

Veridical

Generated by Doxygen 1.8.5

Mon Mar 24 2014 20:09:04

Contents

1	Veridical - An RNA-Seq Variant Validation Program	1
2	Namespace Index	3
2.1	Namespace List	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	bgzf_add_eof Namespace Reference	9
5.1.1	Function Documentation	9
5.1.1.1	fix_bam	9
5.1.1.2	sys_exit	9
6	Class Documentation	11
6.1	ValidationReasons Class Reference	11
6.1.1	Member Typedef Documentation	15
6.1.1.1	InnerKeyType	15
6.1.1.2	InnerKeyType	15
6.1.1.3	SortedValReasonsMapType	15
6.1.1.4	SortedValReasonsMapType	15
6.1.1.5	ValidatedInnerMapType	16
6.1.1.6	ValidatedInnerMapType	16
6.1.1.7	ValidatedMapType	16
6.1.1.8	ValidatedMapType	16
6.1.2	Constructor & Destructor Documentation	16
6.1.2.1	ValidationReasons	16
6.1.2.2	ValidationReasons	16
6.1.3	Member Function Documentation	16
6.1.3.1	addNonVarContValReason	16

6.1.3.2	addNonVarContValReason	16
6.1.3.3	addNonVarContValReasons	16
6.1.3.4	addNonVarContValReasons	17
6.1.3.5	addSummaryValDetails	17
6.1.3.6	addSummaryValDetails	17
6.1.3.7	addToTotal	17
6.1.3.8	addToTotal	17
6.1.3.9	addValidationReason	18
6.1.3.10	addValidationReason	18
6.1.3.11	addValidationReasons	18
6.1.3.12	addValidationReasons	18
6.1.3.13	addVarContValidationReason	19
6.1.3.14	addVarContValidationReason	19
6.1.3.15	addVarContValidationReasons	19
6.1.3.16	addVarContValidationReasons	19
6.1.3.17	getAllSpliceConsequencesCount	19
6.1.3.18	getAllSpliceConsequencesCount	19
6.1.3.19	getCrypticValNumber	20
6.1.3.20	getCrypticValNumber	21
6.1.3.21	getNonVarContCrypticSiteUseValScore	21
6.1.3.22	getNonVarContCrypticSiteUseValScore	21
6.1.3.23	getNonVarContDetails	21
6.1.3.24	getNonVarContDetails	21
6.1.3.25	getNonVarContDirectEvidenceValCount	22
6.1.3.26	getNonVarContDirectEvidenceValCount	22
6.1.3.27	getNonVarContExonSkippingValScore	22
6.1.3.28	getNonVarContExonSkippingValScore	22
6.1.3.29	getNonVarContIndirectEvidenceValCount	22
6.1.3.30	getNonVarContIndirectEvidenceValCount	22
6.1.3.31	getNonVarContIntronInclDirectNumber	22
6.1.3.32	getNonVarContIntronInclDirectNumber	22
6.1.3.33	getNonVarContIntronInclIndirectNumber	23
6.1.3.34	getNonVarContIntronInclIndirectNumber	23
6.1.3.35	getNonVarContIntronInclValScore	23
6.1.3.36	getNonVarContIntronInclValScore	23
6.1.3.37	getNonVarContOverallValCount	23
6.1.3.38	getNonVarContOverallValCount	23
6.1.3.39	getNonVarContSampleSize	23
6.1.3.40	getNonVarContSampleSize	23
6.1.3.41	getPredominantValidationReason	24

6.1.3.42	getPredominantValidationReason	24
6.1.3.43	getPValue	24
6.1.3.44	getPValue	24
6.1.3.45	getValidationDetails	24
6.1.3.46	getValidationDetails	25
6.1.3.47	getValNumber	25
6.1.3.48	getValNumber	26
6.1.3.49	getValNumber	26
6.1.3.50	getValNumber	26
6.1.3.51	getValScore	26
6.1.3.52	getValScore	26
6.1.3.53	getVarContCrypticSiteUseValScore	27
6.1.3.54	getVarContCrypticSiteUseValScore	27
6.1.3.55	getVarContDetails	27
6.1.3.56	getVarContDetails	27
6.1.3.57	getVarContDirectEvidenceValCount	27
6.1.3.58	getVarContDirectEvidenceValCount	27
6.1.3.59	getVarContExonSkippingValScore	28
6.1.3.60	getVarContExonSkippingValScore	28
6.1.3.61	getVarContIndirectEvidenceValCount	28
6.1.3.62	getVarContIndirectEvidenceValCount	28
6.1.3.63	getVarContIntronInclDirectNumber	28
6.1.3.64	getVarContIntronInclDirectNumber	28
6.1.3.65	getVarContIntronInclIndirectNumber	28
6.1.3.66	getVarContIntronInclIndirectNumber	28
6.1.3.67	getVarContIntronInclValScore	29
6.1.3.68	getVarContIntronInclValScore	29
6.1.3.69	getVarContOverallValCount	29
6.1.3.70	getVarContOverallValCount	29
6.1.3.71	getVarContPredominantValidationReasonName	29
6.1.3.72	getVarContPredominantValidationReasonName	29
6.1.3.73	getVarContSampleSize	29
6.1.3.74	getVarContSampleSize	29
6.1.3.75	getZScore	30
6.1.3.76	getZScore	30
6.1.3.77	getZScores	30
6.1.3.78	getZScores	31
6.1.3.79	isStronglyCorroborating	31
6.1.3.80	op_YJTransform	31
6.1.3.81	op_YJTransform	31

6.1.3.82	YJTransform	32
6.1.3.83	YJTransform	32
6.1.4	Member Data Documentation	32
6.1.4.1	nonVarContValMap	32
6.1.4.2	varContValMap	32
7	File Documentation	33
7.1	/home/cviner2/RNA-seq/bgzf_add_eof.py File Reference	33
7.2	/home/cviner2/RNA-seq/Veridical.cpp File Reference	33
7.2.1	Detailed Description	37
7.2.2	Typedef Documentation	38
7.2.2.1	BAMFileListType	38
7.2.2.2	ExonMapType	38
7.2.2.3	MultiExonMapType	38
7.2.2.4	ReadFractionsType	38
7.2.2.5	TeeDevice	38
7.2.2.6	TeeStream	38
7.2.2.7	ValDetailsType	38
7.2.2.8	ValidationReasonsToAddType	38
7.2.3	Enumeration Type Documentation	38
7.2.3.1	Directionality	38
7.2.3.2	EvidenceType	39
7.2.3.3	SampleType	39
7.2.3.4	SplicingConsequence	39
7.2.4	Function Documentation	40
7.2.4.1	checkExonSkipping	40
7.2.4.2	getTimestamp	41
7.2.4.3	isDNANucleotide	41
7.2.4.4	main	41
7.2.4.5	noFileErr	41
7.2.4.6	SPAN_PATTERN	42
7.2.4.7	VAR_PATTERN	42
7.2.5	Variable Documentation	42
7.2.5.1	ACCEPTOR	42
7.2.5.2	CHR_PREFIX	42
7.2.5.3	COMMENT_CHAR	42
7.2.5.4	countCoeff	42
7.2.5.5	DECIMAL_PRECISION	42
7.2.5.6	DEFAULT_P_VALUE	42
7.2.5.7	directCoeff	42

7.2.5.8	DONOR	42
7.2.5.9	evidenceTypeNames	42
7.2.5.10	EXON	43
7.2.5.11	fullBAMFileList	43
7.2.5.12	H_CHR	43
7.2.5.13	H_CODING	43
7.2.5.14	H_COORD_PART	43
7.2.5.15	H_COORD_SS	43
7.2.5.16	H_CRPTORNAT	43
7.2.5.17	H_E_CHR	43
7.2.5.18	H_E_ECOORD	43
7.2.5.19	H_E_GENE	43
7.2.5.20	H_E_NUM	43
7.2.5.21	H_E_SCOORD	43
7.2.5.22	H_E_TID	43
7.2.5.23	H_FILE	43
7.2.5.24	H_GENE	43
7.2.5.25	H_HET	43
7.2.5.26	H_IN_PART	43
7.2.5.27	H_STRAND	43
7.2.5.28	H_TYPE	43
7.2.5.29	H_VAR_PART	43
7.2.5.30	INTRON	43
7.2.5.31	MIN_MAP_Q	43
7.2.5.32	MIN_READ_CUTOFF_FOR_VAL	43
7.2.5.33	NEG	44
7.2.5.34	NORMAL_SAMPLE	44
7.2.5.35	OUTPUT_PROG EVERY	44
7.2.5.36	POS	44
7.2.5.37	sampleTypeNames	44
7.2.5.38	splicingConsequenceNames	44
7.2.5.39	TOKEN_SEPERATOR_CD	44
7.2.5.40	TOKEN_SEPERATOR_WS	44
7.2.5.41	TYPE_CRYPTIC	44
7.2.5.42	TYPE_EXONIC	44
7.2.5.43	YJ_LAMBDA	44
7.3	/home/cviner2/RNA-seq/VeridicalTrial.cpp File Reference	45
7.3.1	Typedef Documentation	48
7.3.1.1	BAMFileListType	48
7.3.1.2	ExonMapType	49

7.3.1.3	MultiExonMapType	49
7.3.1.4	ReadFractionsType	49
7.3.1.5	TeeDevice	49
7.3.1.6	TeeStream	49
7.3.1.7	ValDetailsType	49
7.3.1.8	ValidationReasonsToAddType	49
7.3.2	Enumeration Type Documentation	49
7.3.2.1	Directionality	49
7.3.2.2	EvidenceType	50
7.3.2.3	SampleType	50
7.3.2.4	SplicingConsequence	50
7.3.3	Function Documentation	51
7.3.3.1	checkExonSkipping	51
7.3.3.2	getTimestamp	52
7.3.3.3	isDNANucleotide	52
7.3.3.4	main	52
7.3.3.5	noFileErr	52
7.3.3.6	SPAN_PATTERN	53
7.3.3.7	VAR_PATTERN	53
7.3.4	Variable Documentation	53
7.3.4.1	ACCEPTOR	53
7.3.4.2	CHR_PREFIX	53
7.3.4.3	COMMENT_CHAR	53
7.3.4.4	countCoeff	53
7.3.4.5	DECIMAL_PRECISION	53
7.3.4.6	DEFAULT_P_VALUE	53
7.3.4.7	directCoeff	53
7.3.4.8	DONOR	53
7.3.4.9	evidenceTypeNames	53
7.3.4.10	EXON	54
7.3.4.11	fullBAMFileList	54
7.3.4.12	H_CHR	54
7.3.4.13	H_CODING	54
7.3.4.14	H_COORD_PART	54
7.3.4.15	H_COORD_SS	54
7.3.4.16	H_CRPTORNAT	54
7.3.4.17	H_E_CHR	54
7.3.4.18	H_E_ECOORD	54
7.3.4.19	H_E_GENE	54
7.3.4.20	H_E_NUM	54

7.3.4.21	H_E_SCOORD	54
7.3.4.22	H_E_TID	54
7.3.4.23	H_FILE	54
7.3.4.24	H_GENE	54
7.3.4.25	H_HET	54
7.3.4.26	H_IN_PART	54
7.3.4.27	H_STRAND	54
7.3.4.28	H_TYPE	54
7.3.4.29	H_VAR_PART	54
7.3.4.30	INTRON	54
7.3.4.31	MIN_MAP_Q	54
7.3.4.32	MIN_READ_CUTOFF_FOR_VAL	54
7.3.4.33	NEG	55
7.3.4.34	NORMAL_SAMPLE	55
7.3.4.35	OUTPUT_PROG EVERY	55
7.3.4.36	POS	55
7.3.4.37	sampleTypeNames	55
7.3.4.38	splicingConsequenceNames	55
7.3.4.39	TOKEN_SEPERATOR_CD	55
7.3.4.40	TOKEN_SEPERATOR_WS	55
7.3.4.41	TYPE_CRYPTIC	55
7.3.4.42	TYPE_EXONIC	55
7.3.4.43	VAL_JSON_PATH	55
7.3.4.44	VAL_SITE	55
7.3.4.45	YJ_LAMBDA	55
Bibliographic References		56
Index		58

Chapter 1

Veridical - An RNA-Seq Variant Validation Program

This program attempts to validate putative splicing variants by checking for concordance between splicing predictions and corresponding RNA-Seq data. This program utilizes all non-variant containing files as controls and can be provided with an additional set of normal samples to be used as controls as well. This program creates a log file and a file filtered by p-value (defaulting to 0.05).

Usage:

- **-h** [**-help**] Produce help message and exit.
- **-m** [**-mutations**] arg *REQUIRED*: provide full path to the mutation (variant) file.
- **-e** [**-exomeAn**] arg *REQUIRED*: provide full path to the exome annotation file.
 - NB: variants should be derived and analyzed with respect to a conservative exome (i.e. one in which exon-intron boundaries require a moderately high level of biological evidence as opposed to those with purely EST-based transcripts or those reliant upon in silico methods. This will serve to limit the number of "boutique exons" and decrease the number of biologically-irrelevant validation events. For example, RefSeq would be appropriate, while Ensembl would not.
- **-t** [**-tumour**] arg *REQUIRED*: provide full path to the list of tumour RNA-Seq BAM files.
- **-n** [**-normals**] arg Provide full path to the list of normal RNA-Seq BAM files.
- **-b** [**-base**] arg Provide the desired base output full path. Defaults to `./VeridicalOut` if not specified.
- **-p** [**-filterPVal**] arg Override the default p-value cutoff for filtered data.
- **-d** [**-direct**] Specify that only junction-spanning evidence-based variants should be output.
- **-P** [**-nopvals**] Specify that no p-values should be output. Useful for downstream programs that directly parse output tables.
- **-T** [**-transformed**] Specify that all output data should be transformed via the Yeo-Johnson transformation [3]. Overrides the default wherein only the p-values output use the transformed data.
- **-v** [**-verbose**] arg Specify the verbosity of the program's output.
 - An argument is optional and this option may be specified multiple times. If the argument is a single digit it will be used, otherwise the length of any given argument will be used as the verbosity level (i.e. "-vvv", "-v ab", and "-v 3" => verbosity level 3).

Copyright

BSD-3 License, reproduced below.

