

FastqArazketa

Generated by Doxygen 1.8.8

Thu Aug 31 2017 16:58:07

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	_adapter Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Data Documentation	5
3.1.2.1	adapter_fa	5
3.1.2.2	mismatches	5
3.1.2.3	threshold	5
3.2	_fa_data Struct Reference	5
3.2.1	Detailed Description	6
3.2.2	Member Data Documentation	6
3.2.2.1	entry	6
3.2.2.2	entrylen	6
3.2.2.3	linelen	6
3.2.2.4	nentries	6
3.2.2.5	nlines	7
3.3	_fa_entry Struct Reference	7
3.3.1	Detailed Description	7
3.3.2	Member Data Documentation	7
3.3.2.1	N	7
3.3.2.2	seq	7
3.4	_fq_read Struct Reference	7
3.4.1	Detailed Description	8
3.4.2	Member Data Documentation	8
3.4.2.1	L	8
3.4.2.2	start	8
3.5	_iparam_makeTree Struct Reference	8

3.5.1	Detailed Description	8
3.5.2	Member Data Documentation	8
3.5.2.1	inputfasta	8
3.5.2.2	L	8
3.5.2.3	outputfile	8
3.6	_iparam_Qreport Struct Reference	9
3.6.1	Detailed Description	9
3.6.2	Member Data Documentation	9
3.6.2.1	filter	9
3.6.2.2	inputfile	9
3.6.2.3	minQ	9
3.6.2.4	nQ	9
3.6.2.5	ntiles	9
3.6.2.6	one_read_len	9
3.6.2.7	outputfilebin	10
3.6.2.8	outputfilehtml	10
3.6.2.9	outputfileinfo	10
3.6.2.10	read_len	10
3.7	_iparam_Sreport Struct Reference	10
3.7.1	Detailed Description	10
3.7.2	Member Data Documentation	10
3.7.2.1	inputfolder	10
3.7.2.2	outputfile	10
3.8	_iparam_trimFilter Struct Reference	11
3.8.1	Detailed Description	11
3.8.2	Member Data Documentation	11
3.8.2.1	ad	11
3.8.2.2	globleft	12
3.8.2.3	globright	12
3.8.2.4	lfa	12
3.8.2.5	lfq	12
3.8.2.6	lidx	12
3.8.2.7	is_adapter	12
3.8.2.8	is_fa	12
3.8.2.9	is_idx	12
3.8.2.10	L	12
3.8.2.11	Lmer_len	12
3.8.2.12	method	12
3.8.2.13	minL	12
3.8.2.14	minQ	13

3.8.2.15	nlowQ	13
3.8.2.16	Oprefix	13
3.8.2.17	percent	13
3.8.2.18	score	13
3.8.2.19	trimN	13
3.8.2.20	trimQ	13
3.9	_node Struct Reference	13
3.9.1	Detailed Description	14
3.9.2	Member Data Documentation	14
3.9.2.1	children	14
3.10	_split Struct Reference	14
3.10.1	Detailed Description	14
3.10.2	Member Data Documentation	14
3.10.2.1	N	14
3.10.2.2	s	14
3.11	_stats_TF Struct Reference	14
3.11.1	Detailed Description	15
3.11.2	Member Data Documentation	15
3.11.2.1	discarded	15
3.11.2.2	filters	15
3.11.2.3	good	15
3.11.2.4	nreads	15
3.11.2.5	trimmed	15
3.12	_tree Struct Reference	15
3.12.1	Detailed Description	16
3.12.2	Member Data Documentation	16
3.12.2.1	L	16
3.12.2.2	nnodes	16
3.12.2.3	pool_2D	16
3.12.2.4	pool_available	17
3.12.2.5	pool_count	17
3.13	statsinfo Struct Reference	17
3.13.1	Detailed Description	17
3.13.2	Member Data Documentation	17
3.13.2.1	ACGT_pos	17
3.13.2.2	ACGT_tile	18
3.13.2.3	lane_tags	18
3.13.2.4	lowQ_ACGT_tile	18
3.13.2.5	minQ	18
3.13.2.6	nQ	18

3.13.2.7	nreads	18
3.13.2.8	ntiles	18
3.13.2.9	QPosTile_table	18
3.13.2.10	qual_tags	18
3.13.2.11	read_len	18
3.13.2.12	reads_MlowQ	18
3.13.2.13	reads_wN	18
3.13.2.14	sz_ACGT_pos	19
3.13.2.15	sz_ACGT_tile	19
3.13.2.16	sz_lowQ_ACGT_tile	19
3.13.2.17	sz_QPosTile_table	19
3.13.2.18	sz_reads_MlowQ	19
3.13.2.19	tile_pos	19
3.13.2.20	tile_tags	19
4	File Documentation	21
4.1	include/defines.h File Reference	21
4.1.1	Detailed Description	22
4.1.2	Macro Definition Documentation	22
4.1.2.1	ADAP	22
4.1.2.2	ALL	22
4.1.2.3	B_LEN	22
4.1.2.4	BLOOM	22
4.1.2.5	bool	22
4.1.2.6	CONT	22
4.1.2.7	DEFAULT_MINL	23
4.1.2.8	DEFAULT_MINQ	23
4.1.2.9	DEFAULT_NQ	23
4.1.2.10	DEFAULT_NTILES	23
4.1.2.11	ENDS	23
4.1.2.12	ENDSFRAC	23
4.1.2.13	ERROR	23
4.1.2.14	FA_ENTRY_BUF	23
4.1.2.15	false	23
4.1.2.16	FRAC	23
4.1.2.17	GLOBAL	23
4.1.2.18	GOOD	23
4.1.2.19	LOWQ	24
4.1.2.20	max	24
4.1.2.21	MAX_FASZ_TREE	24

4.1.2.22	MAX_FILENAME	24
4.1.2.23	MAX_RCOMMAND	24
4.1.2.24	mem_usage	24
4.1.2.25	mem_usageMB	24
4.1.2.26	min	24
4.1.2.27	N_ACGT	24
4.1.2.28	NFILTERS	24
4.1.2.29	NNNN	25
4.1.2.30	NO	25
4.1.2.31	NPOOL_1D	25
4.1.2.32	NPOOL_2D	25
4.1.2.33	SA	25
4.1.2.34	STRIP	25
4.1.2.35	T_ACGT	25
4.1.2.36	TREE	25
4.1.2.37	true	25
4.1.2.38	ZEROQ	25
4.2	include/fa_read.h File Reference	25
4.2.1	Detailed Description	27
4.2.2	Function Documentation	27
4.2.2.1	free_fasta	27
4.2.2.2	read_fasta	27
4.2.2.3	size_fasta	28
4.3	include/fopen_gen.h File Reference	29
4.3.1	Detailed Description	30
4.3.2	Function Documentation	30
4.3.2.1	fopen_gen	30
4.4	include/fq_read.h File Reference	30
4.4.1	Detailed Description	31
4.4.2	Function Documentation	31
4.4.2.1	get_fqread	31
4.4.2.2	string_seq	32
4.5	include/init_makeTree.h File Reference	32
4.5.1	Detailed Description	33
4.6	include/init_Qreport.h File Reference	33
4.6.1	Detailed Description	35
4.7	include/init_Sreport.h File Reference	35
4.7.1	Detailed Description	36
4.8	include/init_trimFilter.h File Reference	36
4.8.1	Detailed Description	37

4.8.2	Typedef Documentation	38
4.8.2.1	Adapter	38
4.9	include/io_trimFilter.h File Reference	38
4.9.1	Detailed Description	39
4.9.2	Function Documentation	39
4.9.2.1	buffer_output	39
4.10	include/Lmer.h File Reference	39
4.10.1	Detailed Description	40
4.10.2	Function Documentation	40
4.10.2.1	init_map	40
4.10.2.2	init_map_SA	40
4.11	include/Rcommand_Qreport.h File Reference	40
4.11.1	Detailed Description	41
4.12	include/Rcommand_Sreport.h File Reference	41
4.12.1	Detailed Description	42
4.13	include/stats_info.h File Reference	42
4.13.1	Detailed Description	44
4.13.2	Function Documentation	44
4.13.2.1	init_info	44
4.13.2.2	resize_info	44
4.13.2.3	update_ACGT_counts	44
4.14	include/str_manip.h File Reference	45
4.14.1	Detailed Description	45
4.14.2	Function Documentation	45
4.14.2.1	strindex	45
4.14.2.2	strsplit	46
4.15	include/tree.h File Reference	46
4.15.1	Detailed Description	48
4.15.2	Typedef Documentation	48
4.15.2.1	Tree	48
4.15.3	Function Documentation	48
4.15.3.1	check_path	48
4.15.3.2	free_all_nodes	48
4.15.3.3	get_new_pool	48
4.15.3.4	new_node_buf	49
4.15.3.5	read_tree	49
4.15.3.6	save_tree	49
4.15.3.7	tree_from_fasta	49
4.16	include/trim.h File Reference	50
4.16.1	Detailed Description	51

4.16.2	Function Documentation	51
4.16.2.1	is_read_inTree	51
4.16.2.2	trim_sequenceN	51
4.16.2.3	trim_sequenceQ	52
4.17	src/fa_read.c File Reference	53
4.17.1	Detailed Description	54
4.17.2	Function Documentation	54
4.17.2.1	free_fasta	54
4.17.2.2	ignore_line	54
4.17.2.3	init_entries	54
4.17.2.4	init_fa	55
4.17.2.5	read_fasta	55
4.17.2.6	realloc_fa	55
4.17.2.7	size_fasta	55
4.17.2.8	sweep_fa	56
4.17.3	Variable Documentation	56
4.17.3.1	alloc_mem	56
4.18	src/fopen_gen.c File Reference	56
4.18.1	Detailed Description	57
4.18.2	Function Documentation	57
4.18.2.1	compress	57
4.18.2.2	fcompress	58
4.18.2.3	fopen_gen	58
4.18.2.4	funcompress	58
4.18.2.5	uncompress	58
4.19	src/fq_read.c File Reference	58
4.19.1	Detailed Description	59
4.19.2	Function Documentation	59
4.19.2.1	get_fqread	59
4.19.2.2	string_seq	60
4.20	src/init_makeTree.c File Reference	60
4.20.1	Detailed Description	61
4.20.2	Variable Documentation	61
4.20.2.1	par_MT	61
4.21	src/init_Qreport.c File Reference	61
4.21.1	Detailed Description	62
4.21.2	Variable Documentation	62
4.21.2.1	par_QR	62
4.22	src/init_Sreport.c File Reference	62
4.22.1	Detailed Description	63

4.22.2	Variable Documentation	63
4.22.2.1	par_SR	63
4.23	src/init_trimFilter.c File Reference	63
4.23.1	Detailed Description	64
4.23.2	Variable Documentation	64
4.23.2.1	par_TF	64
4.24	src/io_trimFilter.c File Reference	64
4.24.1	Detailed Description	65
4.24.2	Function Documentation	65
4.24.2.1	buffer_output	65
4.25	src/Lmer.c File Reference	65
4.25.1	Detailed Description	66
4.25.2	Function Documentation	66
4.25.2.1	init_map	66
4.25.2.2	init_map_SA	66
4.25.3	Variable Documentation	66
4.25.3.1	LT	66
4.25.3.2	Nencode	66
4.26	src/makeTree.c File Reference	67
4.26.1	Detailed Description	67
4.26.2	Variable Documentation	68
4.26.2.1	alloc_mem	68
4.26.2.2	par_MT	68
4.27	src/Qreport.c File Reference	68
4.27.1	Detailed Description	68
4.27.2	Variable Documentation	69
4.27.2.1	par_QR	69
4.28	src/Rcommand_Sreport.c File Reference	69
4.28.1	Detailed Description	69
4.28.2	Variable Documentation	70
4.28.2.1	par_SR	70
4.29	src/Sreport.c File Reference	70
4.29.1	Detailed Description	70
4.29.2	Variable Documentation	71
4.29.2.1	par_SR	71
4.30	src/stats_info.c File Reference	71
4.30.1	Detailed Description	72
4.30.2	Function Documentation	72
4.30.2.1	get_tile_lane	72
4.30.2.2	init_info	73

4.30.2.3	resize_info	73
4.30.2.4	update_ACGT_counts	73
4.30.3	Variable Documentation	73
4.30.3.1	par_QR	73
4.31	src/str_manip.c File Reference	73
4.31.1	Detailed Description	74
4.31.2	Function Documentation	74
4.31.2.1	strindex	74
4.31.2.2	strindexC	74
4.31.2.3	strsplit	74
4.32	src/tree.c File Reference	75
4.32.1	Detailed Description	76
4.32.2	Function Documentation	76
4.32.2.1	check_path	76
4.32.2.2	free_all_nodes	76
4.32.2.3	get_new_pool	76
4.32.2.4	new_node_buf	76
4.32.2.5	read_tree	77
4.32.2.6	save_tree	77
4.32.2.7	tree_from_fasta	77
4.32.3	Variable Documentation	77
4.32.3.1	alloc_mem	77
4.33	src/trim.c File Reference	78
4.33.1	Detailed Description	79
4.33.2	Macro Definition Documentation	79
4.33.2.1	TRIM_STRING	79
4.33.3	Function Documentation	79
4.33.3.1	is_read_inTree	79
4.33.3.2	Nfree_Lmer	79
4.33.3.3	no_lowQ	80
4.33.3.4	no_N	80
4.33.3.5	Ntrim_ends	80
4.33.3.6	Qtrim_ends	80
4.33.3.7	Qtrim_endsfrac	81
4.33.3.8	Qtrim_frac	81
4.33.3.9	Qtrim_global	81
4.33.3.10	trim_sequenceN	81
4.33.3.11	trim_sequenceQ	82
4.33.4	Variable Documentation	82
4.33.4.1	LT	82

4.33.4.2	Nencode	82
4.33.4.3	par_TF	82
4.34	src/trimFilter.c File Reference	83
4.34.1	Detailed Description	83
4.34.2	Variable Documentation	84
4.34.2.1	alloc_mem	84
4.34.2.2	par_TF	84
Index		85

Chapter 1

Class Index

1.1 Class List

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

_adapter	5
_fa_data	5
Stores sequences of a fasta file	
_fa_entry	7
Fasta entry	
_fq_read	7
Stores a fastq entry	
_iparam_makeTree	8
MakeTree input parameters	
_iparam_Qreport	9
Qreport input parameters	
_iparam_Sreport	10
Sreport input parameters	
_iparam_trimFilter	11
TrimFilter input parameters	
_node	13
Node structure: formed out of T_ACGT pointers to Node structure	
_split	14
Splitted string and the number of splitted fields	
_stats_TF	14
Collects stats info from the filtering procedure	
_tree	15
Structure containing a T_ACGT-tree	
statsinfo	17
Stores info needed to create the summary graphs	

