FastqArazketa

Generated by Doxygen 1.8.8

Wed Aug 23 2017 17:06:01

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

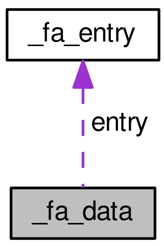
Class Documentation

3.1 _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

3.1.1 Detailed Description

stores sequences of a fasta file

3.1.2 Member Data Documentation

3.1.2.1 Fa_entry∗ _fa_data::entry

Array with fasta entries (see Fa_entry)

3.1.2.2 uint64_t∗ _fa_data::entrylen

Array containing the length of the entries

3.1.2.3 int _fa_data::linelen

Line length of the ∗fa file entries

3.1.2.4 int _fa_data::nentries

Number of entries in ∗fa file

3.1.2.5 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

3.2 _fa_entry Struct Reference

fasta entry

`#include <fa_read.h>`

Public Attributes

- uint64_t N
- char ∗ seq

3.2.1 Detailed Description

fasta entry

3.2.2 Member Data Documentation

3.2.2.1 uint64_t _fa_entry::N

Entry length (chars)

3.2.2.2 char∗ _fa_entry::seq

sequence

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

- include/fa_read.h

3.3 _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

3.3.1 Detailed Description

stores a fastq entry

3.3.2 Member Data Documentation

3.3.2.1 int _fq_read::L

read length

3.3.2.2 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

3.4 _iparam_makeTree Struct Reference

contains makeTree input parameters

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

Public Attributes

- char ∗ inputfasta
- char outputfile [MAX_FILENAME]
- int L

3.4.1 Detailed Description

contains makeTree input parameters

3.4.2 Member Data Documentation

3.4.2.1 char∗ _iparam_makeTree::inputfasta

fasta input file

3.4.2.2 int _iparam_makeTree::L

tree depth

3.4.2.3 char _iparam_makeTree::outputfile[MAX_FILENAME]

outputfile path

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

- include/init_makeTree.h

3.5 _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

3.5.1 Detailed Description

contains Qreport input parameters

3.5.2 Member Data Documentation

3.5.2.1 int _iparam_Qreport::filter

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

3.5.2.2 char∗ _iparam_Qreport::inputfile

Inputfile name

3.5.2.3 int _iparam_Qreport::minQ

minimum Quality allowed 0 - 45

3.5.2.4 int _iparam_Qreport::nQ

# different quality values (default is 46)

3.5.2.5 int _iparam_Qreport::ntiles

# tiles (default is 96)

3.5.2.6 int _iparam_Qreport::one_read_len

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

3.5.2.7 char _iparam_Qreport::outputfilebin[MAX_FILENAME]

Binary outputfile name.

3.5.2.8 char _iparam_Qreport::outputfilehtml[MAX_FILENAME]

html outputfile name

3.5.2.9 char _iparam_Qreport::outputfileinfo[MAX_FILENAME]

Info outputfile name

3.5.2.10 int _iparam_Qreport::read_len

original read length

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

- include/init_Qreport.h

3.6 _iparam_Sreport Struct Reference

contains Sreport input parameters

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

Public Attributes

- char ∗ inputfolder
- char outputfile [MAX_FILENAME]

3.6.1 Detailed Description

contains Sreport input parameters

3.6.2 Member Data Documentation

3.6.2.1 char∗ _iparam_Sreport::inputfolder

input folder

3.6.2.2 char _iparam_Sreport::outputfile[MAX_FILENAME]

html outputfile name

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

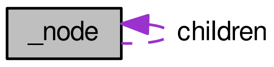
- include/init_Sreport.h

3.7 _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]

3.7.1 Detailed Description

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

3.7.2 Member Data Documentation

3.7.2.1 struct _node∗ _node::children[T_ACGT]

T_ACGT pointers to Node structure

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

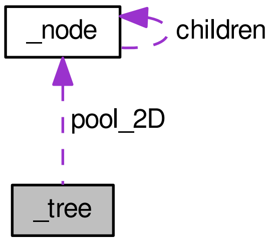
- include/tree.h

3.8 _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

3.8.1 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!).