Copyright (c) 2013, The Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the University of Western Ontario nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Author

Coby Viner - conceptual design and implementation
Stephanie N. Dorman - conceptual design
Peter K. Rogan - conceptual design and initial idea for the program

Version

0.4

Date

March, 2014

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

[bgzf_add_eof](#) 9

Chapter 3

Class Index

3.1 Class List

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

[ValidationReasons](#) 11

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

/home/cviner2/RNA-seq/bgzf_add_eof.py	33
/home/cviner2/RNA-seq/Veridical.cpp	33
/home/cviner2/RNA-seq/VeridicalTrial.cpp	45

Chapter 5

Namespace Documentation

5.1 bgzf_add_eof Namespace Reference

Functions

- def [sys_exit](#)
- def [fix_bam](#)

5.1.1 Function Documentation

5.1.1.1 def `bgzf_add_eof.fix_bam (filename)`

5.1.1.2 def `bgzf_add_eof.sys_exit (msg, return_code = 1)`

Chapter 6

Class Documentation

6.1 ValidationReasons Class Reference

Public Member Functions

- [ValidationReasons](#) ()
Default constructor.
- void [addVarContValidationReason](#) (const std::string file, const [EvidenceType](#) e, const [SplicingConsequence](#) c, const double readFraction)
Add a validation reasons for a variant to the set of variant-containing files.
- void [addNonVarContValReason](#) (const std::string file, const [EvidenceType](#) e, const [SplicingConsequence](#) c, const double readFraction)
Add a validation reasons for a variant to the set of non-variant containing files.
- void [addVarContValidationReasons](#) (const std::string file, const [ValidationReasonsToAddType](#) reasonsToAdd, const [ReadFractionsType](#) readFractions)
Add multiple validation reasons for a variant to the set of variant-containing files.
- void [addNonVarContValReasons](#) (const std::string file, const [ValidationReasonsToAddType](#) reasonsToAdd, const [ReadFractionsType](#) readFractions)
Add multiple validation reasons for a variant to the set of non-variant containing files.
- double [getVarContCrypticSiteUseValScore](#) ()
- double [getVarContExonSkippingValScore](#) ()
- double [getVarContIntronInclValScore](#) ()
- double [getVarContIntronInclDirectNumber](#) ()
- double [getVarContIntronInclIndirectNumber](#) ()
- double [getNonVarContCrypticSiteUseValScore](#) ()
- double [getNonVarContExonSkippingValScore](#) ()
- double [getNonVarContIntronInclValScore](#) ()
- double [getNonVarContIntronInclDirectNumber](#) ()
- double [getNonVarContIntronInclIndirectNumber](#) ()
- double [getVarContDirectEvidenceValCount](#) ()
- double [getVarContIndirectEvidenceValCount](#) ()
- double [getNonVarContDirectEvidenceValCount](#) ()
- double [getNonVarContIndirectEvidenceValCount](#) ()
- int [getVarContSampleSize](#) ()
- int [getNonVarContSampleSize](#) ()
- double [getVarContOverallValCount](#) ()
- double [getNonVarContOverallValCount](#) ()
- std::string [getVarContPredominantValidationReasonName](#) ()
- [ValDetailsType](#) [getVarContDetails](#) (bool outputIndirectlyValidatedVariants, bool generatePVals, bool alwaysTransform, bool isCrypticSite)

Retrieves all validation details, for variant-containing files, in an output suitable format.

- [ValDetailsType](#) `getNonVarContDetails` (bool outputIndirectlyValidatedVariants, bool alwaysTransform, bool isCrypticSite)

Retrieves all validation details, for non-variant containing files, in an output suitable format.

- `std::vector< double >` `getZScores` ([EvidenceType](#) e, [SplicingConsequence](#) c)

Retrieves all of the z-scores for the variant-containing files using non-variant containing files, corresponding to e and c, as the background population.

- [ValidationReasons](#) ()

Default constructor.

- void `addVarContValidationReason` (const std::string file, const [EvidenceType](#) e, const [SplicingConsequence](#) c, const double readFraction)

Add a validation reasons for a variant to the set of variant-containing files.

- void `addNonVarContValReason` (const std::string file, const [EvidenceType](#) e, const [SplicingConsequence](#) c, const double readFraction)

Add a validation reasons for a variant to the set of non-variant containing files.

- void `addVarContValidationReasons` (const std::string file, const [ValidationReasonsToAddType](#) reasonsToAdd, const [ReadFractionsType](#) readFractions)

Add multiple validation reasons for a variant to the set of variant-containing files.

- void `addNonVarContValReasons` (const std::string file, const [ValidationReasonsToAddType](#) reasonsToAdd, const [ReadFractionsType](#) readFractions)

Add multiple validation reasons for a variant to the set of non-variant containing files.

- double `getVarContCrypticSiteUseValScore` ()
- double `getVarContExonSkippingValScore` ()
- double `getVarContIntronInclValScore` ()
- double `getVarContIntronInclDirectNumber` ()
- double `getVarContIntronInclIndirectNumber` ()
- double `getNonVarContCrypticSiteUseValScore` ()
- double `getNonVarContExonSkippingValScore` ()
- double `getNonVarContIntronInclValScore` ()
- double `getNonVarContIntronInclDirectNumber` ()
- double `getNonVarContIntronInclIndirectNumber` ()
- double `getVarContDirectEvidenceValCount` ()
- double `getVarContIndirectEvidenceValCount` ()
- double `getNonVarContDirectEvidenceValCount` ()
- double `getNonVarContIndirectEvidenceValCount` ()
- int `getVarContSampleSize` ()
- int `getNonVarContSampleSize` ()
- double `getVarContOverallValCount` ()
- double `getNonVarContOverallValCount` ()
- std::string `getVarContPredominantValidationReasonName` ()
- [ValDetailsType](#) `getVarContDetails` (bool outputIndirectlyValidatedVariants, bool generatePVals, bool alwaysTransform)

Retrieves all validation details, for variant-containing files, in an output suitable format.

- [ValDetailsType](#) `getNonVarContDetails` (bool outputIndirectlyValidatedVariants, bool alwaysTransform)

Retrieves all validation details, for non-variant containing files, in an output suitable format.

- `std::vector< double >` `getZScores` ([EvidenceType](#) e, [SplicingConsequence](#) c)

Retrieves all of the z-scores for the variant-containing files using non-variant containing files, corresponding to e and c, as the background population.

Static Public Member Functions

- static double [getPValue](#) (double z)
Retrieves the p-value, corresponding to a one-sided z-test, for x using normal samples of `map` corresponding to e and c as the background population.
- static double [getPValue](#) (double z)
Retrieves the p-value, corresponding to a one-sided z-test, for x using normal samples of `map` corresponding to e and c as the background population.

Private Types

- typedef std::map< double, [SplicingConsequence](#) > [SortedValReasonsMapType](#)
Data type for storage of splicing consequences, ordered by thier abundance, in order to facilitate the determination of the primary validation reason.
- typedef std::pair < [EvidenceType](#), [SplicingConsequence](#) > [InnerKeyType](#)
Data type for storage of associated validation information, the [EvidenceType](#) and [SplicingConsequence](#), for a specific validation instance.
- typedef boost::unordered_map < [InnerKeyType](#), double > [ValidatedInnerMapType](#)
Data type to associate a validation instance, defined by [InnerKeyType](#), with a read fraction and to aggregate events of this nature.
- typedef boost::unordered_map < std::string, [ValidatedInnerMapType](#) > [ValidatedMapType](#)
Data type to associate validation instances, defined by [ValidatedInnerMapType](#), to the files in which they occur.
- typedef std::map< double, [SplicingConsequence](#) > [SortedValReasonsMapType](#)
Data type for storage of splicing consequences, ordered by thier abundance, in order to facilitate the determination of the primary validation reason.
- typedef std::pair < [EvidenceType](#), [SplicingConsequence](#) > [InnerKeyType](#)
Data type for storage of associated validation information, the [EvidenceType](#) and [SplicingConsequence](#), for a specific validation instance.
- typedef boost::unordered_map < [InnerKeyType](#), double > [ValidatedInnerMapType](#)
Data type to associate a validation instance, defined by [InnerKeyType](#), with a read fraction and to aggregate events of this nature.
- typedef boost::unordered_map < std::string, [ValidatedInnerMapType](#) > [ValidatedMapType](#)
Data type to associate validation instances, defined by [ValidatedInnerMapType](#), to the files in which they occur.

Private Member Functions

- bool [isStronglyCorroborating](#) ([EvidenceType](#) e, [SplicingConsequence](#) sc, bool isCrypticSite)
Determine if the given splicing consequences of a specific evidence type are strongly corroborating.
- [SplicingConsequence](#) [getPredominantValidationReason](#) ()
Use a [SortedValReasonsMapType](#) to determine the main reason that the variant was validated.
- void [addValidationReasons](#) ([ValidatedMapType](#) &map, const std::string file, const [ValidationReasonsToAddType](#) reasonsToAdd, const [ReadFractionsType](#) readFractions)

- Add multiple validation reasons for a variant.*

 - void `addValidationReason` (`ValidatedMapType` &map, const std::string file, const `EvidenceType` e, const `SplicingConsequence` c, const double readFraction)
- Add a validation reasons for a variant.*

 - double `getValScore` (`ValidatedMapType` &map, `SplicingConsequence` c)

Gets the score associated with all validated instances of c in map.
- double `getValNumber` (`ValidatedMapType` &map, `EvidenceType` e, `SplicingConsequence` c, `SampleType` s)

Retrieves the number of validation events matching the given criteria.
- double `getValNumber` (`ValidatedMapType` &map, `EvidenceType` e, `SplicingConsequence` c)

Retrieves the number of validation events matching the given criteria.
- double `getCrypticValNumber` (`ValidatedMapType` &map, `EvidenceType` e, `SampleType` s)

Retrieves the number of cryptic site usage read fractions.
- double `getAllSpliceConsequencesCount` (`ValidatedMapType` &map, `EvidenceType` e)

Retrieves the sum of all read fractions of the given EvidenceType within map.
- double `getZScore` (double x, `ValidatedMapType` &map, `EvidenceType` e, `SplicingConsequence` c, bool norm-Only)

Retrieves the z-score for x of map corresponding to e and c as the background population.
- void `addSummaryValDetails` (std::stringstream &summary, `ValidatedMapType` &map, `EvidenceType` e, bool suppressNormalOutput)

Adds to the summary of validation details for all applicable sample types.
- `ValDetailsType` `getValidationDetails` (`ValidatedMapType` &map, std::string sampleName, bool suppressNormalOutput, bool outputIndirectlyValidatedVariants, bool outputZScores, bool outputPVals, bool alwaysTransform, bool isCrypticSite)

Retrieves all validation details, in an output suitable format.
- `SplicingConsequence` `getPredominantValidationReason` ()

Use a SortedValReasonsMapType to determine the main reason that the variant was validated.
- void `addValidationReasons` (`ValidatedMapType` &map, const std::string file, const `ValidationReasonsToAddType` reasonsToAdd, const `ReadFractionsType` readFractions)

Add multiple validation reasons for a variant.
- void `addValidationReason` (`ValidatedMapType` &map, const std::string file, const `EvidenceType` e, const `SplicingConsequence` c, const double readFraction)

Add a validation reasons for a variant.
- double `getValScore` (`ValidatedMapType` &map, `SplicingConsequence` c)

Gets the score associated with all validated instances of c in map.
- double `getValNumber` (`ValidatedMapType` &map, `EvidenceType` e, `SplicingConsequence` c, `SampleType` s)

Retrieves the number of validation events matching the given criteria.
- double `getValNumber` (`ValidatedMapType` &map, `EvidenceType` e, `SplicingConsequence` c)

Retrieves the number of validation events matching the given criteria.
- double `getCrypticValNumber` (`ValidatedMapType` &map, `EvidenceType` e, `SampleType` s)

Retrieves the number of cryptic site usage read fractions.
- double `getAllSpliceConsequencesCount` (`ValidatedMapType` &map, `EvidenceType` e)

Retrieves the sum of all read fractions of the given EvidenceType within map.
- double `getZScore` (double x, `ValidatedMapType` &map, `EvidenceType` e, `SplicingConsequence` c, bool norm-Only)

Retrieves the z-score for x of map corresponding to e and c as the background population.
- void `addSummaryValDetails` (std::stringstream &summary, `ValidatedMapType` &map, `EvidenceType` e, bool suppressNormalOutput)

Adds to the summary of validation details for all applicable sample types.
- `ValDetailsType` `getValidationDetails` (`ValidatedMapType` &map, std::string sampleName, bool suppressNormalOutput, bool outputIndirectlyValidatedVariants, bool outputZScores, bool outputPVals, bool alwaysTransform)

Retrieves all validation details, in an output suitable format.

Static Private Member Functions

- static double [addToTotal](#) (double sum, const ValidatedMapType::value_type &data)
Binary operation for use by accumulators to sum the values of a [ValidatedMapType](#).
- static double [YJTransform](#) (double x, double lambda)
Computes the Yeo-Johnson transformation for x , given parameter $\lambda = lambda$.
- static double [op_YJTransform](#) (double x)
Operator for [YJTransform\(\)](#).
- static double [addToTotal](#) (double sum, const ValidatedMapType::value_type &data)
Binary operation for use by accumulators to sum the values of a [ValidatedMapType](#).
- static double [YJTransform](#) (double x, double lambda)
Computes the Yeo-Johnson transformation for x , given parameter $\lambda = lambda$.
- static double [op_YJTransform](#) (double x)
Operator for [YJTransform\(\)](#).

Private Attributes

- [ValidatedMapType varContValMap](#)
Stores validation events occurring within variant-containing files (i.e. evidence corroborating the variant)
- [ValidatedMapType nonVarContValMap](#)
Stores validation events occurring within non-variant containing (control) files (i.e. evidence that the variant may not be causative)

6.1.1 Member Typedef Documentation

6.1.1.1 `typedef std::pair<EvidenceType, SplicingConsequence> ValidationReasons::InnerKeyType`
[private]

Data type for storage of associated validation information, the [EvidenceType](#) and [SplicingConsequence](#), for a specific validation instance.

6.1.1.2 `typedef std::pair<EvidenceType, SplicingConsequence> ValidationReasons::InnerKeyType`
[private]

Data type for storage of associated validation information, the [EvidenceType](#) and [SplicingConsequence](#), for a specific validation instance.

6.1.1.3 `typedef std::map<double, SplicingConsequence> ValidationReasons::SortedValReasonsMapType`
[private]

Data type for storage of splicing consequences, ordered by their abundance, in order to facilitate the determination of the primary validation reason.

6.1.1.4 `typedef std::map<double, SplicingConsequence> ValidationReasons::SortedValReasonsMapType`
[private]

Data type for storage of splicing consequences, ordered by their abundance, in order to facilitate the determination of the primary validation reason.

6.1.1.5 `typedef boost::unordered_map<InnerKeyType, double> ValidationReasons::ValidatedInnerMapType`
`[private]`

Data type to associate a validation instance, defined by [InnerKeyType](#), with a read fraction and to aggregate events of this nature.

6.1.1.6 `typedef boost::unordered_map<InnerKeyType, double> ValidationReasons::ValidatedInnerMapType`
`[private]`

Data type to associate a validation instance, defined by [InnerKeyType](#), with a read fraction and to aggregate events of this nature.