Chapter 2

File Index

2.1 File List

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

include/defines.h	
Macro definitions	21
include/fa_read.h	
Reads in and stores fasta files	25
include/fopen_gen.h	
Uncompress/compress input/output files using pipes	29
include/fq_read.h	
Fastq entries manipulations (read/write)	30
include/init_makeTree.h	
Help dialog for makeTree and initialization of the command line arguments	32
include/init_Qreport.h	
Header file: help dialog for Qreport and initialization of the command line arguments	33
include/init_Sreport.h	
Help dialog for Sreport and initialization of the command line arguments	35
include/init_trimFilter.h	
Help dialog for trimFilter and initialization of the command line arguments	36
include/io_trimFilter.h	
Buffer fq output, write summary file	38
include/Lmer.h	
Manipulation of Lmers and sequences	39
include/Rcommand_Qreport.h	
Get Rscript command for Qreport	40
include/Rcommand_Sreport.h	
Get Rscript command for Sreport	41
include/stats_info.h	
Construct the quality report variables and update them	42
include/str_manip.h	
Functions that do string manipulation	45
include/tree.h	
Construction of tree, check paths, write tree, read in tree	46
include/trim.h	
Trims/filter sequences after Quality, N's contaminations	50
src/fa_read.c	
Reads in and stores fasta files	53
src/fopen_gen.c	
Uncompress/compress input/output files using pipes	56
src/fq_read.c	
Fastq entries manipulations (read/write)	58

src/init_makeTree.c	Help dialog for makeTree and initialization of the command line arguments	60
src/init_Qreport.c	Help dialog for Qreport and initialization of the command line arguments	61
src/init_Sreport.c	Help dialog for Sreport and initialization of the command line arguments	62
src/init_trimFilter.c	Help dialog for trimFilter and initialization of the command line arguments	63
src/io_trimFilter.c	Buffer fq output, write summary file	64
src/Lmer.c	Manipulation of Lmers and sequences	65
src/makeTree.c	MakeTree main function	67
src/Qreport.c	QReport main function	68
src/Rcommand_Sreport.c	Get Rscript command for Sreport	69
src/Sreport.c	Sreport main function	70
src/stats_info.c	Construct the quality report variables and update them	71
src/str_manip.c	Functions that do string manipulation	73
src/tree.c	Construction of tree, check paths, write tree, read in tree	75
src/trim.c	Trims/filter sequences after Quality, N's contaminations	78
src/trimFilter.c	TrimFilter main function	83

Chapter 3

Class Documentation

3.1 `_adapter` Struct Reference

```
#include <init_trimFilter.h>
```

Public Attributes

- char * [adapter_fa](#)
- int [mismatches](#)
- int [threshold](#)

3.1.1 Detailed Description

@ brief adapter struct @ note UNFINISHED!

3.1.2 Member Data Documentation

3.1.2.1 char* `_adapter::adapter_fa`

fasta file containing adapters

3.1.2.2 int `_adapter::mismatches`

Number of allowed mismatches

3.1.2.3 int `_adapter::threshold`

Score threshold

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

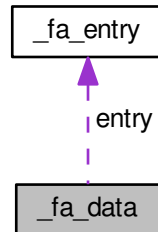
- [include/init_trimFilter.h](#)

3.2 `_fa_data` Struct Reference

stores sequences of a fasta file

```
#include <fa_read.h>
```

Collaboration diagram for `_fa_data`:



Public Attributes

- `uint64_t nlines`
- `int nentries`
- `int linelen`
- `uint64_t * entrylen`
- `Fa_entry * entry`

3.2.1 Detailed Description

stores sequences of a fasta file

3.2.2 Member Data Documentation

3.2.2.1 `Fa_entry* _fa_data::entry`

Array with fasta entries (see `Fa_entry`)

3.2.2.2 `uint64_t* _fa_data::entrylen`

Array containing the length of the entries

3.2.2.3 `int _fa_data::linelen`

Line length of the *fa file entries

3.2.2.4 `int _fa_data::nentries`

Number of entries in *fa file

3.2.2.5 `uint64_t _fa_data::nlines`

Number of lines in `*fa` file

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

- `include/fa_read.h`

3.3 `_fa_entry` Struct Reference

fasta entry

```
#include <fa_read.h>
```

Public Attributes

- `uint64_t N`
- `char * seq`

3.3.1 Detailed Description

fasta entry

3.3.2 Member Data Documentation

3.3.2.1 `uint64_t _fa_entry::N`

Entry length (chars)

3.3.2.2 `char* _fa_entry::seq`

sequence

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

- `include/fa_read.h`

3.4 `_fq_read` Struct Reference

stores a fastq entry

```
#include <fq_read.h>
```

Public Attributes

- `char line1 [READ_MAXLEN]`
- `char line2 [READ_MAXLEN]`
- `char line3 [READ_MAXLEN]`
- `char line4 [READ_MAXLEN]`
- `int L`
- `int start`

3.4.1 Detailed Description

stores a fastq entry

3.4.2 Member Data Documentation

3.4.2.1 `int _fq_read::L`

read length

3.4.2.2 `int _fq_read::start`

nucleotide position start. Can only be different from zero if the read has been filtered with this tool.

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

- [include/fq_read.h](#)

3.5 `_iparam_makeTree` Struct Reference

contains makeTree input parameters

```
#include <init_makeTree.h>
```

Public Attributes

- `char * inputfasta`
- `char outputfile [MAX_FILENAME]`
- `int L`

3.5.1 Detailed Description

contains makeTree input parameters

3.5.2 Member Data Documentation

3.5.2.1 `char* _iparam_makeTree::inputfasta`

fasta input file

3.5.2.2 `int _iparam_makeTree::L`

tree depth

3.5.2.3 `char _iparam_makeTree::outputfile[MAX_FILENAME]`

outputfile path

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

- [include/init_makeTree.h](#)

3.6 _iparam_Qreport Struct Reference

contains Qreport input parameters

```
#include <init_Qreport.h>
```

Public Attributes

- char * [inputfile](#)
- char [outputfilebin](#) [MAX_FILENAME]
- char [outputfilehtml](#) [MAX_FILENAME]
- char [outputfileinfo](#) [MAX_FILENAME]
- int [nQ](#)
- int [ntiles](#)
- int [minQ](#)
- int [read_len](#)
- int [filter](#)
- int [one_read_len](#)

3.6.1 Detailed Description

contains Qreport input parameters

3.6.2 Member Data Documentation

3.6.2.1 int _iparam_Qreport::filter

0 original data, 1 this tool filtered data, 2 other tool filtered data

3.6.2.2 char* _iparam_Qreport::inputfile

Inputfile name

3.6.2.3 int _iparam_Qreport::minQ

minimum Quality allowed 0 - 45

3.6.2.4 int _iparam_Qreport::nQ

different quality values (default is 46)

3.6.2.5 int _iparam_Qreport::ntiles

tiles (default is 96)

3.6.2.6 int _iparam_Qreport::one_read_len

1 all reads of equal length 0 reads have different lengths.

3.6.2.7 char _iparam_Qreport::outputfilebin[MAX_FILENAME]

Binary outputfile name.

3.6.2.8 char _iparam_Qreport::outputfilehtml[MAX_FILENAME]

html outputfile name

3.6.2.9 char _iparam_Qreport::outputfileinfo[MAX_FILENAME]

Info outputfile name

3.6.2.10 int _iparam_Qreport::read_len

original read length

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

- include/[init_Qreport.h](#)

3.7 _iparam_Sreport Struct Reference

contains Sreport input parameters

```
#include <init_Sreport.h>
```

Public Attributes

- char * [inputfolder](#)
- char [outputfile](#) [MAX_FILENAME]

3.7.1 Detailed Description

contains Sreport input parameters

3.7.2 Member Data Documentation

3.7.2.1 char* _iparam_Sreport::inputfolder

input folder

3.7.2.2 char _iparam_Sreport::outputfile[MAX_FILENAME]

html outputfile name

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

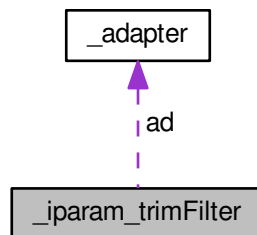
- include/[init_Sreport.h](#)

3.8 _iparam_trimFilter Struct Reference

trimFilter input parameters

```
#include <init_trimFilter.h>
```

Collaboration diagram for _iparam_trimFilter:



Public Attributes

- char * `lfq`
- char * `lfa`
- char * `lidx`
- char * `Oprefix`
- Adapter `ad`
- int `trimQ`
- int `trimN`
- int `method`
- bool `is_fa`
- bool `is_idx`
- bool `is_adapter`
- double `score`
- int `minQ`
- int `L`
- int `minL`
- int `nlowQ`
- int `Lmer_len`
- int `globleft`
- int `globright`
- int `percent`

3.8.1 Detailed Description

trimFilter input parameters

3.8.2 Member Data Documentation

3.8.2.1 Adapter _iparam_trimFilter::ad

Adapter trimming parameters

3.8.2.2 int _iparam_trimFilter::globleft

number of bases globally trimming from the left

3.8.2.3 int _iparam_trimFilter::globright

number of bases globally trimming from the right

3.8.2.4 char* _iparam_trimFilter::lfa

Input fa file (containing contamination sequences)

3.8.2.5 char* _iparam_trimFilter::lfq

Input fq file

3.8.2.6 char* _iparam_trimFilter::lidx

Input index file (constructed from an input.fa cont file)

3.8.2.7 bool _iparam_trimFilter::is_adapter

true if filtering adapter sequences

3.8.2.8 bool _iparam_trimFilter::is_fa

true if a fasta file was passed as a parameter

3.8.2.9 bool _iparam_trimFilter::is_idx

true if an index file was passed as a parameter

3.8.2.10 int _iparam_trimFilter::L

read length

3.8.2.11 int _iparam_trimFilter::Lmer_len

Lmer length to look for contamination

3.8.2.12 int _iparam_trimFilter::method

[TREE\(1\)](#), [SA\(2\)](#), [BLOOM\(3\)](#), 0, when not looking for cont

3.8.2.13 int _iparam_trimFilter::minL

minimum read length accepted before discarding a read

3.8.2.14 int _iparam_trimFilter::minQ

minimum quality threshold

3.8.2.15 int _iparam_trimFilter::nlowQ

maximum number of lowQ bases accepted before discarding

3.8.2.16 char* _iparam_trimFilter::Oprefix

Output files prefix (PATH/prefix)

3.8.2.17 int _iparam_trimFilter::percent

percentage of lowQ bases allowed in a read

3.8.2.18 double _iparam_trimFilter::score

score threshold for matching reads in sequences

3.8.2.19 int _iparam_trimFilter::trimN

[NO\(0\)](#), [ALL\(1\)](#), [ENDS\(2\)](#), [STRIP\(3\)](#)

3.8.2.20 int _iparam_trimFilter::trimQ

[NO\(0\)](#), [FRAC\(1\)](#), [ENDS\(2\)](#), [ENDSFRAC\(3\)](#), [GLOBAL\(4\)](#)

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

- [include/init_trimFilter.h](#)

3.9 _node Struct Reference

Node structure: formed out of T_ACGT pointers to Node structure.

```
#include <tree.h>
```

Collaboration diagram for _node:



Public Attributes

- [struct _node * children](#) [[T_ACGT](#)]

3.9.1 Detailed Description

Node structure: formed out of T_ACGT pointers to Node structure.

3.9.2 Member Data Documentation

3.9.2.1 struct _node* _node::children[T_ACGT]

T_ACGT pointers to Node structure

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

- include/[tree.h](#)

3.10 _split Struct Reference

contains a splitted string and the number or splitted fields

```
#include <str_manip.h>
```

Public Attributes

- int [N](#)
- char ** [s](#)

3.10.1 Detailed Description

contains a splitted string and the number or splitted fields

3.10.2 Member Data Documentation

3.10.2.1 int _split::N

Number of substrings in which the string was splitted

3.10.2.2 char** _split::s

Substring array containing the splitted substrings

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

- include/[str_manip.h](#)

3.11 _stats_TF Struct Reference

collects stats info from the filtering procedure

```
#include <io_trimFilter.h>
```

Public Attributes

- [bool filters](#) [NFILTERS]
- [int trimmed](#) [NFILTERS]
- [int discarded](#) [NFILTERS]
- [int good](#)
- [int nreads](#)

3.11.1 Detailed Description

collects stats info from the filtering procedure

3.11.2 Member Data Documentation

3.11.2.1 `int _stats_TF::discarded[NFILTERS]`

discarded reads: ADAP, CONT, LOWQ, NNNN

3.11.2.2 `bool _stats_TF::filters[NFILTERS]`

Using filters for: ADAP, CONT, LOWQ, NNNN

3.11.2.3 `int _stats_TF::good`

good reads

3.11.2.4 `int _stats_TF::nreads`

total number of reads in the fq file

3.11.2.5 `int _stats_TF::trimmed[NFILTERS]`

trimmed reads by: ADAP, CONT, LOWQ, NNNN

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

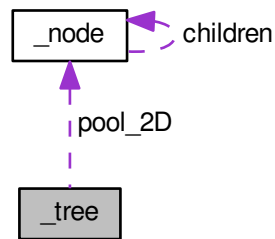
- [include/io_trimFilter.h](#)

3.12 _tree Struct Reference

structure containing a T_ACGT-tree.

```
#include <tree.h>
```

Collaboration diagram for `_tree`:



Public Attributes

- `uint32_t L`
- `uint32_t pool_count`
- `uint32_t pool_available`
- `uint32_t nnodes`
- `Node ** pool_2D`

3.12.1 Detailed Description

structure containing a T_ACGT-tree.

The tree structure is stored in a pointer to pointer to Node. We grow the structure on the flight as we need more memory. In the outer direction, we start by allocating `NPOOL_2D` pointers to Node. In the inner direction, we allocate `NPOOL_1D` Nodes and fill them as we read the fasta file. When all of them are allocated, we allocate again `NPOOL_1D`. If `NPOOL_2D` pointers to Node are allocated, the outer dimension is reallocated with `+NPOOL_2D` extra elements. `L` is the depth of the tree, `pool_count` is the number on Node* elements used so far, `pool_available` is the number of Nodes available in every moment, and `nnodes` is the total number of nodes filled in. We limit the number of allocated nodes to `UINT_MAX` (we cannot count more nodes!).

3.12.2 Member Data Documentation

3.12.2.1 `uint32_t _tree::L`

depth of the tree

3.12.2.2 `uint32_t _tree::nnodes`

Number of nodes in the tree

3.12.2.3 `Node** _tree::pool_2D`

2D pool containing the nodes that form the tree

3.12.2.4 uint32_t tree::pool_available

Number of empty nodes available in the pool

3.12.2.5 uint32_t tree::pool_count

Number of elements in the second dimension

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

- include/[tree.h](#)

3.13 statsinfo Struct Reference

stores info needed to create the summary graphs

```
#include <stats_info.h>
```

Public Attributes

- int [read_len](#)
- int [ntiles](#)
- int [nQ](#)
- int [minQ](#)
- int [tile_pos](#)
- int [nreads](#)
- int [reads_wN](#)
- int [sz_lowQ_ACGT_tile](#)
- int [sz_ACGT_tile](#)
- int [sz_reads_MlowQ](#)
- int [sz_QPosTile_table](#)
- int [sz_ACGT_pos](#)
- int * [tile_tags](#)
- int * [lane_tags](#)
- int * [qual_tags](#)
- uint64_t * [lowQ_ACGT_tile](#)
- uint64_t * [ACGT_tile](#)
- uint64_t * [reads_MlowQ](#)
- uint64_t * [QPosTile_table](#)
- uint64_t * [ACGT_pos](#)

3.13.1 Detailed Description

stores info needed to create the summary graphs

3.13.2 Member Data Documentation

3.13.2.1 uint64_t* statsinfo::ACGT_pos

A, C, G, T, N per position

3.13.2.2 uint64_t* statsinfo::ACGT_tile

A, C, G, T, N per tile, to compute the fraction of lowQuality bases per tile and per nucleotide.

