FragScanTibo

0.1

Gegenereerd door Doxygen 1.8.18

Hoofdstuk 1

Klasse Index

1.1 Klasse Lijst

Hieronder volgen de klassen, structs en unions met voor elk een korte beschrijving:

Hoofdstuk 2

Bestand Index

2.1 Bestandslijst

Hieronder volgt de lijst met alle bestanden, elk met een korte beschrijving:

Hoofdstuk 3

Klassen Documentatie

3.1 HMM Struct Referentie

`#include <hmm.h>`

Public Attributen

- double pi [29]
- int N
- double tr [14]
- double e_M_1 [6][16][4]
- double e_M [6][16][4]
- double tr_R_R [4][4]
- double tr_I_I [4][4]
- double tr_M_I [4][4]
- double tr_S [61][64]
- double tr_E [61][64]
- double tr_S_1 [61][64]
- double tr_E_1 [61][64]
- double S_dist [6]
- double E_dist [6]
- double S1_dist [6]
- double E1_dist [6]

3.1.1 Documentatie van data members

3.1.1.1 E1_dist

`double E1_dist[6]`

3.1.1.2 E_dist

```
double E_dist[6]
```

3.1.1.3 e_M

```
double e_M[6][16][4]
```

3.1.1.4 e_M_1

```
double e_M_1[6][16][4]
```

3.1.1.5 N

```
int N
```

3.1.1.6 pi

```
double pi[29]
```

3.1.1.7 S1_dist

```
double S1_dist[6]
```

3.1.1.8 S_dist

```
double S_dist[6]
```

3.1.1.9 tr

```
double tr[14]
```

3.1.1.10 tr_E

```
double tr_E[61][64]
```

3.1.1.11 tr_E_1

```
double tr_E_1[61][64]
```

3.1.1.12 tr_I_I

```
double tr_I_I[4][4]
```

3.1.1.13 tr_M_I

```
double tr_M_I[4][4]
```

3.1.1.14 tr_R_R

```
double tr_R_R[4][4]
```

3.1.1.15 tr_S

```
double tr_S[61][64]
```

3.1.1.16 tr_S_1

```
double tr_S_1[61][64]
```

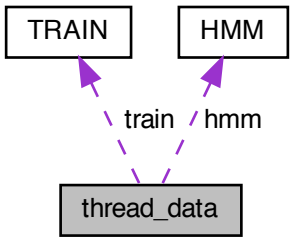
De documentatie voor deze struct is gegenereerd op grond van het volgende bestand:

- src/hmm.h

3.2 thread_data Struct Referentie

`#include <run_hmm.h>`

Collaboratie diagram voor thread_data:



Public Attributen

- FILE ∗ out
- FILE ∗ aa
- FILE ∗ dna
- char ∗ obs_head
- char ∗ obs_seq
- int wholegenome
- int cg
- int format
- HMM ∗ hmm
- TRAIN ∗ train

3.2.1 Documentatie van data members

3.2.1.1 aa

`FILE* aa`

3.2.1.2 cg

`int cg`

3.2.1.3 dna

```
FILE* dna
```

3.2.1.4 format

```
int format
```

3.2.1.5 hmm

```
HMM* hmm
```

3.2.1.6 obs_head

```
char* obs_head
```

3.2.1.7 obs_seq

```
char* obs_seq
```

3.2.1.8 out

```
FILE* out
```

3.2.1.9 train

```
TRAIN* train
```

3.2.1.10 wholegenome

```
int wholegenome
```

De documentatie voor deze struct is gegenereerd op grond van het volgende bestand:

- src/run_hmm.h

3.3 TRAIN Struct Referentie

```
#include <hmm.h>
```

Public Attributen

- double trans [44][6][16][4]
- double rtrans [44][6][16][4]
- double noncoding [44][4][4]
- double start [44][61][64]
- double stop [44][61][64]
- double start1 [44][61][64]
- double stop1 [44][61][64]
- double S_dist [44][6]
- double E_dist [44][6]
- double S1_dist [44][6]
- double E1_dist [44][6]

3.3.1 Documentatie van data members

3.3.1.1 E1_dist

```
double E1_dist[44][6]
```

3.3.1.2 E_dist

```
double E_dist[44][6]
```

3.3.1.3 noncoding

```
double noncoding[44][4][4]
```

3.3.1.4 rtrans

```
double rtrans[44][6][16][4]
```

3.3.1.5 S1_dist

```
double S1_dist[44][6]
```

3.3.1.6 S_dist

```
double S_dist[44][6]
```

3.3.1.7 start

```
double start[44][61][64]
```

3.3.1.8 start1

```
double start1[44][61][64]
```

3.3.1.9 stop

```
double stop[44][61][64]
```

3.3.1.10 stop1

```
double stop1[44][61][64]
```

3.3.1.11 trans

```
double trans[44][6][16][4]
```

De documentatie voor deze struct is gegenereerd op grond van het volgende bestand:

- src/hmm.h

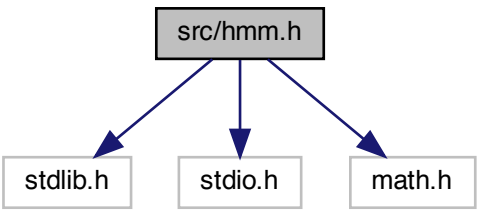
Hoofdstuk 4

Bestand Documentatie

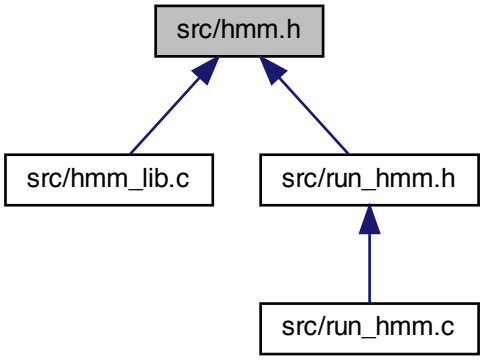
4.1 src/hmm.h Bestand Referentie

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

Include afhankelijkheidsgraaf voor hmm.h:



Deze graaf geeft aan welke bestanden direct of indirect afhankelijk zijn van dit bestand:



Klassen

- struct HMM
- struct TRAIN

Macros

- #define A 0
- #define C 1
- #define G 2
- #define T 3
- #define NUM_STATE 29
- #define NOSTATE -1
- #define S_STATE 0
- #define E_STATE 1
- #define R_STATE 2
- #define S_STATE_1 3
- #define E_STATE_1 4
- #define M1_STATE 5
- #define M2_STATE 6
- #define M3_STATE 7
- #define M4_STATE 8
- #define M5_STATE 9
- #define M6_STATE 10
- #define M1_STATE_1 11
- #define M2_STATE_1 12
- #define M3_STATE_1 13
- #define M4_STATE_1 14
- #define M5_STATE_1 15
- #define M6_STATE_1 16
- #define I1_STATE 17
- #define I2_STATE 18

- #define I3_STATE 19
- #define I4_STATE 20
- #define I5_STATE 21
- #define I6_STATE 22
- #define I1_STATE_1 23
- #define I2_STATE_1 24
- #define I3_STATE_1 25
- #define I4_STATE_1 26
- #define I5_STATE_1 27
- #define I6_STATE_1 28
- #define TR_MM 0
- #define TR_MI 1
- #define TR_MD 2
- #define TR_II 3
- #define TR_IM 4
- #define TR_DD 5
- #define TR_DM 6
- #define TR_GE 7
- #define TR_GG 8
- #define TR_ER 9
- #define TR_RS 10
- #define TR_RR 11
- #define TR_ES 12
- #define TR_ES1 13

Functies

- int get_prob_from_cg (HMM *hmm, TRAIN *train, char *O)
- void get_train_from_file (char *filename, HMM *hmm_ptr, char *mfilename, char *mfilename1, char *nfilename, char *sfilename, char *pfilename, char *s1filename, char *p1filename, char *dfilename, TRAIN *train_ptr)
- void viterbi (HMM *hmm_ptr, TRAIN *train_ptr, char *O, FILE *out_filename, FILE *log_filename, FILE *dna_filename, char *head, int metagene, int cg, int format)
- void free_hmm (HMM *hmm)
- void get_protein (char *dna, char *protein, int strand, int whole_genome)
- void get_rc_dna (char *dna, char *dna1)
- void get_corrected_dna (char *dna, char *dna_f)

4.1.1 Gedetailleerde Beschrijving

This is the header file for the hmm datastructure.

4.1.2 Documentatie van macro's

4.1.2.1 A

```
#define A 0
```

4.1.2.2 C

```
#define C 1
```

4.1.2.3 E_STATE

```
#define E_STATE 1
```

4.1.2.4 E_STATE_1

```
#define E_STATE_1 4
```

4.1.2.5 G

```
#define G 2
```

4.1.2.6 I1_STATE

```
#define I1_STATE 17
```

4.1.2.7 I1_STATE_1

```
#define I1_STATE_1 23
```

4.1.2.8 I2_STATE

```
#define I2_STATE 18
```

4.1.2.9 I2_STATE_1

```
#define I2_STATE_1 24
```

4.1.2.10 I3_STATE

```
#define I3_STATE 19
```

4.1.2.11 I3_STATE_1

```
#define I3_STATE_1 25
```

4.1.2.12 I4_STATE

```
#define I4_STATE 20
```

4.1.2.13 I4_STATE_1

```
#define I4_STATE_1 26
```

4.1.2.14 I5_STATE

```
#define I5_STATE 21
```

4.1.2.15 I5_STATE_1

```
#define I5_STATE_1 27
```

4.1.2.16 I6_STATE

```
#define I6_STATE 22
```

4.1.2.17 I6_STATE_1

```
#define I6_STATE_1 28
```

4.1.2.18 M1_STATE

```
#define M1_STATE 5
```

4.1.2.19 M1_STATE_1

```
#define M1_STATE_1 11
```

4.1.2.20 M2_STATE

```
#define M2_STATE 6
```

4.1.2.21 M2_STATE_1

```
#define M2_STATE_1 12
```

4.1.2.22 M3_STATE

```
#define M3_STATE 7
```

4.1.2.23 M3_STATE_1

```
#define M3_STATE_1 13
```

4.1.2.24 M4_STATE

```
#define M4_STATE 8
```

4.1.2.25 M4_STATE_1

```
#define M4_STATE_1 14
```

4.1.2.26 M5_STATE

```
#define M5_STATE 9
```

4.1.2.27 M5_STATE_1

```
#define M5_STATE_1 15
```

4.1.2.28 M6_STATE

```
#define M6_STATE 10
```

4.1.2.29 M6_STATE_1

```
#define M6_STATE_1 16
```

4.1.2.30 NOSTATE

```
#define NOSTATE -1
```

4.1.2.31 NUM_STATE

```
#define NUM_STATE 29
```

Total number of states, mainly used in for loops.

