FastqArazketa

Generated by Doxygen 1.8.8

Fri Sep 8 2017 11:45:55

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 _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 _bfilter Struct Reference

Bloom filter structure.

`#include <bloom.h>`

Public Attributes

- int kmersize
- int hashNum
- int kmersizeBytes
- double falsePosRate
- uint64_t bfsizeBits
- uint64_t bfsizeBytes
- uint64_t nelem
- unsigned char * filter

3.2.1 Detailed Description

Bloom filter structure.

3.2.2 Member Data Documentation

3.2.2.1 uint64_t _bfilter::bfsizeBits

bloom filter size (bits) (m)

3.2.2.2 uint64_t _bfilter::bfsizeBytes

bloom filter size (bytes)

3.2.2.3 double _bfilter::falsePosRate

False positive rate

3.2.2.4 unsigned char* _bfilter::filter

filter sequence

3.2.2.5 int _bfilter::hashNum

number of hash functions used to construct the filter

3.2.2.6 int _bfilter::kmersize

kmer size (number of elements)

3.2.2.7 int _bfilter::kmersizeBytes

Bytes needed to store the kmer (4bases $\sim$ 1byte)

3.2.2.8 uint64_t _bfilter::nelem

number of elements encoded in the bloom filter (n)

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

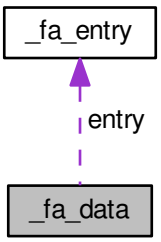
- include/bloom.h

3.3 _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

3.3.1 Detailed Description

stores sequences of a fasta file

3.3.2 Member Data Documentation

3.3.2.1 Fa_entry* _fa_data::entry

Array with fasta entries (see Fa_entry)

3.3.2.2 uint64_t* _fa_data::entrylen

Array containing the length of the entries

3.3.2.3 int _fa_data::linelen

Line length of the *fa file entries

3.3.2.4 int _fa_data::nentries

Number of entries in *fa file

3.3.2.5 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

3.4 _fa_entry Struct Reference

fasta entry

`#include <fa_read.h>`

Public Attributes

- uint64_t N
- char * seq

3.4.1 Detailed Description

fasta entry

3.4.2 Member Data Documentation

3.4.2.1 uint64_t _fa_entry::N

Entry length (chars)

3.4.2.2 char* _fa_entry::seq

sequence

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

- include/fa_read.h

3.5 _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

3.5.1 Detailed Description

stores a fastq entry

3.5.2 Member Data Documentation

3.5.2.1 int _fq_read::L

read length

3.5.2.2 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

3.6 _iparam_makeBloom Struct Reference

contains makeBloom input parameters

`#include <init_makeBloom.h>`

Public Attributes

- char * inputfasta
- char filterfile [MAX_FILENAME]
- char paramfile [MAX_FILENAME]
- int kmersize
- int hashNum
- double falsePosRate
- uint64_t bfsizeBits
- uint64_t nelem

3.6.1 Detailed Description

contains makeBloom input parameters

Note

nelemen will be computed once the fasta file is read and loaded.

3.6.2 Member Data Documentation

3.6.2.1 uint64_t _iparam_makeBloom::bfsizeBits

bloom filter size (bits)

3.6.2.2 double _iparam_makeBloom::falsePosRate

false positive rate

3.6.2.3 char _iparam_makeBloom::filterfile[MAX_FILENAME]

filter file path

3.6.2.4 int _iparam_makeBloom::hashNum

number of hash functions used to construct the filter

3.6.2.5 char* _iparam_makeBloom::inputfasta

fasta input file

3.6.2.6 int _iparam_makeBloom::kmersize

kmer size (number of elements)

3.6.2.7 uint64_t _iparam_makeBloom::nelem

number of elements that the bloomfilter will contain

3.6.2.8 char _iparam_makeBloom::paramfile[MAX_FILENAME]

param file path

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

- include/init_makeBloom.h

3.7 _iparam_makeTree Struct Reference

contains makeTree input parameters

`#include <init_makeTree.h>`

Public Attributes

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

3.7.1 Detailed Description

contains makeTree input parameters

3.7.2 Member Data Documentation

3.7.2.1 char* _iparam_makeTree::inputfasta

fasta input file

3.7.2.2 int _iparam_makeTree::L

tree depth

3.7.2.3 char _iparam_makeTree::outputfile[MAX_FILENAME]

outputfile path

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

- include/init_makeTree.h

3.8 _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

3.8.1 Detailed Description

contains Qreport input parameters

3.8.2 Member Data Documentation

3.8.2.1 int _iparam_Qreport::filter

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

3.8.2.2 char* _iparam_Qreport::inputfile

Inputfile name

3.8.2.3 int _iparam_Qreport::minQ

minimum Quality allowed 0 - 45

3.8.2.4 int _iparam_Qreport::nQ

different quality values (default is 46)

3.8.2.5 int _iparam_Qreport::ntiles

tiles (default is 96)

3.8.2.6 int _iparam_Qreport::one_read_len

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

3.8.2.7 char _iparam_Qreport::outputfilebin[MAX_FILENAME]

Binary outputfile name.

3.8.2.8 char _iparam_Qreport::outputfilehtml[MAX_FILENAME]

html outputfile name

3.8.2.9 char _iparam_Qreport::outputfileinfo[MAX_FILENAME]

Info outputfile name

3.8.2.10 int _iparam_Qreport::read_len

original read length

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

- include/init_Qreport.h

3.9 _iparam_Sreport Struct Reference

contains Sreport input parameters

`#include <init_Sreport.h>`

Public Attributes

- char * inputfolder
- char outputfile [MAX_FILENAME]
- char * Rmd_file

3.9.1 Detailed Description

contains Sreport input parameters

3.9.2 Member Data Documentation

3.9.2.1 char* _iparam_Sreport::inputfolder

input folder

3.9.2.2 char _iparam_Sreport::outputfile[MAX_FILENAME]

html outputfile path

3.9.2.3 char* _iparam_Sreport::Rmd_file

Rmd file path

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

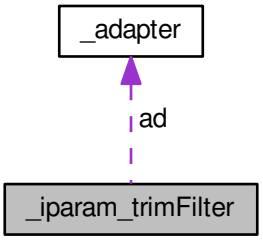
- include/init_Sreport.h

3.10 _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

3.10.1 Detailed Description

trimFilter input parameters

3.10.2 Member Data Documentation

3.10.2.1 Adapter _iparam_trimFilter::ad

Adapter trimming parameters

3.10.2.2 int _iparam_trimFilter::globleft

number of bases globally trimming from the left

3.10.2.3 int _iparam_trimFilter::globright

number of bases globally trimming from the right

3.10.2.4 char* _iparam_trimFilter::lfa

Input fa file (containing contamination sequences)

3.10.2.5 char* _iparam_trimFilter::lfq

Input fq file

3.10.2.6 char* _iparam_trimFilter::lidx

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

3.10.2.7 bool _iparam_trimFilter::is_adapter

true if filtering adapter sequences

3.10.2.8 bool _iparam_trimFilter::is_fa

true if a fasta file was passed as a parameter

3.10.2.9 bool _iparam_trimFilter::is_idx

true if an index file was passed as a parameter

3.10.2.10 int _iparam_trimFilter::L

read length

3.10.2.11 int _iparam_trimFilter::Lmer_len

Lmer length to look for contamination

3.10.2.12 int _iparam_trimFilter::method

TREE(1), SA(2), BLOOM(3), 0, when not looking for cont

3.10.2.13 int _iparam_trimFilter::minL

minimum read length accepted before discarding a read

3.10.2.14 int _iparam_trimFilter::minQ

minimum quality threshold

3.10.2.15 int _iparam_trimFilter::nlowQ

maximum number of lowQ bases accepted before discarding

3.10.2.16 char* _iparam_trimFilter::Oprefix

Output files prefix (PATH/prefix)

3.10.2.17 int _iparam_trimFilter::percent

percentage of lowQ bases allowed in a read

3.10.2.18 double _iparam_trimFilter::score

score threshold for matching reads in sequences

3.10.2.19 int _iparam_trimFilter::trimN

NO(0), ALL(1), ENDS(2), STRIP(3)

3.10.2.20 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:

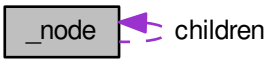
- include/init_trimFilter.h

3.11 _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]

3.11.1 Detailed Description

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

3.11.2 Member Data Documentation

3.11.2.1 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

3.12 _procs_kmer Struct Reference

stores a processed kmer (2 bits pro nucleotide)

`#include <bloom.h>`

Public Attributes

- int kmersize
- int hashNum
- int kmersizeBytes
- int halfsizeBytes
- int hangingBases
- int hasOverhead
- unsigned char * compact
- uint64_t * hashValues

3.12.1 Detailed Description

stores a processed kmer (2 bits pro nucleotide)

3.12.2 Member Data Documentation

3.12.2.1 unsigned char* _procs_kmer::compact

encoded compactified sequence

3.12.2.2 int _procs_kmer::halfsizeBytes

half size in Bytes (needed to decide whether to store a kmer or its reverse complement)

3.12.2.3 int _procs_kmer::hangingBases

number of hanging bases that don't complete a byte

3.12.2.4 int _procs_kmer::hashNum

number of hash functions used to construct the filter

3.12.2.5 uint64_t* _procs_kmer::hashValues

Values of the hash functions

3.12.2.6 int _procs_kmer::hasOverhead

kmer has overhead when kmersize % 4!=0

3.12.2.7 int _procs_kmer::kmersize

kmer size (number of elements)

3.12.2.8 int _procs_kmer::kmersizeBytes

Bytes needed to store the kmer (4bases $\sim$ 1byte)

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

- include/bloom.h

3.13 _split Struct Reference

contains a splitted string and the number or splitted fields

`#include <str_manip.h>`

Public Attributes

- int N
- char ** s

3.13.1 Detailed Description

contains a splitted string and the number or splitted fields

3.13.2 Member Data Documentation

3.13.2.1 int _split::N

Number of substrings in which the string was splitted

3.13.2.2 char** _split::s

Substring array containing the splitted substrings

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

- include/str_manip.h

3.14 _stats_TF Struct Reference

collects stats info from the filtering procedure

`#include <io_trimFilter.h>`

Public Attributes

- int filters [NFILTERS]
- int trimmed [NFILTERS]
- int discarded [NFILTERS]
- int good
- int nreads

3.14.1 Detailed Description

collects stats info from the filtering procedure

3.14.2 Member Data Documentation

3.14.2.1 int _stats_TF::discarded[NFILTERS]

discarded reads: ADAP, CONT, LOWQ, NNNN

3.14.2.2 int _stats_TF::filters[NFILTERS]

Using filters for: ADAP, CONT, LOWQ, NNNN

3.14.2.3 int _stats_TF::good

good reads

3.14.2.4 int _stats_TF::nreads

total number of reads in the fq file

3.14.2.5 int _stats_TF::trimmed[NFILTERS]

trimmed reads by: ADAP, CONT, LOWQ, NNNN

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

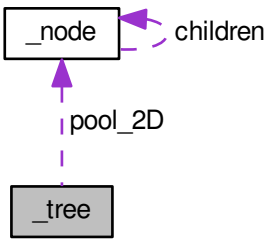
- include/io_trimFilter.h

3.15 _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

3.15.1 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!).