3.13.2.3 int* statsinfo::lane_tags

Names of the existing tiles

3.13.2.4 uint64_t* statsinfo::lowQ_ACGT_tile

low Quality A, C, G, T, N per tile

3.13.2.5 int statsinfo::minQ

Minimum quality threshold

3.13.2.6 int statsinfo::nQ

possible quality values

3.13.2.7 int statsinfo::nreads

reads read till current position.

3.13.2.8 int statsinfo::ntiles

tiles

3.13.2.9 uint64_t* statsinfo::QPosTile_table

bases of a given quality per tile.

3.13.2.10 int* statsinfo::qual_tags

Names of the existing qualities

3.13.2.11 int statsinfo::read_len

Maximum length of a read

3.13.2.12 uint64_t* statsinfo::reads_MlowQ

reads with M(position) lowQuality bases.

3.13.2.13 int statsinfo::reads_wN

reads with N's found till current position

3.13.2.14 int statsinfo::sz_ACGT_pos

ACGT_pos size = read_len * N_ACGT

3.13.2.15 int statsinfo::sz_ACGT_tile

ACGT_tile size = ntiles * N_ACGT

3.13.2.16 int statsinfo::sz_lowQ_ACGT_tile

lowQ_ACGT_tile size = ntiles * N_ACGT

3.13.2.17 int statsinfo::sz_QPosTile_table

QposTile_Table size = ntiles * nQ * read_len

3.13.2.18 int statsinfo::sz_reads_MlowQ

reads_MlowQ size = read_len + 1

3.13.2.19 int statsinfo::tile_pos

current tile position

3.13.2.20 int* statsinfo::tile_tags

Names of the existing tiles

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

- include/[stats_info.h](#)

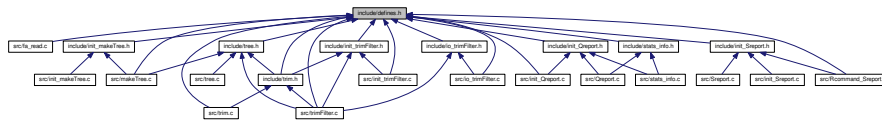
Chapter 4

File Documentation

4.1 include/defines.h File Reference

Macro definitions.

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



Macros

- `#define B_LEN 131072`
- `#define MAX_FILENAME 300`
- `#define bool short`
- `#define true 1`
- `#define false 0`
- `#define max(a, b) (((a) > (b)) ? (a) : (b))`
- `#define min(a, b) (((a) < (b)) ? (a) : (b))`
- `#define mem_usageMB()`
- `#define mem_usage()`
- `#define DEFAULT_MINQ 27`
- `#define DEFAULT_NTILES 96`
- `#define DEFAULT_NQ 46`
- `#define ZEROQ 33`
- `#define N_ACGT 5`
- `#define MAX_RCOMMAND 4000`
- `#define FA_ENTRY_BUF 20`
- `#define T_ACGT 4`
- `#define NPOOL_1D 1048576`
- `#define NPOOL_2D 16`
- `#define MAX_FASZ_TREE 1e7`
- `#define NO 0`
- `#define ALL 1`
- `#define ENDS 2`
- `#define STRIP 3`

- `#define FRAC 3`
- `#define ENDSFRAC 4`
- `#define GLOBAL 5`
- `#define TREE 1`
- `#define SA 2`
- `#define BLOOM 3`
- `#define ERROR 1000`
- `#define DEFAULT_MINL 25`
- `#define ADAP 0`
- `#define CONT 1`
- `#define LOWQ 2`
- `#define NNNN 3`
- `#define GOOD 4`
- `#define NFILTERS 4`

4.1.1 Detailed Description

Macro definitions.

Author

Paula Perez paulaperezrubio@gmail.com

Date

07.08.2017

4.1.2 Macro Definition Documentation

4.1.2.1 `#define ADAP 0`

Adapter filter

4.1.2.2 `#define ALL 1`

Trims if a lowQ base calling | N is found

4.1.2.3 `#define B_LEN 131072`

buffer size

4.1.2.4 `#define BLOOM 3`

Use a bloom filter to look for contaminations

4.1.2.5 `#define bool short`

define a bool type

4.1.2.6 `#define CONT 1`

Contamination filter

4.1.2.7 #define DEFAULT_MINL 25

Default minimum length under which we discard the reads

4.1.2.8 #define DEFAULT_MINQ 27

Minimum quality threshold

4.1.2.9 #define DEFAULT_NQ 46

Default number of different quality values

4.1.2.10 #define DEFAULT_NTILES 96

Default number of tiles

4.1.2.11 #define ENDS 2

Trims at the ends

4.1.2.12 #define ENDSFRAC 4

trims at the ends and discards a read if the remaining part has more than > percent lowQ bases

4.1.2.13 #define ERROR 1000

Encodes an error when reading in trimN, trimQ, method options in trimFilter

4.1.2.14 #define FA_ENTRY_BUF 20

buffer for fasta entries

4.1.2.15 #define false 0

assign false to 0

4.1.2.16 #define FRAC 3

Discards a read if it contains > percent lowQ bases

4.1.2.17 #define GLOBAL 5

Trims a fixed # bases from e left and right

4.1.2.18 #define GOOD 4

Good reads

4.1.2.19 #define LOWQ 2

Low quality filter

4.1.2.20 #define max(a, b) (((a) > (b)) ? (a) : (b))

max function

4.1.2.21 #define MAX_FASZ_TREE 1e7

Maximum fasta size for constructing a tree. DECIDE A SENSIBLE SIZE!

4.1.2.22 #define MAX_FILENAME 300

Maximum # chars in a filename

4.1.2.23 #define MAX_RCOMMAND 4000

Maximum # chars in R command

4.1.2.24 #define mem_usage()

Value:

```
fprintf(stderr, \
    "- Current allocated memory: %ld Bytes.\n", \
    alloc_mem)
```

returns allocated memory in Bytes

4.1.2.25 #define mem_usageMB()

Value:

```
fprintf(stderr, \
    "- Current allocated memory: %ld MB.\n", \
    alloc_mem >> 20)
```

returns allocated memory in MB

4.1.2.26 #define min(a, b) (((a) < (b)) ? (a) : (b))

min function

4.1.2.27 #define N_ACGT 5

Number of different nucleotides in the fq file

4.1.2.28 #define NFILTERS 4

total number of filters

4.1.2.29 #define NNNN 3

N's presence filter

4.1.2.30 #define NO 0

No trimming

4.1.2.31 #define NPOOL_1D 1048576

Number of Node structs allocated in inner dim

4.1.2.32 #define NPOOL_2D 16

Number of *Node allocated in outer dim

4.1.2.33 #define SA 2

Use a suffix array to look for contaminations

4.1.2.34 #define STRIP 3

Looks for the largest N-free sequence

4.1.2.35 #define T_ACGT 4

Number of children per node in tree

4.1.2.36 #define TREE 1

Use a tree to look for contaminations

4.1.2.37 #define true 1

assign true to 1

4.1.2.38 #define ZEROQ 33

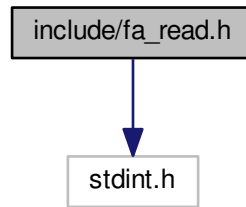
ASCII code of lowest quality value (!)

4.2 include/fa_read.h File Reference

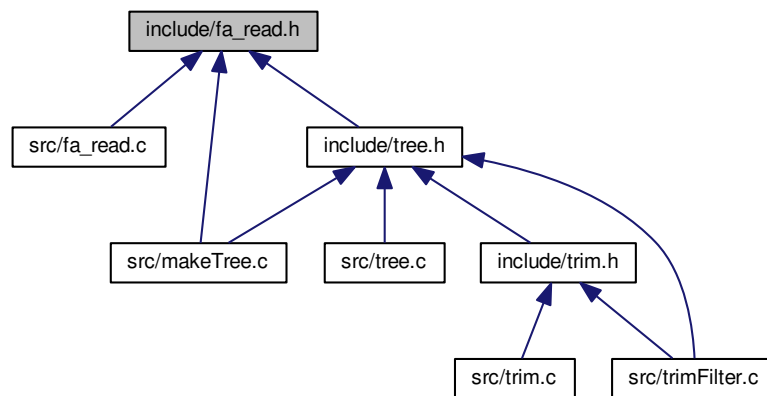
reads in and stores fasta files

```
#include <stdint.h>
```

Include dependency graph for fa_read.h:



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



Classes

- struct [_fa_entry](#)
fasta entry
- struct [_fa_data](#)
stores sequences of a fasta file

Typedefs

- typedef struct [_fa_entry](#) [Fa_entry](#)
fasta entry
- typedef struct [_fa_data](#) [Fa_data](#)
stores sequences of a fasta file

Functions

- int [read_fasta](#) (char *filename, [Fa_data](#) *ptr_fa)
reads a fasta file and stores the contents in a Fa_data structure.
- uint64_t [size_fasta](#) ([Fa_data](#) *ptr_fa)
computes length of genome in fasta structure
- void [free_fasta](#) ([Fa_data](#) *ptr_fa)
free fasta file

4.2.1 Detailed Description

reads in and stores fasta files

Author

Paula Perez paulaperezrubio@gmail.com

Date

16.08.2017

4.2.2 Function Documentation

4.2.2.1 void free_fasta ([Fa_data](#) * *ptr_fa*)

free fasta file

Parameters

<i>ptr_fa</i>	pointer to Fa_data structure.
---------------	---

The dynamically allocated memory in a [Fa_data](#) struct is deallocated and counted, so that we can

4.2.2.2 int read_fasta (char * *filename*, [Fa_data](#) * *ptr_fa*)

reads a fasta file and stores the contents in a [Fa_data](#) structure.

Parameters

<i>filename</i>	path to a fasta input file.
<i>ptr_fa</i>	pointer to Fa_data structure.

Returns

number of entries in the fasta file.

A fasta file is read and stored in a structure [Fa_data](#). The basic problem with reading FASTA files is that there is no end-of-record indicator. When you're reading sequence n, you don't know you're done until you've read the header line for sequence n+1, which you won't parse 'til later (when you're reading in the sequence n+1). The solution implemented here is to read the file twice. The first time, ([sweep_fa](#)), we initialize [Fa_data](#) and store the parameters:

- nlines: number of lines of the fasta file.
- nentries: number of entries in the fasta file.
- linelen: length of a line in the considered fasta file.
- entrylen: array containing the lengths of every entry. With this information, the pointer to [Fa_entry](#) can be allocated and the file is read again and the entries are stored in the structure.

4.2.2.3 `uint64_t size_fasta (Fa_data * ptr_fa)`

computes length of genome in fasta structure

Parameters

<i>ptr_fa</i>	pointer to Fa_data
---------------	--------------------

Returns

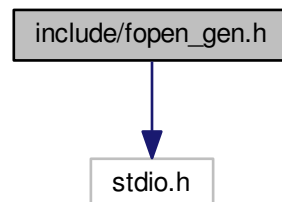
total number of nucleotides

4.3 include/fopen_gen.h File Reference

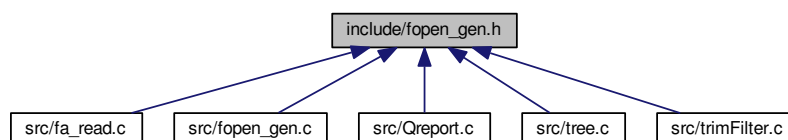
Uncompress/compress input/output files using pipes.

```
#include <stdio.h>
```

Include dependency graph for fopen_gen.h:



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



Macros

- `#define READ_END 0`
- `#define WRITE_END 1`
- `#define PERMISSIONS 0640`

Functions

- `int setCloexec (int fd)`
- `FILE * fopen_gen (const char *path, const char *mode)`

Generalized fopen function. fopen_gen is to be used as fopen. Can be used in read and in write mode. When used in read mode with a compressed extension, the file will be first decompressed and then read. When used in write mode with a compressed extension, the output will be compressed.

4.3.1 Detailed Description

Uncompress/compress input/output files using pipes.

Hook the standard file opening functions, `open`, `fopen` and `fopen64`. If the extension of the file being opened indicates the file is compressed (.gz, .bz2, .xz), when opening in the reading mode a pipe to a program is opened that decompresses that file (gunzip, bunzip2 or xzdec) and return a handle to the open pipe. When opening in the writing mode (only for .gz, .bam), a pipe to a program is opened that compresses the output.

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

Warning

`vfork` vs `fork` to be checked!

Note

- original copyright note - (reading mode, original C++ code) author: Shaun Jackman sjackman@bcgsc.ca, <https://github.com/bcgsc>, filename: Uncompress.cpp

4.3.2 Function Documentation

4.3.2.1 FILE* fopen_gen (const char * *path*, const char * *mode*)

Generalized `fopen` function. `fopen_gen` is to be used as `fopen`. Can be used in read and in write mode. When used in read mode with a compressed extension, the file will be first decompressed and then read. When used in write mode with a compressed extension, the output will be compressed.

Returns

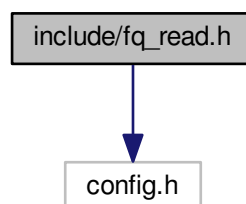
a FILE pointer

4.4 include/fq_read.h File Reference

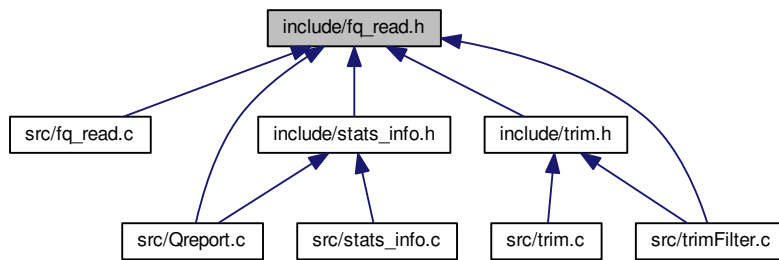
fastq entries manipulations (read/write)

```
#include "config.h"
```

Include dependency graph for `fq_read.h`:



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



Classes

- `struct _fq_read`
stores a fastq entry

Typedefs

- `typedef struct _fq_read Fq_read`
stores a fastq entry

Functions

- `int get_fqread (Fq_read *seq, char *buffer, int pos1, int pos2, int nline, int read_len, int filter)`
reads fastq line from a buffer
- `int string_seq (Fq_read *seq, char *char_seq)`
writes the fq entry in a string

4.4.1 Detailed Description

fastq entries manipulations (read/write)

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.4.2 Function Documentation

4.4.2.1 `int get_fqread (Fq_read * seq, char * buffer, int pos1, int pos2, int nline, int read_len, int filter)`

reads fastq line from a buffer

a fastq line is read from a buffer and the relevant information is stored in a structure **Fq_read**. Depending on the value of **filter**, information about whether the read was trimmed is stored.