4.1.2.32 R_STATE

```
#define R_STATE 2
```

4.1.2.33 S_STATE

```
#define S_STATE 0
```

4.1.2.34 S_STATE_1

```
#define S_STATE_1 3
```

4.1.2.35 T

```
#define T 3
```

4.1.2.36 TR_DD

```
#define TR_DD 5
```

4.1.2.37 TR_DM

```
#define TR_DM 6
```

4.1.2.38 TR_ER

```
#define TR_ER 9
```

4.1.2.39 TR_ES

```
#define TR_ES 12
```

4.1.2.40 TR_ES1

```
#define TR_ES1 13
```

4.1.2.41 TR_GE

```
#define TR_GE 7
```

4.1.2.42 TR_GG

```
#define TR_GG 8
```

4.1.2.43 TR_II

```
#define TR_II 3
```

4.1.2.44 TR_IM

```
#define TR_IM 4
```

4.1.2.45 TR_MD

```
#define TR_MD 2
```

4.1.2.46 TR_MI

```
#define TR_MI 1
```

4.1.2.47 TR_MM

```
#define TR_MM 0
```

4.1.2.48 TR_RR

```
#define TR_RR 11
```

4.1.2.49 TR_RS

```
#define TR_RS 10
```

4.1.3 Documentatie van functies

4.1.3.1 free_hmm()

```
void free_hmm (
            HMM * hmm )
```

Hier is de call graaf voor deze functie:

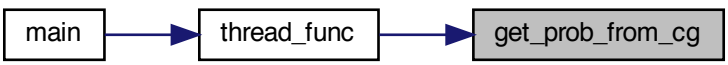
4.1.3.2 get_corrected_dna()

```
void get_corrected_dna (
            char * dna,
            char * dna_f )
```

4.1.3.3 get_prob_from_cg()

```
int get_prob_from_cg (
            HMM * hmm,
            TRAIN * train,
            char * O )
```

Hier is de caller graaf voor deze functie:



4.1.3.4 get_protein()

```
void get_protein (
            char * dna,
            char * protein,
            int strand,
            int whole_genome )
```

Get a protein of dna if Whole_genome equals to zero, then we want a short read and stop early. Hier is de caller graaf voor deze functie:

4.1.3.5 get_rc_dna()

```
void get_rc_dna (
            char * dna,
            char * dna1 )
```

copies dna to dna1 in reverse. and Hier is de call graaf voor deze functie:

Hier is de caller graaf voor deze functie:

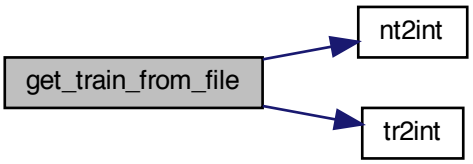
4.1.3.6 get_train_from_file()

```
void get_train_from_file (
              char * filename,
              HMM * hmm_ptr,
              char * mfilename,
              char * mfilename1,
              char * nfilename,
              char * sfilename,
              char * pfilename,
              char * s1filename,
              char * p1filename,
              char * dfilename,
              TRAIN * train_ptr )
```

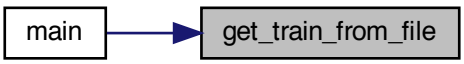
Reads files.

1. Reads trasition file and store in hmm datastructure

Hier is de call graaf voor deze functie:



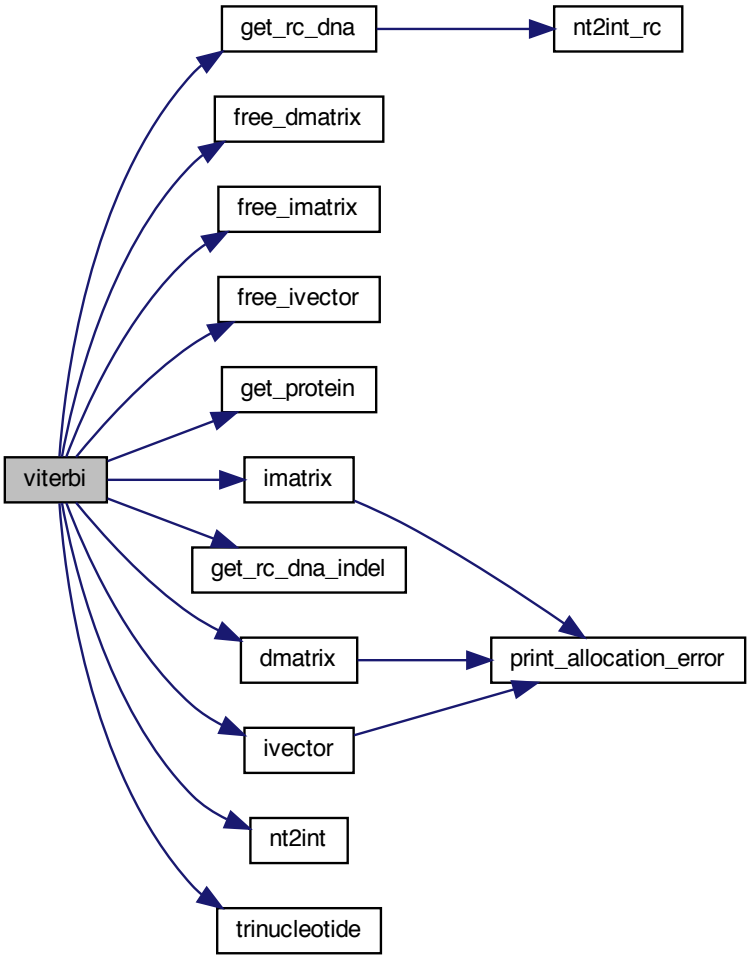
Hier is de caller graaf voor deze functie:



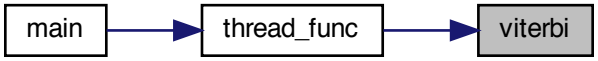
4.1.3.7 viterbi()

```
void viterbi (
              HMM * hmm_ptr,
              TRAIN * train_ptr,
              char * O,
              FILE * out_filename,
              FILE * log_filename,
              FILE * dna_filename,
              char * head,
              int metagene,
              int cg,
              int format )
```

Hier is de call graaf voor deze functie:



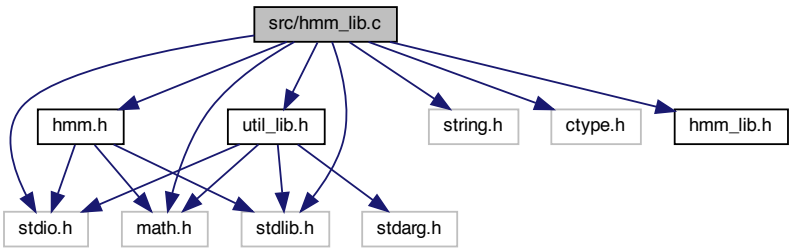
Hier is de caller graaf voor deze functie:



4.2 src/hmm_lib.c Bestand Referentie

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include "hmm.h"
#include "util_lib.h"
#include "hmm_lib.h"
```

Include afhankelijkheidsgraaf voor hmm_lib.c:



Functies

- void viterbi (HMM *hmm_ptr, TRAIN *train_ptr, char *O, FILE *fp_out, FILE *fp_aa, FILE *fp_dna, char *head, int whole_genome, int cg, int format)
- int get_prob_from_cg (HMM *hmm_ptr, TRAIN *train_ptr, char *O)
- void get_train_from_file (char *filename, HMM *hmm_ptr, char *mfilename, char *mfilename1, char *nfilename, char *sfilename, char *pfilename, char *s1filename, char *p1filename, char *dfilename, TRAIN *train_ptr)
- void free_hmm (HMM *hmm_ptr)
- void dump_memory (void *p, int size)

4.2.1 Documentatie van functies

4.2.1.1 dump_memory()

```
void dump_memory (
             void * p,
             int size )
```

4.2.1.2 free_hmm()

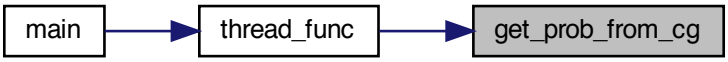
```
void free_hmm (
             HMM * hmm_ptr )
```

Hier is de call graaf voor deze functie:

4.2.1.3 get_prob_from_cg()

```
int get_prob_from_cg (
             HMM * hmm_ptr,
             TRAIN * train_ptr,
             char * O )
```

Hier is de caller graaf voor deze functie:



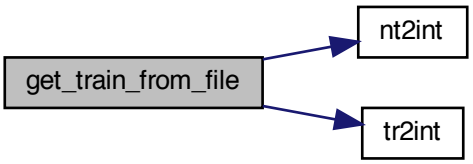
4.2.1.4 get_train_from_file()

```
void get_train_from_file (
              char * filename,
              HMM * hmm_ptr,
              char * mfilename,
              char * mfilename1,
              char * nfilename,
              char * sfilename,
              char * pfilename,
              char * s1filename,
              char * p1filename,
              char * dfilename,
              TRAIN * train_ptr )
```

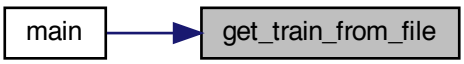
Reads files.

1. Reads trasition file and store in hmm datastructure

Hier is de call graaf voor deze functie:



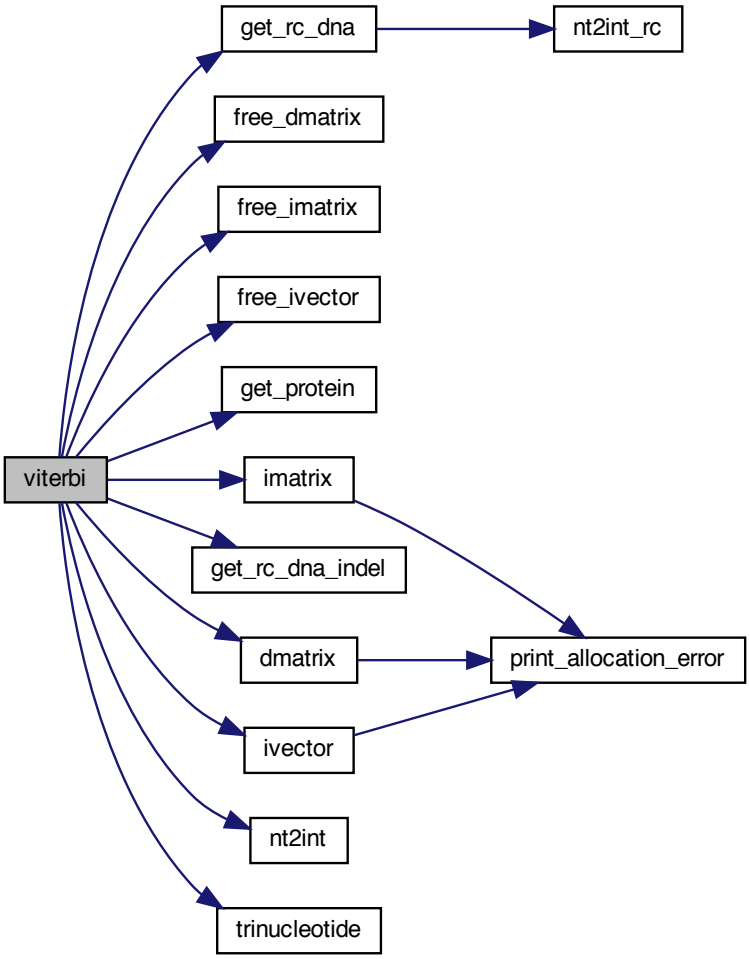
Hier is de caller graaf voor deze functie:



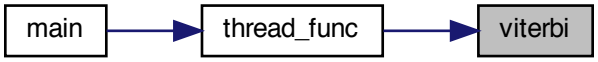
4.2.1.5 viterbi()

```
void viterbi (
              HMM * hmm_ptr,
              TRAIN * train_ptr,
              char * O,
              FILE * fp_out,
              FILE * fp_aa,
              FILE * fp_dna,
              char * head,
              int whole_genome,
              int cg,
              int format )
```

Hier is de call graaf voor deze functie:

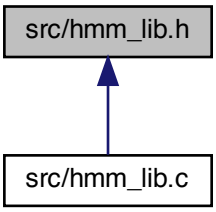


Hier is de caller graaf voor deze functie:



4.3 src/hmm_lib.h Bestand Referentie

Deze graaf geeft aan welke bestanden direct of indirect afhankelijk zijn van dit bestand:



Functies

- void dump_memory (void ∗p, int size)
- void get_rc_dna_indel (char dna_f[300000], char dna_f1[300000])

4.3.1 Documentatie van functies

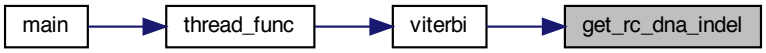
4.3.1.1 dump_memory()