3.15.2 Member Data Documentation

3.15.2.1 uint32_t _tree::L

depth of the tree

3.15.2.2 uint32_t _tree::nnodes

Number of nodes in the tree

3.15.2.3 Node** _tree::pool_2D

2D pool containing the nodes that form the tree

3.15.2.4 uint32_t _tree::pool_available

Number of empty nodes available in the pool

3.15.2.5 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

3.16 _uint128 Struct Reference

Public Attributes

- uint64 **first**
- uint64 **second**

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

- include/city.h

3.17 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

3.17.1 Detailed Description

stores info needed to create the summary graphs

3.17.2 Member Data Documentation

3.17.2.1 uint64_t* statsinfo::ACGT_pos

A, C, G, T, N per position

3.17.2.2 uint64_t* statsinfo::ACGT_tile

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

3.17.2.3 int* statsinfo::lane_tags

Names of the existing tiles

3.17.2.4 uint64_t* statsinfo::lowQ_ACGT_tile

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

3.17.2.5 int statsinfo::minQ

Minimum quality threshold

3.17.2.6 int statsinfo::nQ

possible quality values

3.17.2.7 int statsinfo::nreads

reads read till current position.

3.17.2.8 int statsinfo::ntiles

tiles

3.17.2.9 uint64_t* statsinfo::QPosTile_table

bases of a given quality per tile.

3.17.2.10 int* statsinfo::qual_tags

Names of the existing qualities

3.17.2.11 int statsinfo::read_len

Maximum length of a read

3.17.2.12 uint64_t* statsinfo::reads_MlowQ

reads with M(position) lowQuality bases.

3.17.2.13 int statsinfo::reads_wN

reads with N's found till current position

3.17.2.14 int statsinfo::sz_ACGT_pos

ACGT_pos size = read_len * N_ACGT

3.17.2.15 int statsinfo::sz_ACGT_tile

ACGT_tile size = ntiles * NACGT

3.17.2.16 int statsinfo::sz_lowQ_ACGT_tile

lowQ_ACGT_tile size = ntiles * N_ACGT

3.17.2.17 int statsinfo::sz_QPosTile_table

QposTile_Table size = ntiles * nQ * read_len

3.17.2.18 int statsinfo::sz_reads_MlowQ

reads_MlowQ size = read_len + 1

3.17.2.19 int statsinfo::tile_pos

current tile position

3.17.2.20 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

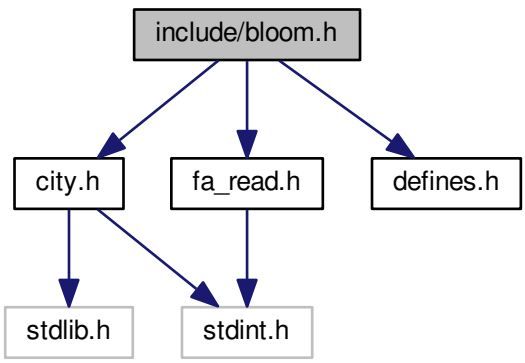
File Documentation

4.1 include/bloom.h File Reference

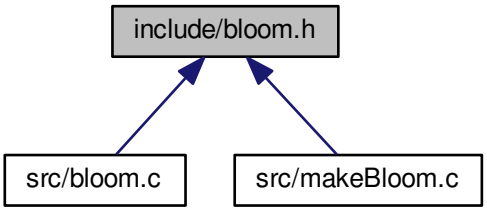
functions that implement the bloom filter

```
#include "city.h"
#include "fa_read.h"
#include "defines.h"
```

Include dependency graph for bloom.h:



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



Classes

- struct _bfilter

  Bloom filter structure.
- struct _procs_kmer

  stores a processed kmer (2 bits pro nucleotide)

Typedefs

- typedef struct _bfilter Bfilter

  Bloom filter structure.
- typedef struct _procs_kmer Procs_kmer

  stores a processed kmer (2 bits pro nucleotide)

Functions

- void init_LUTs ()

  look up table initialization
- Bfilter * init_Bfilter (int kmersize, uint64_t bfsizeBits, int hashNum, double falsePosRate, uint64_t nelem)

  initialization of a Bfilter structure
- void free_Bfilter (Bfilter *ptr_bf)

  free Bfilter memory
- Procs_kmer * init_procs (int kmersize, int hashNum)

  initializes a Procs structure, given the kmersize and the number of hash functions
- void free_procs (Procs_kmer *procs)

  free Procs_kmer
- double score_read_in_filter (unsigned char *read, int L, Procs_kmer *procs, Bfilter *ptr_bf)

  computes a score for a read being in the bloom filter
- Bfilter * create_Bfilter (Fa_data *ptr_fasta, int kmersize, uint64_t bfsizeBits, int hashNum, double falsePos↩ Rate, uint64_t nelem)

  creates a bloom filter from a fasta structure.
- void save_Bfilter (Bfilter *ptr_bf, char *filterfile, char *paramfile)

  saves a bloomfilter to disk
- Bfilter * read_Bfilter (char *filterfile, char *paramfile)

  reads a bloom filter from a file

Variables

- static const unsigned char bitMask [0x08]

  bitMask, ith bit set to 1 in position i

4.1.1 Detailed Description

functions that implement the bloom filter

Author

Paula Perez paulaperezrubio@gmail.com

Date

04.09.2017

4.1.2 Function Documentation

4.1.2.1 Bfilter* create_Bfilter (Fa_data * *ptr_fasta,* int *kmersize,* uint64_t *bfsizeBits,* int *hashNum,* double *falsePosRate,* uint64_t *nelem*)

creates a bloom filter from a fasta structure.

Parameters

ptr_fasta	pointer to fasta structure
kmersize	length of kmers to be inserted in the filter
bfsizeBits	size of Bloom filter in bits
hashNum	number of hash functions to be used
falsePosRate	false positive rate
nelem	number of elemens (kmers in the sequece) contained in the filter

Returns

pointer to Bloom filter structure, where the fasta file was encoded.

4.1.2.2 Bfilter* init_Bfilter (int *kmersize,* uint64_t *bfsizeBits,* int *hashNum,* double *falsePosRate,* uint64_t *nelem*)

initialization of a Bfilter structure

Parameters

kmersize	number of elements of the kmer
bfsizeBits	size of the bloomfilter (in Bits)
hashNum	number of hash functions to be computed
falsePosRate	false positive rate
nelem	number of elemens (kmers in the sequece) contained in the filter

Returns

pointer to initialized Bfilter structure

Given a kmersize, bfsizeBits, number of hash functions, we assign these values to the struture and the two additional values: kmersizeBytes = (kmersize + BASESINCHAR - 1)/BASESINCHAR

4.1.2.3 void init_LUTs ( )

look up table initialization

It initializes: fw0, fw1, fw2, fw3, bw0, bw2, bw3, bw4. They are uint8_t arrays with 256 elements. All elements are set to 0xFF excepting the ones corresponding to 'a', 'A', 'c', 'C', 'g', 'G', 't', 'T':

Var	a,A	c,C	g,G	t,T	Var	a,A	c,C	g,G	t,T
fw0	0x00	0x40	0x80	0xC0	bw0	0xC0	0x80	0x40	0x00
fw1	0x00	0x10	0x20	0x30	bw1	0x30	0x20	0x10	0x00
fw2	0x00	0x04	0x08	0x0C	bw2	0x0C	0x08	0x04	0x00
fw3	0x00	0x01	0x02	0x03	bw3	0x03	0x02	0x01	0x00

With these variables, we will be able to encode a Sequence using 2 bits per nucleotide.

4.1.2.4 Procs_kmer* init_procs (int *kmersize,* int *hashNum*)

initializes a Procs structure, given the kmersize and the number of hash functions

Parameters

kmersize	number of elements of the kmer
hashNum	number of hash functions to be computed

Returns

pointer to a Procs_kmer structure

kmersizeBytes, halfsizeBytes, hangingBases, hasOverhead hashNum are assigned and memory is allocated and set to 0 for compact and hashValues

4.1.2.5 Bfilter* read_Bfilter (char * *filterfile,* char * *paramfile*)

reads a bloom filter from a file

Parameters

inputfile	path to input file

Returns

a pointer to a filter structure containing the bloomfilter

This function reads two files, the auxiliar inputfile where kmersize, hashNum and bfsizeBits are stored, and the actual filter file. If one of them is missing, the program exits with an error. If successful, a pointer to a Bfilter structure with the bloom filter is return

4.1.2.6 void save_Bfilter (Bfilter * *ptr_bf,* char * *filterfile,* char * *paramfile*)

saves a bloomfilter to disk

Parameters

ptr_bf	pointer to Bfilter structure (contains the filter)
filterfile	path to file where the output will be stored

paramfile	path to file where the prameters will be stored

This function will save the bloomfilter in the path filterfile. The paramfile will store the following data:

- kmersize
- hashNum
- bfsizeBits
- falsePosRate
- nelem

4.1.2.7 double score_read_in_filter (unsigned char * *read*, int *L*, Procs_kmer * *procs*, Bfilter * *ptr_bf*)

computes a score for a read being in the bloom filter

Parameters

read	sequence from fastq file
L	sequence length
procs	pointer to Procs structure
ptr_bf	pointer to Bfilter The score is computed by ... DECIDE!!!