3.8.2 Member Data Documentation

3.8.2.1 uint32_t _tree::L

depth of the tree

3.8.2.2 uint32_t _tree::nnodes

Number of nodes in the tree

3.8.2.3 Node∗∗ _tree::pool_2D

2D pool containing the nodes that form the tree

3.8.2.4 uint32_t _tree::pool_available

Number of empty nodes available in the pool

3.8.2.5 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

3.9 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

3.9.1 Detailed Description

stores info needed to create the summary graphs

3.9.2 Member Data Documentation

3.9.2.1 uint64_t∗ statsinfo::ACGT_pos

# A, C, G, T, N per position

3.9.2.2 uint64_t∗ statsinfo::ACGT_tile

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

3.9.2.3 int∗ statsinfo::lane_tags

Names of the existing tiles

3.9.2.4 uint64_t∗ statsinfo::lowQ_ACGT_tile

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

3.9.2.5 int statsinfo::minQ

Minimum quality threshold

3.9.2.6 int statsinfo::nQ

# possible quality values

3.9.2.7 int statsinfo::nreads

# reads read till current position.

3.9.2.8 int statsinfo::ntiles

# tiles

3.9.2.9 uint64_t∗ statsinfo::QPosTile_table

# bases of a given quality per tile.

3.9.2.10 int∗ statsinfo::qual_tags

Names of the existing qualities

3.9.2.11 int statsinfo::read_len

Maximum length of a read

3.9.2.12 uint64_t∗ statsinfo::reads_MlowQ

# reads with M(position) lowQuality bases.

3.9.2.13 int statsinfo::reads_wN

# reads with N's found till current position

3.9.2.14 int statsinfo::sz_ACGT_pos

ACGT_pos size = read_len $*$ N_ACGT

3.9.2.15 int statsinfo::sz_ACGT_tile

ACGT_tile size = ntiles $*$ NACGT

3.9.2.16 int statsinfo::sz_lowQ_ACGT_tile

lowQ_ACGT_tile size = ntiles $*$ N_ACGT

3.9.2.17 int statsinfo::sz_QPosTile_table

QposTile_Table size = ntiles $*$ nQ $*$ read_len

3.9.2.18 int statsinfo::sz_reads_MlowQ

reads_MlowQ size = read_len + 1

3.9.2.19 int statsinfo::tile_pos

current tile position

3.9.2.20 int$*$ statsinfo::tile_tags

Names of the existing tiles

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

- include/stats_info.h

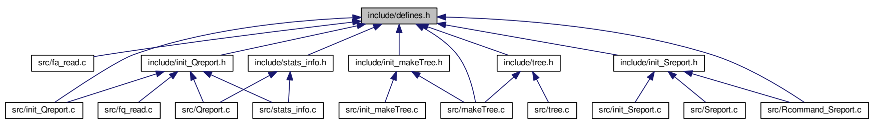
Chapter 4

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

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 B_LEN 131072

buffer size

4.1.2.2 #define bool short

define a bool type

4.1.2.3 #define DEFAULT_MINQ 27

Minimum quality threshold

4.1.2.4 #define DEFAULT_NQ 46

Default number of different quality values

4.1.2.5 #define DEFAULT_NTILES 96

Default number of tiles

4.1.2.6 #define FA_ENTRY_BUF 20

buffer for fasta entries

4.1.2.7 #define false 0

assign false to 0

4.1.2.8 #define max(*a*, *b*) (((a) > (b)) ? (a) : (b))

max function

4.1.2.9 #define MAX_FASZ_TREE 1e7

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

4.1.2.10 #define MAX_FILENAME 300

Maximum # chars in a filename

4.1.2.11 #define MAX_RCOMMAND 4000

Maximum # chars in R command

4.1.2.12 #define mem_usage( )

Value:

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

returns allocated memory in Bytes

4.1.2.13 #define mem_usageMB( )

Value:

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

returns allocated memory in MB

4.1.2.14 #define min(*a*, *b*) (((a) < (b)) ? (a) : (b))

min function

4.1.2.15 #define N_ACGT 5

Number of different nucleotides in the fq file

4.1.2.16 #define NPOOL_1D 1048576

Number of Node structs allocated in inner dim

4.1.2.17 #define NPOOL_2D 16

Number of $*$Node allocated in outer dim

4.1.2.18 #define T_ACGT 4

Number of children per node in tree

4.1.2.19 #define true 1

assign true to 1

4.1.2.20 #define ZEROQ 33

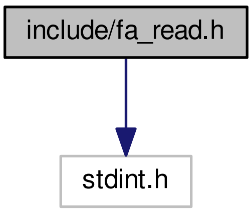
ASCII code of lowest quality value (!)

4.2 include/fa_read.h File Reference

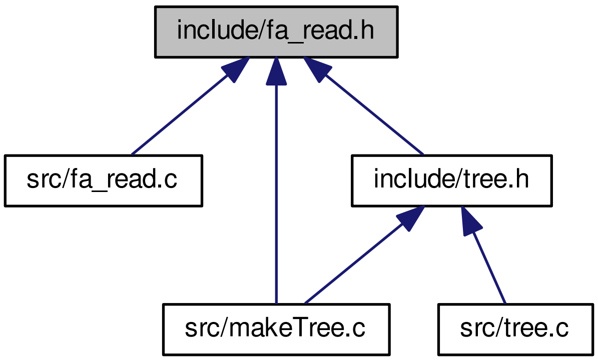
reads in and stores fasta files

```
#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

ptr_fa	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

filename	path to a fasta input file.
ptr_fa	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

ptr_fa	pointer to Fa_data

Returns

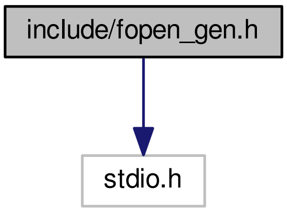
total number of nucleotides

4.3 include/fopen_gen.h File Reference

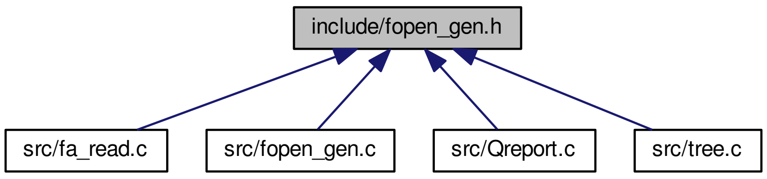
Uncompress/compress input/output files using pipes.