Parameters

<i>*seq</i>	pointer to Fq_read , where the info will be stored.
<i>buffer</i>	variable where the file being read is stored.
<i>pos1</i>	buffer start position of the line.
<i>pos2</i>	buffer end position of the line.
<i>nline</i>	file line number being read.
<i>read_len</i>	predefined read length
<i>filter</i>	0 original file, 1 file filtered with filter_trim, 2 file filtered with another tool

4.4.2.2 int string_seq (Fq_read * seq, char * char_seq)

writes the fq entry in a string

Parameters

<i>*seq</i>	pointer to Fq_read , where the info will be stored.
<i>char_seq</i>	pointer to buffer, where the sequence will be stored

Warning

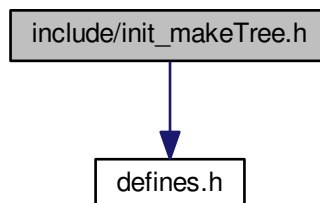
change the call to sprintf to snprintf

4.5 include/init_makeTree.h File Reference

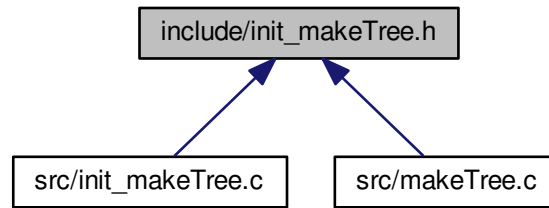
Help dialog for makeTree and initialization of the command line arguments.

```
#include "defines.h"
```

Include dependency graph for init_makeTree.h:



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



Classes

- struct [_iparam_makeTree](#)
contains makeTree input parameters

Typedefs

- typedef struct [_iparam_makeTree](#) [iparam_makeTree](#)
contains makeTree input parameters

Functions

- void [printHelpDialog_makeTree](#) ()
Function that prints makeTree help dialog when called.
- void [getarg_makeTree](#) (int argc, char **argv)
Reads in the arguments passed through the command line to makeTree. and stores them in the global variable par_MT.

4.5.1 Detailed Description

Help dialog for makeTree and initialization of the command line arguments.

Author

Paula Perez paulaperezrubio@gmail.com

Date

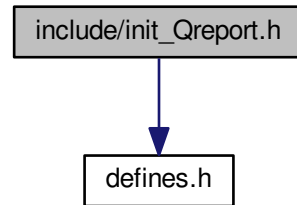
23.08.2017

4.6 include/init_Qreport.h File Reference

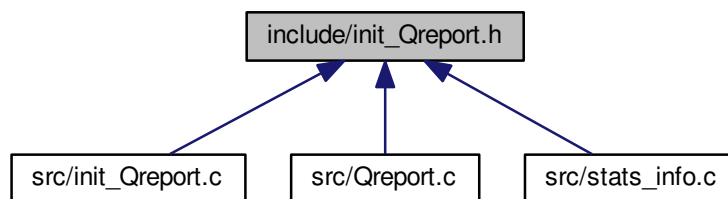
Header file: help dialog for Qreport and initialization of the command line arguments.

```
#include "defines.h"
```

Include dependency graph for init_Qreport.h:



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



Classes

- struct [_iparam_Qreport](#)
contains Qreport input parameters

Typedefs

- typedef struct [_iparam_Qreport](#) [Iparam_Qreport](#)
contains Qreport input parameters

Functions

- void [printHelpDialog_Qreport](#) ()
Function that prints Qreport help dialog when called.
- void [getarg_Qreport](#) (int argc, char **argv)
Reads in the arguments passed through the command line to Qreport. and stores them in the global variable par_QR.

4.6.1 Detailed Description

Header file: help dialog for Qreport and initialization of the command line arguments.

Author

Paula Perez paulaperezrubio@gmail.com

Date

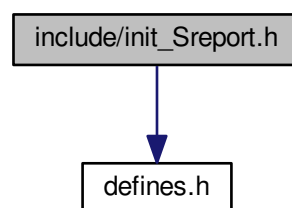
03.08.2017

4.7 include/init_Sreport.h File Reference

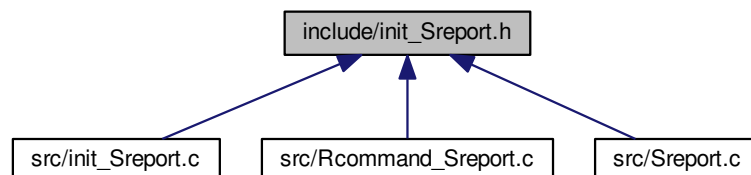
Help dialog for Sreport and initialization of the command line arguments.

```
#include "defines.h"
```

Include dependency graph for init_Sreport.h:



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



Classes

- [struct _iparam_Sreport](#)
contains Sreport input parameters

Typedefs

- typedef struct `_iparam_Sreport` `lparam_Sreport`

contains Sreport input parameters

Functions

- void `printHelpDialog_Sreport` ()

Function that prints Sreport help dialog when called.

- void `getarg_Sreport` (int argc, char **argv)

Reads in the arguments passed through the command line to Sreport. and stores them in the global variable `par_SR`.

4.7.1 Detailed Description

Help dialog for Sreport and initialization of the command line arguments.

Author

Paula Perez paulaperezrubio@gmail.com

Date

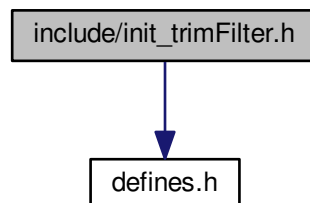
09.08.2017

4.8 include/init_trimFilter.h File Reference

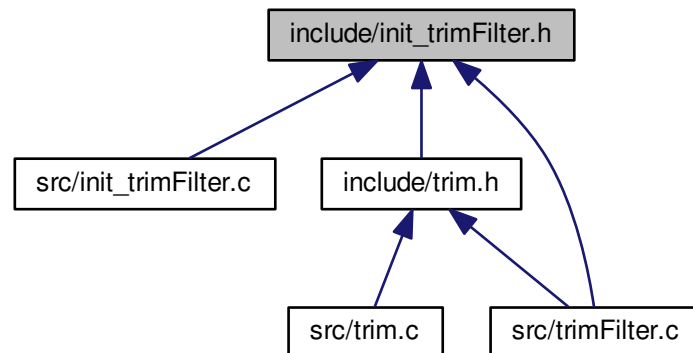
help dialog for trimFilter and initialization of the command line arguments.

```
#include "defines.h"
```

Include dependency graph for `init_trimFilter.h`:



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



Classes

- struct [_adapter](#)
- struct [_iparam_trimFilter](#)
trimFilter input parameters

Typedefs

- typedef struct [_adapter](#) Adapter
- typedef struct [_iparam_trimFilter](#) Iparam_trimFilter
trimFilter input parameters

Functions

- void [printHelpDialog_trimFilter](#) ()
Function that prints trimFilter help dialog when called.
- void [getarg_trimFilter](#) (int argc, char **argv)
Reads in the arguments passed through the command line to trimFilter. and stores them in the global variable par_TF.

4.8.1 Detailed Description

help dialog for trimFilter and initialization of the command line arguments.

Author

Paula Perez paulaperezrubio@gmail.com

Date

24.08.2017

4.8.2 Typedef Documentation

4.8.2.1 typedef struct _adapter Adapter

@ brief adapter struct @ note UNFINISHED!

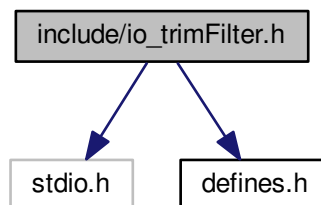
4.9 include/io_trimFilter.h File Reference

buffer fq output, write summary file

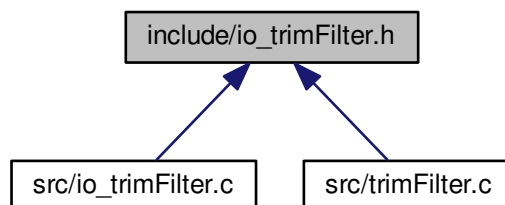
```
#include <stdio.h>
```

```
#include "defines.h"
```

Include dependency graph for io_trimFilter.h:



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



Classes

- struct [_stats_TF](#)
collects stats info from the filtering procedure

Typedefs

- typedef struct [_stats_TF](#) Stats_TF
collects stats info from the filtering procedure

Functions

- void `buffer_output` (FILE *fout, const char *a, const int len, const int fd_i)
buffers the output before writing to disk, writes out summary
- void `write_summary_TF` (Stats_TF tf_stats, char *filename)
writes stats of filtering to summary file (binary)

4.9.1 Detailed Description

buffer fq output, write summary file

Author

Paula Perez paulaperezrubio@gmail.com

Date

29.08.2017

4.9.2 Function Documentation

4.9.2.1 void `buffer_output` (FILE * *fout*, const char * *str*, const int *len*, const int *fd_i*)

buffers the output before writing to disk, writes out summary

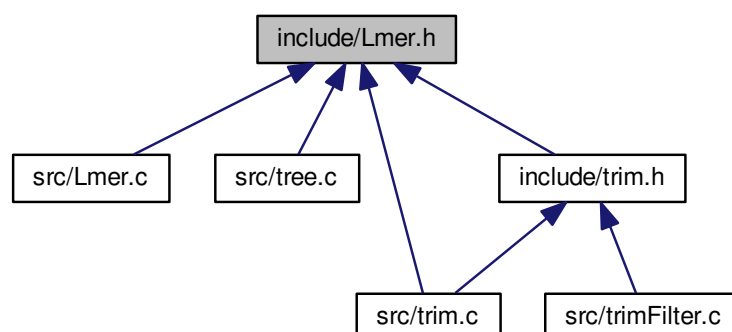
Parameters

<i>fout</i>	FILE pointer where we might write to disk;
<i>str</i>	string we want to add
<i>len</i>	length of the string we want to add
<i>fd_i</i>	identifier: GOOD, ADAP, CONT, LOWQ, NNNN

4.10 include/Lmer.h File Reference

Manipulation of Lmers and sequences.

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



Functions

- void `init_map` ()
Initialize lookup table LT.
- void `init_map_SA` ()
Initialize lookup table LT (for SA)
- void `Lmer_sLmer` (char *Lmer, int L)
Transforms an Lmer to the convention stored in the lookup table LT.
- void `rev_comp` (char *sLmer, int L)
Obtains the reverse complement, for {'\000','\001','\002','\003'}.
- void `rev_comp2` (char *sLmer, int L)
Obtains the reverse complement, for {'\001','\002','\003','\004'}.

4.10.1 Detailed Description

Manipulation of Lmers and sequences.

Author

Paula Perez paulaperezrubi@gmail.com

Date

18.08.2017

Note

I have to try to merge the two versions of conversions!

Basically, and depending on the method used, nucleotides {'a', 'c', 'g', 't'} are shifted to the characters {'\000','\001','\002','\003'} or to {'\001','\002','\003','\004'} in a Lmer. A function to provide the reverse complement is also provided.

4.10.2 Function Documentation

4.10.2.1 void `init_map` ()

Initialize lookup table LT.

{'a','c','g','t'} → {'\000','\001','\002','\003'}, rest '\004'.

4.10.2.2 void `init_map_SA` ()

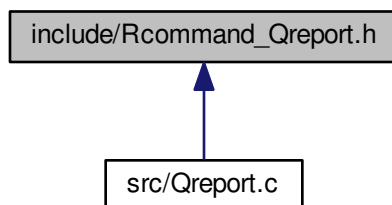
Initialize lookup table LT (for SA)

{'a','c','g','t'} → {'\001','\002','\003','\004'}, rest '\005'.

4.11 include/Rcommand_Qreport.h File Reference

get Rscript command for Qreport

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



Functions

- char * [command_Qreport](#) ()

returns Rscript command that generates the quality report in html

4.11.1 Detailed Description

get Rscript command for Qreport

Author

Paula Perez paulaperezrubio@gmail.com

Date

07.08.2017

Author

Paula Perez paulaperezrubio@gmail.com

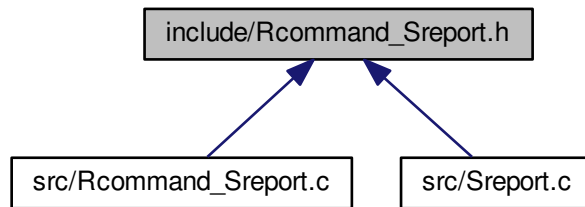
Date

09.08.2017

4.12 include/Rcommand_Sreport.h File Reference

get Rscript command for Sreport

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



Functions

- `char * command_Sreport ()`

returns Rscript command that generates the summary report in html

4.12.1 Detailed Description

get Rscript command for Sreport

Author

Paula Perez paulaperezrubio@gmail.com

Date

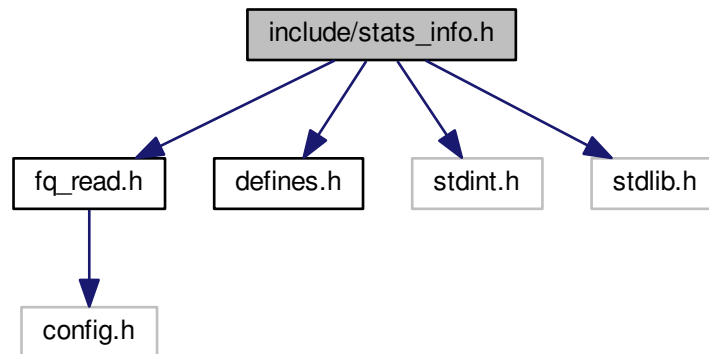
09.08.2017

4.13 include/stats_info.h File Reference

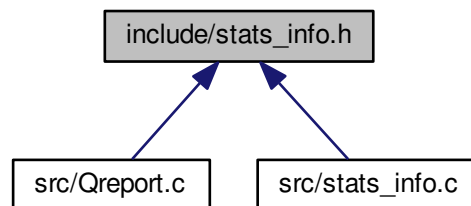
Construct the quality report variables and update them.

```
#include "fq_read.h"
#include "defines.h"
#include <stdint.h>
#include <stdlib.h>
```

Include dependency graph for stats_info.h:



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



Classes

- struct [statsinfo](#)
stores info needed to create the summary graphs

Typedefs

- typedef struct [statsinfo](#) [Info](#)
stores info needed to create the summary graphs

Functions

- void [init_info](#) ([Info](#) *res)
Initialization of a Info type.
- void [free_info](#) ([Info](#) *res)
frees allocated memory in Info

- void `read_info` (`Info *res`, `char *file`)
Read Info from binary file.
- void `write_info` (`Info *res`, `char *file`)
Write info to binary file.
- void `print_info` (`Info *res`, `char *infofile`)
print Info to a textfile
- void `get_first_tile` (`Info *res`, `Fq_read *seq`)
gets first tile
- void `update_info` (`Info *res`, `Fq_read *seq`)
updates Info with Fq_read
- int `update_ACGT_counts` (`uint64_t *ACGT_low`, `char ACGT`)
update, for current tile, ACGT counts.
- void `update_QPosTile_table` (`Info *res`, `Fq_read *seq`)
update QPostile table
- void `update_ACGT_pos` (`uint64_t *ACGT_pos`, `Fq_read *seq`, `int read_len`)
update ACGT_pos
- void `resize_info` (`Info *res`)
resize Info

4.13.1 Detailed Description

Construct the quality report variables and update them.