6.1.1.7 `typedef boost::unordered_map<std::string, ValidatedInnerMapType> ValidationReasons::ValidatedMap-`
`Type [private]`

Data type to associate validation instances, defined by [ValidatedInnerMapType](#), to the files in which they occur.

6.1.1.8 `typedef boost::unordered_map<std::string, ValidatedInnerMapType> ValidationReasons::ValidatedMap-`
`Type [private]`

Data type to associate validation instances, defined by [ValidatedInnerMapType](#), to the files in which they occur.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `ValidationReasons::ValidationReasons () [inline]`

Default constructor.

6.1.2.2 `ValidationReasons::ValidationReasons () [inline]`

Default constructor.

6.1.3 Member Function Documentation

6.1.3.1 `void ValidationReasons::addNonVarContValReason (const std::string file, const EvidenceType e, const SplicingConsequence c, const double readFraction) [inline]`

Add a validation reasons for a variant to the set of non-variant containing files.

6.1.3.2 `void ValidationReasons::addNonVarContValReason (const std::string file, const EvidenceType e, const SplicingConsequence c, const double readFraction) [inline]`

Add a validation reasons for a variant to the set of non-variant containing files.

6.1.3.3 `void ValidationReasons::addNonVarContValReasons (const std::string file, const ValidationReasonsToAddType reasonsToAdd, const ReadFractionsType readFractions) [inline]`

Add multiple validation reasons for a variant to the set of non-variant containing files.

6.1.3.4 void ValidationReasons::addNonVarContValReasons (const std::string *file*, const ValidationReasonsToAddType *reasonsToAdd*, const ReadFractionsType *readFractions*) [inline]

Add multiple validation reasons for a variant to the set of non-variant containing files.

6.1.3.5 void ValidationReasons::addSummaryValDetails (std::stringstream & *summary*, ValidatedMapType & *map*, EvidenceType *e*, bool *suppressNormalOutput*) [inline],[private]

Adds to the summary of validation details for all applicable sample types.

Parameters

<i>summary</i>	The summary object to add to
<i>map</i>	The map to retrieve from
<i>e</i>	The evidence type of interest
<i>suppress-NormalOutput</i>	If set to true, prevents output of validations in files which are NORMAL . Used to output the experimental sample(s).

6.1.3.6 void ValidationReasons::addSummaryValDetails (std::stringstream & *summary*, ValidatedMapType & *map*, EvidenceType *e*, bool *suppressNormalOutput*) [inline],[private]

Adds to the summary of validation details for all applicable sample types.

Parameters

<i>summary</i>	The summary object to add to
<i>map</i>	The map to retrieve from
<i>e</i>	The evidence type of interest
<i>suppress-NormalOutput</i>	If set to true, prevents output of validations in files which are NORMAL . Used to output the experimental sample(s).

6.1.3.7 static double ValidationReasons::addToTotal (double *sum*, const ValidatedMapType::value_type & *data*) [inline],[static],[private]

Binary operation for use by accumulators to sum the values of a [ValidatedMapType](#).

Sums across all constituent values of the underlying [ValidatedInnerMapType](#).

Parameters

<i>sum</i>	The running sum of values
<i>data</i>	The map whose values are summed

Returns

The sum of *sum* and all substituent values of the underlying [ValidatedInnerMapType](#) of *data*.

6.1.3.8 static double ValidationReasons::addToTotal (double *sum*, const ValidatedMapType::value_type & *data*) [inline],[static],[private]

Binary operation for use by accumulators to sum the values of a [ValidatedMapType](#).

Sums across all constituent values of the underlying [ValidatedInnerMapType](#).

Parameters

<i>sum</i>	The running sum of values
<i>data</i>	The map whose values are summed

Returns

The sum of *sum* and all substituent values of the underlying [ValidatedInnerMapType](#) of *data*.

6.1.3.9 `void ValidationReasons::addValidationReason (ValidatedMapType & map, const std::string file, const EvidenceType e, const SplicingConsequence c, const double readFraction) [inline],[private]`

Add a validation reasons for a variant.

Parameters

<i>map</i>	The map to which the validated read fraction information should be added
<i>file</i>	The file containing the validating read
<i>e</i>	The evidence type associated with this validation event
<i>c</i>	The splicing consequence associated with this validation event
<i>readFraction</i>	The fraction of the read which was validated

6.1.3.10 `void ValidationReasons::addValidationReason (ValidatedMapType & map, const std::string file, const EvidenceType e, const SplicingConsequence c, const double readFraction) [inline],[private]`

Add a validation reasons for a variant.

Parameters

<i>map</i>	The map to which the validated read fraction information should be added
<i>file</i>	The file containing the validating read
<i>e</i>	The evidence type associated with this validation event
<i>c</i>	The splicing consequence associated with this validation event
<i>readFraction</i>	The fraction of the read which was validated

6.1.3.11 `void ValidationReasons::addValidationReasons (ValidatedMapType & map, const std::string file, const ValidationReasonsToAddType reasonsToAdd, const ReadFractionsType readFractions) [inline],[private]`

Add multiple validation reasons for a variant.

Parameters

<i>map</i>	The structure to which the validated read fraction information should be added
<i>file</i>	The file containing the validating read
<i>reasonsToAdd</i>	The information pertaining to the validated read fraction (N of them)
<i>readFractions</i>	The fraction of reads pertaining to each reason to add (N of them)

6.1.3.12 `void ValidationReasons::addValidationReasons (ValidatedMapType & map, const std::string file, const ValidationReasonsToAddType reasonsToAdd, const ReadFractionsType readFractions) [inline],[private]`

Add multiple validation reasons for a variant.

Parameters

<i>map</i>	The structure to which the validated read fraction information should be added
<i>file</i>	The file containing the validating read
<i>reasonsToAdd</i>	The information pertaining to the validated read fraction (N of them)
<i>readFractions</i>	The fraction of reads pertaining to each reason to add (N of them)

6.1.3.13 `void ValidationReasons::addVarContValidationReason (const std::string file, const EvidenceType e, const SplicingConsequence c, const double readFraction) [inline]`

Add a validation reasons for a variant to the set of variant-containing files.

6.1.3.14 `void ValidationReasons::addVarContValidationReason (const std::string file, const EvidenceType e, const SplicingConsequence c, const double readFraction) [inline]`

Add a validation reasons for a variant to the set of variant-containing files.

6.1.3.15 `void ValidationReasons::addVarContValidationReasons (const std::string file, const ValidationReasonsToAddType reasonsToAdd, const ReadFractionsType readFractions) [inline]`

Add multiple validation reasons for a variant to the set of variant-containing files.

6.1.3.16 `void ValidationReasons::addVarContValidationReasons (const std::string file, const ValidationReasonsToAddType reasonsToAdd, const ReadFractionsType readFractions) [inline]`

Add multiple validation reasons for a variant to the set of variant-containing files.

6.1.3.17 `double ValidationReasons::getAllSpliceConsequencesCount (ValidatedMapType & map, EvidenceType e) [inline], [private]`

Retrieves the sum of all read fractions of the given [EvidenceType](#) within `map`.

Parameters

<i>map</i>	The map to retrieve from
<i>e</i>	The evidence type of interest

Returns

The sum of read fractions contained within `map`, matching `e`.

6.1.3.18 `double ValidationReasons::getAllSpliceConsequencesCount (ValidatedMapType & map, EvidenceType e) [inline], [private]`

Retrieves the sum of all read fractions of the given [EvidenceType](#) within `map`.

Parameters

<i>map</i>	The map to retrieve from
<i>e</i>	The evidence type of interest

Returns

The sum of read fractions contained within `map`, matching `e`.

6.1.3.19 `double ValidationReasons::getCrypticValNumber (ValidatedMapType & map, EvidenceType e, SampleType s) [inline],[private]`

Retrieves the number of cryptic site usage read fractions.

Parameters

<i>map</i>	The map to retrieve from
<i>e</i>	The evidence type of interest
<i>s</i>	The sample type of interest

Returns

The sum of read fractions contained within `map`, matching `s`, that are `CRYPTIC_SITE_USE` if there is no anti-cryptic site use, otherwise the sum of `CRYPTIC_SITE_USE` divided by the sum of `ANTI_CRYPTIC_SITE_USE`.

6.1.3.20 `double ValidationReasons::getCrypticValNumber (ValidatedMapType & map, EvidenceType e, SampleType s) [inline],[private]`

Retrieves the number of cryptic site usage read fractions.

Parameters

<i>map</i>	The map to retrieve from
<i>e</i>	The evidence type of interest
<i>s</i>	The sample type of interest

Returns

The sum of read fractions contained within `map`, matching `s`, that are `CRYPTIC_SITE_USE` if there is no anti-cryptic site use, otherwise the sum of `CRYPTIC_SITE_USE` divided by the sum of `ANTI_CRYPTIC_SITE_USE`.

6.1.3.21 `double ValidationReasons::getNonVarContCrypticSiteUseValScore () [inline]`

Returns

Score of validating reads retrieved by `getValScore()` within non-variant containing files validated due to `CRYPTIC_SITE_USE`.

6.1.3.22 `double ValidationReasons::getNonVarContCrypticSiteUseValScore () [inline]`

Returns

Score of validating reads retrieved by `getValScore()` within non-variant containing files validated due to `CRYPTIC_SITE_USE`.

6.1.3.23 `ValDetailsType ValidationReasons::getNonVarContDetails (bool outputIndirectlyValidatedVariants, bool alwaysTransform) [inline]`

Retrieves all validation details, for non-variant containing files, in an output suitable format.

Includes normal sample data.

6.1.3.24 `ValDetailsType ValidationReasons::getNonVarContDetails (bool outputIndirectlyValidatedVariants, bool alwaysTransform, bool isCrypticSite) [inline]`

Retrieves all validation details, for non-variant containing files, in an output suitable format.

Includes normal sample data.

6.1.3.25 `double ValidationReasons::getNonVarContDirectEvidenceValCount () [inline]`

Returns

Number of EvidenceType::DIRECT validating reads retrieved by [getAllSpliceConsequencesCount\(\)](#) within non-variant containing files.

6.1.3.26 `double ValidationReasons::getNonVarContDirectEvidenceValCount () [inline]`

Returns

Number of EvidenceType::DIRECT validating reads retrieved by [getAllSpliceConsequencesCount\(\)](#) within non-variant containing files.

6.1.3.27 `double ValidationReasons::getNonVarContExonSkippingValScore () [inline]`

Returns

Score of validating reads retrieved by [getValScore\(\)](#) within non-variant containing files validated due to [EXON_SKIPPING](#).

6.1.3.28 `double ValidationReasons::getNonVarContExonSkippingValScore () [inline]`

Returns

Score of validating reads retrieved by [getValScore\(\)](#) within non-variant containing files validated due to [EXON_SKIPPING](#).

6.1.3.29 `double ValidationReasons::getNonVarContIndirectEvidenceValCount () [inline]`

Returns

Number of EvidenceType::COUNT validating reads retrieved by [getAllSpliceConsequencesCount\(\)](#) within non-variant containing files.

6.1.3.30 `double ValidationReasons::getNonVarContIndirectEvidenceValCount () [inline]`

Returns

Number of EvidenceType::COUNT validating reads retrieved by [getAllSpliceConsequencesCount\(\)](#) within non-variant containing files.

6.1.3.31 `double ValidationReasons::getNonVarContIntronInclDirectNumber () [inline]`

Returns

Number of validating reads retrieved by [getValNumber\(\)](#) within non-variant containing files matching [DIRECT](#) and [INTRON_INCLUSION](#).

6.1.3.32 `double ValidationReasons::getNonVarContIntronInclDirectNumber () [inline]`

Returns

Number of validating reads retrieved by [getValNumber\(\)](#) within non-variant containing files matching [DIRECT](#) and [INTRON_INCLUSION](#).

6.1.3.33 `double ValidationReasons::getNonVarContIntronInclIndirectNumber () [inline]`

Returns

Number of validating reads retrieved by [getValNumber\(\)](#) within non-variant containing files matching [COUNT](#) and [INTRON_INCLUSION](#).

6.1.3.34 `double ValidationReasons::getNonVarContIntronInclIndirectNumber () [inline]`

Returns

Number of validating reads retrieved by [getValNumber\(\)](#) within non-variant containing files matching [COUNT](#) and [INTRON_INCLUSION](#).

6.1.3.35 `double ValidationReasons::getNonVarContIntronInclValScore () [inline]`

Returns

Score of validating reads retrieved by [getValScore\(\)](#) within non-variant containing files validated due to [INTRON_INCLUSION](#).

6.1.3.36 `double ValidationReasons::getNonVarContIntronInclValScore () [inline]`

Returns

Score of validating reads retrieved by [getValScore\(\)](#) within non-variant containing files validated due to [INTRON_INCLUSION](#).

6.1.3.37 `double ValidationReasons::getNonVarContOverallValCount () [inline]`

Returns

The total sum of validating read fractions across all non-variant containing files.

6.1.3.38 `double ValidationReasons::getNonVarContOverallValCount () [inline]`

Returns

The total sum of validating read fractions across all non-variant containing files.

6.1.3.39 `int ValidationReasons::getNonVarContSampleSize () [inline]`

Returns

The number of non-variant containing files which have validating reads.

6.1.3.40 `int ValidationReasons::getNonVarContSampleSize () [inline]`

Returns

The number of non-variant containing files which have validating reads.

6.1.3.41 `SplicingConsequence ValidationReasons::getPredominantValidationReason () [inline],[private]`

Use a [SortedValReasonsMapType](#) to determine the main reason that the variant was validated.

Returns

A [SplicingConsequence](#) indicating the main validation reason.

6.1.3.42 `SplicingConsequence ValidationReasons::getPredominantValidationReason () [inline],[private]`

Use a [SortedValReasonsMapType](#) to determine the main reason that the variant was validated.

Returns

A [SplicingConsequence](#) indicating the main validation reason.

6.1.3.43 `static double ValidationReasons::getPValue (double z) [inline],[static]`

Retrieves the p-value, corresponding to a one-sided z-test, for x using normal samples of `map` corresponding to `e` and `c` as the background population.

Parameters

<code>z</code>	The z-score to which the p-value should correspond
----------------	--

Returns

The one-sided (right-tailed) p-value corresponding to given z-score, from the standard normal CDF. This corresponds to the integral from z to Inf of the Gaussian density. Returns NaN if z is not finite.

6.1.3.44 `static double ValidationReasons::getPValue (double z) [inline],[static]`

Retrieves the p-value, corresponding to a one-sided z-test, for x using normal samples of `map` corresponding to `e` and `c` as the background population.

Parameters

<code>z</code>	The z-score to which the p-value should correspond
----------------	--

Returns

The one-sided (right-tailed) p-value corresponding to given z-score, from the standard normal CDF. This corresponds to the integral from z to Inf of the Gaussian density. Returns NaN if z is not finite.