```
#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

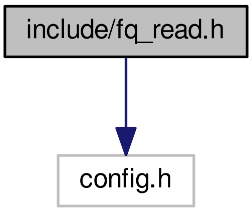
a FILE pointer

4.4 include/fq_read.h File Reference

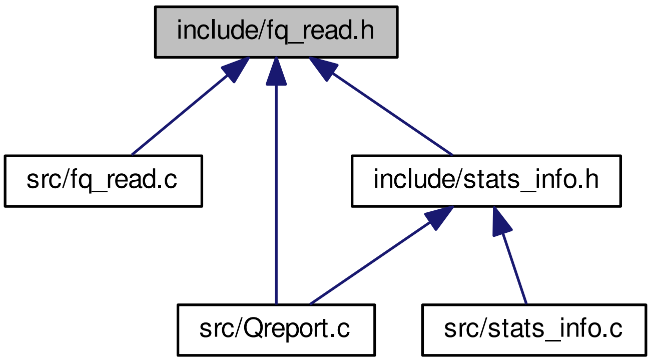
fastq entries manipulations (read/write)

```
#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

- void get_fqread (Fq_read ∗seq, char ∗buffer, int c1, int c2, int k)

  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 void get_fqread (**Fq_read** ∗ *seq,* char ∗ *buffer,* int *pos1,* int *pos2,* int *nline*)

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 variable **par_QR** values, information about whether the read was trimmed is stored.

Parameters

∗seq	pointer to **Fq_read**, where the info will be stored.
buffer	variable where the file being read is stored.
pos1	buffer start position of the line.
pos2	buffer end position of the line.
nline	file line number being read.

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

writes the fq entry in a string

Parameters

∗seq	pointer to **Fq_read**, where the info will be stored.
char_seq	pointer to buffer, where the sequence will be stored

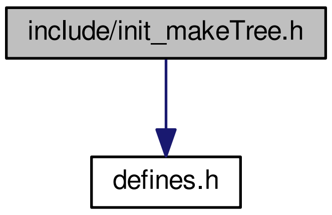
Warning

change the call to sprintf to snprintf

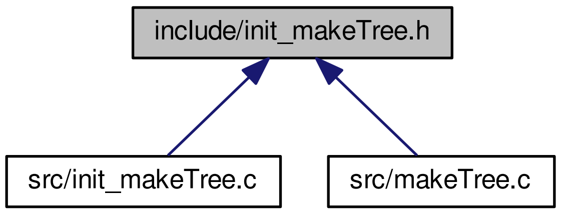
4.5 include/init_makeTree.h File Reference

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`

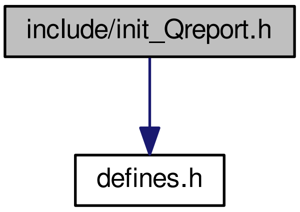
Date

23.08.2017

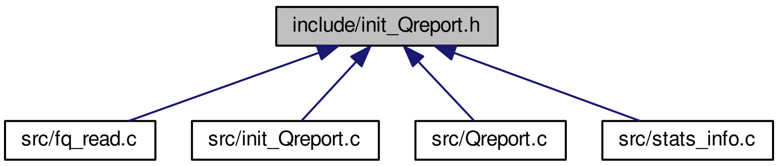
4.6 include/init_Qreport.h File Reference

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

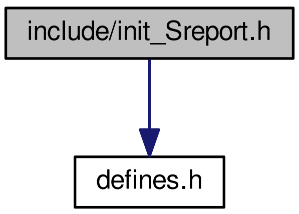
Date

03.08.2017

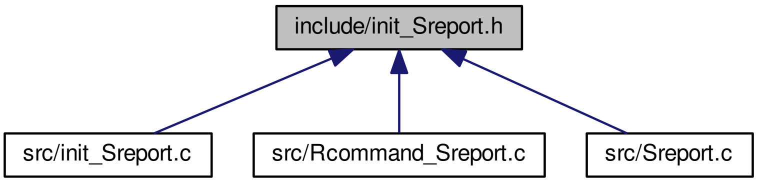
4.7 include/init_Sreport.h File Reference

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 Iparam_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`

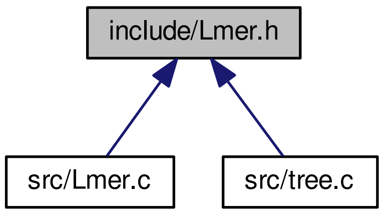
Date

09.08.2017

4.8 include/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'}.

4.8.1 Detailed Description

Manipulation of Lmers and sequences.

Author

Paula Perez paulaperezrubio@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.

4.8.2 Function Documentation

4.8.2.1 void init_map ( )

Initialize lookup table LT.

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

4.8.2.2 void init_map_SA ( )

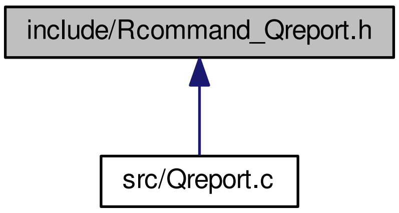
Initialize lookup table LT (for SA)

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

4.9 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

4.9.1 Detailed Description

get Rscript command for Qreport

Author

Paula Perez paulaperezrubio@gmail.com

Date

07.08.2017

Author

Paula Perez paulaperezrubio@gmail.com

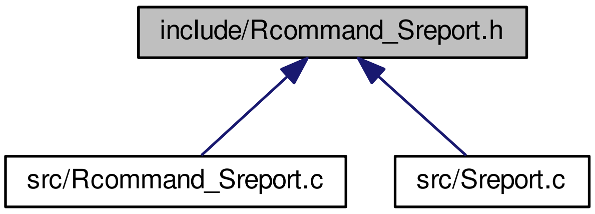
Date

09.08.2017

4.10 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

4.10.1 Detailed Description

get Rscript command for Sreport

Author

Paula Perez paulaperezrubio@gmail.com

Date

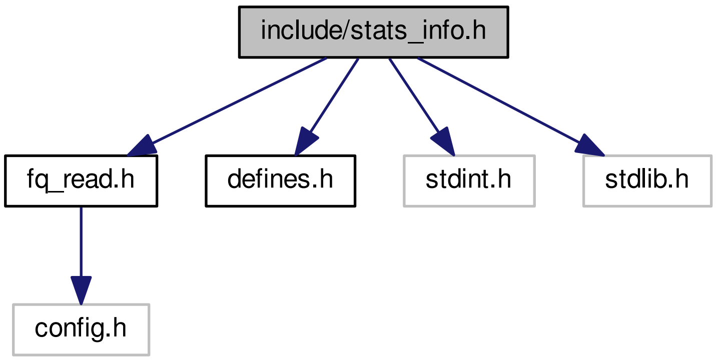
09.08.2017

4.11 include/stats_info.h File Reference

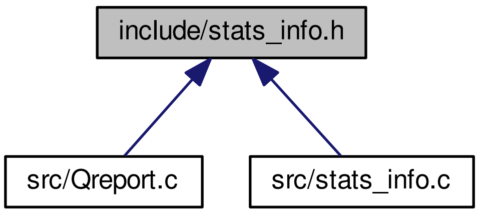
Construct the quality report variables and update them.