Author

Paula Perez paulaperezrubio@gmail.com

Date

04.08.2017

4.13.2 Function Documentation

4.13.2.1 void init_info (Info * res)

Initialization of a Info type.

It sets: nQ, read_len, ntiles, minQ and the dimensions of the arrays. Initializes the rest of the variables to zero and allocates memory to the arrays initializing them to 0 (calloc).

4.13.2.2 void resize_info (Info * res)

resize Info

At the end of the program, resize the structure Info, and adapt it to the actual number of tiles and the actual number of different quality values present.

4.13.2.3 int update_ACGT_counts (uint64_t * ACGT_low, char ACGT)

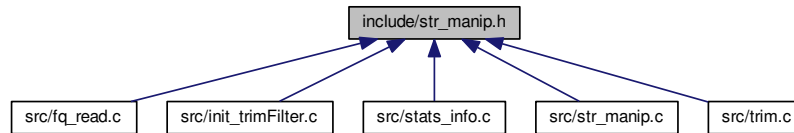
update, for current tile, ACGT counts.

Makes update of ACGT counts for the current tile. Can be used with variables: lowQ_ACGT_tile and ACGT_tile

4.14 include/str_manip.h File Reference

functions that do string manipulation

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



Classes

- [struct _split](#)
contains a splitted string and the number of splitted fields

Typedefs

- [typedef struct _split Split](#)
contains a splitted string and the number of splitted fields

Functions

- [int strindex](#) (char *s, char *t)
returns index of t in s (start, first occurrence)
- [int count_char](#) (char *s, char c)
returns the # of occurrences of char c in string s
- [Split strsplit](#) (char *str, char sep)
Separates strings by a separator.

4.14.1 Detailed Description

functions that do string manipulation

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.14.2 Function Documentation

4.14.2.1 int strindex (char * s, char * t)

returns index of t in s (start, first occurrence)

Parameters

<i>s</i>	string to be checked.
<i>t</i>	substring to be found in s.

4.14.2.2 Split strsplit (char * *str*, char *sep*)

Separates strings by a separator.

Parameters

<i>str</i>	input string
<i>sep</i>	separator (char)

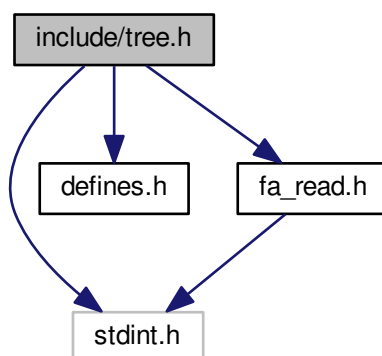
Returns

array of strings containing the substrings in the input separated

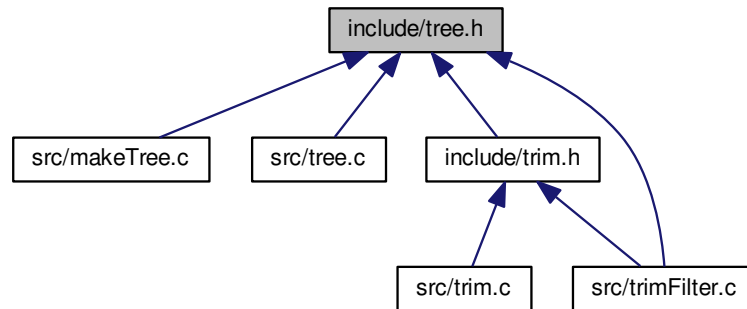
4.15 include/tree.h File Reference

Construction of tree, check paths, write tree, read in tree.

```
#include <stdint.h>
#include "defines.h"
#include "fa_read.h"
Include dependency graph for tree.h:
```



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



Classes

- `struct _node`
Node structure: formed out of T_ACGT pointers to Node structure.
- `struct _tree`
structure containing a T_ACGT-tree.

Typedefs

- `typedef struct _node Node`
Node structure: formed out of T_ACGT pointers to Node structure.
- `typedef struct _tree Tree`
structure containing a T_ACGT-tree.

Functions

- `Node * get_new_pool (Tree *tree_ptr)`
reallocs pool_2D (++NPOOL_2D) if all existing nodes have been used
- `Node * new_node_buf (Tree *tree_ptr)`
moves to the next node (allocating new memory if necessary)
- `void free_all_nodes (Tree *tree_ptr)`
frees the whole tree structure
- `void insert_Lmer (Tree *tree_ptr, char *Lmer)`
Lmer insertion in the tree (depth L).
- `void insert_entry (Tree *tree_ptr, Fa_entry *entry)`
fasta entry insertion in the tree (depth L).
- `double check_path (Tree *tree_ptr, char *read, int Lread)`
checks if read is found in tree and outputs a score
- `Tree * tree_from_fasta (Fa_data *fasta, int L)`
create Tree structure from fasta structure.
- `void save_tree (Tree *tree_ptr, char *filename)`
saves Tree to disk in filename
- `Tree * read_tree (char *filename)`
read tree from file

4.15.1 Detailed Description

Construction of tree, check paths, write tree, read in tree.

Author

Paula Perez paulaperezrubio@gmail.com

Date

18.08.2017

4.15.2 Typedef Documentation

4.15.2.1 typedef struct _tree Tree

structure containing a T_ACGT-tree.

The tree structure is stored in a pointer to pointer to Node. We grow the structure on the flight as we need more memory. In the outer direction, we start by allocating NPOOL_2D pointers to Node. In the inner direction, we allocate NPOOL_1D Nodes and fill them as we read the fasta file. When all of them are allocated, we allocate again NPOOL_1D. If NPOOL_2D pointers to Node are allocated, the outer dimension is reallocated with +NPOOL_2D extra elements. L is the depth of the tree, pool_count is the number on Node* elements used so far, pool_available is the number of Nodes available in every moment, and nnodes is the total number of nodes filled in. We limit the number of allocated nodes to UINT_MAX (we cannot count more nodes!).

4.15.3 Function Documentation

4.15.3.1 double check_path (Tree * tree_ptr, char * read, int Lread)

checks if read is found in tree and outputs a score

Parameters

<i>tree_ptr</i>	pointer to Tree structure
<i>read</i>	Read or reverse complement
<i>Lread</i>	length of read

Returns

score = (number of Lmers of reads found in read) / (Lread-L+1)

4.15.3.2 void free_all_nodes (Tree * tree_ptr)

frees the whole tree structure

Parameters

<i>tree_ptr</i>	pointer to Tree structure
-----------------	---------------------------

This function deallocates the memory allocated in a Tree structure.

4.15.3.3 Node* get_new_pool (Tree * tree_ptr)

reallocs pool_2D (++NPOOL_2D) if all existing nodes have been used

Parameters

<i>tree_ptr</i>	pointer to Tree structure
-----------------	---------------------------

4.15.3.4 Node* new_node_buf (Tree * tree_ptr)

moves to the next node (allocating new memory if necessary)

Parameters

<i>tree_ptr</i>	pointer to Tree structure
-----------------	---------------------------

Returns

address to next node

The function checks if there are available nodes (information stored in the variable `tree_ptr -> pool_available`) and goes to the next node. If there is no nodes left, it allocates a new `pool_1D`, and if there is no room left in the outer dimension, it reallocates `NPOOL_2D` more `Node*`'s. If the number of nodes reaches `UINT_MAX`, the program returns an error message and exits.

4.15.3.5 Tree* read_tree (char * filename)

read tree from file

Parameters

<i>filename</i>	string with the filename
-----------------	--------------------------

Returns

pointer to Tree structure

This function unwinds the process carried out in `save_tree` and assigns addresses to the children of every given node.

4.15.3.6 void save_tree (Tree * tree_ptr, char * filename)

saves Tree to disk in filename

Parameters

<i>tree_ptr</i>	pointer to Tree structure
<i>filename</i>	string containing filename

The tree structure is stored as follows: every address is stored in a `uint32_t` (we are not allowing trees with more than `UINT_MAX` nodes). For every node, the addresses of the children are stored in the following fashion:

- If it is pointing to NULL: 0.
- Otherwise: `i2`, the index in the outer dimension of `pool_2D` is identified, and the difference `jump = pool_2D[i2][j].children[k] - pool_2D[i2]` is computed. `i2*NPOOL_D1 + jump` is then stored for child `k`.

4.15.3.7 Tree* tree_from_fasta (Fa_data * fasta, int L)

create Tree structure from fasta structure.

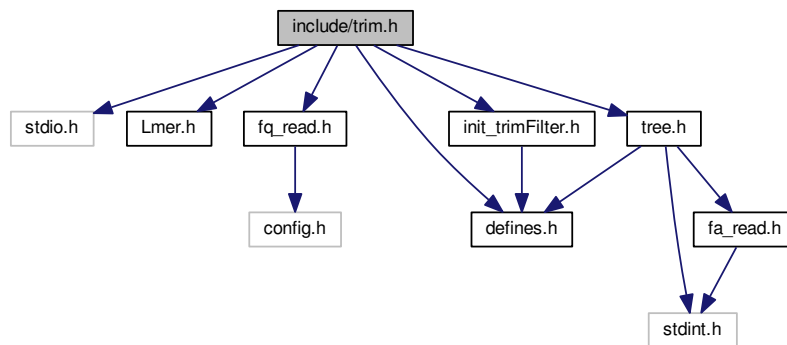
Parameters

<i>fasta</i>	pointer to fasta structure
<i>L</i>	tree length

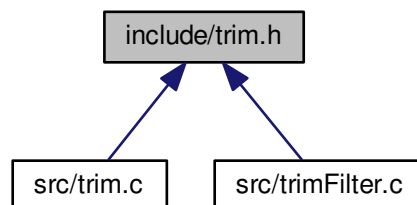
4.16 include/trim.h File Reference

trims/filter sequences after Quality, N's contaminations.

```
#include <stdio.h>
#include "Lmer.h"
#include "fq_read.h"
#include "defines.h"
#include "tree.h"
#include "init_trimFilter.h"
Include dependency graph for trim.h:
```



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



Functions

- int `trim_sequenceN` (`Fq_read` *seq)
trims sequence based on presence of N nucleotides

- `int trim_sequenceQ (Fq_read *seq)`
trims sequence based on lowQ base callings
- `bool is_read_inTree (Tree *tree_ptr, Fq_read *seq)`
check if Lread is contained in tree. It computes the score for the read and its reverse complement; if one of them exceeds the user selected threshold, it returns true. Otherwise, it returns false.

4.16.1 Detailed Description

trims/filter sequences after Quality, N's contaminations.

Author

Paula Perez paulaperezrubio@gmail.com

Date

24.08.2017

4.16.2 Function Documentation

4.16.2.1 `bool is_read_inTree (Tree * tree_ptr, Fq_read * seq)`

check if Lread is contained in tree. It computes the score for the read and its reverse complement; if one of them exceeds the user selected threshold, it returns true. Otherwise, it returns false.

Parameters

<i>tree_ptr</i>	pointer to Tree structure
<i>seq</i>	fastq read

Returns

true if read was found, false otherwise

4.16.2.2 `int trim_sequenceN (Fq_read * seq)`

trims sequence based on presence of N nucleotides

Parameters

<i>seq</i>	fastq read
------------	------------

Returns

-1 error, 0 discarded, 1 accepted as is, 2 accepted and trimmed

This function calls a different function depending on the method passed as input `par_TF.trimN`:

- `NO(0)`: accepts it as is, (1),
- `ALL(1)`: accepts it as is if NO N's found (1), rejects it otherwise (0),
- `ENDS(2)`: trims the ends and accepts it if it is longer than minL (2 if trimming, 1 if no trimming), rejects it otherwise (0),
- `STRIP(3)`: finds the longest N-free subsequence and trims it if it is at least minL nucleotides long (2 if trimming, 1 if no N's are found), rejects it otherwise (0).

4.16.2.3 int trim_sequenceQ (Fq_read * seq)

trims sequence based on lowQ base callings

Parameters

<i>seq</i>	fastq read
------------	------------

Returns

-1 error, 0 discarded, 1 accepted as is, 2 accepted and trimmed

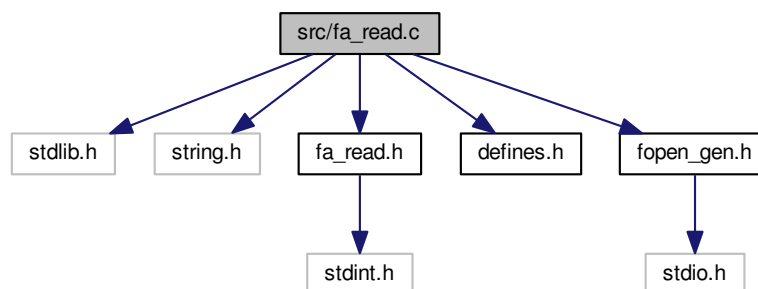
This function calls a different function depending on the method passed as input `par_TF.trimQ`:

- **NO(0)**: accepts is as is , (1),
- **FRAC(1)**: accepts it if less than `par_TF.nlowQ` are found (1), rejects it otherwise (0),
- **ENDS(2)**: trims the ends and accepts it if it is longer than `minL` (2 if trimming, 1 if no trimming), rejects it otherwise (0),
- **ENDSFRAC(3)**: trims the ends and accepts if the remaining sequence is at least `minL` bases long and if it contains less than `nlowQ` lowQ nucleotides (2 if trimming, 1 if no trimming). Otherwise, it is rejected, (0).
- **GLOBAL(4)**: it trims globally `globleft` nucleotides from the left and `globright` from the right, (returns 2).

4.17 src/fa_read.c File Reference

reads in and stores fasta files

```
#include <stdlib.h>
#include <string.h>
#include "fa_read.h"
#include "defines.h"
#include "fopen_gen.h"
Include dependency graph for fa_read.c:
```



Functions

- static int **ignore_line** (char *line)
ignore header lines.
- static void **init_fa** (**Fa_data** *ptr_fa)
Initialization of Fa_data.
- static void **realloc_fa** (**Fa_data** *ptr_fa)
Reallocation of Fa_data, in case the length of entrylen is exhausted.

- static void `init_entries (Fa_data *ptr_fa)`
Allocation of Fa_entries.
- static uint64_t `sweep_fa (char *filename, Fa_data *ptr_fa)`
this function sweeps a fasta file to obtain structure details.
- int `read_fasta (char *filename, Fa_data *ptr_fa)`
reads a fasta file and stores the contents in a Fa_data structure.
- uint64_t `size_fasta (Fa_data *ptr_fa)`
computes length of genome in fasta structure
- void `free_fasta (Fa_data *ptr_fa)`
free fasta file

Variables

- uint64_t `alloc_mem`

4.17.1 Detailed Description

reads in and stores fasta files

Author

Paula Perez paulaperezrubio@gmail.com

Date

18.08.2017

4.17.2 Function Documentation

4.17.2.1 void free_fasta (Fa_data * ptr_fa)

free fasta file

Parameters

<code>ptr_fa</code>	pointer to Fa_data structure.
---------------------	-------------------------------

The dynamically allocated memory in a Fa_data struct is deallocated and counted, so that we can

4.17.2.2 static int ignore_line (char * line) [static]

ignore header lines.