6.1.3.45 `ValDetailsType ValidationReasons::getValidationDetails (ValidatedMapType & map, std::string sampleName, bool suppressNormalOutput, bool outputIndirectlyValidatedVariants, bool outputZScores, bool outputPVals, bool alwaysTransform) [inline],[private]`

Retrieves all validation details, in an output suitable format.

Parameters

<i>map</i>	The map to retrieve from
<i>sampleName</i>	A name for the sample type
<i>suppressNormalOutput</i>	If set to true, prevents output of validations in files which are NORMAL . Used to output the experimental sample(s).
<i>outputIndirectlyValidatedVariants</i>	Outputs information concerning indirectly validated variants if and only if this is set to <code>true</code>
<i>outputZScores</i>	If set to true, outputs z-scores and p-values via getPValue() using getZScore() as its parameter
<i>outputPVals</i>	If set to true, outputs z-scores and p-values via getZScore()
<i>alwaysTransform</i>	Specifies if the YJ transform should be applied to all data and not just for the p-value computation

Returns

A pair of a double (first) and a pair of strings (second). The double represents the minimum p-value over all evidence types for the primary validation reason, as specified by [getPredominantValidationReason\(\)](#). The first string contains an overall summary of the validations that occurred, while the second provides a detailed table quantifying validation events across each file in which they occurred.

6.1.3.46 `ValDetailsType ValidationReasons::getValidationDetails (ValidatedMapType & map, std::string sampleName, bool suppressNormalOutput, bool outputIndirectlyValidatedVariants, bool outputZScores, bool outputPVals, bool alwaysTransform, bool isCrypticSite)` [`inline`], [`private`]

Retrieves all validation details, in an output suitable format.

Parameters

<i>map</i>	The map to retrieve from
<i>sampleName</i>	A name for the sample type
<i>suppressNormalOutput</i>	If set to true, prevents output of validations in files which are NORMAL . Used to output the experimental sample(s).
<i>outputIndirectlyValidatedVariants</i>	Outputs information concerning indirectly validated variants if and only if this is set to <code>true</code>
<i>outputZScores</i>	If set to true, outputs z-scores and p-values via getPValue() using getZScore() as its parameter
<i>outputPVals</i>	If set to true, outputs z-scores and p-values via getZScore()
<i>alwaysTransform</i>	Specifies if the YJ transform should be applied to all data and not just for the p-value computation
<i>isCrypticSite</i>	True iff the variant associated with this validation event is cryptic

Returns

A pair of a double (first) and a pair of strings (second). The double represents the minimum p-value over all evidence types for the primary validation reason, as specified by [getPredominantValidationReason\(\)](#). The first string contains an overall summary of the validations that occurred, while the second provides a detailed table quantifying validation events across each file in which they occurred.

6.1.3.47 `double ValidationReasons::getValNumber (ValidatedMapType & map, EvidenceType e, SplicingConsequence c, SampleType s)` [`inline`], [`private`]

Retrieves the number of validation events matching the given criteria.

Parameters

<i>map</i>	The map to retrieve from
<i>e</i>	The evidence type of interest
<i>c</i>	The splicing consequence of interest
<i>s</i>	The sample type of interest

Returns

The sum of read fractions contained within `map`, matching all of `e`, `c`, and `s`.

6.1.3.48 `double ValidationReasons::getValNumber (ValidatedMapType & map, EvidenceType e, SplicingConsequence c, SampleType s)` `[inline], [private]`

Retrieves the number of validation events matching the given criteria.

Parameters

<i>map</i>	The map to retrieve from
<i>e</i>	The evidence type of interest
<i>c</i>	The splicing consequence of interest
<i>s</i>	The sample type of interest

Returns

The sum of read fractions contained within `map`, matching all of `e`, `c`, and `s`.

6.1.3.49 `double ValidationReasons::getValNumber (ValidatedMapType & map, EvidenceType e, SplicingConsequence c)` `[inline], [private]`

Retrieves the number of validation events matching the given criteria.

6.1.3.50 `double ValidationReasons::getValNumber (ValidatedMapType & map, EvidenceType e, SplicingConsequence c)` `[inline], [private]`

Retrieves the number of validation events matching the given criteria.

6.1.3.51 `double ValidationReasons::getValScore (ValidatedMapType & map, SplicingConsequence c)` `[inline], [private]`

Gets the score associated with all validated instances of `c` in `map`.

Parameters

<i>map</i>	The map from which to check for validated instances of <code>c</code>
<i>c</i>	The splicing consequence of interest

Returns

A score, reflective of the abundance and weight of `in map`. Computed by multiplying relevant entries by [directCoeff](#) or [countCoeff](#).

6.1.3.52 `double ValidationReasons::getValScore (ValidatedMapType & map, SplicingConsequence c)` `[inline], [private]`

Gets the score associated with all validated instances of `c` in `map`.

Parameters

<i>map</i>	The map from which to check for validated instances of <i>c</i>
<i>c</i>	The splicing consequence of interest

Returns

A score, reflective of the abundance and weight of `in map`. Computed by multiplying relevant entries by [directCoeff](#) or [countCoeff](#).

6.1.3.53 `double ValidationReasons::getVarContCrypticSiteUseValScore ()` [\[inline\]](#)

Returns

Score of validating reads retrieved by [getValScore\(\)](#) within variant-containing files validated due to [CRYPTIC_SITE_USE](#).

6.1.3.54 `double ValidationReasons::getVarContCrypticSiteUseValScore ()` [\[inline\]](#)

Returns

Score of validating reads retrieved by [getValScore\(\)](#) within variant-containing files validated due to [CRYPTIC_SITE_USE](#).

6.1.3.55 `ValDetailsType ValidationReasons::getVarContDetails (bool outputIndirectlyValidatedVariants, bool generatePVals, bool alwaysTransform)` [\[inline\]](#)

Retrieves all validation details, for variant-containing files, in an output suitable format.

Includes p-values for each event, as compared with normal sample data (refer to [getPValue\(\)](#) and [getZScore\(\)](#) for further details).

6.1.3.56 `ValDetailsType ValidationReasons::getVarContDetails (bool outputIndirectlyValidatedVariants, bool generatePVals, bool alwaysTransform, bool isCrypticSite)` [\[inline\]](#)

Retrieves all validation details, for variant-containing files, in an output suitable format.

Includes p-values for each event, as compared with normal sample data (refer to [getPValue\(\)](#) and [getZScore\(\)](#) for further details).

6.1.3.57 `double ValidationReasons::getVarContDirectEvidenceValCount ()` [\[inline\]](#)

Returns

Number of `EvidenceType::DIRECT` validating reads retrieved by [getAllSpliceConsequencesCount\(\)](#) within variant-containing files.

6.1.3.58 `double ValidationReasons::getVarContDirectEvidenceValCount ()` [\[inline\]](#)

Returns

Number of `EvidenceType::DIRECT` validating reads retrieved by [getAllSpliceConsequencesCount\(\)](#) within variant-containing files.

6.1.3.59 `double ValidationReasons::getVarContExonSkippingValScore () [inline]`

Returns

Score of validating reads retrieved by [getValScore\(\)](#) within variant-containing files validated due to [EXON_SKIPPING](#).

6.1.3.60 `double ValidationReasons::getVarContExonSkippingValScore () [inline]`

Returns

Score of validating reads retrieved by [getValScore\(\)](#) within variant-containing files validated due to [EXON_SKIPPING](#).

6.1.3.61 `double ValidationReasons::getVarContIndirectEvidenceValCount () [inline]`

Returns

Number of `EvidenceType::COUNT` validating reads retrieved by [getAllSpliceConsequencesCount\(\)](#) within variant-containing files.

6.1.3.62 `double ValidationReasons::getVarContIndirectEvidenceValCount () [inline]`

Returns

Number of `EvidenceType::COUNT` validating reads retrieved by [getAllSpliceConsequencesCount\(\)](#) within variant-containing files.

6.1.3.63 `double ValidationReasons::getVarContIntronInclDirectNumber () [inline]`

Returns

Number of validating reads retrieved by [getValNumber\(\)](#) within variant-containing files matching [DIRECT](#) and [INTRON_INCLUSION](#).

6.1.3.64 `double ValidationReasons::getVarContIntronInclDirectNumber () [inline]`

Returns

Number of validating reads retrieved by [getValNumber\(\)](#) within variant-containing files matching [DIRECT](#) and [INTRON_INCLUSION](#).

6.1.3.65 `double ValidationReasons::getVarContIntronInclIndirectNumber () [inline]`

Returns

Number of validating reads retrieved by [getValNumber\(\)](#) within variant-containing files matching [COUNT](#) and [INTRON_INCLUSION](#).

6.1.3.66 `double ValidationReasons::getVarContIntronInclIndirectNumber () [inline]`

Returns

Number of validating reads retrieved by [getValNumber\(\)](#) within variant-containing files matching [COUNT](#) and [INTRON_INCLUSION](#).

6.1.3.67 `double ValidationReasons::getVarContIntronInclValScore () [inline]`

Returns

Score of validating reads retrieved by `getValScore()` within variant-containing files validated due to [INTRON_INCLUSION](#).

6.1.3.68 `double ValidationReasons::getVarContIntronInclValScore () [inline]`

Returns

Score of validating reads retrieved by `getValScore()` within variant-containing files validated due to [INTRON_INCLUSION](#).

6.1.3.69 `double ValidationReasons::getVarContOverallValCount () [inline]`

Returns

The total sum of validating read fractions across all variant-containing files.

6.1.3.70 `double ValidationReasons::getVarContOverallValCount () [inline]`

Returns

The total sum of validating read fractions across all variant-containing files.

6.1.3.71 `std::string ValidationReasons::getVarContPredominantValidationReasonName () [inline]`

Returns

The name of the predominant validation reason for this variant.

6.1.3.72 `std::string ValidationReasons::getVarContPredominantValidationReasonName () [inline]`

Returns

The name of the predominant validation reason for this variant.

6.1.3.73 `int ValidationReasons::getVarContSampleSize () [inline]`

Returns

The number of variant-containing files which have validating reads.

6.1.3.74 `int ValidationReasons::getVarContSampleSize () [inline]`

Returns

The number of variant-containing files which have validating reads.

6.1.3.75 `double ValidationReasons::getZScore (double x, ValidatedMapType & map, EvidenceType e, SplicingConsequence c, bool normOnly) [inline], [private]`

Retrieves the z-score for `x` of `map` corresponding to `e` and `c` as the background population.

Applies the Yeo-Johnson power transformation, with $\lambda = \text{lambda}$ ([YJTransform](#)). `lambda` can be set to 1, returning the identity transformation, thereby computing the z-score on the raw data alone

Parameters

<code>x</code>	The raw score whose z-score we wish to compute.
<code>map</code>	The map to retrieve from
<code>e</code>	The evidence type of interest
<code>c</code>	The splicing consequence of interest
<code>normOnly</code>	Restrict computation to normal samples only

Returns

The z-score $((x - \text{mean})/\text{SD})$ of `x` WRT all samples of type `e` and `c` or the maximum double value possible if there are no NVC samples.

6.1.3.76 `double ValidationReasons::getZScore (double x, ValidatedMapType & map, EvidenceType e, SplicingConsequence c, bool normOnly) [inline], [private]`

Retrieves the z-score for `x` of `map` corresponding to `e` and `c` as the background population.

Applies the Yeo-Johnson power transformation, with $\lambda = \text{lambda}$ ([YJTransform](#)). `lambda` can be set to 1, returning the identity transformation, thereby computing the z-score on the raw data alone

Parameters

<code>x</code>	The raw score whose z-score we wish to compute.
<code>map</code>	The map to retrieve from
<code>e</code>	The evidence type of interest
<code>c</code>	The splicing consequence of interest
<code>normOnly</code>	Restrict computation to normal samples only

Returns

The z-score $((x - \text{mean})/\text{SD})$ of `x` WRT all samples of type `e` and `c` or the maximum double value possible if there are no NVC samples.

6.1.3.77 `std::vector<double> ValidationReasons::getZScores (EvidenceType e, SplicingConsequence c) [inline]`

Retrieves all of the z-scores for the variant-containing files using non-variant containing files, corresponding to `e` and `c`, as the background population.

All data is transformed via the Yeo-Johnson Transformation [3] to ensure greater normality. See [getZScore\(\)](#) for details.

Parameters

<code>e</code>	The evidence type of interest
<code>c</code>	The splicing consequence of interest

Returns

A vector of z-scores $((x - \text{mean}) / \text{SD})$ containing a z-score for each variant-containing file WRT all normal samples of type `e` and `c`.

6.1.3.78 `std::vector<double> ValidationReasons::getZScores (EvidenceType e, SplicingConsequence c)`
`[inline]`

Retrieves all of the z-scores for the variant-containing files using non-variant containing files, corresponding to `e` and `c`, as the background population.

All data is transformed via the Yeo-Johnson Transformation [3] to ensure greater normality. See [getZScore\(\)](#) for details.

Parameters

<code>e</code>	The evidence type of interest
<code>c</code>	The splicing consequence of interest

Returns

A vector of z-scores $((x - \text{mean}) / \text{SD})$ containing a z-score for each variant-containing file WRT all normal samples of type `e` and `c`.

6.1.3.79 `bool ValidationReasons::isStronglyCorroborating (EvidenceType e, SplicingConsequence sc, bool isCrypticSite)`
`[inline], [private]`

Determine if the given splicing consequences of a specific evidence type are strongly corroborating.

The following are strongly corroborating (JS \leq DIRECT, RA \leq COUNT): JS CS, JS ES, JS II, and RA II. While intron inclusion with mutation is strongly corroborating, it is not an independent consideration, as it must be present with significant II to be relevant. We do not currently automatically integrate its significance into our overall considerations.

Parameters

<code>e</code>	The evidence type associated with this validation event
<code>sc</code>	The splicing consequence associated with this validation event
<code>isCrypticSite</code>	True iff the variant associated with this validation event is cryptic

Returns

True iff this validation event has one of: JS CS, JS ES, JS II, or RA II.

6.1.3.80 `static double ValidationReasons::op_YJTransform (double x)`
`[inline], [static], [private]`

Operator for [YJTransform\(\)](#).

Parameters

<code>x</code>	The value to transform
----------------	------------------------

Returns

The Yeo-Johnson transformation [3] for `x`, given $\lambda: \psi(\lambda, x)$.

6.1.3.81 `static double ValidationReasons::op_YJTransform (double x)`
`[inline], [static], [private]`

Operator for [YJTransform\(\)](#).

Parameters

<i>x</i>	The value to transform
----------	------------------------

Returns

The Yeo-Johnson transformation [3] for x , given λ : $\psi(\lambda, x)$

6.1.3.82 `static double ValidationReasons::YJTransform (double x, double lambda) [inline],[static], [private]`

Computes the Yeo-Johnson transformation for x , given parameter $\lambda = \text{lambda}$.