```
void dump_memory (
            void * p,
            int size )
```

4.3.1.2 get_rc_dna_indel()

```
void get_rc_dna_indel (
            char dna_f[300000],
            char dna_f1[300000] )
```

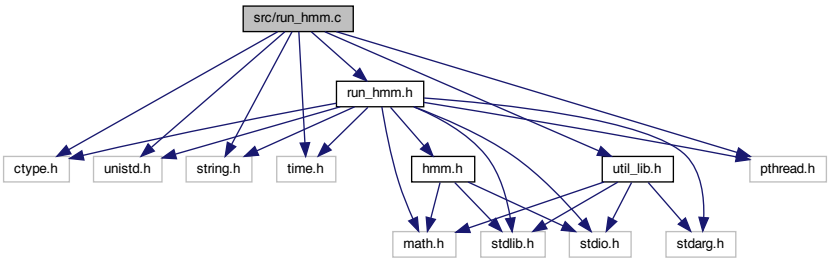
Hier is de caller graaf voor deze functie:



4.4 src/run_hmm.c Bestand Referentie

```
#include <ctype.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include "run_hmm.h"
#include "util_lib.h"
#include <pthread.h>
```

Include afhankelijkheidsgraaf voor run_hmm.c:



Macros

- #define ADD_LEN 1024
- #define STRINGLEN 4096

Functies

- int main (int argc, char ∗∗argv)
- void ∗ thread_func (void ∗threadarr)
- int appendSeq (char ∗input, char ∗∗seq, int input_max)
- void print_error (const char ∗error_message,...)
- void print_file_error (const char ∗error_message, char ∗file)

4.4.1 Documentatie van macro's

4.4.1.1 ADD_LEN

```
#define ADD_LEN 1024
```

4.4.1.2 STRINGLEN

```
#define STRINGLEN 4096
```

4.4.2 Documentatie van functies

4.4.2.1 appendSeq()

```
int appendSeq (
             char * input,
             char ** seq,
             int input_max )
```

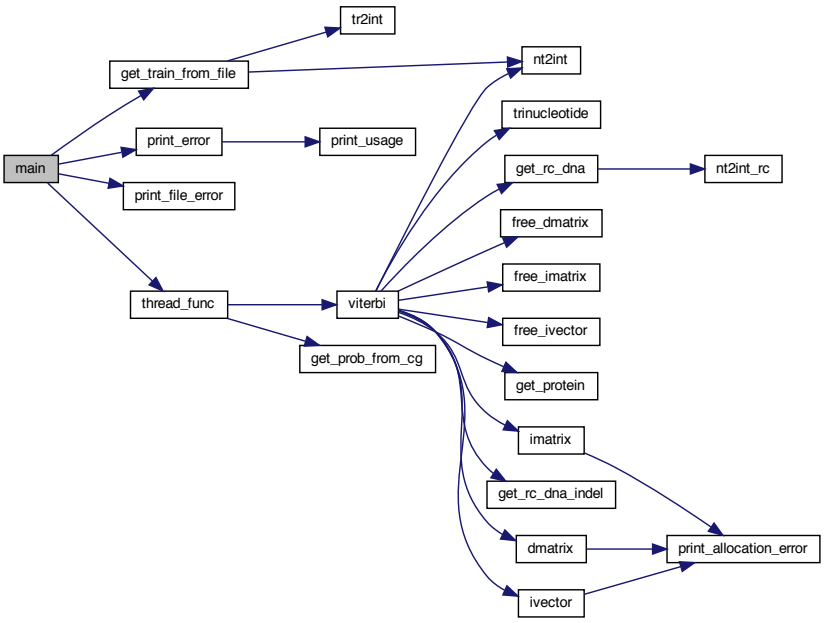
4.4.2.2 main()

```
int main (
             int argc,
             char ** argv )
```

Entry point of program

1. Initialization of variables and datatypes
2. Check File acessiblity

Hier is de call graaf voor deze functie:



4.4.2.3 print_error()

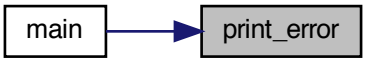
```
void print_error (
             const char * error_message,
              ...  )
```

Error function:

1. Print error message
2. Call print_usage() from util_lib
3. EXIT program

Hier is de call graaf voor deze functie:

Hier is de caller graaf voor deze functie:



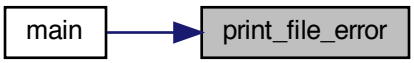
4.4.2.4 print_file_error()