```
#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

4.11.1 Detailed Description

Construct the quality report variables and update them.

Author

Paula Perez `paulaperezrubio@gmail.com`

Date

04.08.2017

4.11.2 Function Documentation

4.11.2.1 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).

4.11.2.2 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.

4.11.2.3 int update_ACGT_counts (uint64_t ∗ *ACGT_low*, char *ACGT*)

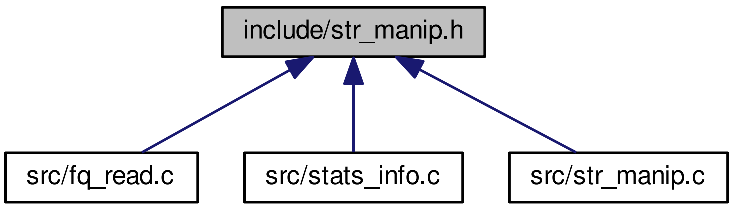
update, for current tile, ACGT counts.

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

4.12 include/str_manip.h File Reference

functions that do string manipulation

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



Functions

- int strindex (char ∗s, char ∗t)

  returns index of t in s (start, first occurence)
- int count_char (char ∗s, char c)

  returns the # of occurences of char c in string s

4.12.1 Detailed Description

functions that do string manipulation

Author

Paula Perez paulaperezrubio@gmail.com

Date

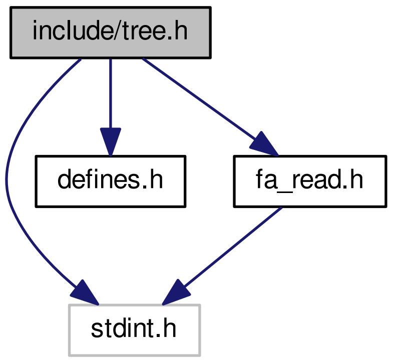
03.08.2017

4.13 include/tree.h File Reference

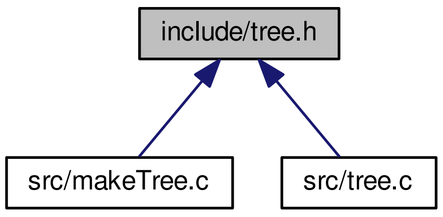
Construction of tree, check paths, write tree, read in tree.

```
#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).
- bool check_path (Node ∗tree, char ∗Lmer, int L, int Lread)

  check if Lread is contained in tree.
- 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

4.13.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

18.08.2017

4.13.2 Typedef Documentation

4.13.2.1 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!).

4.13.3 Function Documentation

4.13.3.1 bool check_path (Node ∗ *tree,* char ∗ *Lmer,* int *L,* int *Lread*)

check if Lread is contained in tree.

change it so that it returns a score!

4.13.3.2 void free_all_nodes (Tree ∗ *tree_ptr*)

frees the whole tree structure

Parameters

tree_ptr	pointer to Tree structure

This function deallocates the memory allocated in a Tree structure.

4.13.3.3 Node∗ get_new_pool (Tree ∗ *tree_ptr*)

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

Parameters

tree_ptr	pointer to Tree structure

4.13.3.4 Node∗ new_node_buf (Tree ∗ *tree_ptr*)

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

Parameters

tree_ptr	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 outter dimension, it reallocates NPOOL_2D more Node∗'s. If the number of nodes reaches UINT_MAX, the program returns an error message and exits.

4.13.3.5 Tree∗ read_tree (char ∗ *filename*)

read tree from file

Parameters

filename	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.

4.13.3.6 void save_tree (Tree ∗ *tree_ptr,* char ∗ *filename*)

saves Tree to disk in filename

Parameters

tree_ptr	pointer to Tree structure