Uses equation (2•1) within: [3].

Parameters

<i>x</i>	The value to transform
<i>lambda</i>	Parameter affecting the tranfrotation. A value below zero will skew towards the right and greater than zero, towards the left. A value of one represents the identity transformation.

Returns

The Yeo-Johnson transformation [3] for x , given λ : $\psi(\lambda, x)$.

6.1.3.83 `static double ValidationReasons::YJTransform (double x, double lambda) [inline],[static], [private]`

Computes the Yeo-Johnson transformation for x , given parameter $\lambda = \text{lambda}$.

Uses equation (2•1) within: [3].

Parameters

<i>x</i>	The value to transform
<i>lambda</i>	Parameter affecting the tranfrotation. A value below zero will skew towards the right and greater than zero, towards the left. A value of one represents the identity transformation.

Returns

The Yeo-Johnson transformation [3] for x , given λ : $\psi(\lambda, x)$.

6.1.4 Member Data Documentation

6.1.4.1 `ValidatedMapType ValidationReasons::nonVarContValMap [private]`

Stores validation events occuring within non-variant containing (control) files (i.e. evidence that the variant may not be causitive)

6.1.4.2 `ValidatedMapType ValidationReasons::varContValMap [private]`

Stores validation events occuring within variant-containing files (i.e. evidence corroborating the varaint)

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

- [/home/cviner2/RNA-seq/Veridical.cpp](#)
- [/home/cviner2/RNA-seq/VeridicalTrial.cpp](#)

Chapter 7

File Documentation

7.1 /home/cviner2/RNA-seq/bgzf_add_eof.py File Reference

Namespaces

- [bgzf_add_eof](#)

Functions

- [def bgzf_add_eof.sys_exit](#)
- [def bgzf_add_eof.fix_bam](#)

7.2 /home/cviner2/RNA-seq/Veridical.cpp File Reference

```
#include <cstdlib>
```

```

#include <cmath>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <map>
#include <limits>
#include <algorithm>
#include <numeric>
#include <iterator>
#include <utility>
#include <ctime>
#include <stdexcept>
#include <boost/program_options.hpp>
#include <boost/array.hpp>
#include <boost/lexical_cast.hpp>
#include <boost/foreach.hpp>
#include <boost/tokenizer.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/algorithm/string/find.hpp>
#include <boost/algorithm/string/erase.hpp>
#include <boost/bind.hpp>
#include <boost/unordered_map.hpp>
#include <boost/swap.hpp>
#include <boost/assign/list_of.hpp>
#include <boost/filesystem.hpp>
#include <boost/regex.hpp>
#include <boost/iostreams/tee.hpp>
#include <boost/iostreams/stream.hpp>
#include <boost/math/distributions/normal.hpp>
#include <boost/assert.hpp>
#include "api/BamMultiReader.h"
#include "api/BamAlignment.h"

```

Classes

- class [ValidationReasons](#)

Typedefs

- typedef std::vector< std::pair
< [EvidenceType](#),
[SplicingConsequence](#) > > [ValidationReasonsToAddType](#)
Data type for storage for each validated read to be added as a group to [ValidationReasons](#) (or portions thereof) and its associated validation information.
- typedef std::vector< double > [ReadFractionsType](#)
Data type for storage of read fractions, used to add validated reads to [ValidationReasons](#). Stores different fractions of a read (which need not sum to unity), as having validated for various reasons.
- typedef boost::unordered_map
< std::string, [SampleType](#) > [BAMFileListType](#)
Data type for storage of paths (key) to RNA-Seq BAM files and their associated [SampleType](#).
- typedef std::pair< double,
std::pair< std::string,
std::string > > [ValDetailsType](#)

Data type including a p-value and two strings of different levels of verbosity. Used to pass details of a validated variant.

- typedef

```
boost::unordered_multimap
< std::string, std::vector
< std::string > > MultiExonMapType
```

Data type for the storage of all exome data (from all transcript variants) of the exons of a given gene (key)

- typedef std::map< std::pair

```
< std::string, int >
, std::vector< std::string > > ExonMapType
```

Data type for the storage of exons to be used in discriminating between transcript variants.

Output Modulators

BOOST Tee types to facilitate multiple output redirection.

- typedef

```
boost::iostreams::tee_device
< std::ostream, std::ofstream > TeeDevice
```

- typedef

```
boost::iostreams::stream
< TeeDevice > TeeStream
```

Enumerations

- enum SampleType {
TUMOUR, NORMAL, _STcardinality, ANY,
TUMOUR, NORMAL, _STcardinality, ANY }

Enumeration describing the sample type of a given BAM file.

- enum Directionality { RIGHT, LEFT, RIGHT, LEFT }

The directionality of a given variant.

- enum EvidenceType {
DIRECT, COUNT, _ETcardinality, DIRECT,
COUNT, _ETcardinality }

The type of evidence used to validate a variant.

- enum SplicingConsequence {
NONE, CRYPTIC_SITE_USE, ANTI_CRYPTIC_SITE_USE, EXON_SKIPPING,
INTRON_INCLUSION, INTRON_INCLUSION_WITH_MUT, _SCcardinality, NONE,
CRYPTIC_SITE_USE, ANTI_CRYPTIC_SITE_USE, EXON_SKIPPING, INTRON_INCLUSION,
INTRON_INCLUSION_WITH_MUT, _SCcardinality }

The splicing consequence of a given variant.

Functions

- const boost::regex SPAN_PATTERN ("^[^ATGC]++")

Regular expression defining what is considered to be a spanning region in alignments.

- const boost::regex VAR_PATTERN ("(?<REFERENCE>[ATGC])/(?<VARIANT>[ATGC])")

Regular expression for the identification of the reference and mutant base in a variant.

- std::string getTimestamp ()

Retrieves the current timestamp for logging and benchmarking purposes.

- void noFileErr (boost::program_options::options_description desc)

This method is used to terminate the program and return `EXIT_FAILURE` if the required arguments are not provided.

- bool isDNANucleotide (char base)

Determines if the given character is a specific DNA nucleotide (i.e.

- bool `checkExonSkipping` (const int alnPos, const boost::smatch &matches, const int rightReadBoundary, const int leftReadBoundary)
Determines if the given read is indicative of exon skipping.
- int `main` (int argc, char *argv[])
The main method of the Veridical program.

Variables

- const int `OUTPUT_PROG_EVERY` = 100
The integer specifying how often to output progress information (every X variants).
- const int `DECIMAL_PRECISION` = 4
The number of decimal points to use when reporting read fractions.
- const double `MIN_READ_CUTOFF_FOR_VAL` = 0.01
The minimum value which is considered a reportable read fraction. Read fractions lower than this value are considered to be equivalent to 0. Note that this number can be used for both YJ transformed data or untransformed data as both are considered equivalent to 0 ($YJ(x = 0.01, \lambda = 0.5) = 0.0976177$).
- const double `DEFAULT_P_VALUE` = 0.05
The default p-value to use for the filtering of validated variants. Variants are kept in the filtered set iff their p-value is less than this threshold (or a user-provided threshold).
- const char `COMMENT_CHAR` = '#'
The character used to comment out lines in input files.
- boost::array< std::string, `_STcardinality` > `sampleTypeNames` = { "tumour", "normal" }
- boost::array< std::string, `_ETcardinality` > `evidenceTypeNames` = { "junction-spanning", "read-abundance" }
Stores the corresponding name for each EvidenceType.
- boost::array< std::string, `_SCcardinality` > `splicingConsequenceNames` = { "none", "cryptic site use", "anti cryptic site use", "exon skipping", "total intron inclusion", "intron inclusion with mutation" }
Stores the corresponding name for each SplicingConsequence.
- `BAMFileListType` `fullBAMFileList`
Data structure storing all of the RNA-Seq BAM files within a BAMFileListType data structure.
- static const double `directCoeff` = 1
This class serves as a container for the reasons and classification of a set of validation events for a given variant.
- static const double `countCoeff` = 0.01
coefficient used to weight indirect validation events - set to ensure junction-spanning evidence always takes precedence
- static const double `YJ_LAMBDA` = 0.5
The λ parameter for use in the Yeo-Johnson transformation. This value makes the distribution more right-skewed, correcting for left-skewness.
- const std::string `TYPE_CRYPTIC` = "cryptic"
Used in parsing. Cryptic variants are those which contain this string.
- const std::string `TYPE_EXONIC` = "exon"
Used in parsing. Exonic variants are those which contain this string.
- const std::string `CHR_PREFIX` = "chr"
Prefix for the chromosome column which is detected and inserted if needed.
- const int `MIN_MAP_Q` = 1
The minimum permitted quality for a read to be evaluated.

Token Separators

Used in parsing.

- const char *const `TOKEN_SEPERATOR_WS` = "\t"

- `const char *const TOKEN_SEPERATOR_CD = ";`

Variant Properties

- `const std::string NEG = "-"`
- `const std::string POS = "+"`
- `const std::string EXON = "exon"`
- `const std::string INTRON = "intron"`
- `const std::string DONOR = "donor"`
- `const std::string ACCEPTOR = "acceptor"`
- `const std::string NORMAL_SAMPLE = "normal"`

Variant file header column names

These strings are used when parsing the variant file to identify what each column pertains to.

- `const std::string H_CHR = "chromosome"`
- `const std::string H_COORD_SS = "splice"`
- `const std::string H_COORD_PART = "coordinate"`
- `const std::string H_STRAND = "strand"`
- `const std::string H_TYPE = "type"`
- `const std::string H_GENE = "gene"`
- `const std::string H_CRPTORNAT = "location"`
- `const std::string H_CODING = "location_type"`
- `const std::string H_HET = "heterozygosity"`
- `const std::string H_VAR_PART = "variant"`
- `const std::string H_IN_PART = "input"`
- `const std::string H_FILE = "key"`

Exome file header column names

These strings are used when parsing the exome file to identify what each column pertains to.

- `const std::string H_E_TID = "transcript id"`
- `const std::string H_E_CHR = H_CHR`
- `const std::string H_E_SCOORD = "exon chr start"`
- `const std::string H_E_ECOORD = "exon chr end"`
- `const std::string H_E_NUM = "exon rank"`
- `const std::string H_E_GENE = H_GENE`

7.2.1 Detailed Description

Definitions

We use the terms read and alignment interchangeably. Ns – Nucleotides in reads split across connecting exons, comprised of a sequence of unknown nucleotides Spanning Reads – Reads containing one or more 'N' (we term the sequence of these 'Ns' to be spanning bases). Validated Count – The number of reads satisfying the given criteria

Libraries

This program utilizes numerous BOOST libraries [1] and makes use of the BamTools API [2].

Definitions

We use the terms read and alignment interchangeably. Ns - Nucleotides in reads split across connecting exons, comprised of a sequence of unknown nucleotides Spanning Reads - Reads containing one or more 'N' (we term the sequence of these 'Ns' to be spanning bases). Validated Count - The number of reads satisfying the given criteria

Libraries

This program utilizes numerous BOOST libraries [1] and makes use of the BamTools API [2].

7.2.2 Typedef Documentation

7.2.2.1 typedef boost::unordered_map<std::string, SampleType> BAMFileListType

Data type for storage of paths (key) to RNA-Seq BAM files and their associated [SampleType](#).

7.2.2.2 typedef std::map< std::pair<std::string, int>, std::vector<std::string> > ExonMapType

Data type for the storage of exons to be used in discriminating between transcript variants.

This ordered map is keyed by the pair of transcript ID and exon start coordinate, and has as its value all columns from the exon file pertaining to the exon in question. This allows exons to be retrieved in their natural genomic order, seperated by transcript variant.

7.2.2.3 typedef boost::unordered_multimap< std::string, std::vector<std::string> > MultiExonMapType

Data type for the storage of all exome data (from all transcript variants) of the exons of a given gene (key)

7.2.2.4 typedef std::vector<double> ReadFractionsType

Data type for storage of read fractions, used to add validated reads to [ValidationReasons](#). Stores different fractions of a read (which need not sum to unity), as having validated for various reasons.

7.2.2.5 typedef boost::iostreams::tee_device<std::ostream, std::ofstream> TeeDevice

7.2.2.6 typedef boost::iostreams::stream<TeeDevice> TeeStream

7.2.2.7 typedef std::pair< double, std::pair<std::string, std::string> > ValDetailsType

Data type including a p-value and two strings of different levels of verbosity. Used to pass details of a validated variant.

7.2.2.8 typedef std::vector< std::pair<EvidenceType, SplicingConsequence> > ValidationReasonsToAddType

Data type for storage for each validated read to be added as a group to [ValidationReasons](#) (or portions thereof) and its associated validation information.

7.2.3 Enumeration Type Documentation

7.2.3.1 enum Directionality

The directionality of a given variant.

This is defined as follows:

+	Donor/Exonic	RIGHT
-	Donor/Exonic	LEFT
+	Acceptor/Intronic	LEFT
-	Acceptor/Intronic	RIGHT

Enumerator

RIGHT

LEFT

RIGHT

LEFT

7.2.3.2 enum EvidenceType

The type of evidence used to validate a variant.

Enumerator

DIRECT
COUNT
_ETcardinality
DIRECT
COUNT
_ETcardinality

7.2.3.3 enum SampleType

Enumeration describing the sample type of a given BAM file.

Enumerator

TUMOUR
NORMAL
_STcardinality
ANY
TUMOUR
NORMAL
_STcardinality
ANY

7.2.3.4 enum SplicingConsequence

The splicing consequence of a given variant.

Enumerator

NONE
CRYPTIC_SITE_USE
ANTI_CRYPTIC_SITE_USE
EXON_SKIPPING
INTRON_INCLUSION
INTRON_INCLUSION_WITH_MUT
_SCcardinality
NONE
CRYPTIC_SITE_USE
ANTI_CRYPTIC_SITE_USE
EXON_SKIPPING
INTRON_INCLUSION
INTRON_INCLUSION_WITH_MUT
_SCcardinality

7.2.4 Function Documentation

7.2.4.1 `bool checkExonSkipping (const int alnPos, const boost::smatch & matches, const int rightReadBoundary, const int leftReadBoundary)`

Determines if the given read is indicative of exon skipping.

Parameters

<i>alnPos</i>	The genomic coordinate of the read start
<i>matches</i>	The <code>boost::smatch</code> object containing information
<i>rightReadBoundary</i>	The right genomic boundary for the exon skipping check
<i>leftReadBoundary</i>	The left genomic boundary for the exon skipping check

Returns

True if and only if the alignment's spanning region starts at `leftReadBoundary` and the number of spanning bases is equal to the difference between `rightReadBoundary` and `leftReadBoundary`.

7.2.4.2 `std::string getTimestamp ()`

Retrieves the current timestamp for logging and benchmarking purposes.