```
void print_file_error (
            const char * error_message,
            char * file )
```

Error function:

1. Print error message
2. EXIT program

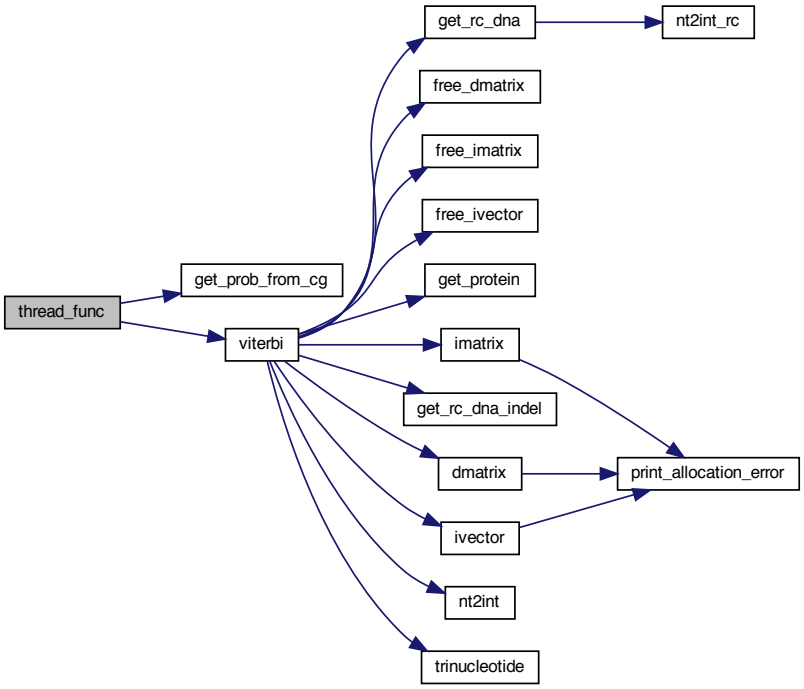
Hier is de caller graaf voor deze functie:



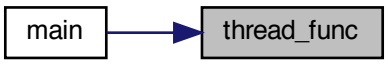
4.4.2.5 thread_func()

```
void* thread_func (
            void * threadarr )
```

Hier is de call graaf voor deze functie:



Hier is de caller graaf voor deze functie:

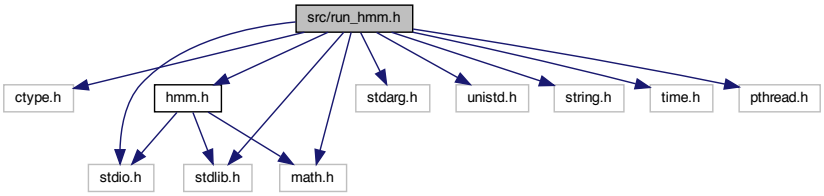


4.5 src/run_hmm.h Bestand Referentie

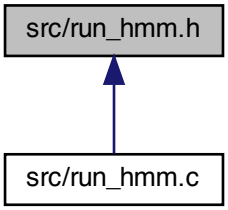
```
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include "hmm.h"
```

```
#include <pthread.h>
```

Include afhankelijkheidsgraaf voor run_hmm.h:



Deze graaf geeft aan welke bestanden direct of indirect afhankelijk zijn van dit bestand:



Klassen

- struct thread_data

Typedefs

- typedef struct thread_data thread_data

Functies

- void ∗ thread_func (void ∗threadarr)
- void print_error (const char ∗error_message,...)
- void print_file_error (const char ∗error_message, char ∗file)

4.5.1 Documentatie van typedefs

4.5.1.1 thread_data

```
typedef struct thread_data thread_data
```

4.5.2 Documentatie van functies

4.5.2.1 print_error()

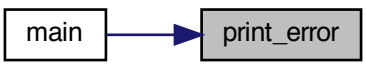
```
void print_error (
             const char * error_message,
              ...  )
```

Error function:

1. Print error message
2. Call print_usage() from util_lib
3. EXIT program

Hier is de call graaf voor deze functie:

Hier is de caller graaf voor deze functie:



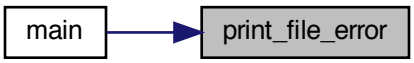
4.5.2.2 print_file_error()

```
void print_file_error (
            const char * error_message,
            char * file )
```

Error function:

1. Print error message
2. EXIT program

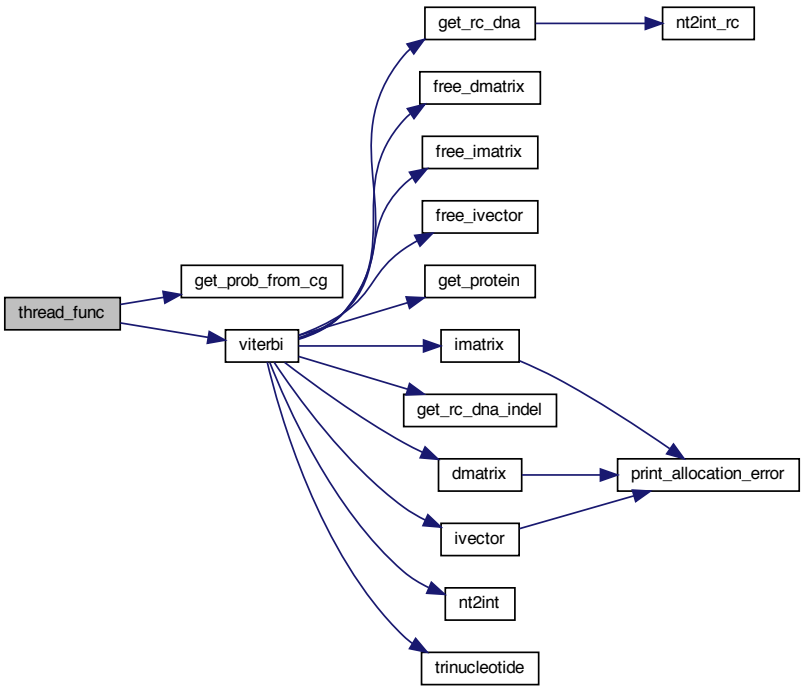
Hier is de caller graaf voor deze functie:



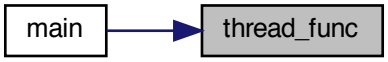
4.5.2.3 thread_func()