Parameters

<code>line</code>	string of characters.
-------------------	-----------------------

Returns

number of characters to jump until a
is found.

4.17.2.3 static void init_entries (Fa_data * ptr_fa) [static]

Allocation of Fa_entries.

Parameters

<i>ptr_fa</i>	pointer to Fa_data structure.
---------------	-------------------------------

When we have swept the fasta file once, we can proceed to allocate the memory for the entries (now we have registered their length).

4.17.2.4 static void init_fa (Fa_data * ptr_fa) [static]

Initialization of Fa_data.

Parameters

<i>ptr_fa</i>	pointer to Fa_data structure.
---------------	-------------------------------

Initializes nlines, linelen, nentries to 0 and allocates memory for entrylen (FA_ENTRY_BUF entries).

4.17.2.5 int read_fasta (char * filename, Fa_data * ptr_fa)

reads a fasta file and stores the contents in a Fa_data structure.

Parameters

<i>filename</i>	path to a fasta input file.
<i>ptr_fa</i>	pointer to Fa_data structure.

Returns

number of entries in the fasta file.

A fasta file is read and stored in a structure Fa_data. The basic problem with reading FASTA files is that there is no end-of-record indicator. When you're reading sequence n, you don't know you're done until you've read the header line for sequence n+1, which you won't parse 'til later (when you're reading in the sequence n+1). The solution implemented here is to read the file twice. The first time, (sweep_fa), we initialize Fa_data and store the parameters:

- nlines: number of lines of the fasta file.
- nentries: number of entries in the fasta file.
- linelen: length of a line in the considered fasta file.
- entrylen: array containing the lengths of every entry. With this information, the pointer to Fa_entry can be allocated and the file is read again and the entries are stored in the structure.

4.17.2.6 static void realloc_fa (Fa_data * ptr_fa) [static]

Reallocation of Fa_data, in case the length of entrylen is exhausted.

Parameters

<i>ptr_fa</i>	pointer to Fa_data structure.
---------------	-------------------------------

4.17.2.7 uint64_t size_fasta (Fa_data * ptr_fa)

computes length of genome in fasta structure

Parameters

<i>ptr_fa</i>	pointer to Fa_data
---------------	--------------------

Returns

total number of nucleotides

4.17.2.8 `static uint64_t sweep_fa (char * filename, Fa_data * ptr_fa) [static]`

this function sweeps a fasta file to obtain structure details.

Parameters

<i>filename</i>	path to a fasta input file.
<i>ptr_fa</i>	pointer to Fa_data structure.

Returns

size of fasta file.

This function sweeps over the fasta file once to annotate how many entries there are, how long they are, how many characters there are per line, and how many lines the file has.

4.17.3 Variable Documentation**4.17.3.1** `uint64_t alloc_mem`

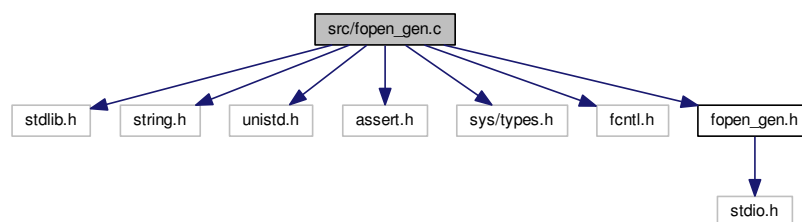
global variable. Memory allocated in the heap.

4.18 src/fopen_gen.c File Reference

Uncompress/compress input/output files using pipes.

```
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <assert.h>
#include <sys/types.h>
#include <fcntl.h>
#include "fopen_gen.h"
```

Include dependency graph for fopen_gen.c:



Functions

- static const char * **zcatExec** (const char *path)
- static const char * **catExec** (const char *path)
Commands to compress files. To be done in output.
- static int **uncompress** (const char *path)
Open a pipe to uncompress file. Open a pipe to uncompress the specified file. Not thread safe.
- static int **compress** (const char *path)
Open a pipe to compress output. Open a pipe to uncompress the specified file. Not thread safe.
- int **setCloexec** (int fd)
- static FILE * **funcompress** (const char *path)
Open a pipe to uncompress the specified file.
- static FILE * **fcompress** (const char *path)
Open a pipe to compress the specified file.
- FILE * **fopen_gen** (const char *path, const char *mode)
Generalized fopen function. fopen_gen is to be used as fopen. Can be used in read and in write mode. When used in read mode with a compressed extension, the file will be first decompressed and then read. When used in write mode with a compressed extension, the output will be compressed.

4.18.1 Detailed Description

Uncompress/compress input/output files using pipes.

Hook the standard file opening functions, open, fopen and fopen64. If the extension of the file being opened indicates the file is compressed (.gz, .bz2, .xz), when opening in the reading mode a pipe to a program is opened that decompresses that file (gunzip, bunzip2 or xzdec) and return a handle to the open pipe. When opening in the writing mode (only for .gz, .bam), a pipe to a program is opened that compresses the output.

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

Warning

vfork vs fork to be checked!

Note

- original copyright note - (reading mode, original C++ code) author: Shaun Jackman sjackman@bcgsc.ca, <https://github.com/bcgsc>, filename: Uncompress.cpp

4.18.2 Function Documentation

4.18.2.1 static int compress (const char * path) [static]

Open a pipe to compress output. Open a pipe to uncompress the specified file. Not thread safe.

Returns

a file descriptor

4.18.2.2 static FILE* fcompress (const char * *path*) [static]

Open a pipe to compress the specified file.

Returns

a FILE pointer

4.18.2.3 FILE* fopen_gen (const char * *path*, const char * *mode*)

Generalized fopen function. fopen_gen is to be used as fopen. Can be used in read and in write mode. When used in read mode with a compressed extension, the file will be first decompressed and then read. When used in write mode with a compressed extension, the output will be compressed.

Returns

a FILE pointer

4.18.2.4 static FILE* funcompress (const char * *path*) [static]

Open a pipe to uncompress the specified file.

Returns

a FILE pointer

4.18.2.5 static int uncompress (const char * *path*) [static]

Open a pipe to uncompress file. Open a pipe to uncompress the specified file. Not thread safe.

Returns

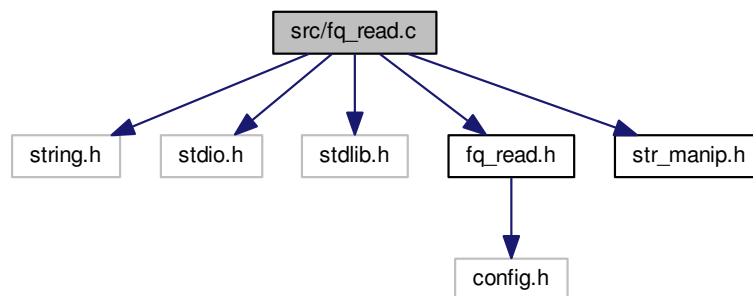
a file descriptor

4.19 src/fq_read.c File Reference

fastq entries manipulations (read/write)

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "fq_read.h"
#include "str_manip.h"
```

Include dependency graph for fq_read.c:



Functions

- int [get_fqread](#) ([Fq_read](#) *seq, char *buffer, int pos1, int pos2, int nline, int read_len, int filter)
reads fastq line from a buffer
- int [string_seq](#) ([Fq_read](#) *seq, char *char_seq)
writes the fq entry in a string

4.19.1 Detailed Description

fastq entries manipulations (read/write)

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.19.2 Function Documentation

4.19.2.1 int get_fqread ([Fq_read](#) * seq, char * buffer, int pos1, int pos2, int nline, int read_len, int filter)

reads fastq line from a buffer

a fastq line is read from a buffer and the relevant information is stored in a structure **Fq_read**. Depending on the value of **filter**, information about whether the read was trimmed is stored.

Parameters

<i>*seq</i>	pointer to Fq_read , where the info will be stored.
<i>buffer</i>	variable where the file being read is stored.
<i>pos1</i>	buffer start position of the line.

<i>pos2</i>	buffer end position of the line.
<i>nline</i>	file line number being read.
<i>read_len</i>	predefined read length
<i>filter</i>	0 original file, 1 file filtered with filter_trim, 2 file filtered with another tool

4.19.2.2 int string_seq (Fq_read * seq, char * char_seq)

writes the fq entry in a string

Parameters

<i>*seq</i>	pointer to Fq_read , where the info will be stored.
<i>char_seq</i>	pointer to buffer, where the sequence will be stored

Warning

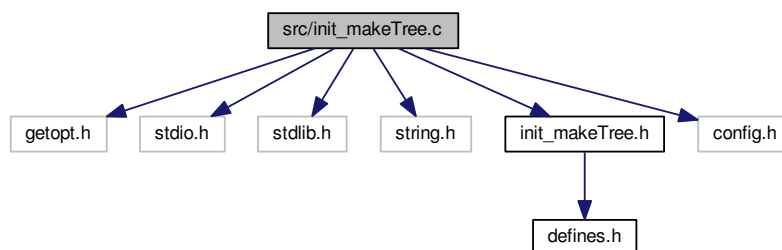
change the call to sprintf to snprintf

4.20 src/init_makeTree.c File Reference

Help dialog for makeTree and initialization of the command line arguments.

```
#include <getopt.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "init_makeTree.h"
#include "config.h"
```

Include dependency graph for init_makeTree.c:



Functions

- void [printHelpDialog_makeTree](#) ()
Function that prints makeTree help dialog when called.
- void [getarg_makeTree](#) (int argc, char **argv)
Reads in the arguments passed through the command line to makeTree. and stores them in the global variable par_MT.

Variables

- [lparam_makeTree](#) par_MT

4.20.1 Detailed Description

Help dialog for makeTree and initialization of the command line arguments.

Author

Paula Perez paulaperezrubio@gmail.com

Date

23.08.2017

4.20.2 Variable Documentation

4.20.2.1 lparam_makeTree par_MT

Input parameters of makeTree

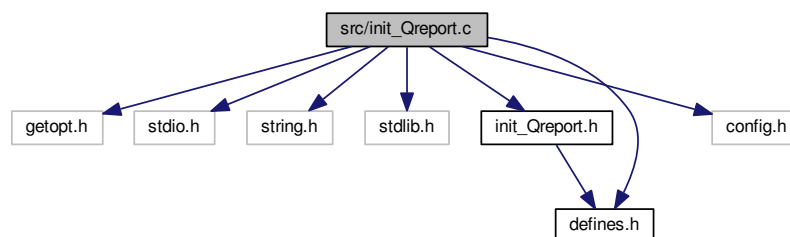
global variable: Input parameters of makeTree.

4.21 src/init_Qreport.c File Reference

Help dialog for Qreport and initialization of the command line arguments.

```
#include <getopt.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "init_Qreport.h"
#include "config.h"
#include "defines.h"
```

Include dependency graph for init_Qreport.c:



Functions

- void [printHelpDialog_Qreport](#) ()
Function that prints Qreport help dialog when called.
- void [getarg_Qreport](#) (int argc, char **argv)
Reads in the arguments passed through the command line to Qreport. and stores them in the global variable `par_QR`.

Variables

- [lparam_Qreport par_QR](#)

4.21.1 Detailed Description

Help dialog for Qreport and initialization of the command line arguments.

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.21.2 Variable Documentation

4.21.2.1 lparam_Qreport par_QR

Input parameters of Qreport

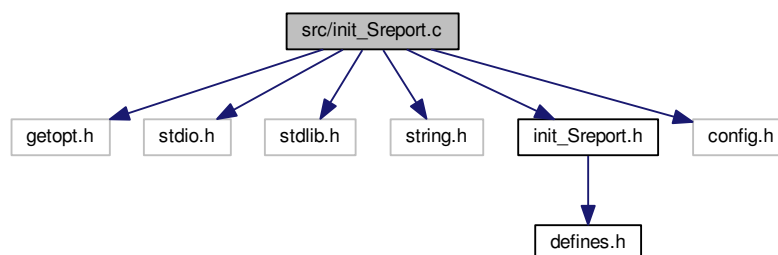
global variable: input parameters for Qreport

4.22 src/init_Sreport.c File Reference

Help dialog for Sreport and initialization of the command line arguments.

```
#include <getopt.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "init_Sreport.h"
#include "config.h"
```

Include dependency graph for init_Sreport.c:



Functions

- void [printHelpDialog_Sreport](#) ()
Function that prints Sreport help dialog when called.
- void [getarg_Sreport](#) (int argc, char **argv)
Reads in the arguments passed through the command line to Sreport. and stores them in the global variable `par_SR`.

Variables

- [lparam_Sreport par_SR](#)

4.22.1 Detailed Description

Help dialog for Sreport and initialization of the command line arguments.

Author

Paula Perez paulaperezrubio@gmail.com

Date

09.08.2017

4.22.2 Variable Documentation

4.22.2.1 lparam_Sreport par_SR

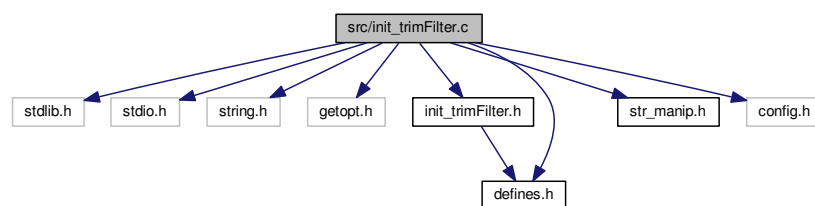
input parameters Sreport

4.23 src/init_trimFilter.c File Reference

help dialog for trimFilter and initialization of the command line arguments.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <getopt.h>
#include "init_trimFilter.h"
#include "defines.h"
#include "str_manip.h"
#include "config.h"
```

Include dependency graph for init_trimFilter.c:



Functions

- void [printHelpDialog_trimFilter](#) ()
Function that prints trimFilter help dialog when called.
- void [getarg_trimFilter](#) (int argc, char **argv)
Reads in the arguments passed through the command line to trimFilter. and stores them in the global variable par_TF.

Variables

- [lparam_trimFilter par_TF](#)

4.23.1 Detailed Description

help dialog for trimFilter and initialization of the command line arguments.

Author

Paula Perez paulaperezrubio@gmail.com

Date

24.08.2017

4.23.2 Variable Documentation

4.23.2.1 lparam_trimFilter par_TF

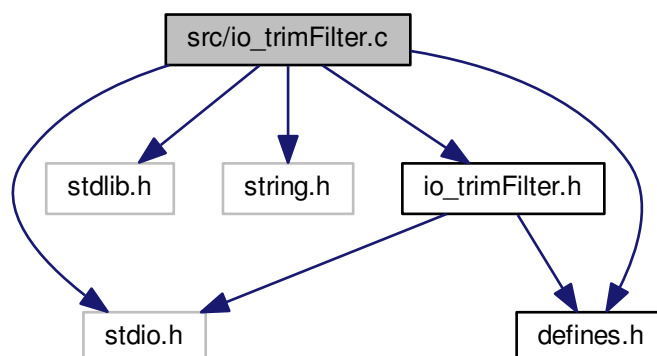
Input parameters of makeTree

global variable: Input parameters of makeTree.