filename	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:

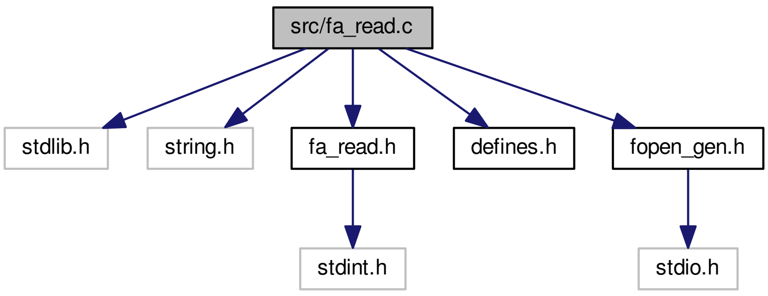
- If it is pointing to NULL: 0.
- Otherwise: i2, the index in the outer dimension of pool_2D is identified, and the difference jump = pool_2↵D[i][j].children[k] - pool_2D[i2] is computed. i2∗NPOOL_D1 + jump is then stored for child k.

4.14 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

4.14.1 Detailed Description

reads in and stores fasta files

Author

Paula Perez paulaperezrubio@gmail.com

Date

18.08.2017

4.14.2 Function Documentation

4.14.2.1 void free_fasta (**Fa_data** ∗ *ptr_fa*)

free fasta file

Parameters

ptr_fa	pointer to Fa_data structure.

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

4.14.2.2 static int ignore_line (char ∗ *line*) `[static]`

ignore header lines.

Parameters

line	string of characters.

Returns

number of characters to jump until a
is found.

4.14.2.3 static void init_entries (**Fa_data** ∗ *ptr_fa*) `[static]`

Allocation of Fa_entries.

Parameters

ptr_fa	pointer to Fa_data structure.

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

4.14.2.4 static void init_fa (**Fa_data** ∗ *ptr_fa*) `[static]`

Initialization of Fa_data.

Parameters

ptr_fa	pointer to Fa_data structure.

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

4.14.2.5 int read_fasta (char ∗ *filename,* **Fa_data** ∗ *ptr_fa*)

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

Parameters

filename	path to a fasta input file.
ptr_fa	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.

4.14.2.6 static void realloc_fa (**Fa_data** ∗ *ptr_fa*) `[static]`

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

Parameters

ptr_fa	pointer to Fa_data structure.

4.14.2.7 uint64_t size_fasta (**Fa_data** ∗ *ptr_fa*)

computes length of genome in fasta structure

Parameters

ptr_fa	pointer to Fa_data

Returns

total number of nucleotides

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

this function sweeps a fasta file to obtain structure details.

Parameters

filename	path to a fasta input file.
ptr_fa	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.

4.14.3 Variable Documentation

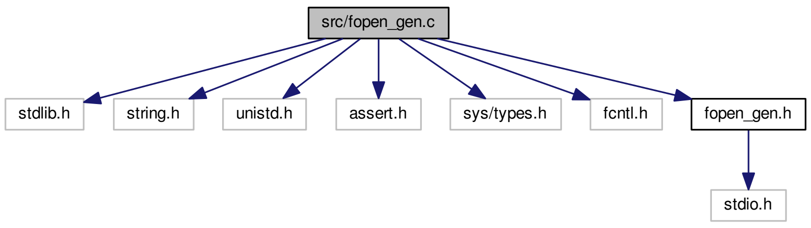
4.14.3.1 uint64_t alloc_mem

global variable. Memory allocated in the heap.

4.15 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.

4.15.1 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

4.15.2 Function Documentation

4.15.2.1 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

4.15.2.2 static FILE∗ fcompress (const char ∗ *path*) `[static]`

Open a pipe to compress the specified file.

Returns

a FILE pointer

4.15.2.3 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

4.15.2.4 static FILE∗ funcompress (const char ∗ *path*) [static]

Open a pipe to uncompress the specified file.

