function that returns the transcript name



**Returns**

string containing transcript name

Definition at line 166 of file transcripts.h.

**3.14.3.11 const std::string& Transcript::seq( ) const [inline]**

a member function that returns the transcript sequence

**Returns**

string containing transcript sequence

Definition at line 177 of file transcripts.h.

**3.14.3.12 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 201 of file transcripts.h.

**3.14.3.13 double Transcript::unbiased\_effective\_length( ) const**

a member function that calculates and returns the effective length of the transcript (non-logged) ignoring bias and using the prior [FLD](#) distribution

**Returns**

the effective length of the transcript calculated as  $\tilde{l} = \sum_{l=1}^{L(t)} D_{prior}(l)(L(t) - l + 1)$

Definition at line 105 of file transcripts.cpp.

**3.14.3.14 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 207 of file transcripts.h.

### 3.14.3.15 void Transcript::update\_transcript\_bias ( )

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

Definition at line 134 of file `transcripts.cpp`.

### 3.14.3.16 double Transcript::var ( ) const [inline]

a member function that returns the current (logged) variance

#### Returns

logged mass variance

Definition at line 195 of file `transcripts.h`.

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

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

## 3.15 TranscriptTable Class Reference

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

### Public Member Functions

- [TranscriptTable](#) (const std::string &trans\_fasta\_file, double alpha, const [FLD](#) \*fld, [BiasBoss](#) \*bias\_table, const [MismatchTable](#) \*mismatch\_table)
- [~TranscriptTable](#) ()
- [Transcript](#) \* [get\\_trans](#) (TransID id)
- size\_t [size](#) () const
- void [update\\_covar](#) (TransID trans1, TransID trans2, double covar)
- double [get\\_covar](#) (TransID trans1, TransID trans2)
- size\_t [covar\\_size](#) () const
- TransID [get\\_trans\\_rep](#) (TransID trans)
- TransID [merge\\_bundles](#) (TransID rep1, TransID rep2)
- size\_t [num\\_bundles](#) ()
- void [output\\_bundles](#) (std::string output\_dir)
- void [threaded\\_bias\\_update](#) ()
- void [output\\_results](#) (std::string output\_dir, size\_t tot\_counts, bool output\_varcov)

### 3.15.1 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 285 of file transcripts.h.

### 3.15.2 Constructor & Destructor Documentation

**3.15.2.1** [TranscriptTable::TranscriptTable](#) ( const std::string & *trans\_fasta\_file*, double *alpha*, const FLD \* *fld*, BiasBoss \* *bias\_table*, const MismatchTable \* *mismatch\_table* )

[TranscriptTable](#) Constructor

#### Parameters

<i>trans_fasta_file</i>	a string storing the path to the fasta file from which to load transcripts
<i>alpha</i>	a double that specifies the initial pseudo-counts for each bp of the transcripts (non-logged)
<i>fld</i>	a pointer to the global <a href="#">Fragment Length Distribution (FLD)</a> object
<i>bias_table</i>	a pointer to the global <a href="#">BiasBoss</a> object
<i>mismatch_table</i>	a pointer to the global <a href="#">MismatchTable</a> object

Definition at line 142 of file transcripts.cpp.

**3.15.2.2** [TranscriptTable::~~TranscriptTable](#) ( )

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

Definition at line 199 of file transcripts.cpp.

### 3.15.3 Member Function Documentation

**3.15.3.1** `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 368 of file transcripts.h.

**3.15.3.2 double TranscriptTable::get\_covar ( TransID *trans1*, TransID *trans2* )**

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 log of the negative of the pair's covariance

Definition at line 248 of file transcripts.cpp.

**3.15.3.3 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 227 of file transcripts.cpp.

**3.15.3.4 TransID TranscriptTable::get\_trans\_rep ( TransID *trans* )**

a member function that returns the bundle representative of the given transcript in the partitioning

**Parameters**

<i>trans</i>	the TransID of the transcript whose representative is requested
--------------	---

**Returns**

the TransID of the representative for the bundle the given transcript is in

Definition at line 262 of file transcripts.cpp.