4.1.3 Variable Documentation

4.1.3.1 const unsigned char bitMask[0x08] [static]

Initial value:

```
= {0x01, 0x02, 0x04, 0x08,
                                        0x10, 0x20, 0x40, 0x80}
```

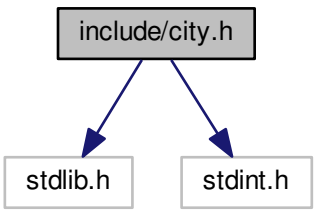
bitMask, ith bit set to 1 in position i

4.2 include/city.h File Reference

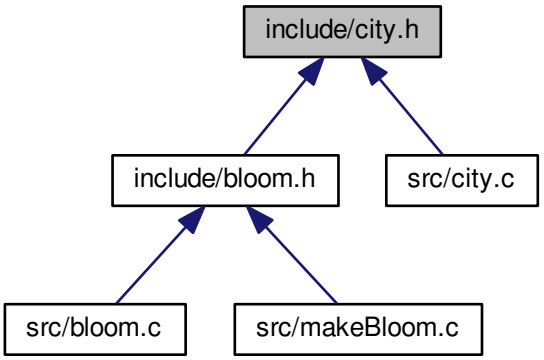
functions for hashin strings, C translation of cityhash (C++, google)

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

Include dependency graph for city.h:



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



Classes

- struct _uint128

Macros

- #define **Uint128Low64**(x) (x).first
- #define **Uint128High64**(x) (x).second

Typedefs

- typedef uint8_t **uint8**
- typedef uint32_t **uint32**
- typedef uint64_t **uint64**
- typedef struct _uint128 **uint128**

Functions

- uint64 **CityHash64** (const char ∗buf, size_t len)
- uint64 **CityHash64WithSeed** (const char ∗buf, size_t len, uint64 seed)
- uint64 **CityHash64WithSeeds** (const char ∗buf, size_t len, uint64 seed0, uint64 seed1)
- uint128 **CityHash128** (const char ∗s, size_t len)
- uint128 **CityHash128WithSeed** (const char ∗s, size_t len, uint128 seed)

4.2.1 Detailed Description

functions for hashin strings, C translation of cityhash (C++, google)

Author

Alexander Nusov

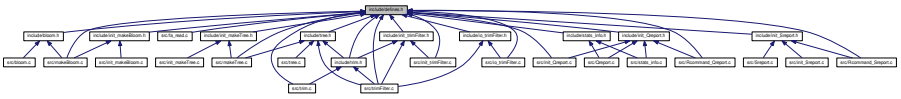
See also

https://github.com/nusov/cityhash-c
https://github.com/google/cityhash

4.3 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 BITSPERCHAR 8
- #define BASESPERCHAR 4
- #define KMER_LEN 25
- #define FALSE_POS_RATE 0.05
- #define ZERO_POS_RATE 1e-14
- #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

4.3.1 Detailed Description

Macro definitions.

Author

Paula Perez paulaperezrubio@gmail.com

Date

07.08.2017

4.3.2 Macro Definition Documentation

4.3.2.1 #define ADAP 0

Adapter filter

4.3.2.2 #define ALL 1

Trims if a lowQ base calling | N is found

4.3.2.3 #define B_LEN 131072

buffer size

4.3.2.4 #define BASESPERCHAR 4

number of nucleotides that can fit in a char

4.3.2.5 #define BITSPERCHAR 8

number of bits in a char

4.3.2.6 #define BLOOM 3

Use a bloom filter to look for contaminations

4.3.2.7 #define bool short

define a bool type

4.3.2.8 #define CONT 1

Contamination filter

4.3.2.9 #define DEFAULT_MINL 25

Default minimum length under which we discard the reads

4.3.2.10 #define DEFAULT_MINQ 27

Minimum quality threshold

4.3.2.11 #define DEFAULT_NQ 46

Default number of different quality values

4.3.2.12 #define DEFAULT_NTILES 96

Default number of tiles

4.3.2.13 #define ENDS 2

Trims at the ends

4.3.2.14 #define ENDSFRAC 4

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

4.3.2.15 #define ERROR 1000

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

4.3.2.16 #define FA_ENTRY_BUF 20

buffer for fasta entries

4.3.2.17 #define false 0

assign false to 0

4.3.2.18 #define FALSE_POS_RATE 0.05

default false positive rate

4.3.2.19 #define FRAC 3

Discards a read if it contains $>$ percent lowQ bases

4.3.2.20 #define GLOBAL 5

Trims a fixed # bases from e left and right

4.3.2.21 #define GOOD 4

Good reads

4.3.2.22 #define KMER_LEN 25

default kmer length

4.3.2.23 #define LOWQ 2

Low quality filter

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

max function

4.3.2.25 #define MAX_FASZ_TREE 1e7

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

4.3.2.26 #define MAX_FILENAME 300

Maximum # chars in a filename

4.3.2.27 #define MAX_RCOMMAND 4000

Maximum # chars in R command

4.3.2.28 #define mem_usage( )

Value:

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

returns allocated memory in Bytes

4.3.2.29 #define mem_usageMB( )

Value:

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

returns allocated memory in MB

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

min function

4.3.2.31 #define N_ACGT 5

Number of different nucleotides in the fq file

4.3.2.32 #define NFILTERS 4

total number of filters

4.3.2.33 #define NNNN 3

N's presence filter

4.3.2.34 #define NO 0

No trimming

4.3.2.35 #define NPOOL_1D 1048576

Number of Node structs allocated in inner dim

4.3.2.36 #define NPOOL_2D 16

Number of ∗Node allocated in outer dim

4.3.2.37 #define SA 2

Use a suffix array to look for contaminations

4.3.2.38 #define STRIP 3

Looks for the largest N-free sequence

4.3.2.39 #define T_ACGT 4

Number of children per node in tree

4.3.2.40 #define TREE 1

Use a tree to look for contaminations

4.3.2.41 #define true 1

assign true to 1

4.3.2.42 #define ZERO_POS_RATE 1e-14

0 threshold for a double

4.3.2.43 #define ZEROQ 33

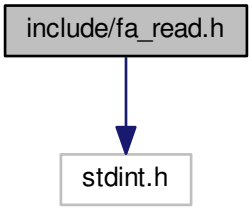
ASCII code of lowest quality value (!)

4.4 include/fa_read.h File Reference

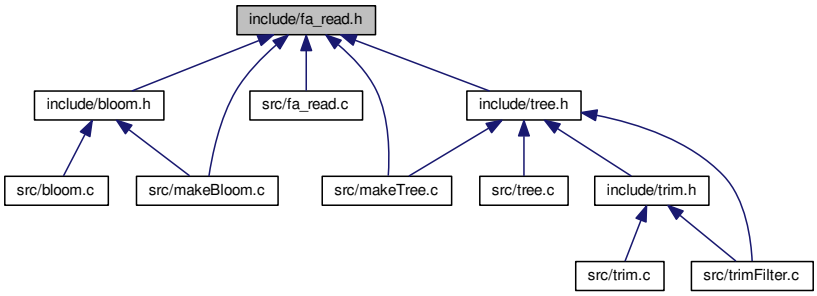
reads in and stores fasta files

```
#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
- uint64_t nkmers (Fa_data *ptr_fa, int kmersize)
  number of kmers of length kmersize contained in a fasta structure
- void free_fasta (Fa_data *ptr_fa)
  free fasta file

4.4.1 Detailed Description

reads in and stores fasta files

Author

Paula Perez paulaperezrubio@gmail.com

Date

16.08.2017

4.4.2 Function Documentation

4.4.2.1 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

4.4.2.2 uint64_t nkmers (Fa_data * *ptr_fa,* int *kmersize*)

number of kmers of length kmersize contained in a fasta structure

Returns

number of kmers of length kmersize contained in a fasta structure

4.4.2.3 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.

4.4.2.4 uint64_t size_fasta (Fa_data * *ptr_fa*)

computes length of genome in fasta structure

Parameters

ptr_fa	pointer to Fa_data

Returns

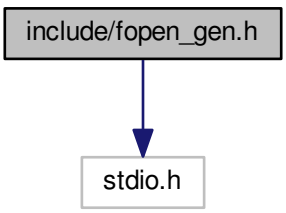
total number of nucleotides

4.5 include/fopen_gen.h File Reference

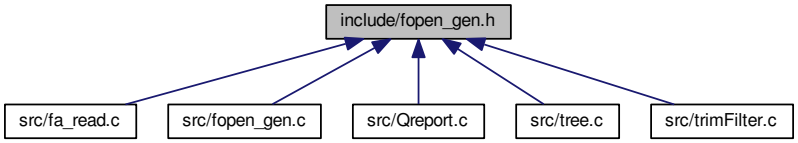
Uncompress/compress input/output files using pipes.

```
#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.

4.5.1 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

4.5.2 Function Documentation

4.5.2.1 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

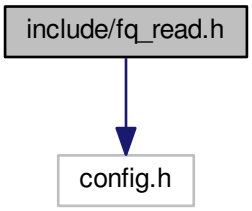
a FILE pointer

4.6 include/fq_read.h File Reference

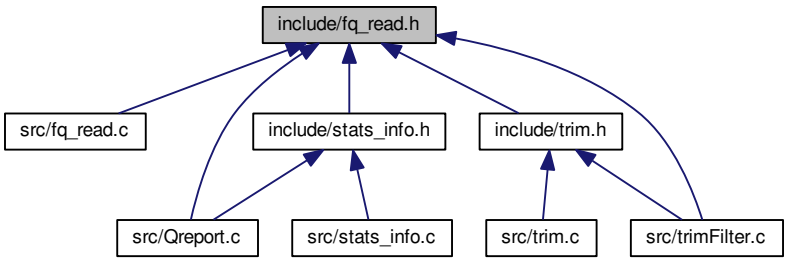
fastq entries manipulations (read/write)

```
#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

4.6.1 Detailed Description

fastq entries manipulations (read/write)

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.6.2 Function Documentation

4.6.2.1 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

*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.
read_len	predefined read length
filter	0 original file, 1 file filtered with filter_trim, 2 file filtered with another tool

4.6.2.2 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

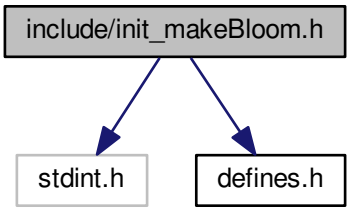
change the call to sprintf to snprintf

4.7 include/init_makeBloom.h File Reference

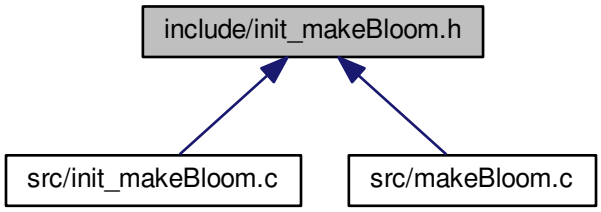
Help dialog for makeBloom and initialization of the command line arguments.

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

Include dependency graph for init_makeBloom.h:



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



Classes

- struct _iparam_makeBloom

  contains makeBloom input parameters

Typedefs

- typedef struct _iparam_makeBloom lparam_makeBloom

  contains makeBloom input parameters

Functions

- void printHelpDialog_makeBloom ()

  Function that prints makeBloom help dialog when called.
- void getarg_makeBloom (int argc, char **argv)

Reads in the arguments passed through the command line to makeBloom. and stores them in the global variable par_MB.

4.7.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

05.09.2017

4.7.2 Typedef Documentation

4.7.2.1 typedef struct _iparam_makeBloom Iparam_makeBloom

contains makeBloom input parameters

Note

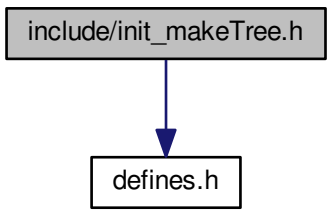
nelemen will be computed once the fasta file is read and loaded.

4.8 include/init_makeTree.h File Reference

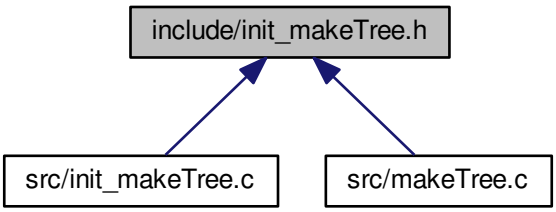
Help dialog for makeTree and initialization of the command line arguments.