Returns

a FILE pointer

4.15.2.5 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

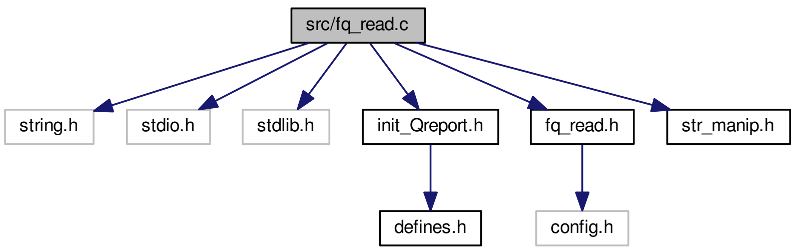
a file descriptor

4.16 src/fq_read.c File Reference

fastq entries manipulations (read/write)

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

Include dependency graph for fq_read.c:



Functions

- void get_fqread (Fq_read ∗seq, char ∗buffer, int pos1, int pos2, int nline)

  reads fastq line from a buffer
- int string_seq (Fq_read ∗seq, char ∗char_seq)

  writes the fq entry in a string

Variables

- Iparam_Qreport par_QR

4.16.1 Detailed Description

fastq entries manipulations (read/write)

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.16.2 Function Documentation

4.16.2.1 void get_fqread (**Fq_read** ∗ *seq,* char ∗ *buffer,* int *pos1,* int *pos2,* int *nline*)

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 variable **par_QR** values, information about whether the read was trimmed is stored.

Parameters

∗*seq*	pointer to **Fq_read**, where the info will be stored.
buffer	variable where the file being read is stored.
pos1	buffer start position of the line.
pos2	buffer end position of the line.
nline	file line number being read.

4.16.2.2 int string_seq (**Fq_read** ∗ *seq,* char ∗ *char_seq*)

writes the fq entry in a string

Parameters

∗*seq*	pointer to **Fq_read**, where the info will be stored.
char_seq	pointer to buffer, where the sequence will be stored

Warning

change the call to sprintf to snprintf

4.16.3 Variable Documentation

4.16.3.1 **Iparam_Qreport** par_QR

input parameters

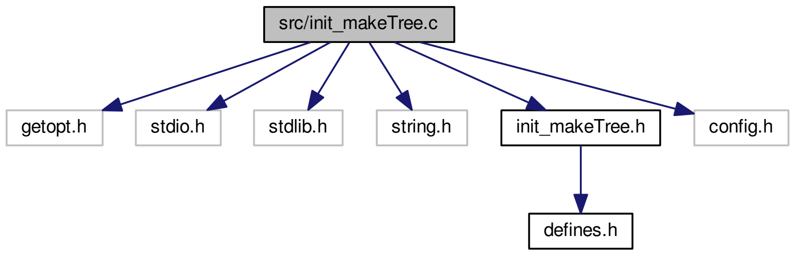
global variable: input parameters for Qreport

4.17 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

- Iparam_makeTree par_MT

4.17.1 Detailed Description

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

Author

Paula Perez `paulaperezrubio@gmail.com`

Date

23.08.2017

4.17.2 Variable Documentation

4.17.2.1 Iparam_makeTree par_MT

Input parameters of makeTree

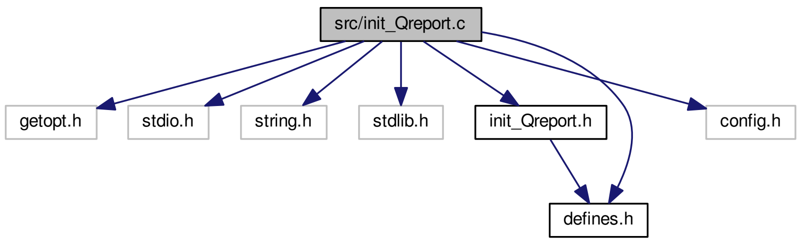
global variable: Input parameters of makeTree.

4.18 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

- Iparam_Qreport par_QR

4.18.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.18.2 Variable Documentation

4.18.2.1 Iparam_Qreport par_QR

Input parameters of Qreport

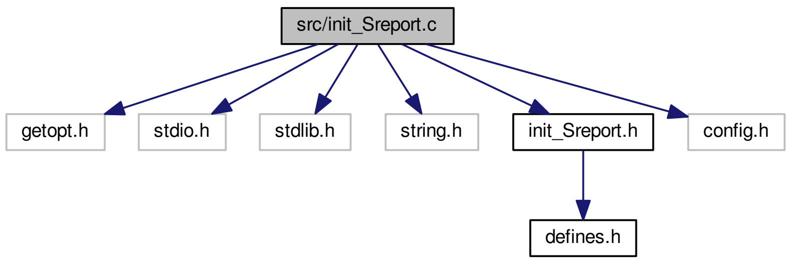
global variable: input parameters for Qreport

4.19 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

- Iparam_Sreport par_SR

4.19.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

09.08.2017

4.19.2 Variable Documentation

4.19.2.1 Iparam_Sreport par_SR

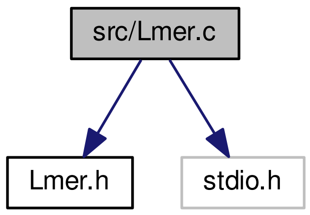
input parameters Sreport

4.20 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]