Returns

The current system time formatted as: YYYY-MM-DD.HH:MM:SS.

7.2.4.3 `bool isDNANucleotide (char base)`

Determines if the given character is a specific DNA nucleotide (i.e. one of: A, T, G, or C).

Parameters

<i>base</i>	The putative nucleotide base
-------------	------------------------------

Returns

True if and only if the given character is one of: A, T, G, or C.

7.2.4.4 `int main (int argc, char * argv[])`

The main method of the Veridical program.

Parameters

<i>argc</i>	The number of command line arguments
<i>argv</i>	Command line arguments

Returns

The exit status, which should equal `EXIT_SUCCESS` upon completion.

7.2.4.5 `void noFileErr (boost::program_options::options_description desc)`

This method is used to terminate the program and return `EXIT_FAILURE` if the required arguments are not provided.

Parameters

<i>desc</i>	The <code>boost::program_options::options_description</code> object describing the program's correct usage
-------------	--

7.2.4.6 `const boost::regex SPAN_PATTERN ("++" [^ ATGC])`

Regular expression defining what is considered to be a spanning region in alignments.

7.2.4.7 `const boost::regex VAR_PATTERN ("(?<REFERENCE>[ATGC])(?<VARIANT>[ATGC])")`

Regular expression for the identification of the reference and mutant base in a variant.

7.2.5 Variable Documentation

7.2.5.1 `const std::string ACCEPTOR = "acceptor"`7.2.5.2 `const std::string CHR_PREFIX = "chr"`

Prefix for the chromosome column which is detected and inserted if needed.

7.2.5.3 `const char COMMENT_CHAR = '#'`

The character used to comment out lines in input files.

7.2.5.4 `const double countCoeff = 0.01 [static]`

coefficient used to weight indirect validation events - set to ensure junction-spanning evidence always takes precedence

7.2.5.5 `const int DECIMAL_PRECISION = 4`

The number of decimal points to use when reporting read fractions.

7.2.5.6 `const double DEFAULT_P_VALUE = 0.05`

The default p-value to use for the filtering of validated variants. Variants are kept in the filtered set iff their p-value is less than this threshold (or a user-provided threshold).

7.2.5.7 `const double directCoeff = 1 [static]`

This class serves as a container for the reasons and classification of a set of validation events for a given variant.
coefficient used to weight direct validation events

7.2.5.8 `const std::string DONOR = "donor"`7.2.5.9 `boost::array<std::string, _ETcardinality> evidenceTypeNames = { "junction-spanning", "read-abundance" }`

Stores the corresponding name for each [EvidenceType](#).

7.2.5.10 `const std::string EXON = "exon"`

7.2.5.11 `BAMFileListType fullBAMFileList`

Data structure storing all of the RNA-Seq BAM files within a [BAMFileListType](#) data structure.

7.2.5.12 `const std::string H_CHR = "chromosome"`

7.2.5.13 `const std::string H_CODING = "location_type"`

7.2.5.14 `const std::string H_COORD_PART = "coordinate"`

7.2.5.15 `const std::string H_COORD_SS = "splice"`

7.2.5.16 `const std::string H_CRPTORNAT = "location"`

7.2.5.17 `const std::string H_E_CHR = H_CHR`

7.2.5.18 `const std::string H_E_ECOORD = "exon chr end"`

7.2.5.19 `const std::string H_E_GENE = H_GENE`

7.2.5.20 `const std::string H_E_NUM = "exon rank"`

7.2.5.21 `const std::string H_E_SCOORD = "exon chr start"`

7.2.5.22 `const std::string H_E_TID = "transcript id"`

7.2.5.23 `const std::string H_FILE = "key"`

7.2.5.24 `const std::string H_GENE = "gene"`

7.2.5.25 `const std::string H_HET = "heterozygosity"`

7.2.5.26 `const std::string H_IN_PART = "input"`

7.2.5.27 `const std::string H_STRAND = "strand"`

7.2.5.28 `const std::string H_TYPE = "type"`

7.2.5.29 `const std::string H_VAR_PART = "variant"`

7.2.5.30 `const std::string INTRON = "intron"`

7.2.5.31 `const int MIN_MAP_Q = 1`

The minimum permitted quality for a read to be evaluated.

Remarks

Reads with quality 0 tend to map randomly so we always wish to exclude them.

7.2.5.32 `const double MIN_READ_CUTOFF_FOR_VAL = 0.01`

The minimum value which is considered a reportable read fraction. Read fractions lower than this value are considered to be equivalent to 0. Note that this number can be used for both YJ transformed data or untransformed data

as both are considered equivalent to 0 ($YJ(x = 0.01, \lambda = 0.5) = 0.0976177$)

7.2.5.33 `const std::string NEG = "-"`

7.2.5.34 `const std::string NORMAL_SAMPLE = "normal"`

7.2.5.35 `const int OUTPUT_PROG_EVERY = 100`

The integer specifying how often to output progress information (every X variants).

7.2.5.36 `const std::string POS = "+"`

7.2.5.37 `boost::array<std::string, _STcardinality> sampleTypeNames = { "tumour", "normal" }`

7.2.5.38 `boost::array<std::string, _SCcardinality> splicingConsequenceNames = { "none", "cryptic site use", "anti cryptic site use", "exon skipping", "total intron inclusion", "intron inclusion with mutation" }`

Stores the corresponding name for each [SplicingConsequence](#).

7.2.5.39 `const char* const TOKEN_SEPERATOR_CD = ","`

7.2.5.40 `const char* const TOKEN_SEPERATOR_WS = "\t"`

7.2.5.41 `const std::string TYPE_CRYPTIC = "cryptic"`

Used in parsing. Cryptic variants are those which contain this string.

7.2.5.42 `const std::string TYPE_EXONIC = "exon"`

Used in parsing. Exonic variants are those which contain this string.

7.2.5.43 `const double YJ_LAMBDA = 0.5` `[static]`

The λ parameter for use in the Yeo-Johnson transformation. This value makes the distribution more right-skewed, correcting for left-skewness.

7.3 /home/cviner2/RNA-seq/VeridicalTrial.cpp File Reference

```
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <map>
#include <limits>
#include <algorithm>
#include <numeric>
#include <iterator>
#include <utility>
#include <ctime>
#include <stdexcept>
#include <boost/program_options.hpp>
#include <boost/array.hpp>
#include <boost/lexical_cast.hpp>
#include <boost/foreach.hpp>
#include <boost/tokenizer.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/algorithm/string/find.hpp>
#include <boost/algorithm/string/erase.hpp>
#include <boost/bind.hpp>
#include <boost/unordered_map.hpp>
#include <boost/swap.hpp>
#include <boost/assign/list_of.hpp>
#include <boost/filesystem.hpp>
#include <boost/regex.hpp>
#include <boost/iostreams/tee.hpp>
#include <boost/iostreams/stream.hpp>
#include <boost/math/distributions/normal.hpp>
#include <boost/assert.hpp>
#include <boost/network/protocol/http/client.hpp>
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#include "api/BamMultiReader.h"
#include "api/BamAlignment.h"
```

Classes

- class [ValidationReasons](#)

Typedefs

- typedef std::vector< std::pair
< [EvidenceType](#),
[SplicingConsequence](#) > > [ValidationReasonsToAddType](#)

Data type for storage for each validated read to be added as a group to [ValidationReasons](#) (or portions thereof) and its associated validation information.

- typedef std::vector< double > [ReadFractionsType](#)

Data type for storage of read fractions, used to add validated reads to [ValidationReasons](#). Stores different fractions of a read (which need not sum to unity), as having validated for various reasons.

- typedef boost::unordered_map
< std::string, [SampleType](#) > [BAMFileListType](#)

Data type for storage of paths (key) to RNA-Seq BAM files and their associated [SampleType](#).

- typedef std::pair< double,
std::pair< std::string,
std::string > > [ValDetailsType](#)

Data type including a p-value and two strings of different levels of verbosity. Used to pass details of a validated variant.

- typedef
boost::unordered_multimap
< std::string, std::vector
< std::string > > [MultiExonMapType](#)

Data type for the storage of all exome data (from all transcript variants) of the exons of a given gene (key)

- typedef std::map< std::pair
< std::string, int >
, std::vector< std::string > > [ExonMapType](#)

Data type for the storage of exons to be used in discriminating between transcript variants.

Output Modulators

BOOST Tee types to facilitate multiple output redirection.

- typedef
boost::iostreams::tee_device
< std::ostream, std::ofstream > [TeeDevice](#)
- typedef
boost::iostreams::stream
< [TeeDevice](#) > [TeeStream](#)

Enumerations

- enum [SampleType](#) {
[TUMOUR](#), [NORMAL](#), [_STcardinality](#), [ANY](#),
[TUMOUR](#), [NORMAL](#), [_STcardinality](#), [ANY](#) }

Enumeration describing the sample type of a given BAM file.

- enum [Directionality](#) { [RIGHT](#), [LEFT](#), [RIGHT](#), [LEFT](#) }

The directionality of a given variant.

- enum [EvidenceType](#) {
[DIRECT](#), [COUNT](#), [_ETcardinality](#), [DIRECT](#),
[COUNT](#), [_ETcardinality](#) }

The type of evidence used to validate a variant.

- enum [SplicingConsequence](#) {
[NONE](#), [CRYPTIC_SITE_USE](#), [ANTI_CRYPTIC_SITE_USE](#), [EXON_SKIPPING](#),
[INTRON_INCLUSION](#), [INTRON_INCLUSION_WITH_MUT](#), [_SCcardinality](#), [NONE](#),
[CRYPTIC_SITE_USE](#), [ANTI_CRYPTIC_SITE_USE](#), [EXON_SKIPPING](#), [INTRON_INCLUSION](#),
[INTRON_INCLUSION_WITH_MUT](#), [_SCcardinality](#) }

The splicing consequence of a given variant.

Functions

- const boost::regex [SPAN_PATTERN](#) ("^[^ATGC]++")
Regular expression defining what is considered to be a spanning region in alignments.
- const boost::regex [VAR_PATTERN](#) ("(?<REFERENCE>[ATGC])/(?<VARIANT>[ATGC])")
Regular expression for the identification of the reference and mutant base in a variant.

- `std::string` `getTimestamp` ()
Retrieves the current timestamp for logging and benchmarking purposes.
- `void` `noFileErr` (boost::program_options::options_description desc)
This method is used to terminate the program and return `EXIT_FAILURE` if the required arguments are not provided.
- `bool` `isDNANucleotide` (char base)
Determines if the given character is a specific DNA nucleotide (i.e.
- `bool` `checkExonSkipping` (const int alnPos, const boost::smatch &matches, const int rightReadBoundary, const int leftReadBoundary)
Determines if the given read is indicative of exon skipping.
- `int` `main` (int argc, char *argv[])
The main method of the Veridical program.

Variables

- `const std::string` `VAL_SITE` = "http://www.veridical.org/XS44yNqLd8Jopeqy1CqJ.php"
Site used to validate trial software copy.
- `const std::string` `VAL_JSON_PATH` = "allowAccess"
JSON path to the validation element.
- `const int` `OUTPUT_PROG EVERY` = 100
The integer specifying how often to output progress information (every X variants).
- `const int` `DECIMAL_PRECISION` = 2
The number of decimal points to use when reporting read fractions.
- `const double` `MIN_READ_CUTOFF_FOR_VAL` = 0.01
The minimum value which is considered a reportable read fraction. Read fractions lower than this value are considered to be equivalent to 0. Note that this number can be used for both YJ transformed data or untransformed data as both are considered equivalent to 0 ($YJ(x = 0.01, \lambda = 0.5) = 0.0976177$)
- `const double` `DEFAULT_P_VALUE` = 0.05
The default p-value to use for the filtering of validated variants. Variants are kept in the filtered set iff their p-value is less than this threshold (or a user-provided threshold).
- `const char` `COMMENT_CHAR` = '#'
The character used to comment out lines in input files.
- `boost::array< std::string, _STcardinality >` `sampleTypeNames` = { "tumour", "normal" }
- `boost::array< std::string, _ETcardinality >` `evidenceTypeNames` = { "junction-spanning", "read-abundance" }
*Stores the corresponding name for each *EvidenceType*.*
- `boost::array< std::string, _SCcardinality >` `splicingConsequenceNames` = { "none", "cryptic site use", "anti cryptic site use", "exon skipping", "total intron inclusion", "intron inclusion with mutation" }
*Stores the corresponding name for each *SplicingConsequence*.*
- `BAMFileListType` `fullBAMFileList`
*Data structure storing all of the RNA-Seq BAM files within a *BAMFileListType* data structure.*
- `static const double` `directCoeff` = 1
This class serves as a container for the reasons and classification of a set of validation events for a given variant.
- `static const double` `countCoeff` = 0.01
coefficient used to weight indirect validation events - set to ensure junction-spanning evidence always takes precedence
- `static const double` `YJ_LAMBDA` = 0.5
The λ parameter for use in the Yeo-Johnson transformation. This value makes the distribution more right-skewed, correcting for left-skewness.
- `const std::string` `TYPE_CRYPTIC` = "cryptic"
Used in parsing. Cryptic variants are those which contain this string.

- const std::string `TYPE_EXONIC` = "exon"
Used in parsing. Exonic variants are those which contain this string.
- const std::string `CHR_PREFIX` = "chr"
Prefix for the chromosome column which is detected and inserted if needed.
- const int `MIN_MAP_Q` = 1
The minimum permitted quality for a read to be evaluated.

Token Separators

Used in parsing.

- const char *const `TOKEN_SEPERATOR_WS` = "\t"
- const char *const `TOKEN_SEPERATOR_CD` = ","

Variant Properties

- const std::string `NEG` = "-"
- const std::string `POS` = "+"
- const std::string `EXON` = "exon"
- const std::string `INTRON` = "intron"
- const std::string `DONOR` = "donor"
- const std::string `ACCEPTOR` = "acceptor"
- const std::string `NORMAL_SAMPLE` = "normal"

Variant file header column names

These strings are used when parsing the variant file to identify what each column pertains to.

- const std::string `H_CHR` = "chromosome"
- const std::string `H_COORD_SS` = "splice"
- const std::string `H_COORD_PART` = "coordinate"
- const std::string `H_STRAND` = "strand"
- const std::string `H_TYPE` = "type"
- const std::string `H_GENE` = "gene"
- const std::string `H_CRPTORNAT` = "location"
- const std::string `H_CODING` = "location_type"
- const std::string `H_HET` = "heterozygosity"
- const std::string `H_VAR_PART` = "variant"
- const std::string `H_IN_PART` = "input"
- const std::string `H_FILE` = "key"

Exome file header column names

These strings are used when parsing the exome file to identify what each column pertains to.