4.24 src/io_trimFilter.c File Reference

buffer fq output, write summary file

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "io_trimFilter.h"
#include "defines.h"
Include dependency graph for io_trimFilter.c:
```



Functions

- void `buffer_output` (FILE **fout*, const char **str*, const int *len*, const int *fd_i*)
buffers the output before writing to disk, writes out summary
- void `write_summary_TF` (Stats_TF *tf_stats*, char **filename*)
writes stats of filtering to summary file (binary)

4.24.1 Detailed Description

buffer fq output, write summary file

Author

Paula Perez paulaperezrubio@gmail.com

Date

29.08.2017

4.24.2 Function Documentation

4.24.2.1 void `buffer_output` (FILE * *fout*, const char * *str*, const int *len*, const int *fd_i*)

buffers the output before writing to disk, writes out summary

Parameters

<i>fout</i>	FILE pointer where we might write to disk;
<i>str</i>	string we want to add
<i>len</i>	length of the string we want to add
<i>fd_i</i>	identifier: GOOD, ADAP, CONT, LOWQ, NNNN

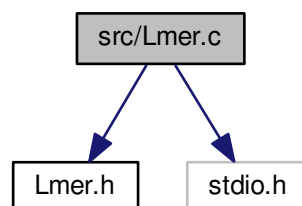
4.25 src/Lmer.c File Reference

Manipulation of Lmers and sequences.

```
#include "Lmer.h"
```

```
#include <stdio.h>
```

Include dependency graph for Lmer.c:



Functions

- void `init_map` ()
Initialize lookup table LT.
- void `init_map_SA` ()
Initialize lookup table LT (for SA)
- void `Lmer_sLmer` (char *Lmer, int L)
Transforms an Lmer to the convention stored in the lookup table LT.
- void `rev_comp` (char *sLmer, int L)
Obtains the reverse complement, for {'\000','\001','\002','\003'}.
- void `rev_comp2` (char *sLmer, int L)
Obtains the reverse complement, for {'\001','\002','\003','\004'}.

Variables

- char `LT` [256]
- char `Nencode`

4.25.1 Detailed Description

Manipulation of Lmers and sequences.

Author

Paula Perez paulaperezrubio@gmail.com

Date

18.08.2017

4.25.2 Function Documentation

4.25.2.1 void `init_map` ()

Initialize lookup table LT.

{'a','c','g','t'} → {'\000','\001','\002','\003'}, rest '\004'.

4.25.2.2 void `init_map_SA` ()

Initialize lookup table LT (for SA)

{'a','c','g','t'} → {'\001','\002','\003','\004'}, rest '\005'.

4.25.3 Variable Documentation

4.25.3.1 char `LT`[256]

global variable. Lookup table.

4.25.3.2 char `Nencode`

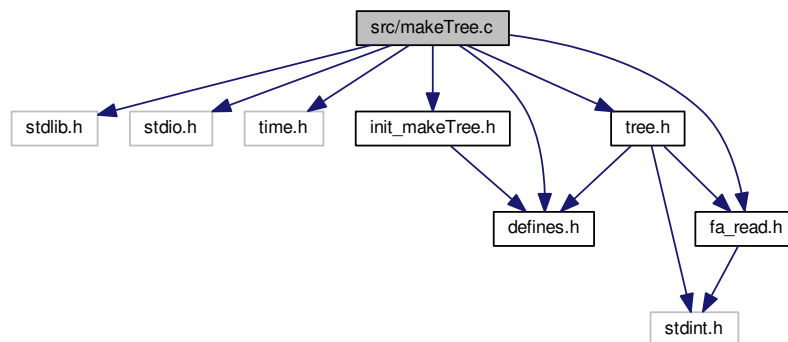
global variable. Encoding for N's(\004, or \005)

4.26 src/makeTree.c File Reference

makeTree main function

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "defines.h"
#include "fa_read.h"
#include "tree.h"
#include "init_makeTree.h"
```

Include dependency graph for makeTree.c:



Functions

- int [main](#) (int argc, char *argv[])

makeTree main function

Variables

- uint64_t [alloc_mem](#) = 0
- [lparam_makeTree](#) [par_MT](#)

4.26.1 Detailed Description

makeTree main function

Author

Paula Perez paulaperezrubio@gmail.com

Date

23.08.2017 This file contains the makeTree main function. It reads a fasta file, constructs a 4-tree of depth L and stores it compressed in a file. See README_makeTree.md for more details.

4.26.2 Variable Documentation

4.26.2.1 uint64_t alloc_mem = 0

global variable. Memory allocated in the heap.

4.26.2.2 lparam_makeTree par_MT

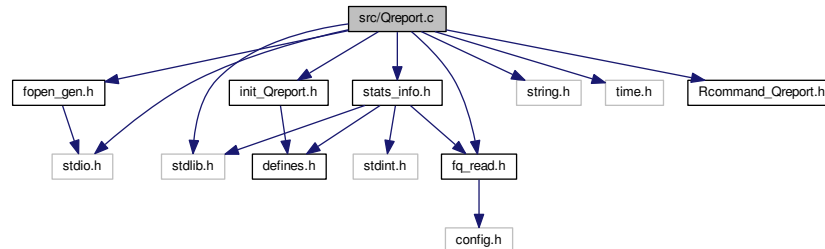
global variable: Input parameters of makeTree.

4.27 src/Qreport.c File Reference

QReport main function.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "init_Qreport.h"
#include "fopen_gen.h"
#include "fq_read.h"
#include "stats_info.h"
#include "Rcommand_Qreport.h"
```

Include dependency graph for Qreport.c:



Functions

- int [main](#) (int argc, char *argv[])
Qreport main function.

Variables

- [lparam_Qreport](#) par_QR

4.27.1 Detailed Description

QReport main function.

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017 This file contains the quality report main function. It reads a fastq file and creates a html quality report. See README_Qreport.md for more details.

4.27.2 Variable Documentation**4.27.2.1 lparam_Qreport par_QR**

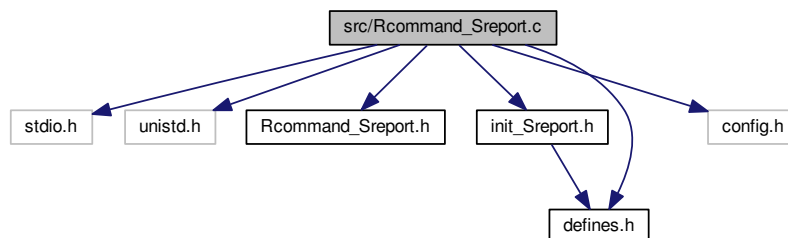
global variable: input parameters for Qreport

4.28 src/Rcommand_Sreport.c File Reference

get Rscript command for Sreport

```
#include <stdio.h>
#include <unistd.h>
#include "Rcommand_Sreport.h"
#include "init_Sreport.h"
#include "defines.h"
#include "config.h"
```

Include dependency graph for Rcommand_Sreport.c:

**Functions**

- `char * command_Sreport ()`
returns Rscript command that generates the summary report in html

Variables

- `lparam_Sreport par_SR`

4.28.1 Detailed Description

get Rscript command for Sreport

Author

Paula Perez paulaperezrubio@gmail.com

Date

09.08.2017

4.28.2 Variable Documentation**4.28.2.1 lparam_Sreport par_SR**

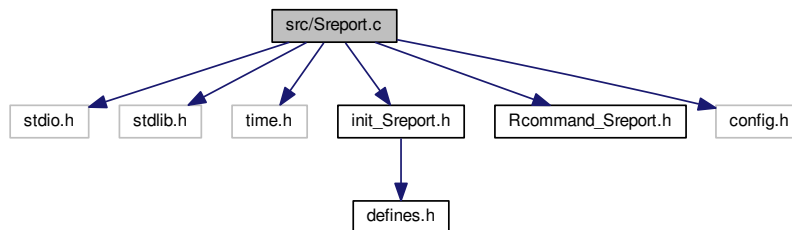
input parameters Sreport

4.29 src/Sreport.c File Reference

Sreport main function.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "init_Sreport.h"
#include "Rcommand_Sreport.h"
#include "config.h"
```

Include dependency graph for Sreport.c:

**Functions**

- `int main (int argc, char *argv[])`
Qreport main function.

Variables

- `lparam_Sreport par_SR`

4.29.1 Detailed Description

Sreport main function.

Author

Paula Perez paulaperezrubio@gmail.com

Date

09.08.2017 This file contains the summary report main function. Given a folder containing *bin as from Qreport output, Sreport generates a summary report in html format. See README_Sreport.md for more details.

4.29.2 Variable Documentation**4.29.2.1 lparam_Sreport par_SR**

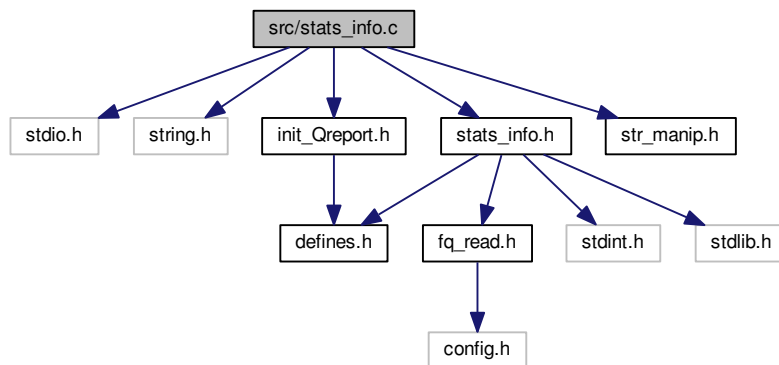
input parameters Sreport

4.30 src/stats_info.c File Reference

Construct the quality report variables and update them.

```
#include <stdio.h>
#include <string.h>
#include "stats_info.h"
#include "init_Qreport.h"
#include "str_manip.h"
```

Include dependency graph for stats_info.c:

**Functions**

- void [get_tile_lane](#) (char *line1, int *tile, int *lane)
get tile number from first line in fastq entry.
- static int [belongsto](#) (int k, int *qual_tags, int nQ)
returns 1 if k is in qual_tags, 0 otherwise.
- static int [cmpfunc](#) (const void *a, const void *b)
comparison function for qsort
- void [init_info](#) ([Info](#) *res)
Initialization of a Info type.

- void `free_info` (`Info *res`)
frees allocated memory in Info
- void `read_info` (`Info *res`, `char *file`)
Read Info from binary file.
- void `write_info` (`Info *res`, `char *file`)
Write info to binary file.
- void `print_info` (`Info *res`, `char *infofile`)
print Info to a textfile
- void `get_first_tile` (`Info *res`, `Fq_read *seq`)
gets first tile
- void `update_info` (`Info *res`, `Fq_read *seq`)
updates Info with Fq_read
- int `update_ACGT_counts` (`uint64_t *ACGT_low`, `char ACGT`)
update, for current tile, ACGT counts.
- void `update_QPosTile_table` (`Info *res`, `Fq_read *seq`)
update QPostile table
- void `update_ACGT_pos` (`uint64_t *ACGT_pos`, `Fq_read *seq`, `int read_len`)
update ACGT_pos
- void `resize_info` (`Info *res`)
resize Info

Variables

- `lparam_Qreport par_QR`

4.30.1 Detailed Description

Construct the quality report variables and update them.

Author

Paula Perez paulaperezrubio@gmail.com

Date

04.08.2017

4.30.2 Function Documentation

4.30.2.1 void `get_tile_lane` (`char *line1`, `int *tile`, `int *lane`)

get tile number from first line in fastq entry.

Parameters

<i>line1</i>	first line of a fastq entry
<i>tile</i>	int* where the tile will be stored
<i>lane</i>	int* where the lane will be stored

See also

http://wiki.christophchamp.com/index.php?title=FASTQ_format

Only Illumina sequence identifiers are allowed. The line is inspected, and the number of ':' is obtained. The function exits with an error if the number of semicolons is different from 4 or 9.

4.30.2.2 void init_info (Info * res)

Initialization of a Info type.

It sets: nQ, read_len, ntiles, minQ and the dimensions of the arrays. Initializes the rest of the variables to zero and allocates memory to the arrays initializing them to 0 (calloc).

4.30.2.3 void resize_info (Info * res)

resize Info

At the end of the program, resize the structure Info, and adapt it to the actual number of tiles and the actual number of different quality values present.

4.30.2.4 int update_ACGT_counts (uint64_t * ACGT_low, char ACGT)

update, for current tile, ACGT counts.

Makes update of ACGT counts for the current tile. Can be used with variables: lowQ_ACGT_tile and ACGT_tile

4.30.3 Variable Documentation

4.30.3.1 lparam_Qreport par_QR

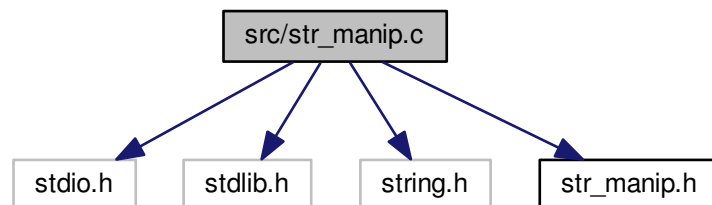
global variable: input parameters for Qreport

4.31 src/str_manip.c File Reference

functions that do string manipulation

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "str_manip.h"
```

Include dependency graph for str_manip.c:



Functions

- int [strindex](#) (char *s, char *t)
returns index of t in s (start, first occurrence)

- `int count_char (char *str, char sep)`
returns the # of occurrences of char c in string s
- `int strindexC (char *s, char sep)`
returns index of t in s (start, first occurrence)
- `Split strsplit (char *str, char sep)`
Separates strings by a separator.

4.31.1 Detailed Description

functions that do string manipulation

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.31.2 Function Documentation

4.31.2.1 `int strindex (char * s, char * t)`

returns index of t in s (start, first occurrence)

Parameters

<i>s</i>	string to be checked.
<i>t</i>	substring to be found in s.

4.31.2.2 `int strindexC (char * s, char sep)`

returns index of t in s (start, first occurrence)

Parameters

<i>s</i>	string to be checked.
<i>sep</i>	char, separator

4.31.2.3 `Split strsplit (char * str, char sep)`

Separates strings by a separator.

Parameters

<i>str</i>	input string
<i>sep</i>	separator (char)

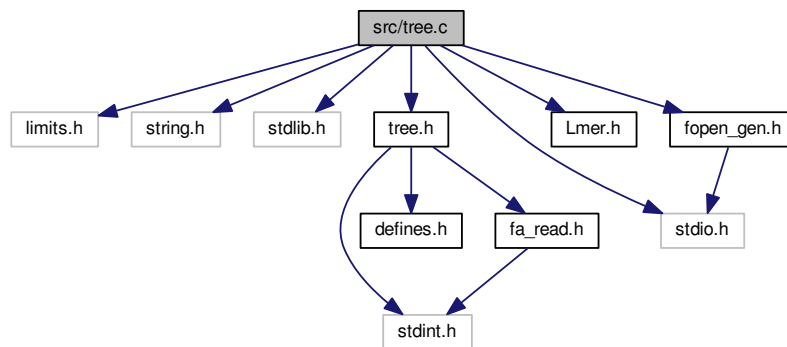
Returns

array of strings containing the substrings in the input separated

4.32 src/tree.c File Reference

Construction of tree, check paths, write tree, read in tree.