4.20.1 Detailed Description

Manipulation of Lmers and sequences.

Author

Paula Perez paulaperezrubio@gmail.com

Date

18.08.2017

4.20.2 Function Documentation

4.20.2.1 void init_map ( )

Initialize lookup table LT.

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

4.20.2.2 void init_map_SA ( )

Initialize lookup table LT (for SA)

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

4.20.3 Variable Documentation

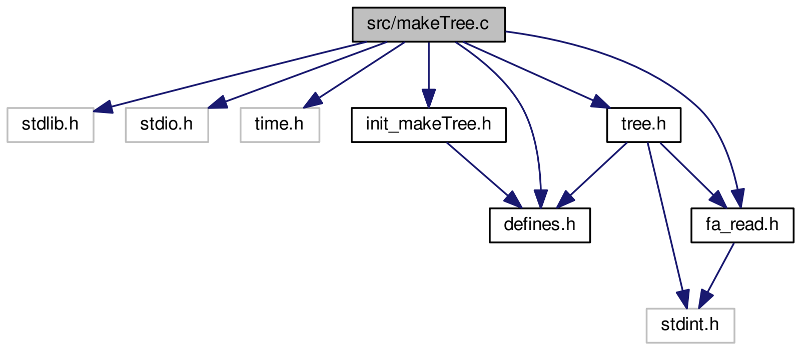
4.20.3.1 char LT[256]

global variable. Lookup table.

4.21 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
- Iparam_makeTree par_MT

4.21.1 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.

4.21.2 Variable Documentation

4.21.2.1 uint64_t alloc_mem = 0

global variable. Memory allocated in the heap.

4.21.2.2 Iparam_makeTree par_MT

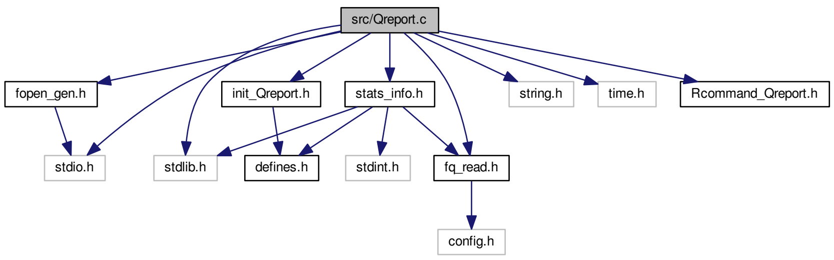
global variable: Input parameters of makeTree.

4.22 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

- Iparam_Qreport par_QR

4.22.1 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.

4.22.2 Variable Documentation

4.22.2.1 Iparam_Qreport par_QR

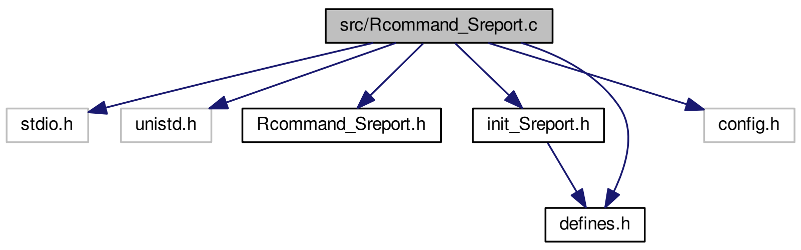
global variable: input parameters for Qreport

4.23 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

- Iparam_Sreport par_SR

4.23.1 Detailed Description

get Rscript command for Sreport

Author

Paula Perez paulaperezrubio@gmail.com

Date

09.08.2017

4.23.2 Variable Documentation

4.23.2.1 Iparam_Sreport par_SR

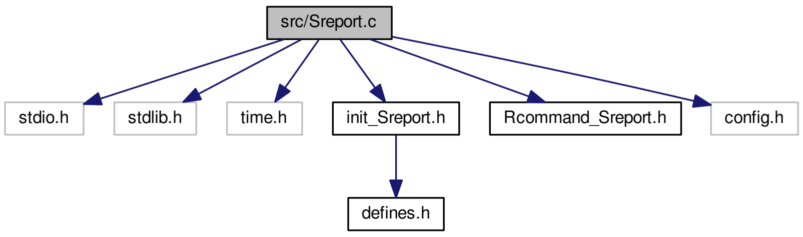
input parameters Sreport

4.24 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

- Iparam_Sreport par_SR

4.24.1 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.

4.24.2 Variable Documentation

4.24.2.1 Iparam_Sreport par_SR

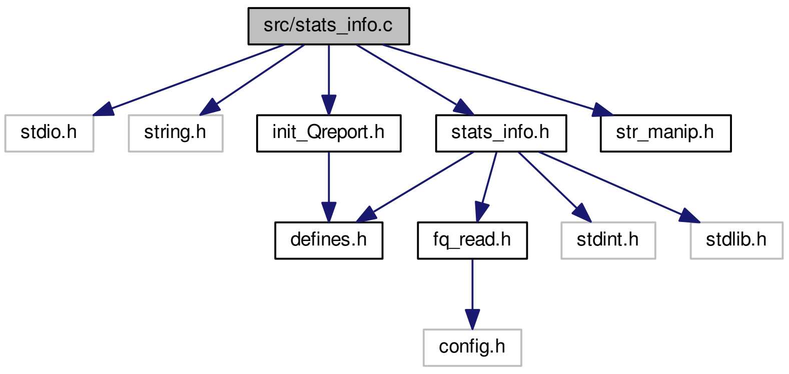
input parameters Sreport

4.25 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

- Iparam_Qreport par_QR

4.25.1 Detailed Description

Construct the quality report variables and update them.

Author

Paula Perez paulaperezrubio@gmail.com

Date

04.08.2017

4.25.2 Function Documentation

4.25.2.1 void get_tile_lane (char ∗ *line1,* int ∗ *tile,* int ∗ *lane*)

get tile number from first line in fastq entry.

Parameters

line1	first line of a fastq entry
tile	int∗ where the tile will be stored
lane	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.

4.25.2.2 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).

4.25.2.3 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.

4.25.2.4 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

4.25.3 Variable Documentation

4.25.3.1 **Iparam_Qreport** par_QR

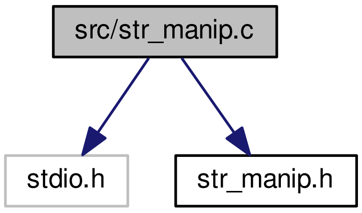
global variable: input parameters for Qreport

4.26 src/str_manip.c File Reference

functions that do string manipulation