```
#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.

4.8.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

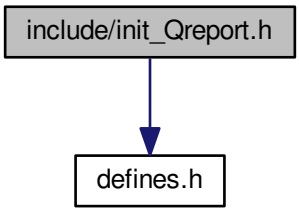
23.08.2017

4.9 include/init_Qreport.h File Reference

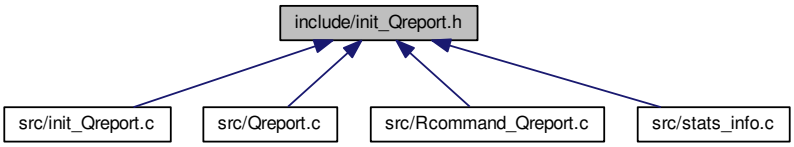
Header file: help dialog for Qreport and initialization of the command line arguments.

```
#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.

4.9.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

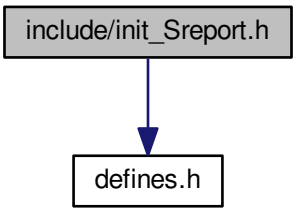
03.08.2017

4.10 include/init_Sreport.h File Reference

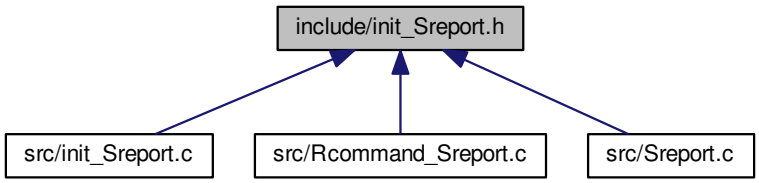
Help dialog for Sreport and initialization of the command line arguments.

```
#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.

4.10.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

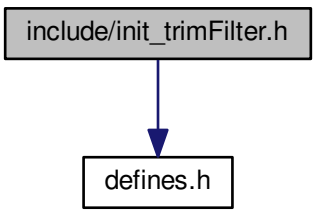
09.08.2017

4.11 include/init_trimFilter.h File Reference

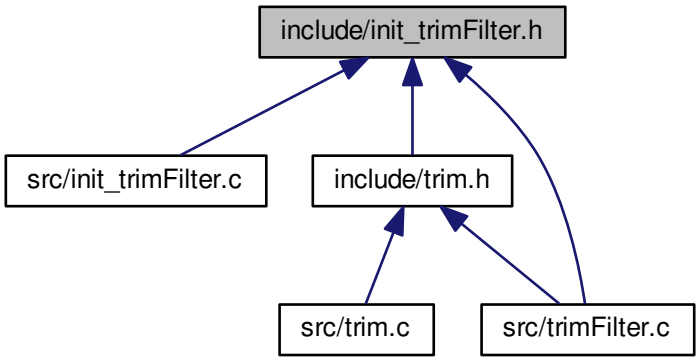
help dialog for trimFilter and initialization of the command line arguments.

```
#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.

4.11.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

24.08.2017

4.11.2 Typedef Documentation

4.11.2.1 typedef struct _adapter Adapter

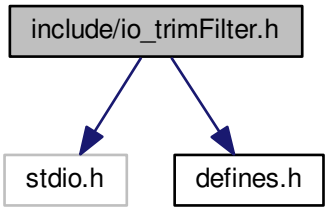
@ brief adapter struct @ note UNFINISHED!

4.12 include/io_trimFilter.h File Reference

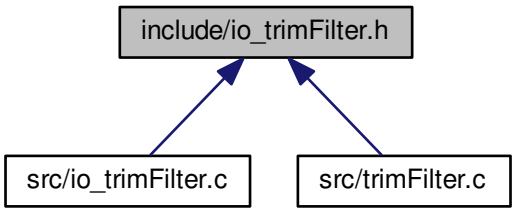
buffer fq output, write summary file

```
#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)

4.12.1 Detailed Description

buffer fq output, write summary file

Author

Paula Perez paulaperezrubio@gmail.com

Date

29.08.2017

4.12.2 Function Documentation

4.12.2.1 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

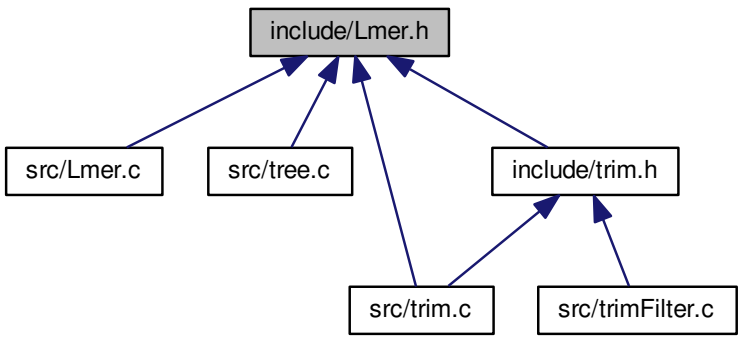
Parameters

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

4.13 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'}.

4.13.1 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.

4.13.2 Function Documentation

4.13.2.1 void init_map ( )

Initialize lookup table LT.

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

4.13.2.2 void init_map_SA ( )

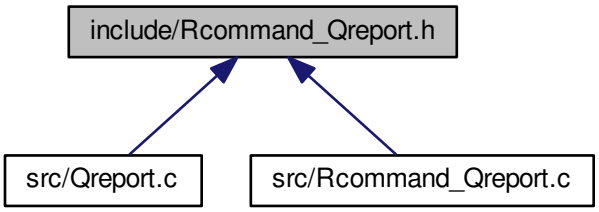
Initialize lookup table LT (for SA)

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

4.14 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

4.14.1 Detailed Description

get Rscript command for Qreport

Author

Paula Perez paulaperezrubio@gmail.com

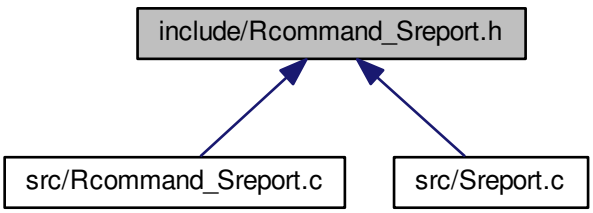
Date

09.08.2017

4.15 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

4.15.1 Detailed Description

get Rscript command for Sreport

Author

Paula Perez paulaperezrubio@gmail.com

Date

09.08.2017

4.15.2 Function Documentation

4.15.2.1 char* command_Sreport ( )

returns Rscript command that generates the summary report in html

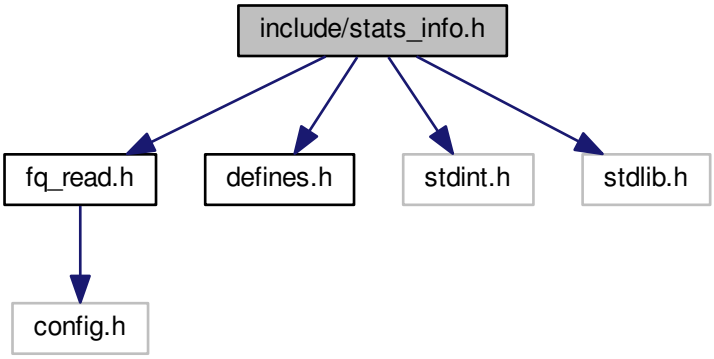
```
1 # To run between quotation marks after:  Rscript_RBioC -e (Rscript)
2 inputfolder = normalizePath( <par.SR.inputfolder>, mustWork = TRUE);
3 output = <par_SR.outputfile>;
4 output_file = gsub('.* /', '', output);
5 path = gsub('[^/]+$', '', output);
6 if (path != '') {
7   outputfile = paste0(normalizePath(path, mustWork = TRUE), '/', outputfile);
8 } else {
9   outputfile = paste0(cwd, '/', output_file); # cwd: current working dir
10 };
11 rmarkdown::render(<par_SR.Rmd_file>,
12                   params = list(inputfolder = inputfolder, version= VERSION),
13                   output_file = output_file)
```

4.16 include/stats_info.h File Reference

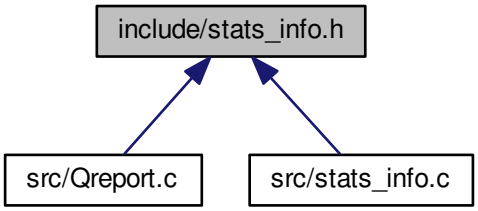
Construct the quality report variables and update them.

```
#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

4.16.1 Detailed Description

Construct the quality report variables and update them.

Author

Paula Perez paulaperezrubio@gmail.com

Date

04.08.2017

4.16.2 Function Documentation

4.16.2.1 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).

4.16.2.2 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.

4.16.2.3 int update_ACGT_counts (uint64_t * *ACGT_low,* char *ACGT*)

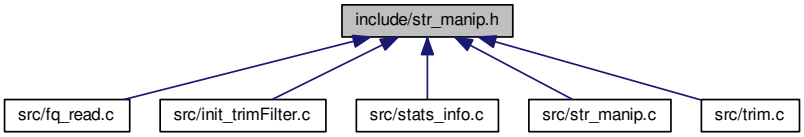
update, for current tile, ACGT counts.

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

4.17 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 or splitted fields

Typedefs

- typedef struct _split Split

  contains a splitted string and the number or splitted fields

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
- Split strsplit (char ∗str, char sep)

  Separates strings by a separator.

4.17.1 Detailed Description

functions that do string manipulation

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.17.2 Function Documentation

4.17.2.1 int strindex (char ∗ *s*, char ∗ *t*)

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

Parameters

s	string to be checked.
t	substring to be found in s.

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

Separates strings by a separator.

Parameters

str	input string
sep	separator (char)

Returns

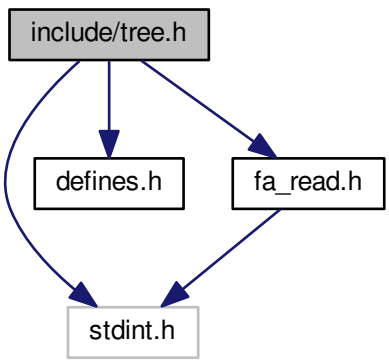
array of strings containing the substrings in the input separated

4.18 include/tree.h File Reference

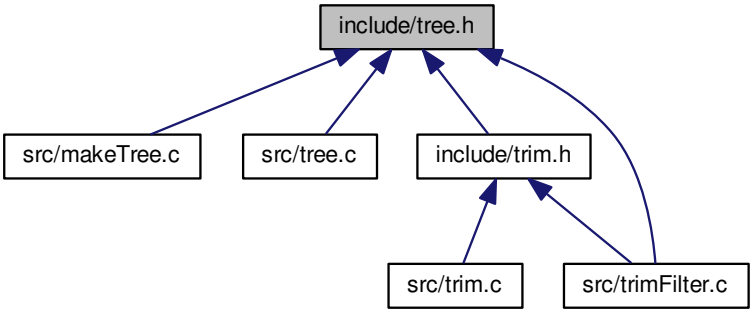
Construction of tree, check paths, write tree, read in tree.