```
#include <limits.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "tree.h"
#include "Lmer.h"
#include "fopen_gen.h"
Include dependency graph for tree.c:
```



Functions

- `Node * get_new_pool (Tree *tree_ptr)`
reallocs pool_2D (++NPOOL_2D) if all existing nodes have been used
- `Node * new_node_buf (Tree *tree_ptr)`
moves to the next node (allocating new memory if necessary)
- `void free_all_nodes (Tree *tree_ptr)`
frees the whole tree structure
- `void insert_Lmer (Tree *tree_ptr, char *Lmer)`
Lmer insertion in the tree (depth L).
- `void insert_entry (Tree *tree_ptr, Fa_entry *entry)`
fasta entry insertion in the tree (depth L).
- `Tree * tree_from_fasta (Fa_data *fasta, int L)`
create Tree structure from fasta structure.
- `double check_path (Tree *tree_ptr, char *read, int Lread)`
checks if read is found in tree and outputs a score
- `void save_tree (Tree *tree_ptr, char *filename)`
saves Tree to disk in filename
- `Tree * read_tree (char *filename)`
read tree from file

Variables

- `uint64_t alloc_mem`

4.32.1 Detailed Description

Construction of tree, check paths, write tree, read in tree.

Author

Paula Perez paulaperezrubio@gmail.com

Date

23.08.2017

4.32.2 Function Documentation

4.32.2.1 double check_path (Tree * tree_ptr, char * read, int Lread)

checks if read is found in tree and outputs a score

Parameters

<i>tree_ptr</i>	pointer to Tree structure
<i>read</i>	Read or reverse complement
<i>Lread</i>	length of read

Returns

score = (number of Lmers of reads found in read) / (Lread-L+1)

4.32.2.2 void free_all_nodes (Tree * tree_ptr)

frees the whole tree structure

Parameters

<i>tree_ptr</i>	pointer to Tree structure
-----------------	---------------------------

This function deallocates the memory allocated in a Tree structure.

4.32.2.3 Node* get_new_pool (Tree * tree_ptr)

reallocs pool_2D (++NPOOL_2D) if all existing nodes have been used

Parameters

<i>tree_ptr</i>	pointer to Tree structure
-----------------	---------------------------

4.32.2.4 Node* new_node_buf (Tree * tree_ptr)

moves to the next node (allocating new memory if necessary)

Parameters

<i>tree_ptr</i>	pointer to Tree structure
-----------------	---------------------------

Returns

address to next node

The function checks if there are available nodes (information stored in the variable `tree_ptr -> pool_available`) and goes to the next node. If there is no nodes left, it allocates a new `pool_1D`, and if there is no room left in the outer dimension, it reallocates `NPOOL_2D` more `Node*`'s. If the number of nodes reaches `UINT_MAX`, the program returns an error message and exits.

4.32.2.5 Tree* read_tree (char * filename)

read tree from file

Parameters

<i>filename</i>	string with the filename
-----------------	--------------------------

Returns

pointer to Tree structure

This function unwinds the process carried out in `save_tree` and assigns addresses to the children of every given node.

4.32.2.6 void save_tree (Tree * tree_ptr, char * filename)

saves Tree to disk in filename

Parameters

<i>tree_ptr</i>	pointer to Tree structure
<i>filename</i>	string containing filename

The tree structure is stored as follows: every address is stored in a `uint32_t` (we are not allowing trees with more than `UINT_MAX` nodes). For every node, the addresses of the children are stored in the following fashion:

- If it is pointing to NULL: 0.
- Otherwise: `i2`, the index in the outer dimension of `pool_2D` is identified, and the difference `jump = pool_2D[i2][j].children[k] - pool_2D[i2]` is computed. `i2*NPOOL_D1 + jump` is then stored for child `k`.

4.32.2.7 Tree* tree_from_fasta (Fa_data * fasta, int L)

create Tree structure from fasta structure.

Parameters

<i>fasta</i>	pointer to fasta structure
<i>L</i>	tree length

4.32.3 Variable Documentation**4.32.3.1 uint64_t alloc_mem**

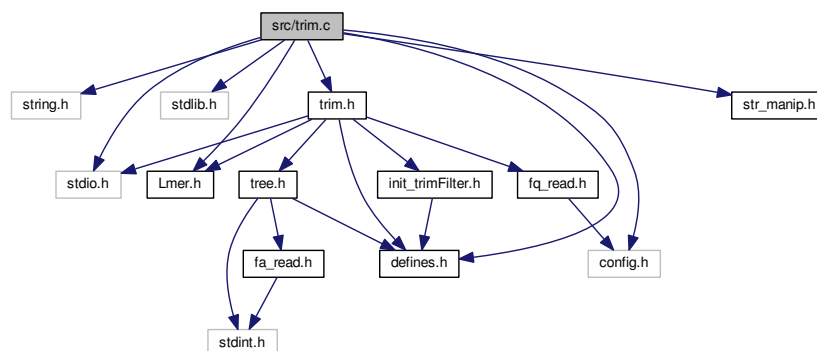
global variable. Memory allocated in the heap.

4.33 src/trim.c File Reference

trims/filter sequences after Quality, N's contaminations.

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "trim.h"
#include "Lmer.h"
#include "str_manip.h"
#include "defines.h"
#include "config.h"
```

Include dependency graph for trim.c:



Macros

- `#define TRIM_STRING 20`

Functions

- static int `no_N (Fq_read *seq)`
checks if a sequence contains any non standard base callings (N's)
- static int `Nfree_Lmer (Fq_read *seq, int minL)`
Finds the largest Nfree sub-seq and keeps it if larger than minL.
- static int `Ntrim_ends (Fq_read *seq, int minL)`
trims a read if N's are at the ends and the remaining sub-seq \geq minL
- static int `no_lowQ (Fq_read *seq, int minQ)`
checks if a sequence contains lowQ nucleotides
- static int `Qtrim_ends (Fq_read *seq, int minQ, int minL)`
trims a read if lowQs are at the ends and remaining sub-seq \geq minL
- static int `Qtrim_frac (Fq_read *seq, int minQ, int nlowQ)`
accepts the sequence as is if there are less than nlowQ
- static int `Qtrim_endsfrac (Fq_read *seq, int minQ, int minL, int nlowQ)`
trims the ends for lowQ. The rest is kept if it contains $<$ nlowQ lowQ
- static int `Qtrim_global (Fq_read *seq, int left, int right)`
trims left from the left and right from the right
- int `trim_sequenceN (Fq_read *seq)`
trims sequence based on presence of N nucleotides

- int `trim_sequenceQ` (`Fq_read *seq`)
trims sequence based on lowQ base callings
- bool `is_read_inTree` (`Tree *tree_ptr`, `Fq_read *seq`)
check if Lread is contained in tree. It computes the score for the read and its reverse complement; if one of them exceeds the user selected threshold, it returns true. Otherwise, it returns false.

Variables

- char `LT` [256]
- char `Nencode`
- `lparam_trimFilter` `par_TF`

4.33.1 Detailed Description

trims/filter sequences after Quality, N's contaminations.

Author

Paula Perez paulaperezrubio@gmail.com

Date

24.08.2017

4.33.2 Macro Definition Documentation

4.33.2.1 `#define TRIM_STRING 20`

maximal length of trimming info string.

4.33.3 Function Documentation

4.33.3.1 `bool is_read_inTree (Tree * tree_ptr, Fq_read * seq)`

check if Lread is contained in tree. It computes the score for the read and its reverse complement; if one of them exceeds the user selected threshold, it returns true. Otherwise, it returns false.

Parameters

<code>tree_ptr</code>	pointer to Tree structure
<code>seq</code>	fastq read

Returns

true if read was found, false otherwise

4.33.3.2 `static int Nfree_Lmer (Fq_read * seq, int minL) [static]`

Finds the largest Nfree sub-seq and keeps it if larger than minL.

Parameters

<i>seq</i>	fastq read
<i>minL</i>	minimum accepted trimmed length

Returns

0 if not used, 1 if accepted as is, 2 if accepted and trimmed

4.33.3.3 `static int no_lowQ (Fq_read * seq, int minQ) [static]`

checks if a sequence contains lowQ nucleotides

Parameters

<i>seq</i>	fastq read
<i>minQ</i>	minimum accepted quality value

Returns

0 if seq contains lowQ nucleotides, 1 otherwise

4.33.3.4 `static int no_N (Fq_read * seq) [static]`

checks if a sequence contains any non standard base callings (N's)

Returns

0 if no N's found, 1 if N's found

This function checks if any of the base callings in a given fastq read is different from A, C, G, T. Basically, any char different from the former ones is classified as N.

4.33.3.5 `static int Ntrim_ends (Fq_read * seq, int minL) [static]`

trims a read if N's are at the ends and the remaining sub-seq >= minL

Parameters

<i>seq</i>	fastq read
<i>minL</i>	minimum accepted trimmed length

Returns

0 if not used, 1 no N's found, 2 if accepted and trimmed

4.33.3.6 `static int Qtrim_ends (Fq_read * seq, int minQ, int minL) [static]`

trims a read if lowQs are at the ends and remaining sub-seq >= minL

Parameters

<i>seq</i>	fastq read
<i>minQ</i>	minimum accepted quality value
<i>minL</i>	minimum accepted trimmed length

Returns

0 if not used, 1 if accepted as is, 2 if accepted and trimmed

4.33.3.7 `static int Qtrim_endsfrac (Fq_read * seq, int minQ, int minL, int nlowQ)` `[static]`

trims the ends for lowQ. The rest is kept if it contains < nlowQ lowQ

Parameters

<i>seq</i>	fastq read
<i>minQ</i>	minimum accepted quality value
<i>minL</i>	minimum accepted trimmed length
<i>nlowQ</i>	threshold on lowQ nucleotides (>= NOT allowed)

Returns

0 if not used, 1 if accepted as is

Trims the ends if there are lowQ bases. From the remaining part, it counts how many lowQ bases there are and keeps it if there are less than nlowQ.

4.33.3.8 `static int Qtrim_frac (Fq_read * seq, int minQ, int nlowQ)` `[static]`

accepts the sequence as is if there are less than nlowQ

Parameters

<i>seq</i>	fastq read
<i>minQ</i>	minimum accepted quality value
<i>nlowQ</i>	threshold on lowQ nucleotides (>= NOT allowed)

Returns

0 if not used, 1 if accepted as is

4.33.3.9 `static int Qtrim_global (Fq_read * seq, int left, int right)` `[static]`

trims left from the left and right from the right

Parameters

<i>seq</i>	fastq read
<i>left</i>	number of nucleotides to be trimmed from the left
<i>right</i>	number of nucleotides to be trimmed from the right

Returns

2, since they are all accepted and trim

4.33.3.10 `int trim_sequenceN (Fq_read * seq)`

trims sequence based on presence of N nucleotides

Parameters

<i>seq</i>	fastq read
------------	------------

Returns

-1 error, 0 discarded, 1 accepted as is, 2 accepted and trimmed

This function calls a different function depending on the method passed as input `par_TF.trimN`:

- **NO(0)**: accepts it as is, (1),
- **ALL(1)**: accepts it as is if NO N's found (1), rejects it otherwise (0),
- **ENDS(2)**: trims the ends and accepts it if it is longer than minL (2 if trimming, 1 if no trimming), rejects it otherwise (0),
- **STRIP(3)**: finds the longest N-free subsequence and trims it if it is at least minL nucleotides long (2 if trimming, 1 if no N's are found), rejects it otherwise (0).

4.33.3.11 int trim_sequenceQ (Fq_read * seq)

trims sequence based on lowQ base callings

Parameters

<i>seq</i>	fastq read
------------	------------

Returns

-1 error, 0 discarded, 1 accepted as is, 2 accepted and trimmed

This function calls a different function depending on the method passed as input `par_TF.trimQ`:

- **NO(0)**: accepts is as is , (1),
- **FRAC(1)**: accepts it if less than `par_TF.nlowQ` are found (1), rejects it otherwise (0),
- **ENDS(2)**: trims the ends and accepts it if it is longer than minL (2 if triming, 1 if no trimming), rejects it otherwise (0),
- **ENDSFRAC(3)**: trims the ends and accepts if the remaining sequence is at least minL bases long and if it contains less than `nlowQ` lowQ nucleotides (2 if trimming, 1 if no trimming). Otherwise, it is rejected, (0).
- **GLOBAL(4)**: it trims globally globleft nucleotides from the left and globright from the right, (returns 2).

4.33.4 Variable Documentation**4.33.4.1 char LT[256]**

global variable. Lookup table.

4.33.4.2 char Nencode

global variable. Encoding for N's(\004, or \005)

4.33.4.3 lparam_trimFilter par_TF

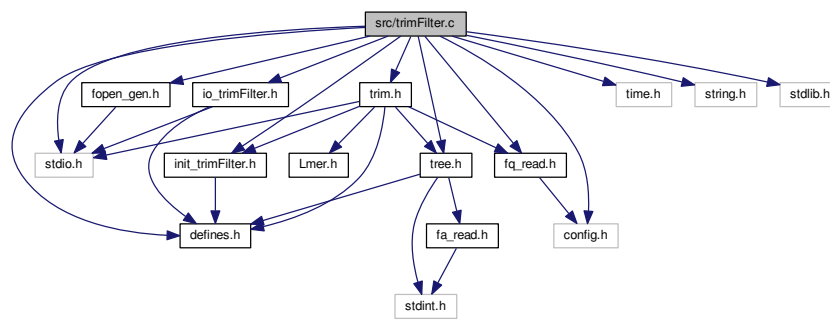
global variable: Input parameters of makeTree.

4.34 src/trimFilter.c File Reference

trimFilter main function

```
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include "fq_read.h"
#include "fopen_gen.h"
#include "config.h"
#include "defines.h"
#include "init_trimFilter.h"
#include "io_trimFilter.h"
#include "tree.h"
#include "trim.h"
```

Include dependency graph for trimFilter.c:



Functions

- int [main](#) (int argc, char *argv[])
makeTree main function

Variables

- uint64_t [alloc_mem](#) = 0
- [lparam_trimFilter](#) [par_TF](#)

4.34.1 Detailed Description

trimFilter main function

Author

Paula Perez paulaperezrubio@gmail.com

Date

25.08.2017 This file contains the trimFilter main function. BLA BLA See README_trimFilter.md for more details.

4.34.2 Variable Documentation

4.34.2.1 `uint64_t alloc_mem = 0`

global variable. Memory allocated in the heap.

4.34.2.2 `lparam_trimFilter par_TF`

global variable: Input parameters of makeTree.

Index

- nreads
 - statsinfo, [18](#)
- ntiles
 - statsinfo, [18](#)
- statsinfo, [17](#)
 - nreads, [18](#)
 - ntiles, [18](#)