**3.15.3.5 TransID TranscriptTable::merge\_bundles ( TransID *rep1*, TransID *rep2* )**

a member function that merges the bundles represented by the two given transcripts

**Parameters**

<i>rep1</i>	the TransID of the first bundle representative
<i>rep2</i>	the TransID of the second bundle representative

**Returns**

the TransID of the representative for the new merged bundle

Definition at line 268 of file transcripts.cpp.

**3.15.3.6 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 277 of file transcripts.cpp.

**3.15.3.7 void TranscriptTable::output\_bundles ( std::string *output\_dir* )**

a member function that outputs the bundles of the partition in a tab-delimited file called 'bundles.tab' in the given output directory each line contains a space-separated list of transcripts in a single bundle

**Parameters**

<i>output_dir</i>	the directory to output the bundle file to
-------------------	--

Definition at line 288 of file transcripts.cpp.

**3.15.3.8 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 368 of file transcripts.cpp.

### 3.15.3.9 `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 344 of file transcripts.h.

### 3.15.3.10 `void TranscriptTable::threaded_bias_update ( )`

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

Definition at line 308 of file transcripts.cpp.

### 3.15.3.11 `void TranscriptTable::update_covar ( TransID trans1, TransID trans2, double covar )`

a member function that increases the covariance between two transcripts by the specified (logged) amount 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 (logged) amount to increase the pair's covariance by

Definition at line 235 of file transcripts.cpp.

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

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

# Index

- ~Fragment
  - Fragment, [15](#)
- ~TranscriptTable
  - TranscriptTable, [37](#)
- activate
  - MismatchTable, [20](#)
- add\_map\_end
  - Fragment, [15](#)
- add\_mass
  - Transcript, [32](#)
- add\_val
  - FLD, [10](#)
- append\_output
  - BiasBoss, [7](#)
  - FLD, [10](#)
  - MismatchTable, [20](#)
  - PosWeightTable, [24](#)
  - SeqWeightTable, [28](#)
- arr
  - FrequencyMatrix, [17](#)
- BAMParser, [5](#)
  - BAMParser, [6](#)
  - next\_fragment, [6](#)
- BiasBoss, [6](#)
  - append\_output, [7](#)
  - BiasBoss, [7](#)
  - get\_transcript\_bias, [7](#)
  - to\_string, [8](#)
  - update\_expectations, [8](#)
  - update\_observed, [8](#)
- bundle\_counts
  - Transcript, [33](#)
- covar\_size
  - TranscriptTable, [37](#)
- effective\_length
  - Transcript, [33](#)
- est\_effective\_length
  - Transcript, [33](#)
- FLD, [9](#)
  - add\_val, [10](#)
  - append\_output, [10](#)
  - FLD, [9](#)
  - max\_val, [10](#)
  - mean, [10](#)
  - pdf, [10](#)
  - to\_string, [11](#)
  - tot\_mass, [11](#)
- FragMap, [11](#)
  - left, [13](#)
  - left\_first, [13](#)
  - length, [12](#)
  - mapped\_trans, [13](#)
  - mate\_l, [13](#)
  - name, [13](#)
  - pair\_status, [12](#)
  - right, [13](#)
  - seq\_l, [13](#)
  - seq\_r, [14](#)
  - trans\_id, [14](#)
- Fragment, [14](#)
  - ~Fragment, [15](#)
  - add\_map\_end, [15](#)
  - maps, [15](#)
  - name, [15](#)
  - num\_maps, [15](#)
- FrequencyMatrix, [16](#)
  - arr, [17](#)
  - FrequencyMatrix, [16](#)
  - increment, [17](#)
  - operator(), [18](#)
  - row, [18](#)
- get\_covar
  - TranscriptTable, [38](#)
- get\_trans
  - TranscriptTable, [38](#)
- get\_trans\_rep
  - TranscriptTable, [38](#)
- get\_transcript\_bias