```
#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

4.18.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

18.08.2017

4.18.2 Typedef Documentation

4.18.2.1 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!).

4.18.3 Function Documentation

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

checks if read is found in tree and outputs a score

Parameters

tree_ptr	pointer to Tree structure
read	Read or reverse complement
Lread	length of read

Returns

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

4.18.3.2 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.

4.18.3.3 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

4.18.3.4 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.

4.18.3.5 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.

4.18.3.6 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.

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

create Tree structure from fasta structure.

Parameters

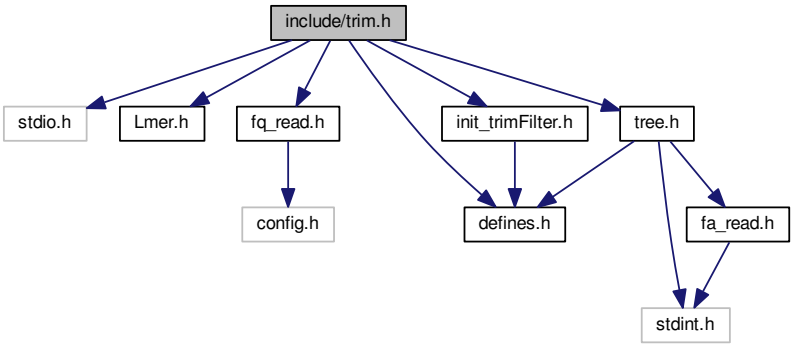
fasta	pointer to fasta structure
L	tree length

4.19 include/trim.h File Reference

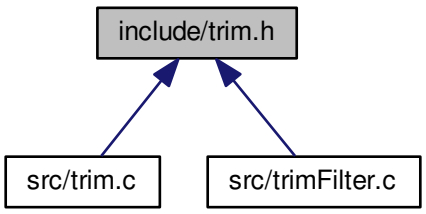
trims/filter sequences after Quality, N's contaminations.

```
#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 ot them exceeds the user selected threshold, it returns true. Otherwise, it returns false.

4.19.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

24.08.2017

4.19.2 Function Documentation

4.19.2.1 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 ot them exceeds the user selected threshold, it returns true. Otherwise, it returns false.

Parameters

tree_ptr	pointer to Tree structure
seq	fastq read

Returns

true if read was found, false otherwise

4.19.2.2 int trim_sequenceN (Fq_read * *seq*)

trims sequence based on presence of N nucleotides

Parameters

seq	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).

4.19.2.3 int trim_sequenceQ (Fq_read * *seq*)

trims sequence based on lowQ base callings

Parameters

seq	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:

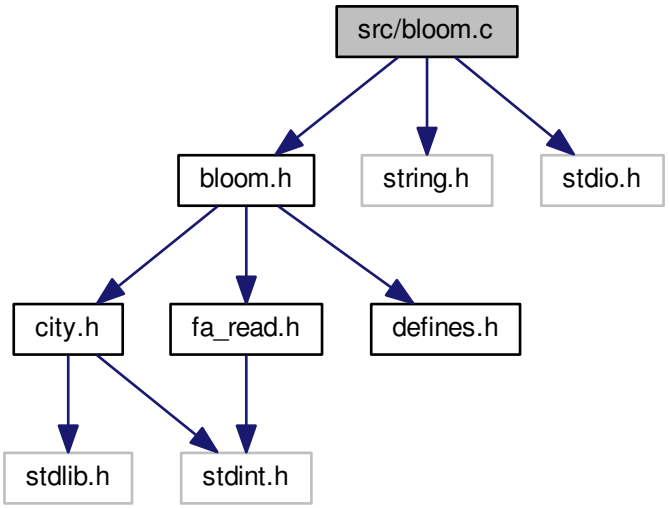
- NO(0): accepts is as is , (1),
- FRAC(1): accepts it if less than par_TF.nlowQ are found (1), rejects it otherwise (0),
- ENDS(2): trims the ends and accepts it if it is longer than minL (2 if triming, 1 if no trimming), rejects it otherwise (0),
- ENDSFRAC(3): trims the ends and accepts if the remaining 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).

4.20 src/bloom.c File Reference

functions that implement the bloom filter

```
#include "bloom.h"
#include <string.h>
#include <stdio.h>
```

Include dependency graph for bloom.c:



Functions

- void init_LUTs ()

look up table initialization

- Bfilter * init_Bfilter (int kmersize, uint64_t bfsizeBits, int hashNum, double falsePosRate, uint64_t nelem)

  initialization of a Bfilter structure
- void free_Bfilter (Bfilter *ptr_bf)

  free Bfilter memory
- Procs_kmer * init_procs (int kmersize, int hashNum)

  initializes a Procs structure, given the kmersize and the number of hash functions
- void free_procs (Procs_kmer *procs)

  free Procs_kmer
- static int compact_kmer (const unsigned char *sequence, uint64_t position, Procs_kmer *procs)

  compactifies a kmer for insertion in the bloomfilter
- static void multiHash (Procs_kmer *procs)

  obtains the hashNum hashvalues for a compactified kmer
- static bool insert_and_fetch (Bfilter *ptr_bf, Procs_kmer *procs)

  inserts the hashvalues of a kmer in filter
- static bool contains (Bfilter *ptr_bf, Procs_kmer *procs)

  check if kmer is contained in the filter
- double score_read_in_filter (unsigned char *read, int L, Procs_kmer *procs, Bfilter *ptr_bf)

  computes a score for a read being in the bloom filter
- Bfilter * create_Bfilter (Fa_data *ptr_fasta, int kmersize, uint64_t bfsizeBits, int hashNum, double falsePos↩ Rate, uint64_t nelem)

  creates a bloom filter from a fasta structure.
- void save_Bfilter (Bfilter *ptr_bf, char *filterfile, char *paramfile)

  saves a bloomfilter to disk
- Bfilter * read_Bfilter (char *filterfile, char *paramfile)

  reads a bloom filter from a file

Variables

- static uint8_t fw0 [256]

  Global variables (lookup table) Used to compactify kmers.
- static uint8_t **fw1** [256]
- static uint8_t **fw2** [256]
- static uint8_t **fw3** [256]
- static uint8_t **bw0** [256]
- static uint8_t **bw1** [256]
- static uint8_t **bw2** [256]
- static uint8_t **bw3** [256]
- uint64_t alloc_mem

4.20.1 Detailed Description

functions that implement the bloom filter

Author

Paula Perez paulaperezrubio@gmail.com

Date

04.09.2017

4.20.2 Function Documentation

4.20.2.1 static int compact_kmer (const unsigned char * *sequence,* uint64_t *position,* Procs_kmer * *procs*) [static]

compactifies a kmer for insertion in the bloomfilter

Parameters

sequence	unsigned char DNA sequence (or cDNA)
position	position in the sequence where the kmer starts
procs	initialized Procs_kmer

The compactified sequence is computed in the following way:

- We start compactifying both, the forward and backward (reverse complement). The outer loop covers up until half of the sequence.
- As soon as one of the two is lexicographically smaller, we continue only with it. In that way, the "smaller" sequence is consistently returned.
- If the sequence is palindromic, we continue with the forward sequence.
- kmersize should be > 3.

  We illustrate the compactification with an example:

```
kmer = TTTT|GGAT
m_fw = 00000000 | 00000000 // 2 bytes
m_bw = 00000000 | 00000000 // 2 bytes
m_fw[0] |= fw0['T'] = 0xC0|0x00;    m_bw[0] |= bw0['T'] = 0x00|0x00;
m_fw[0] |= fw1['T'] = 0xF0|0x00;    m_bw[0] |= bw1['A'] = 0x30|0x00;
m_fw[0] |= fw2['T'] = 0xFC|0x00;    m_bw[0] |= bw2['G'] = 0x34|0x00;
m_fw[0] |= fw3['T'] = 0xFF|0x00;    m_bw[0] |= bw3['G'] = 0x35|0x00;
m_fw[1] |= fw0['G'] = 0xC0|0x80;    m_bw[1] |= bw0['T'] = 0x35|0x00;
m_fw[1] |= fw1['G'] = 0xF0|0xA0;    m_bw[1] |= bw1['T'] = 0x35|0x00;
m_fw[1] |= fw2['A'] = 0xFC|0xA0;    m_bw[1] |= bw2['T'] = 0x35|0x00;
m_fw[1] |= fw3['T'] = 0xFF|0xA3;    m_bw[1] |= bw3['T'] = 0x35|0x00;
```

(In this case, we would store m_bw)

4.20.2.2 static bool contains (Bfilter * *ptr_bf*, Procs_kmer * *procs*) [static]

check if kmer is contained in the filter

Parameters

ptr_bf	pointer to a Bfilter structure, where a bloomfilter is stored
procs	pointer to a Procs_kmer structure containing the hash values

Returns

true if all corresponding bits were set to 1 in the filter

4.20.2.3 Bfilter* create_Bfilter (Fa_data * *ptr_fasta*, int *kmersize*, uint64_t *bfsizeBits*, int *hashNum*, double *falsePosRate*, uint64_t *nelem*)

creates a bloom filter from a fasta structure.

Parameters

ptr_fasta	pointer to fasta structure
kmersize	length of kmers to be inserted in the filter
bfsizeBits	size of Bloom filter in bits
hashNum	number of hash functions to be used

falsePosRate	false positive rate
nelem	number of elemens (kmers in the sequece) contained in the filter

Returns

pointer to Bloom filter structure, where the fasta file was encoded.

4.20.2.4 Bfilter* init_Bfilter (int *kmersize,* uint64_t *bfsizeBits,* int *hashNum,* double *falsePosRate,* uint64_t *nelem*)

initialization of a Bfilter structure

Parameters

kmersize	number of elements of the kmer
bfsizeBits	size of the bloomfilter (in Bits)
hashNum	number of hash functions to be computed
falsePosRate	false positive rate
nelem	number of elemens (kmers in the sequece) contained in the filter

Returns

pointer to initialized Bfilter structure

Given a kmersize, bfsizeBits, number of hash functions, we assign these values to the struture and the two additional values: kmersizeBytes = (kmersize + BASESINCHAR - 1)/BASESINCHAR

4.20.2.5 void init_LUTs ( )

look up table initialization

It initializes: fw0, fw1, fw2, fw3, bw0, bw2, bw3, bw4. They are uint8_t arrays with 256 elements. All elements are set to 0xFF excepting the ones corresponding to 'a', 'A', 'c', 'C', 'g', 'G', 't', 'T':

Var	a,A	c,C	g,G	t,T	Var	a,A	c,C	g,G	t,T
fw0	0x00	0x40	0x80	0xC0	bw0	0xC0	0x80	0x40	0x00
fw1	0x00	0x10	0x20	0x30	bw1	0x30	0x20	0x10	0x00
fw2	0x00	0x04	0x08	0x0C	bw2	0x0C	0x08	0x04	0x00
fw3	0x00	0x01	0x02	0x03	bw3	0x03	0x02	0x01	0x00

With these variables, we will be able to encode a Sequence using 2 bits per nucleotide.

4.20.2.6 Procs_kmer* init_procs (int *kmersize,* int *hashNum*)

initializes a Procs structure, given the kmersize and the number of hash functions

Parameters

kmersize	number of elements of the kmer
hashNum	number of hash functions to be computed

Returns

pointer to a Procs_kmer structure

kmersizeBytes, halfsizeBytes, hangingBases, hasOverhead hashNum are assigned and memory is allocated and set to 0 for compact and hashValues

4.20.2.7 static bool insert_and_fetch (Bfilter * *ptr_bf*, Procs_kmer * *procs*) [static]

inserts the hashvalues of a kmer in filter

Parameters

ptr_bf	pointer to Bfilter structure, where we will include the new entry
procs	pointer to Procs_kmer structure, where the hashvalues are stored

Returns

true if the positions of the hash values were already set to one previously.

The hash values are inserted in the following way.

- modValue = hashvalue mod(filter size) is calculated.
- the bit in position modValue of the filter is set to 1.

4.20.2.8 static void multiHash (Procs_kmer * *procs*) [static]

obtains the hashNum hashvalues for a compactified kmer

The hash values are computed using the CityHash64 hash functions.

4.20.2.9 Bfilter* read_Bfilter (char * *filterfile,* char * *paramfile*)

reads a bloom filter from a file

Parameters

inputfile	path to input file

Returns

a pointer to a filter structure containing the bloomfilter

This function reads two files, the auxiliar inputfile where kmersize, hashNum and bfsizeBits are stored, and the actual filter file. If one of them is missing, the program exits with an error. If successful, a pointer to a Bfilter structure with the bloom filter is return

4.20.2.10 void save_Bfilter (Bfilter * *ptr_bf,* char * *filterfile,* char * *paramfile*)

saves a bloomfilter to disk

Parameters

ptr_bf	pointer to Bfilter structure (contains the filter)
filterfile	path to file where the output will be stored
paramfile	path to file where the prameters will be stored

This function will save the bloomfilter in the path filterfile. The paramfile will store the following data:

- kmersize
- hashNum
- bfsizeBits
- falsePosRate
- nelem

4.20.2.11 double score_read_in_filter (unsigned char * *read,* int *L,* Procs_kmer * *procs,* Bfilter * *ptr_bf*)

computes a score for a read being in the bloom filter

Parameters

read	sequence from fastq file
L	sequence length
procs	pointer to Procs structure
ptr_bf	pointer to Bfilter The score is computed by ... DECIDE!!!

4.20.3 Variable Documentation

4.20.3.1 uint64_t alloc_mem

allocated memory

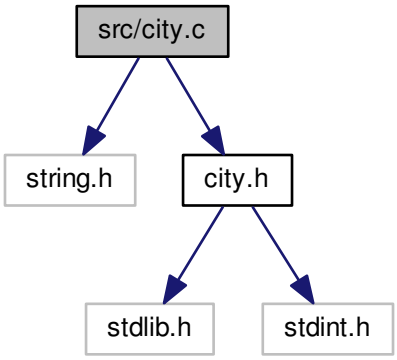
global variable. Memory allocated in the heap.

4.21 src/city.c File Reference

functions for hashin strings, C translation of cityhash (C++, google)

```
#include <string.h>
#include "city.h"
```

Include dependency graph for city.c:



Macros

- #define **uint32_in_expected_order**(x) (x)
- #define **uint64_in_expected_order**(x) (x)
- #define **LIKELY**(x) (x)

Functions

- static uint64 **UNALIGNED_LOAD64** (const char *p)
- static uint32 **UNALIGNED_LOAD32** (const char *p)
- static uint64 **Fetch64** (const char *p)
- static uint32 **Fetch32** (const char *p)

- static uint64 **Hash128to64** (const uint128 x)
- static uint64 **Rotate** (uint64 val, int shift)
- static uint64 **RotateByAtLeast1** (uint64 val, int shift)
- static uint64 **ShiftMix** (uint64 val)
- static uint64 **HashLen16** (uint64 u, uint64 v)
- static uint64 **HashLen0to16** (const char *s, size_t len)
- static uint64 **HashLen17to32** (const char *s, size_t len)
- uint128 **WeakHashLen32WithSeeds6** (uint64 w, uint64 x, uint64 y, uint64 z, uint64 a, uint64 b)
- uint128 **WeakHashLen32WithSeeds** (const char *s, uint64 a, uint64 b)
- static uint64 **HashLen33to64** (const char *s, size_t len)
- uint64 **CityHash64** (const char *s, size_t len)
- uint64 **CityHash64WithSeed** (const char *s, size_t len, uint64 seed)
- uint64 **CityHash64WithSeeds** (const char *s, size_t len, uint64 seed0, uint64 seed1)
- static uint128 **CityMurmur** (const char *s, size_t len, uint128 seed)
- uint128 **CityHash128WithSeed** (const char *s, size_t len, uint128 seed)
- uint128 **CityHash128** (const char *s, size_t len)

Variables

- static const uint64 **k0** = 0xc3a5c85c97cb3127ULL
- static const uint64 **k1** = 0xb492b66fbe98f273ULL
- static const uint64 **k2** = 0x9ae16a3b2f90404fULL
- static const uint64 **k3** = 0xc949d7c7509e6557ULL

4.21.1 Detailed Description

functions for hashin strings, C translation of cityhash (C++, google)

Author

Alexander Nusov

See also

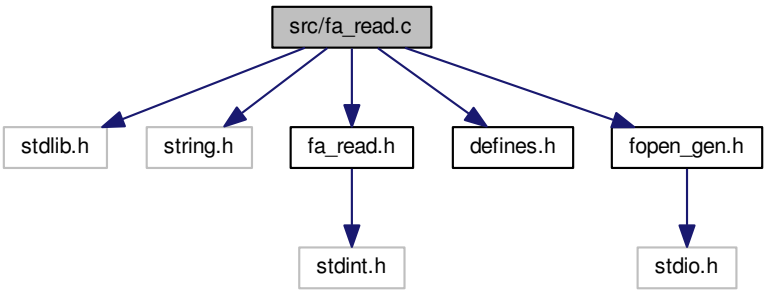
https://github.com/nusov/cityhash-c
https://github.com/google/cityhash

4.22 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
- uint64_t nkmers (Fa_data *ptr_fa, int kmersize)
  number of kmers of length kmersize contained in a fasta structure
- void free_fasta (Fa_data *ptr_fa)
  free fasta file

Variables

- uint64_t alloc_mem

4.22.1 Detailed Description

reads in and stores fasta files

Author

Paula Perez paulaperezrubio@gmail.com

Date

18.08.2017

4.22.2 Function Documentation

4.22.2.1 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

4.22.2.2 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.

4.22.2.3 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).

4.22.2.4 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).

4.22.2.5 uint64_t nkmers (Fa_data * *ptr_fa,* int *kmersize*)

number of kmers of length kmersize contained in a fasta structure

Returns

number of kmers of length kmersize contained in a fasta structure

4.22.2.6 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.

4.22.2.7 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.

4.22.2.8 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

4.22.2.9 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.

4.22.3 Variable Documentation

4.22.3.1 uint64_t alloc_mem

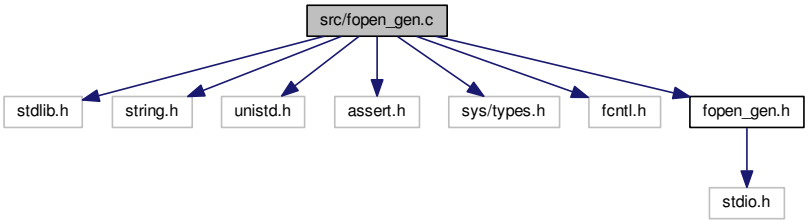
global variable. Memory allocated in the heap.

4.23 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.

4.23.1 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

4.23.2 Function Documentation

4.23.2.1 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

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

Open a pipe to compress the specified file.

Returns

a FILE pointer

4.23.2.3 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

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

Open a pipe to uncompress the specified file.

Returns

a FILE pointer

4.23.2.5 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

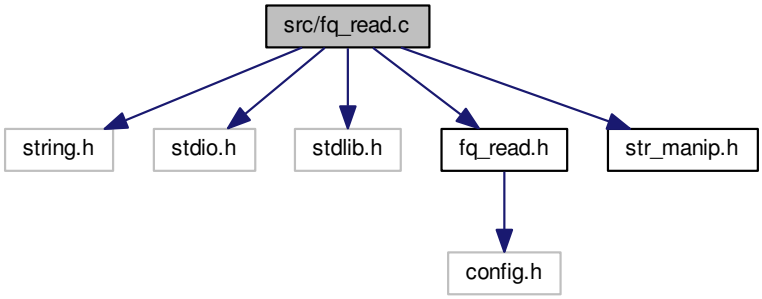
a file descriptor

4.24 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

4.24.1 Detailed Description

fastq entries manipulations (read/write)

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.24.2 Function Documentation

4.24.2.1 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

*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.
read_len	predefined read length
filter	0 original file, 1 file filtered with filter_trim, 2 file filtered with another tool

4.24.2.2 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

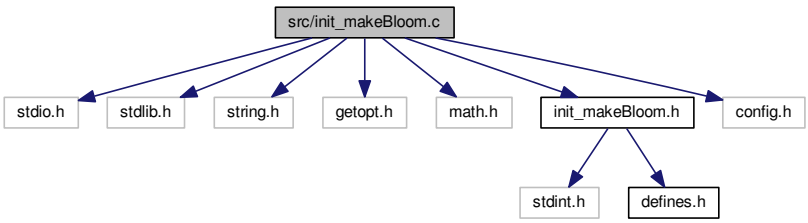
change the call to sprintf to snprintf

4.25 src/init_makeBloom.c File Reference

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

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

Include dependency graph for init_makeBloom.c:



Functions

- void printHelpDialog_makeBloom ()

  Function that prints makeBloom help dialog when called.
- void getarg_makeBloom (int argc, char **argv)

  Reads in the arguments passed through the command line to makeBloom. and stores them in the global variable par_MB.

Variables

- lparam_makeBloom par_MB

4.25.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

05.09.2017

4.25.2 Variable Documentation

4.25.2.1 lparam_makeBloom par_MB

Input parameters of makeBloom

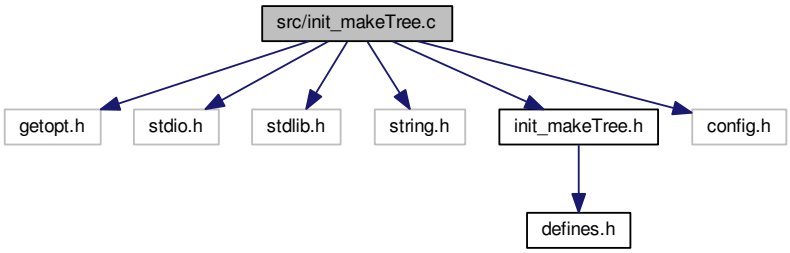
global variable: Input parameters of makeTree.

4.26 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

4.26.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

23.08.2017

4.26.2 Variable Documentation

4.26.2.1 lparam_makeTree par_MT

Input parameters of makeTree

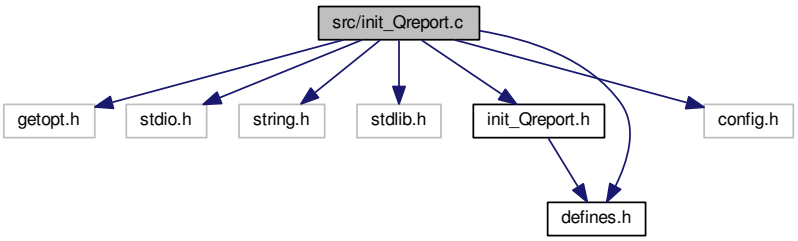
global variable: Input parameters of makeTree.

4.27 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

4.27.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.27.2 Variable Documentation

4.27.2.1 Iparam_Qreport par_QR

Input parameters of Qreport

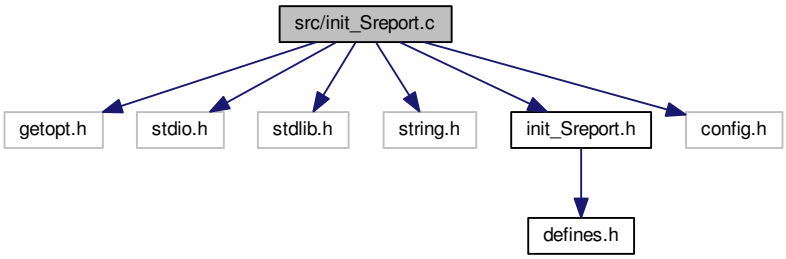
global variable: input parameters for Qreport

4.28 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

4.28.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

09.08.2017

4.28.2 Variable Documentation

4.28.2.1 lparam_Sreport par_SR

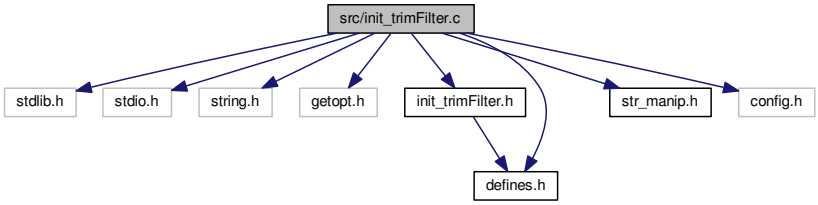
input parameters Sreport

4.29 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

4.29.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

24.08.2017

4.29.2 Variable Documentation

4.29.2.1 lparam_trimFilter par_TF

Input parameters of makeTree

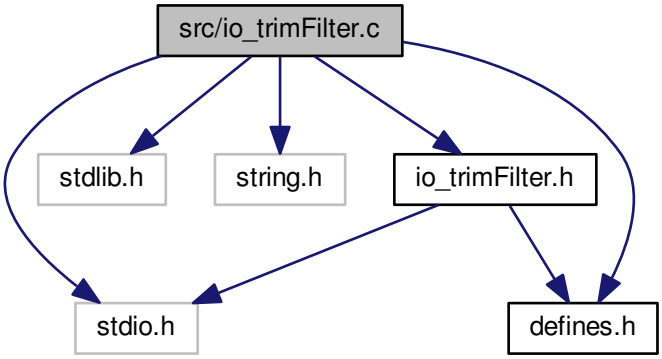
global variable: Input parameters of makeTree.

4.30 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)

4.30.1 Detailed Description

buffer fq output, write summary file

Author

Paula Perez paulaperezrubio@gmail.com

Date

29.08.2017

4.30.2 Function Documentation

4.30.2.1 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

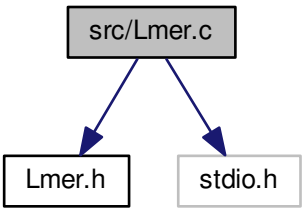
fout	FILE pointer where we might write to disk;
str	string we want to add
len	length of the string we want to add
fd_i	identifier: GOOD, ADAP, CONT, LOWQ, NNNN

4.31 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

4.31.1 Detailed Description

Manipulation of Lmers and sequences.

Author

Paula Perez paulaperezrubio@gmail.com

Date

18.08.2017

4.31.2 Function Documentation

4.31.2.1 void init_map ( )

Initialize lookup table LT.

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

4.31.2.2 void init_map_SA ( )

Initialize lookup table LT (for SA)

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

4.31.3 Variable Documentation

4.31.3.1 char LT[256]

global variable. Lookup table.

4.31.3.2 char Nencode

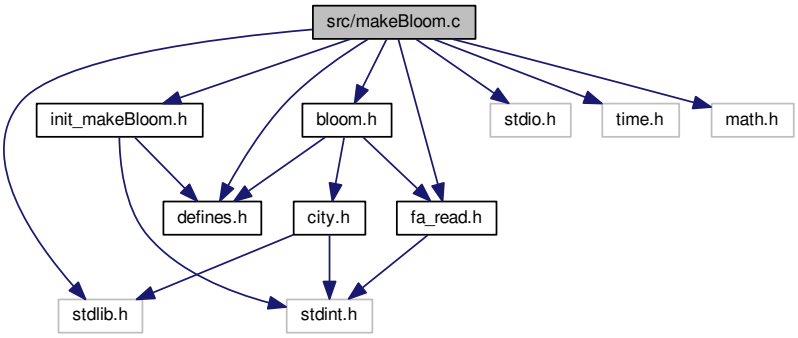
global variable. Encoding for N's(\004, or \005)

4.32 src/makeBloom.c File Reference

makeBloom main function

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <math.h>
#include "defines.h"
#include "fa_read.h"
#include "bloom.h"
#include "init_makeBloom.h"
```