- const std::string `H_E_TID` = "transcript id"
- const std::string `H_E_CHR` = `H_CHR`
- const std::string `H_E_SCOORD` = "exon chr start"
- const std::string `H_E_ECOORD` = "exon chr end"
- const std::string `H_E_NUM` = "exon rank"
- const std::string `H_E_GENE` = `H_GENE`

7.3.1 Typedef Documentation

7.3.1.1 typedef boost::unordered_map<std::string, SampleType> BAMFileListType

Data type for storage of paths (key) to RNA-Seq BAM files and their associated `SampleType`.

7.3.1.2 typedef std::map< std::pair<std::string, int>, std::vector<std::string> > ExonMapType

Data type for the storage of exons to be used in discriminating between transcript variants.

This ordered map is keyed by the pair of transcript ID and exon start coordinate, and has as its value all columns from the exon file pertaining to the exon in question. This allows exons to be retrieved in their natural genomic order, separated by transcript variant.

7.3.1.3 typedef boost::unordered_multimap< std::string, std::vector<std::string> > MultiExonMapType

Data type for the storage of all exome data (from all transcript variants) of the exons of a given gene (key)

7.3.1.4 typedef std::vector<double> ReadFractionsType

Data type for storage of read fractions, used to add validated reads to [ValidationReasons](#). Stores different fractions of a read (which need not sum to unity), as having validated for various reasons.

7.3.1.5 typedef boost::iostreams::tee_device<std::ostream, std::ofstream> TeeDevice**7.3.1.6 typedef boost::iostreams::stream<TeeDevice> TeeStream****7.3.1.7 typedef std::pair< double, std::pair<std::string, std::string> > ValDetailsType**

Data type including a p-value and two strings of different levels of verbosity. Used to pass details of a validated variant.

7.3.1.8 typedef std::vector< std::pair<EvidenceType, SplicingConsequence> > ValidationReasonsToAddType

Data type for storage for each validated read to be added as a group to [ValidationReasons](#) (or portions thereof) and its associated validation information.

7.3.2 Enumeration Type Documentation**7.3.2.1 enum Directionality**

The directionality of a given variant.

This is defined as follows:

+	Donor/Exonic	RIGHT
-	Donor/Exonic	LEFT
+	Acceptor/Intronic	LEFT
-	Acceptor/Intronic	RIGHT

Enumerator

RIGHT

LEFT

RIGHT

LEFT

7.3.2.2 enum EvidenceType

The type of evidence used to validate a variant.

Enumerator

DIRECT
COUNT
_ETcardinality
DIRECT
COUNT
_ETcardinality

7.3.2.3 enum SampleType

Enumeration describing the sample type of a given BAM file.

Enumerator

TUMOUR
NORMAL
_STcardinality
ANY
TUMOUR
NORMAL
_STcardinality
ANY

7.3.2.4 enum SplicingConsequence

The splicing consequence of a given variant.

Enumerator

NONE
CRYPTIC_SITE_USE
ANTI_CRYPTIC_SITE_USE
EXON_SKIPPING
INTRON_INCLUSION
INTRON_INCLUSION_WITH_MUT
_SCcardinality
NONE
CRYPTIC_SITE_USE
ANTI_CRYPTIC_SITE_USE
EXON_SKIPPING
INTRON_INCLUSION
INTRON_INCLUSION_WITH_MUT
_SCcardinality

7.3.3 Function Documentation

7.3.3.1 `bool checkExonSkipping (const int alnPos, const boost::smatch & matches, const int rightReadBoundary, const int leftReadBoundary)`

Determines if the given read is indicative of exon skipping.

Parameters

<i>alnPos</i>	The genomic coordinate of the read start
<i>matches</i>	The <code>boost::smatch</code> object containing information
<i>rightReadBoundary</i>	The right genomic boundary for the exon skipping check
<i>leftReadBoundary</i>	The left genomic boundary for the exon skipping check

Returns

True if and only if the alignment's spanning region starts at `leftReadBoundary` and the number of spanning bases is equal to the difference between `rightReadBoundary` and `leftReadBoundary`.

7.3.3.2 `std::string getTimestamp ()`

Retrieves the current timestamp for logging and benchmarking purposes.

Returns

The current system time formatted as: YYYY-MM-DD.HH:MM:SS.

7.3.3.3 `bool isDNANucleotide (char base)`

Determines if the given character is a specific DNA nucleotide (i.e. one of: A, T, G, or C).

Parameters

<i>base</i>	The putative nucleotide base
-------------	------------------------------

Returns

True if and only if the given character is one of: A, T, G, or C.

7.3.3.4 `int main (int argc, char * argv[])`

The main method of the Veridical program.

Parameters

<i>argc</i>	The number of command line arguments
<i>argv</i>	Command line arguments

Returns

The exit status, which should equal `EXIT_SUCCESS` upon completion.

7.3.3.5 `void noFileErr (boost::program_options::options_description desc)`

This method is used to terminate the program and return `EXIT_FAILURE` if the required arguments are not provided.

Parameters

<i>desc</i>	The <code>boost::program_options::options_description</code> object describing the program's correct usage
-------------	--

7.3.3.6 `const boost::regex SPAN_PATTERN ("++" [^ATGC])`

Regular expression defining what is considered to be a spanning region in alignments.

7.3.3.7 `const boost::regex VAR_PATTERN ("(?<REFERENCE>[ATGC])(?<VARIANT>[ATGC])")`

Regular expression for the identification of the reference and mutant base in a variant.

7.3.4 Variable Documentation

7.3.4.1 `const std::string ACCEPTOR = "acceptor"`

7.3.4.2 `const std::string CHR_PREFIX = "chr"`

Prefix for the chromosome column which is detected and inserted if needed.

7.3.4.3 `const char COMMENT_CHAR = '#'`

The character used to comment out lines in input files.

7.3.4.4 `const double countCoeff = 0.01 [static]`

coefficient used to weight indirect validation events - set to ensure junction-spanning evidence always takes precedence

7.3.4.5 `const int DECIMAL_PRECISION = 2`

The number of decimal points to use when reporting read fractions.

7.3.4.6 `const double DEFAULT_P_VALUE = 0.05`

The default p-value to use for the filtering of validated variants. Variants are kept in the filtered set iff their p-value is less than this threshold (or a user-provided threshold).

7.3.4.7 `const double directCoeff = 1 [static]`

This class serves as a container for the reasons and classification of a set of validation events for a given variant.
coefficient used to weight direct validation events

7.3.4.8 `const std::string DONOR = "donor"`

7.3.4.9 `boost::array<std::string, _ETcardinality> evidenceTypeNames = { "junction-spanning", "read-abundance" }`

Stores the corresponding name for each [EvidenceType](#).

7.3.4.10 `const std::string EXON = "exon"`

7.3.4.11 **BAMFileListType** fullBAMFileList

Data structure storing all of the RNA-Seq BAM files within a [BAMFileListType](#) data structure.

7.3.4.12 `const std::string H_CHR = "chromosome"`

7.3.4.13 `const std::string H_CODING = "location_type"`

7.3.4.14 `const std::string H_COORD_PART = "coordinate"`

7.3.4.15 `const std::string H_COORD_SS = "splice"`

7.3.4.16 `const std::string H_CRPTORNAT = "location"`

7.3.4.17 `const std::string H_E_CHR = H_CHR`

7.3.4.18 `const std::string H_E_ECOORD = "exon chr end"`

7.3.4.19 `const std::string H_E_GENE = H_GENE`

7.3.4.20 `const std::string H_E_NUM = "exon rank"`

7.3.4.21 `const std::string H_E_SCOORD = "exon chr start"`

7.3.4.22 `const std::string H_E_TID = "transcript id"`

7.3.4.23 `const std::string H_FILE = "key"`

7.3.4.24 `const std::string H_GENE = "gene"`

7.3.4.25 `const std::string H_HET = "heterozygosity"`

7.3.4.26 `const std::string H_IN_PART = "input"`

7.3.4.27 `const std::string H_STRAND = "strand"`

7.3.4.28 `const std::string H_TYPE = "type"`

7.3.4.29 `const std::string H_VAR_PART = "variant"`

7.3.4.30 `const std::string INTRON = "intron"`

7.3.4.31 `const int MIN_MAP_Q = 1`

The minimum permitted quality for a read to be evaluated.

Remarks

Reads with quality 0 tend to map randomly so we always wish to exclude them.

7.3.4.32 `const double MIN_READ_CUTOFF_FOR_VAL = 0.01`

The minimum value which is considered a reportable read fraction. Read fractions lower than this value are considered to be equivalent to 0. Note that this number can be used for both YJ transformed data or untransformed data

as both are considered equivalent to 0 ($YJ(x = 0.01, \lambda = 0.5) = 0.0976177$)

7.3.4.33 `const std::string NEG = "-"`

7.3.4.34 `const std::string NORMAL_SAMPLE = "normal"`

7.3.4.35 `const int OUTPUT_PROG_EVERY = 100`

The integer specifying how often to output progress information (every X variants).

7.3.4.36 `const std::string POS = "+"`

7.3.4.37 `boost::array<std::string, _STcardinality> sampleTypeNames = { "tumour", "normal" }`

7.3.4.38 `boost::array<std::string, _SCcardinality> splicingConsequenceNames = { "none", "cryptic site use", "anti cryptic site use", "exon skipping", "total intron inclusion", "intron inclusion with mutation" }`

Stores the corresponding name for each [SplicingConsequence](#).

7.3.4.39 `const char* const TOKEN_SEPERATOR_CD = ","`

7.3.4.40 `const char* const TOKEN_SEPERATOR_WS = "\t"`

7.3.4.41 `const std::string TYPE_CRYPTIC = "cryptic"`

Used in parsing. Cryptic variants are those which contain this string.

7.3.4.42 `const std::string TYPE_EXONIC = "exon"`

Used in parsing. Exonic variants are those which contain this string.

7.3.4.43 `const std::string VAL_JSON_PATH = "allowAccess"`

JSON path to the validation element.

7.3.4.44 `const std::string VAL_SITE = "http://www.veridical.org/XS44yNqLd8Jopeqy1CqJ.php"`

Site used to validate trial software copy.

7.3.4.45 `const double YJ_LAMBDA = 0.5 [static]`

The λ parameter for use in the Yeo-Johnson transformation. This value makes the distribution more right-skewed, correcting for left-skewness.

Bibliography

- [1] *BOOST C++ Libraries*. <http://www.boost.org>. 37
- [2] D. W. Barnett, E. K. Garrison, A. R. Quinlan, M. P. Strömberg, and G. T. Marth. Bamtools: A c++ api and toolkit for analyzing and managing bam files. *Bioinformatics*, 27(12):1691–1692, 2011. <http://bioinformatics.oxfordjournals.org/content/27/12/1691>. 37
- [3] I. N. K Yeo and R. A. Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 2000. Cited By (since 1996):73. 1, 30, 31, 32

Index

- [/home/cviner2/RNA-seq/Veridical.cpp](#), 33
- [/home/cviner2/RNA-seq/VeridicalTrial.cpp](#), 45
- [/home/cviner2/RNA-seq/bgzf_add_eof.py](#), 33
- [_ETcardinality](#)
 - [Veridical.cpp](#), 39
 - [VeridicalTrial.cpp](#), 50
- [_SCcardinality](#)
 - [Veridical.cpp](#), 39
 - [VeridicalTrial.cpp](#), 50
- [_STcardinality](#)
 - [Veridical.cpp](#), 39
 - [VeridicalTrial.cpp](#), 50
- [ANTI_CRYPTIC_SITE_USE](#)
 - [Veridical.cpp](#), 39
 - [VeridicalTrial.cpp](#), 50
- [ANY](#)
 - [Veridical.cpp](#), 39
 - [VeridicalTrial.cpp](#), 50
- [ACCEPTOR](#)
 - [Veridical.cpp](#), 42
 - [VeridicalTrial.cpp](#), 53
- [addNonVarContValReason](#)
 - [ValidationReasons](#), 16
- [addNonVarContValReasons](#)
 - [ValidationReasons](#), 16
- [addSummaryValDetails](#)
 - [ValidationReasons](#), 17
- [addToTotal](#)
 - [ValidationReasons](#), 17
- [addValidationReason](#)
 - [ValidationReasons](#), 18
- [addValidationReasons](#)
 - [ValidationReasons](#), 18
- [addVarContValidationReason](#)
 - [ValidationReasons](#), 19
- [addVarContValidationReasons](#)
 - [ValidationReasons](#), 19
- [BAMFileListType](#)
 - [Veridical.cpp](#), 38
 - [VeridicalTrial.cpp](#), 48
- [bgzf_add_eof](#), 9
 - [fix_bam](#), 9
 - [sys_exit](#), 9
- [COUNT](#)
 - [Veridical.cpp](#), 39
 - [VeridicalTrial.cpp](#), 50
- [CRYPTIC_SITE_USE](#)
 - [Veridical.cpp](#), 39
 - [VeridicalTrial.cpp](#), 50
- [CHR_PREFIX](#)
 - [Veridical.cpp](#), 42
 - [VeridicalTrial.cpp](#), 53
- [COMMENT_CHAR](#)
 - [Veridical.cpp](#), 42
 - [VeridicalTrial.cpp](#), 53
- [checkExonSkipping](#)
 - [Veridical.cpp](#), 40
 - [VeridicalTrial.cpp](#), 51
- [countCoeff](#)
 - [Veridical.cpp](#), 42
 - [VeridicalTrial.cpp](#), 53
- [DIRECT](#)
 - [Veridical.cpp](#), 39
 - [VeridicalTrial.cpp](#), 50
- [DECIMAL_PRECISION](#)
 - [Veridical.cpp](#), 42
 - [VeridicalTrial.cpp](#), 53
- [DEFAULT_P_VALUE](#)
 - [Veridical.cpp](#), 42
 - [VeridicalTrial.cpp](#), 53
- [DONOR](#)
 - [Veridical.cpp](#), 42
 - [VeridicalTrial.cpp](#), 53
- [directCoeff](#)
 - [Veridical.cpp](#), 42
 - [VeridicalTrial.cpp](#), 53
- [Directionality](#)
 - [Veridical.cpp](#), 38
 - [VeridicalTrial.cpp](#), 49
- [EXON_SKIPPING](#)
 - [Veridical.cpp](#), 39
 - [VeridicalTrial.cpp](#), 50
- [EXON](#)
 - [Veridical.cpp](#), 42
 - [VeridicalTrial.cpp](#), 53
- [EvidenceType](#)
 - [Veridical.cpp](#), 39
 - [VeridicalTrial.cpp](#), 49
- [evidenceTypeNames](#)
 - [Veridical.cpp](#), 42
 - [VeridicalTrial.cpp](#), 53
- [ExonMapType](#)
 - [Veridical.cpp](#), 38
 - [VeridicalTrial.cpp](#), 48