```
#include <stdio.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 occurence)
- int count_char (char ∗s, char c)

  returns the # of occurences of char c in string s

4.26.1 Detailed Description

functions that do string manipulation

Author

Paula Perez paulaperezrubio@gmail.com

Date

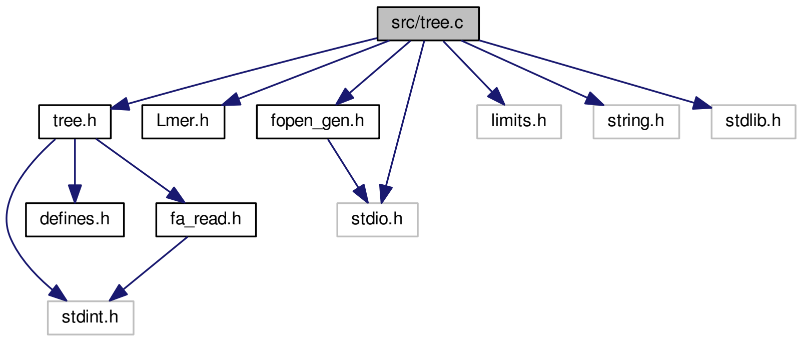
03.08.2017

4.27 src/tree.c File Reference

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

```
#include "tree.h"
#include "Lmer.h"
#include "fopen_gen.h"
#include <limits.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.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.
- bool check_path (Node ∗tree, char ∗Lmer, int L, int Lread)

  check if Lread is contained in tree.
- 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

4.27.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

23.08.2017

4.27.2 Function Documentation

4.27.2.1 bool check_path (Node ∗ *tree,* char ∗ *Lmer,* int *L,* int *Lread*)

check if Lread is contained in tree.

change it so that it returns a score!

4.27.2.2 void free_all_nodes (Tree ∗ *tree_ptr*)

frees the whole tree structure

Parameters

tree_ptr	pointer to Tree structure

This function deallocates the memory allocated in a Tree structure.

4.27.2.3 Node∗ get_new_pool (Tree ∗ *tree_ptr*)

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

Parameters

tree_ptr	pointer to Tree structure

4.27.2.4 Node∗ new_node_buf (Tree ∗ *tree_ptr*)

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

Parameters

tree_ptr	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 outter dimension, it reallocates NPOOL_2D more Node∗'s. If the number of nodes reaches UINT_MAX, the program returns an error message and exits.

4.27.2.5 Tree∗ read_tree (char ∗ *filename*)

read tree from file

Parameters

filename	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.

4.27.2.6 void save_tree (Tree ∗ *tree_ptr,* char ∗ *filename*)

saves Tree to disk in filename

Parameters

tree_ptr	pointer to Tree structure
filename	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_2↩ D[i][j].children[k] - pool_2D[i2] is computed. i2∗NPOOL_D1 + jump is then stored for child k.

4.27.3 Variable Documentation

4.27.3.1 uint64_t alloc_mem

global variable. Memory allocated in the heap.

Index