Include dependency graph for makeBloom.c:



Functions

- int main (int argc, char ∗argv[])

  makeTree main function

Variables

- uint64_t alloc_mem = 0
- lparam_makeBloom par_MB

4.32.1 Detailed Description

makeBloom main function

Author

Paula Perez paulaperezrubio@gmail.com

Date

05.09.2017 This file contains the makeBloom main function. It reads a fasta file, constructs a bloom filter and stores it in a file. See README_makeTree.md for more details.

4.32.2 Variable Documentation

4.32.2.1 uint64_t alloc_mem = 0

global variable. Memory allocated in the heap.

4.32.2.2 lparam_makeBloom par_MB

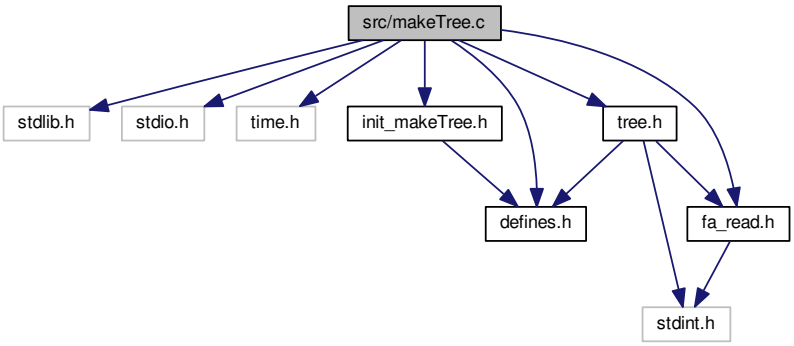
global variable: Input parameters of makeTree.

4.33 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

4.33.1 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.

4.33.2 Variable Documentation

4.33.2.1 uint64_t alloc_mem = 0

global variable. Memory allocated in the heap.

4.33.2.2 lparam_makeTree par_MT

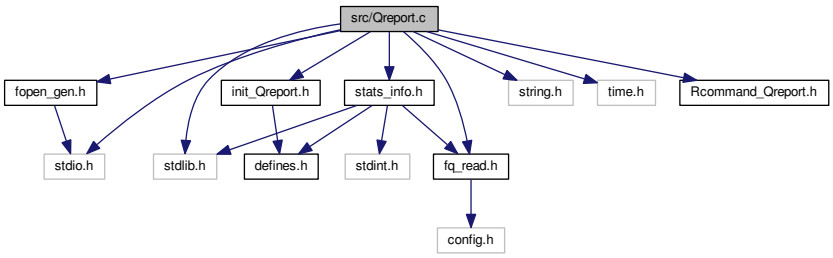
global variable: Input parameters of makeTree.

4.34 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

4.34.1 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.