- BiasBoss, 7
- get\_weight
  - PosWeightTable, 24, 25
  - SeqWeightTable, 29
- id
  - Transcript, 33
- incr\_bundle\_counts
  - Transcript, 33
- increment
  - FrequencyMatrix, 17
- increment\_expected
  - PosWeightTable, 25
  - SeqWeightTable, 29
- increment\_observed
  - PosWeightTable, 25, 26
  - SeqWeightTable, 29
- left
  - FragMap, 13
- left\_first
  - FragMap, 13
- length
  - FragMap, 12
  - Transcript, 34
- log\_likelihood
  - MismatchTable, 20
  - Transcript, 34
- mapped\_trans
  - FragMap, 13
- maps
  - Fragment, 15
- mass
  - Transcript, 34
- mate\_l
  - FragMap, 13
- max\_val
  - FLD, 10
- mean
  - FLD, 10
- merge\_bundles
  - TranscriptTable, 38
- MismatchTable, 19
  - activate, 20
  - append\_output, 20
  - log\_likelihood, 20
  - MismatchTable, 19
  - to\_string, 20
  - update, 21
- name
  - FragMap, 13
  - Fragment, 15
  - Transcript, 34
- next\_frag
  - ParseThreadSafety, 23
- next\_fragment
  - BAMParser, 6
  - Parser, 22
  - SAMParser, 27
- num\_bundles
  - TranscriptTable, 39
- num\_maps
  - Fragment, 15
- operator()
  - FrequencyMatrix, 18
- output\_bundles
  - TranscriptTable, 39
- output\_results
  - TranscriptTable, 39
- pair\_status
  - FragMap, 12
- parse\_lk
  - ParseThreadSafety, 23
- Parser, 21
  - next\_fragment, 22
- ParseThreadSafety, 22
  - next\_frag, 23
  - parse\_lk, 23
  - proc\_lk, 23
- pdf
  - FLD, 10
- PosWeightTable, 23
  - append\_output, 24
  - get\_weight, 24, 25
  - increment\_expected, 25
  - increment\_observed, 25, 26
  - PosWeightTable, 24
- proc\_lk
  - ParseThreadSafety, 23
- right
  - FragMap, 13
- row
  - FrequencyMatrix, 18
- SAMParser, 26
  - next\_fragment, 27



- SAMParser, 27
- seq
  - Transcript, 35
- seq\_l
  - FragMap, 13
- seq\_r
  - FragMap, 14
- SeqWeightTable, 27
  - append\_output, 28
  - get\_weight, 29
  - increment\_expected, 29
  - increment\_observed, 29
  - SeqWeightTable, 28
  - to\_string, 29
- size
  - TranscriptTable, 40
- threaded\_bias\_update
  - TranscriptTable, 40
- threaded\_parse
  - ThreadedMapParser, 31
- ThreadedMapParser, 30
  - threaded\_parse, 31
  - ThreadedMapParser, 30
- to\_string
  - BiasBoss, 8
  - FLD, 11
  - MismatchTable, 20
  - SeqWeightTable, 29
- tot\_counts
  - Transcript, 35
- tot\_mass
  - FLD, 11
- trans\_id
  - FragMap, 14
- Transcript, 31
  - add\_mass, 32
  - bundle\_counts, 33
  - effective\_length, 33
  - est\_effective\_length, 33
  - id, 33
  - incr\_bundle\_counts, 33
  - length, 34
  - log\_likelihood, 34
  - mass, 34
  - name, 34
  - seq, 35
  - tot\_counts, 35
  - Transcript, 32
  - unbiased\_effective\_length, 35
  - uniq\_counts, 35
  - update\_transcript\_bias, 35
  - var, 36
- TranscriptTable, 36
  - ~TranscriptTable, 37
  - covar\_size, 37
  - get\_covar, 38
  - get\_trans, 38
  - get\_trans\_rep, 38
  - merge\_bundles, 38
  - num\_bundles, 39
  - output\_bundles, 39
  - output\_results, 39
  - size, 40
  - threaded\_bias\_update, 40
  - TranscriptTable, 37
  - update\_covar, 40
- unbiased\_effective\_length
  - Transcript, 35
- uniq\_counts
  - Transcript, 35
- update
  - MismatchTable, 21
- update\_covar
  - TranscriptTable, 40
- update\_expectations
  - BiasBoss, 8
- update\_observed
  - BiasBoss, 8
- update\_transcript\_bias
  - Transcript, 35
- var
  - Transcript, 36