- fix_bam
 - bgzf_add_eof, 9
- fullBAMFileList
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- getAllSpliceConsequencesCount
 - ValidationReasons, 19
- getCrypticValNumber
 - ValidationReasons, 19, 21
- getNonVarContCrypticSiteUseValScore
 - ValidationReasons, 21
- getNonVarContDetails
 - ValidationReasons, 21
- getNonVarContDirectEvidenceValCount
 - ValidationReasons, 21, 22
- getNonVarContExonSkippingValScore
 - ValidationReasons, 22
- getNonVarContIndirectEvidenceValCount
 - ValidationReasons, 22
- getNonVarContIntronInclDirectNumber
 - ValidationReasons, 22
- getNonVarContIntronInclIndirectNumber
 - ValidationReasons, 22, 23
- getNonVarContIntronInclValScore
 - ValidationReasons, 23
- getNonVarContOverallValCount
 - ValidationReasons, 23
- getNonVarContSampleSize
 - ValidationReasons, 23
- getPValue
 - ValidationReasons, 24
- getPredominantValidationReason
 - ValidationReasons, 23, 24
- getTimestamp
 - Veridical.cpp, 41
 - VeridicalTrial.cpp, 52
- getValNumber
 - ValidationReasons, 25, 26
- getValScore
 - ValidationReasons, 26
- getValidationDetails
 - ValidationReasons, 24, 25
- getVarContCrypticSiteUseValScore
 - ValidationReasons, 27
- getVarContDetails
 - ValidationReasons, 27
- getVarContDirectEvidenceValCount
 - ValidationReasons, 27
- getVarContExonSkippingValScore
 - ValidationReasons, 27, 28
- getVarContIndirectEvidenceValCount
 - ValidationReasons, 28
- getVarContIntronInclDirectNumber
 - ValidationReasons, 28
- getVarContIntronInclIndirectNumber
 - ValidationReasons, 28
- getVarContIntronInclValScore
 - ValidationReasons, 28, 29
- getVarContOverallValCount
 - ValidationReasons, 29
- getVarContPredominantValidationReasonName
 - ValidationReasons, 29
- getVarContSampleSize
 - ValidationReasons, 29
- getZScore
 - ValidationReasons, 29, 30
- getZScores
 - ValidationReasons, 30
- H_CHR
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_CODING
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_COORD_PART
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_COORD_SS
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_CRPTORNAT
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_E_CHR
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_E_ECOORD
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_E_GENE
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_E_NUM
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_E_SCOORD
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_E_TID
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_FILE
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_GENE
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_HET
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_IN_PART
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54
- H_STRAND
 - Veridical.cpp, 43
 - VeridicalTrial.cpp, 54

- H_TYPE
 - Veridical.cpp, [43](#)
 - VeridicalTrial.cpp, [54](#)
- H_VAR_PART
 - Veridical.cpp, [43](#)
 - VeridicalTrial.cpp, [54](#)
- INTRON_INCLUSION
 - Veridical.cpp, [39](#)
 - VeridicalTrial.cpp, [50](#)
- INTRON_INCLUSION_WITH_MUT
 - Veridical.cpp, [39](#)
 - VeridicalTrial.cpp, [50](#)
- INTRON
 - Veridical.cpp, [43](#)
 - VeridicalTrial.cpp, [54](#)
- InnerKeyType
 - ValidationReasons, [15](#)
- isDNANucleotide
 - Veridical.cpp, [41](#)
 - VeridicalTrial.cpp, [52](#)
- isStronglyCorroborating
 - ValidationReasons, [31](#)
- LEFT
 - Veridical.cpp, [38](#)
 - VeridicalTrial.cpp, [49](#)
- MIN_MAP_Q
 - Veridical.cpp, [43](#)
 - VeridicalTrial.cpp, [54](#)
- main
 - Veridical.cpp, [41](#)
 - VeridicalTrial.cpp, [52](#)
- MultiExonMapType
 - Veridical.cpp, [38](#)
 - VeridicalTrial.cpp, [49](#)
- NONE
 - Veridical.cpp, [39](#)
 - VeridicalTrial.cpp, [50](#)
- NORMAL
 - Veridical.cpp, [39](#)
 - VeridicalTrial.cpp, [50](#)
- NEG
 - Veridical.cpp, [44](#)
 - VeridicalTrial.cpp, [55](#)
- NORMAL_SAMPLE
 - Veridical.cpp, [44](#)
 - VeridicalTrial.cpp, [55](#)
- noFileErr
 - Veridical.cpp, [41](#)
 - VeridicalTrial.cpp, [52](#)
- nonVarContValMap
 - ValidationReasons, [32](#)
- OUTPUT_PROG_EVERY
 - Veridical.cpp, [44](#)
 - VeridicalTrial.cpp, [55](#)
- op_YJTransform
 - ValidationReasons, [31](#)
- POS
 - Veridical.cpp, [44](#)
 - VeridicalTrial.cpp, [55](#)
- RIGHT
 - Veridical.cpp, [38](#)
 - VeridicalTrial.cpp, [49](#)
- ReadFractionsType
 - Veridical.cpp, [38](#)
 - VeridicalTrial.cpp, [49](#)
- SPAN_PATTERN
 - Veridical.cpp, [42](#)
 - VeridicalTrial.cpp, [53](#)
- SampleType
 - Veridical.cpp, [39](#)
 - VeridicalTrial.cpp, [50](#)
- sampleTypeNames
 - Veridical.cpp, [44](#)
 - VeridicalTrial.cpp, [55](#)
- SortedValReasonsMapType
 - ValidationReasons, [15](#)
- SplicingConsequence
 - Veridical.cpp, [39](#)
 - VeridicalTrial.cpp, [50](#)
- splicingConsequenceNames
 - Veridical.cpp, [44](#)
 - VeridicalTrial.cpp, [55](#)
- sys_exit
 - bgzf_add_eof, [9](#)
- TUMOUR
 - Veridical.cpp, [39](#)
 - VeridicalTrial.cpp, [50](#)
- TOKEN_SEPERATOR_CD
 - Veridical.cpp, [44](#)
 - VeridicalTrial.cpp, [55](#)
- TOKEN_SEPERATOR_WS
 - Veridical.cpp, [44](#)
 - VeridicalTrial.cpp, [55](#)
- TYPE_CRYPTIC
 - Veridical.cpp, [44](#)
 - VeridicalTrial.cpp, [55](#)
- TYPE_EXONIC
 - Veridical.cpp, [44](#)
 - VeridicalTrial.cpp, [55](#)
- TeeDevice
 - Veridical.cpp, [38](#)
 - VeridicalTrial.cpp, [49](#)
- TeeStream
 - Veridical.cpp, [38](#)
 - VeridicalTrial.cpp, [49](#)
- VAL_JSON_PATH
 - VeridicalTrial.cpp, [55](#)
- VAL_SITE

- VeridicalTrial.cpp, 55
- VAR_PATTERN
 - Veridical.cpp, 42
 - VeridicalTrial.cpp, 53
- ValDetailsType
 - Veridical.cpp, 38
 - VeridicalTrial.cpp, 49
- ValidatedInnerMapType
 - ValidationReasons, 15, 16
- ValidatedMapType
 - ValidationReasons, 16
- ValidationReasons, 11
 - addNonVarContValReason, 16
 - addNonVarContValReasons, 16
 - addSummaryValDetails, 17
 - addToTotal, 17
 - addValidationReason, 18
 - addValidationReasons, 18
 - addVarContValidationReason, 19
 - addVarContValidationReasons, 19
 - getAllSpliceConsequencesCount, 19
 - getCrypticValNumber, 19, 21
 - getNonVarContCrypticSiteUseValScore, 21
 - getNonVarContDetails, 21
 - getNonVarContDirectEvidenceValCount, 21, 22
 - getNonVarContExonSkippingValScore, 22
 - getNonVarContIndirectEvidenceValCount, 22
 - getNonVarContIntronInclDirectNumber, 22
 - getNonVarContIntronInclIndirectNumber, 22, 23
 - getNonVarContIntronInclValScore, 23
 - getNonVarContOverallValCount, 23
 - getNonVarContSampleSize, 23
 - getPValue, 24
 - getPredominantValidationReason, 23, 24
 - getValNumber, 25, 26
 - getValScore, 26
 - getValidationDetails, 24, 25
 - getVarContCrypticSiteUseValScore, 27
 - getVarContDetails, 27
 - getVarContDirectEvidenceValCount, 27
 - getVarContExonSkippingValScore, 27, 28
 - getVarContIndirectEvidenceValCount, 28
 - getVarContIntronInclDirectNumber, 28
 - getVarContIntronInclIndirectNumber, 28
 - getVarContIntronInclValScore, 28, 29
 - getVarContOverallValCount, 29
 - getVarContPredominantValidationReasonName, 29
 - getVarContSampleSize, 29
 - getZScore, 29, 30
 - getZScores, 30
 - InnerKeyType, 15
 - isStronglyCorroborating, 31
 - nonVarContValMap, 32
 - op_YJTransform, 31
 - SortedValReasonsMapType, 15
 - ValidatedInnerMapType, 15, 16
 - ValidatedMapType, 16
 - ValidationReasons, 16
 - ValidationReasons, 16
 - varContValMap, 32
 - YJTransform, 32
 - ValidationReasonsToAddType
 - Veridical.cpp, 38
 - VeridicalTrial.cpp, 49
 - varContValMap
 - ValidationReasons, 32
 - Veridical.cpp
 - _ETcardinality, 39
 - _SCcardinality, 39
 - _STcardinality, 39
 - ANTI_CRYPTIC_SITE_USE, 39
 - ANY, 39
 - COUNT, 39
 - CRYPTIC_SITE_USE, 39
 - DIRECT, 39
 - EXON_SKIPPING, 39
 - INTRON_INCLUSION, 39
 - INTRON_INCLUSION_WITH_MUT, 39
 - LEFT, 38
 - NONE, 39
 - NORMAL, 39
 - RIGHT, 38
 - TUMOUR, 39
 - Veridical.cpp
 - ACCEPTOR, 42
 - BAMFileListType, 38
 - CHR_PREFIX, 42
 - COMMENT_CHAR, 42
 - checkExonSkipping, 40
 - countCoeff, 42
 - DECIMAL_PRECISION, 42
 - DEFAULT_P_VALUE, 42
 - DONOR, 42
 - directCoeff, 42
 - Directionality, 38
 - EXON, 42
 - EvidenceType, 39
 - evidenceTypeNames, 42
 - ExonMapType, 38
 - fullBAMFileList, 43
 - getTimestamp, 41
 - H_CHR, 43
 - H_CODING, 43
 - H_COORD_PART, 43
 - H_COORD_SS, 43
 - H_CRPTORNAT, 43
 - H_E_CHR, 43
 - H_E_ECOORD, 43
 - H_E_GENE, 43
 - H_E_NUM, 43
 - H_E_SCOORD, 43
 - H_E_TID, 43
 - H_FILE, 43
 - H_GENE, 43
 - H_HET, 43

- H_IN_PART, 43
- H_STRAND, 43
- H_TYPE, 43
- H_VAR_PART, 43
- INTRON, 43
- isDNANucleotide, 41
- MIN_MAP_Q, 43
- main, 41
- MultiExonMapType, 38
- NEG, 44
- NORMAL_SAMPLE, 44
- noFileErr, 41
- OUTPUT_PROG_EVERY, 44
- POS, 44
- ReadFractionsType, 38
- SPAN_PATTERN, 42
- SampleType, 39
- sampleTypeNames, 44
- SplicingConsequence, 39
- splicingConsequenceNames, 44
- TOKEN_SEPERATOR_CD, 44
- TOKEN_SEPERATOR_WS, 44
- TYPE_CRYPTIC, 44
- TYPE_EXONIC, 44
- TeeDevice, 38
- TeeStream, 38
- VAR_PATTERN, 42
- ValDetailsType, 38
- ValidationReasonsToAddType, 38
- YJ_LAMBDA, 44
- VeridicalTrial.cpp
 - _ETcardinality, 50
 - _SCcardinality, 50
 - _STcardinality, 50
 - ANTI_CRYPTIC_SITE_USE, 50
 - ANY, 50
 - COUNT, 50
 - CRYPTIC_SITE_USE, 50
 - DIRECT, 50
 - EXON_SKIPPING, 50
 - INTRON_INCLUSION, 50
 - INTRON_INCLUSION_WITH_MUT, 50
 - LEFT, 49
 - NONE, 50
 - NORMAL, 50
 - RIGHT, 49
 - TUMOUR, 50
- VeridicalTrial.cpp
 - ACCEPTOR, 53
 - BAMFileListType, 48
 - CHR_PREFIX, 53
 - COMMENT_CHAR, 53
 - checkExonSkipping, 51
 - countCoeff, 53
 - DECIMAL_PRECISION, 53
 - DEFAULT_P_VALUE, 53
 - DONOR, 53
 - directCoeff, 53
 - Directionality, 49
 - EXON, 53
 - EvidenceType, 49
 - evidenceTypeNames, 53
 - ExonMapType, 48
 - fullBAMFileList, 54
 - getTimestamp, 52
 - H_CHR, 54
 - H_CODING, 54
 - H_COORD_PART, 54
 - H_COORD_SS, 54
 - H_CRPTORNAT, 54
 - H_E_CHR, 54
 - H_E_ECOORD, 54
 - H_E_GENE, 54
 - H_E_NUM, 54
 - H_E_SCOORD, 54
 - H_E_TID, 54
 - H_FILE, 54
 - H_GENE, 54
 - H_HET, 54
 - H_IN_PART, 54
 - H_STRAND, 54
 - H_TYPE, 54
 - H_VAR_PART, 54
 - INTRON, 54
 - isDNANucleotide, 52
 - MIN_MAP_Q, 54
 - main, 52
 - MultiExonMapType, 49
 - NEG, 55
 - NORMAL_SAMPLE, 55
 - noFileErr, 52
 - OUTPUT_PROG_EVERY, 55
 - POS, 55
 - ReadFractionsType, 49
 - SPAN_PATTERN, 53
 - SampleType, 50
 - sampleTypeNames, 55
 - SplicingConsequence, 50
 - splicingConsequenceNames, 55
 - TOKEN_SEPERATOR_CD, 55
 - TOKEN_SEPERATOR_WS, 55
 - TYPE_CRYPTIC, 55
 - TYPE_EXONIC, 55
 - TeeDevice, 49
 - TeeStream, 49
 - VAL_JSON_PATH, 55
 - VAL_SITE, 55
 - VAR_PATTERN, 53
 - ValDetailsType, 49
 - ValidationReasonsToAddType, 49
 - YJ_LAMBDA, 55
- YJ_LAMBDA
 - Veridical.cpp, 44
 - VeridicalTrial.cpp, 55
- YJTransform
 - ValidationReasons, 32