4.34.2 Variable Documentation

4.34.2.1 lparam_Qreport par_QR

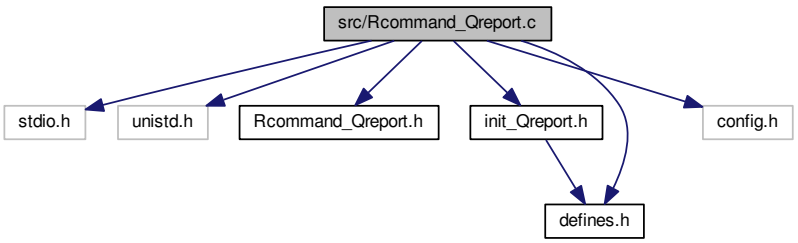
global variable: input parameters for Qreport

4.35 src/Rcommand_Qreport.c File Reference

get Rscript command for Qreport

```
#include <stdio.h>
#include <unistd.h>
#include "Rcommand_Qreport.h"
#include "init_Qreport.h"
#include "defines.h"
#include "config.h"
```

Include dependency graph for Rcommand_Qreport.c:



Functions

- char * command_Qreport ()

  returns Rscript command that generates the quality report in html

Variables

- lparam_Qreport par_QR

4.35.1 Detailed Description

get Rscript command for Qreport

Author

Paula Perez paulaperezrubio@gmail.com

Date

07.08.2017

4.35.2 Variable Documentation

4.35.2.1 lparam_Qreport par_QR

input parameters Qreport

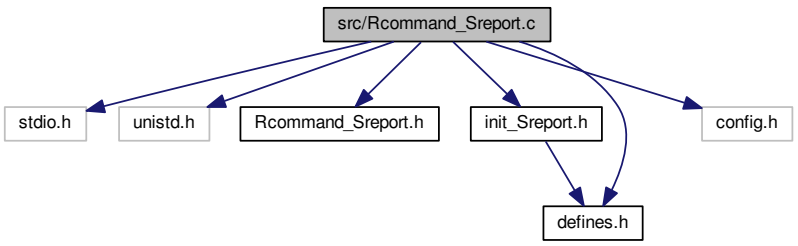
global variable: input parameters for Qreport

4.36 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

4.36.1 Detailed Description

get Rscript command for Sreport

Author

Paula Perez paulaperezrubio@gmail.com

Date

09.08.2017

4.36.2 Function Documentation

4.36.2.1 char* command_Sreport ( )

returns Rscript command that generates the summary report in html

```
1 # To run between quotation marks after:  Rscript_RBioC -e (Rscript)
2 inputfolder = normalizePath( <par.SR.inputfolder>, mustWork = TRUE);
3 output = <par_SR.outputfile>;
4 output_file = gsub('.* /', '', output);
5 path = gsub('[^/]+$', '', output);
6 if (path != '') {
7   outputfile = paste0(normalizePath(path, mustWork = TRUE), '/', outputfile);
8 } else {
9   outputfile = paste0(cwd, '/', output_file); # cwd: current working dir
10 };
11 rmarkdown::render(<par_SR.Rmd_file>,
12                   params = list(inputfolder = inputfolder, version= VERSION),
13                   output_file = output_file)
```

4.36.3 Variable Documentation

4.36.3.1 Iparam_Sreport par_SR

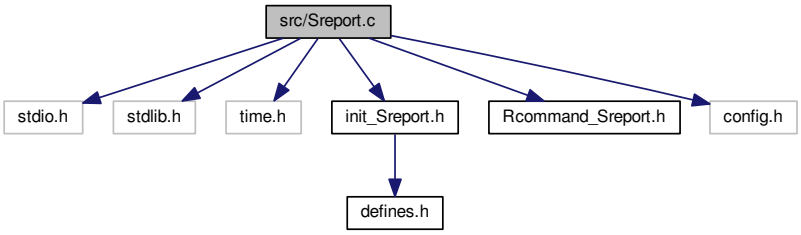
input parameters Sreport

4.37 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

4.37.1 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.

4.37.2 Variable Documentation

4.37.2.1 lparam_Sreport par_SR

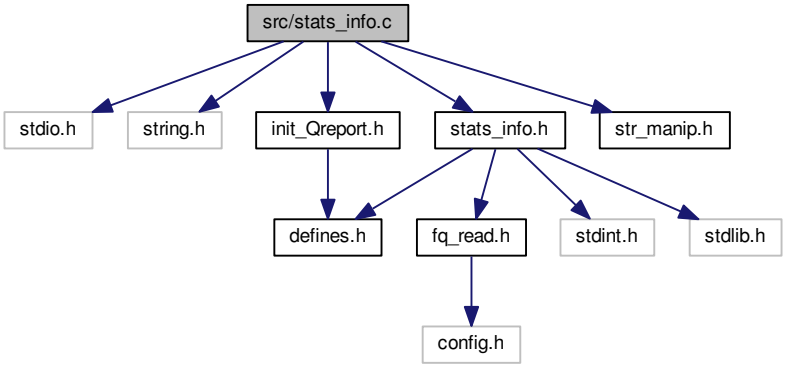
input parameters Sreport

4.38 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

4.38.1 Detailed Description

Construct the quality report variables and update them.

Author

Paula Perez paulaperezrubio@gmail.com

Date

04.08.2017

4.38.2 Function Documentation

4.38.2.1 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.

4.38.2.2 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).

4.38.2.3 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.

4.38.2.4 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

4.38.3 Variable Documentation

4.38.3.1 Iparam_Qreport par_QR

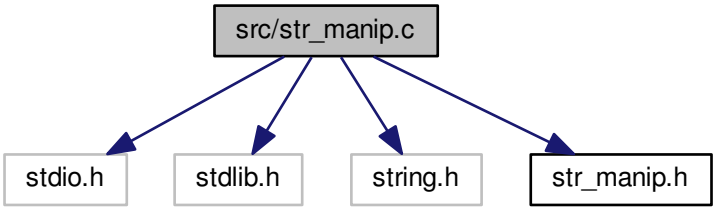
global variable: input parameters for Qreport

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

4.39.1 Detailed Description

functions that do string manipulation

Author

Paula Perez paulaperezrubio@gmail.com

Date

03.08.2017

4.39.2 Function Documentation

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

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

Parameters

s	string to be checked.
t	substring to be found in s.

4.39.2.2 int strindexC (char * *s*, char *sep*)

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

Parameters

s	string to be checked.
sep	char, separator

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

Separates strings by a separator.

Parameters

str	input string
sep	separator (char)

Returns

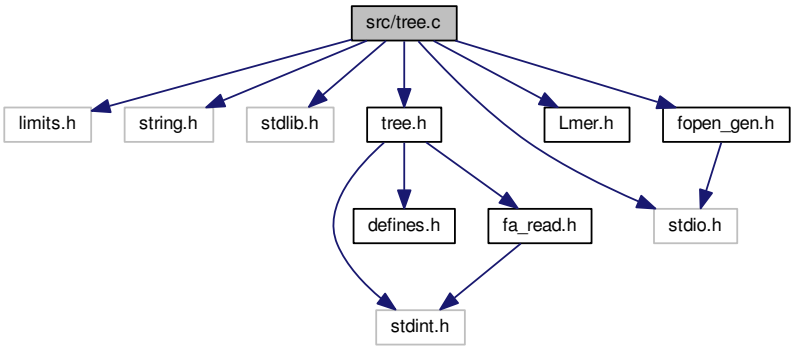
array of strings containing the substrings in the input separated

4.40 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

4.40.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

23.08.2017

4.40.2 Function Documentation

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

checks if read is found in tree and outputs a score

Parameters

tree_ptr	pointer to Tree structure
read	Read or reverse complement
Lread	length of read

Returns

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

4.40.2.2 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.

4.40.2.3 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

4.40.2.4 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.

4.40.2.5 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.

4.40.2.6 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.

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

create Tree structure from fasta structure.

Parameters

fasta	pointer to fasta structure
L	tree length

4.40.3 Variable Documentation

4.40.3.1 uint64_t alloc_mem

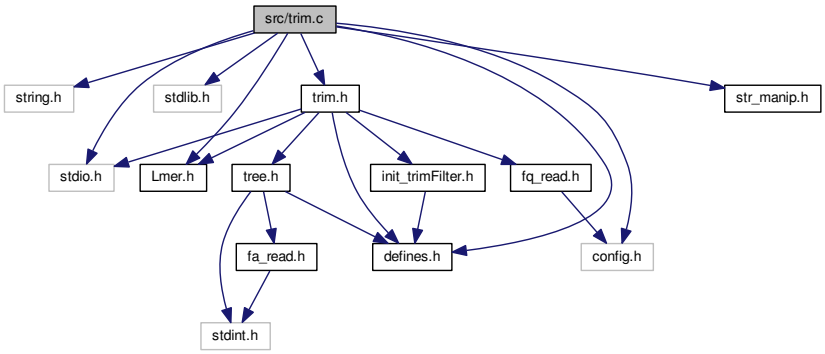
global variable. Memory allocated in the heap.

4.41 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 >= 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 >= 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 ot them exceeds the user selected threshold, it returns true. Otherwise, it returns false.

Variables

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

4.41.1 Detailed Description

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

Author

Paula Perez paulaperezrubio@gmail.com

Date

24.08.2017

4.41.2 Macro Definition Documentation

4.41.2.1 #define TRIM_STRING 20

maximal length of trimming info string.

4.41.3 Function Documentation

4.41.3.1 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 ot them exceeds the user selected threshold, it returns true. Otherwise, it returns false.

Parameters

tree_ptr	pointer to Tree structure
seq	fastq read

Returns

true if read was found, false otherwise

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

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

Parameters

seq	fastq read
minL	minimum accepted trimmed length

Returns

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

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

checks if a sequence contains lowQ nucleotides

Parameters

seq	fastq read
minQ	minimum accepted quality value

Returns

0 if seq contains lowQ nucleotides, 1 otherwise

4.41.3.4 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.

4.41.3.5 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

seq	fastq read
minL	minimum accepted trimmed length

Returns

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

4.41.3.6 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

seq	fastq read
minQ	minimum accepted quality value
minL	minimum accepted trimmed length

Returns

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

4.41.3.7 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

seq	fastq read
minQ	minimum accepted quality value
minL	minimum accepted trimmed length
nlowQ	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.

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

accepts the sequence as is if there are less than nlowQ

Parameters

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

Returns

0 if not used, 1 if accepted as is

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

trims left from the left and right from the right

Parameters

seq	fastq read
left	number of nucleotides to be trimmed from the left
right	number of nucleotides to be trimmed from the right

Returns

2, since they are all accepted and trim

4.41.3.10 int trim_sequenceN (Fq_read * *seq*)

trims sequence based on presence of N nucleotides

Parameters

seq	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).

4.41.3.11 int trim_sequenceQ (Fq_read * *seq*)

trims sequence based on lowQ base callings

Parameters

seq	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).

4.41.4 Variable Documentation

4.41.4.1 char LT[256]

global variable. Lookup table.

4.41.4.2 char Nencode

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

4.41.4.3 lparam_trimFilter par_TF

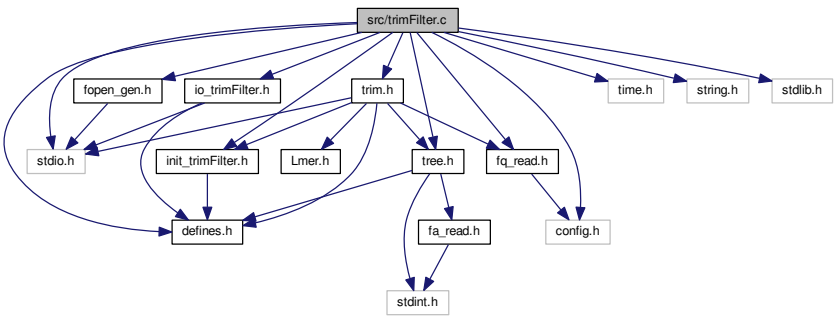
global variable: Input parameters of makeTree.

4.42 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

4.42.1 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.

4.42.2 Variable Documentation

4.42.2.1 uint64_t alloc_mem = 0

global variable. Memory allocated in the heap.

4.42.2.2 lparam_trimFilter par_TF

global variable: Input parameters of makeTree.

Index