```
void* thread_func (
            void * threadarr )
```

Hier is de call graaf voor deze functie:



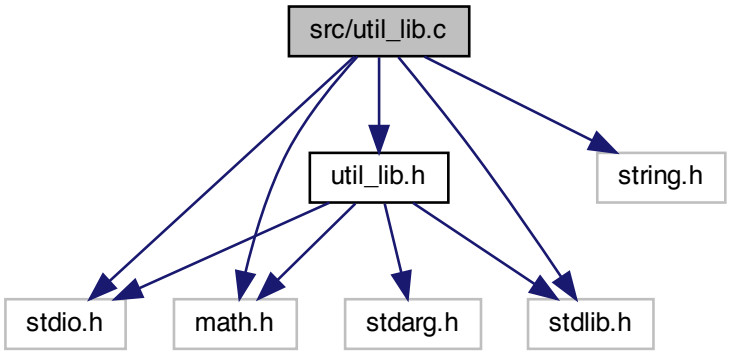
Hier is de caller graaf voor deze functie:



4.6 src/util_lib.c Bestand Referentie

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include "util_lib.h"
```

Include afhankelijkheidsgraaf voor util_lib.c:



Macros

- #define TR_SIZE 14

Functies

- double ∗∗ dmatrix (int num_row, int num_col)
- int ∗∗ imatrix (int num_row, int num_col)
- double ∗ dvector (int nh)
- int ∗ ivector (int nh)
- void free_dvector (double ∗v)
- void free_ivector (int ∗v)

- void free_dmatrix (double **m, int num_row)
- void free_imatrix (int **m, int num_row)
- int tr2int (char *tr)
- int nt2int (char nt)
- int nt2int_rc (char nt)
- int nt2int_rc_indel (char nt)
- int trinucleotide (char a, char b, char c)
- int trinucleotide_pep (char a, char b, char c)
- void get_rc_dna (char *dna, char *dna1)
- void get_rc_dna_indel (char *dna, char *dna1)
- void get_protein (char *dna, char *protein, int strand, int whole_genome)
- void print_usage ()
- void print_allocation_error (const char *format,...)

Variabelen

- const char * tr_list [TR_SIZE] = { "MM","MI","MD","II","IM","DD","DM","GE","GG","ER","RS","RR","ES","ES1" }
- const char codon5 [5] = { 'A', 'C', 'G', 'T', 'N' }
- const char codon11 [11] = { 'A', 'C', 'G', 'T', 'N', 'a', 'c', 'g', 't', 'n', 'x' }
- const char codon_code [65]
- const char anti_codon_code [65]

4.6.1 Documentatie van macro's

4.6.1.1 TR_SIZE

```
#define TR_SIZE 14
```

4.6.2 Documentatie van functies

4.6.2.1 dmatrix()

```
double** dmatrix (
            int num_row,
            int num_col )
```

Makes an matrix with datatype double. Elements are double pointers en matrix is a double double pointer (**pointer). Exits when allocation fails. Hier is de call graaf voor deze functie:

Hier is de caller graaf voor deze functie:

4.6.2.2 dvector()

```
double* dvector (
            int nh )
```

Makes an vector (array) with datatype double. Elements are doubles en vector is a double pointer. Exits when allocation fails. Hier is de call graaf voor deze functie:

4.6.2.3 free_dmatrix()

```
void free_dmatrix (
            double ** m,
            int num_row )
```

Frees the memory allocation of an matrix with datatype double. Hier is de caller graaf voor deze functie:

4.6.2.4 free_dvector()

```
void free_dvector (
            double * v )
```

Frees the memory allocation of an vector with datatype double. Hier is de caller graaf voor deze functie:

4.6.2.5 free_imatrix()

```
void free_imatrix (
            int ** m,
            int num_row )
```

Frees the memory allocation of an matrix with datatype int. Hier is de caller graaf voor deze functie:

4.6.2.6 free_ivector()

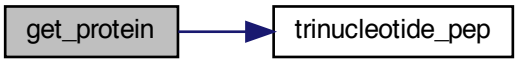
```
void free_ivector (
             int * v )
```

Frees the memory allocation of an vector with datatype int. Hier is de caller graaf voor deze functie:

4.6.2.7 get_protein()

```
void get_protein (
             char * dna,
             char * protein,
             int strand,
             int whole_genome )
```

Get a protein of dna if Whole_genome equals to zero, then we want a short read and stop early. Hier is de call graaf voor deze functie:



Hier is de caller graaf voor deze functie:

4.6.2.8 get_rc_dna()

```
void get_rc_dna (
            char * dna,
            char * dna1 )
```

copies dna to dna1 in reverse. and Hier is de call graaf voor deze functie:

Hier is de caller graaf voor deze functie:

4.6.2.9 get_rc_dna_indel()

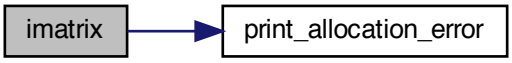
```
void get_rc_dna_indel (
            char * dna,
            char * dna1 )
```

copies dna to dna1 in reverse. and Hier is de call graaf voor deze functie:

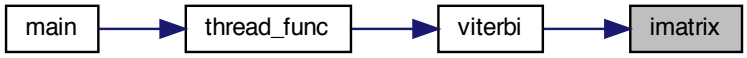
4.6.2.10 imatrix()

```
int** imatrix (
            int num_row,
            int num_col )
```

Makes an matrix with datatype int. Elements are int pointers en matrix is a double int pointer. Exits when allocation fails. Hier is de call graaf voor deze functie:



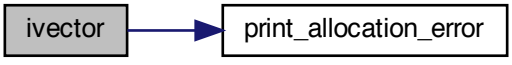
Hier is de caller graaf voor deze functie:



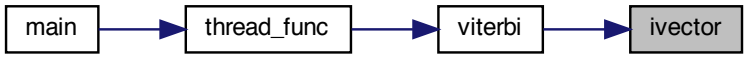
4.6.2.11 ivector()

```
int* ivector (
            int nh )
```

Makes an vector array) with datatype int. Elements are ints en vector is a int pointer. Exits when allocation fails. Hier is de call graaf voor deze functie:



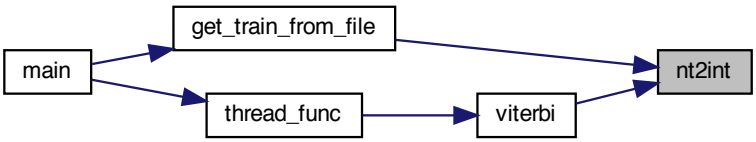
Hier is de caller graaf voor deze functie:



4.6.2.12 nt2int()

```
int nt2int (
            char nt )
```

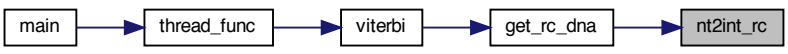
Hier is de caller graaf voor deze functie:



4.6.2.13 nt2int_rc()

```
int nt2int_rc (
            char nt )
```

Hier is de caller graaf voor deze functie:



4.6.2.14 nt2int_rc_indel()

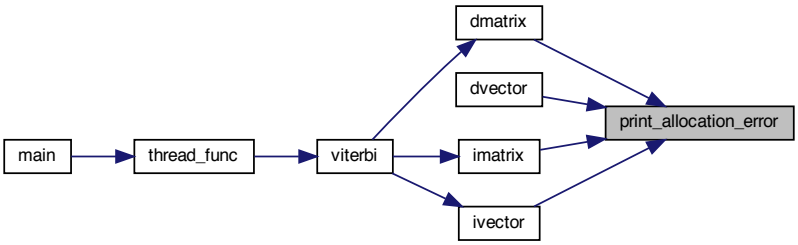
```
int nt2int_rc_indel (
             char nt )
```

Hier is de caller graaf voor deze functie:

4.6.2.15 print_allocation_error()

```
void print_allocation_error (
             const char * format,
              ...  )
```

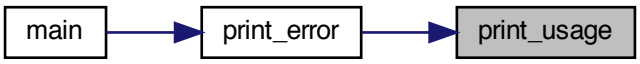
Custom error function to print allocation errors. Mostly called from matrix or vector functions. Hier is de caller graaf voor deze functie:



4.6.2.16 print_usage()

```
void print_usage ( )
```

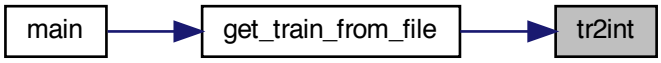
Print how the program should be used. called mainly on help or error. Hier is de caller graaf voor deze functie:



4.6.2.17 tr2int()

```
int tr2int (
            char * tr )
```

Converts a given transition to int. Use for example as indexing. switch case not possible due the fact that strings are not constonant. Hier is de caller graaf voor deze functie:



4.6.2.18 trinucleotide()

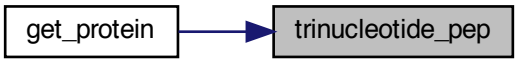
```
int trinucleotide (
            char a,
            char b,
            char c )
```

Hier is de caller graaf voor deze functie:

4.6.2.19 trinucleotide_pep()

```
int trinucleotide_pep (
            char a,
            char b,
            char c )
```

Hier is de caller graaf voor deze functie:



4.6.3 Documentatie van variabelen

4.6.3.1 anti_codon_code

```
const char anti_codon_code[65]
```

Initiële waarde:

```
= { 'F','V','L','I',
            'C','G','R','S',
            'S','A','P','T',
            'Y','D','H','N',
            'L','V','L','M',
            'W','G','R','R',
            'S','A','P','T',
            '*','E','Q','K',
            'F','V','L','I',
            'C','G','R','S',
            'S','A','P','T',
            'Y','D','H','N',
            'L','V','L','I',
            '*','G','R','R',
            'S','A','P','T',
            '*','E','Q','K','X' }
```

4.6.3.2 codon11

```
const char codon11[11] = { 'A', 'C', 'G', 'T', 'N', 'a', 'c', 'g', 't', 'n', 'x' }
```

4.6.3.3 codon5

```
const char codon5[5] = { 'A', 'C', 'G', 'T', 'N' }
```

4.6.3.4 codon_code

```
const char codon_code[65]
```

Initiële waarde:

```
= { 'K','N','K','N',
            'T','T','T','T',
            'R','S','R','S',
            'I','I','M','I',
            'Q','H','Q','H',
            'P','P','P','P',
            'R','R','R','R',
            'L','L','L','L',
            'E','D','E','D',
            'A','A','A','A',
            'G','G','G','G',
            'V','V','V','V',
            '*','Y','*','Y',
            'S','S','S','S',
            '*','C','W','C',
            'L','F','L','F', 'X' }
```

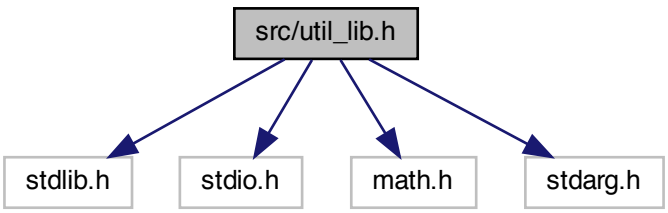
4.6.3.5 tr_list

```
const char* tr_list[TR_SIZE] = { "MM","MI","MD","II","IM","DD","DM","GE","GG","ER","RS","RR","ES","E↩
S1" }
```

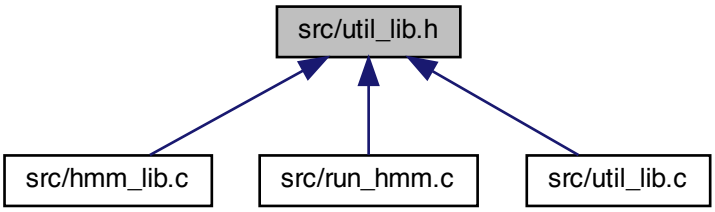
4.7 src/util_lib.h Bestand Referentie

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <stdarg.h>
```

Include afhankelijkheidsgraaf voor util_lib.h:



Deze graaf geeft aan welke bestanden direct of indirect afhankelijk zijn van dit bestand:



Functies

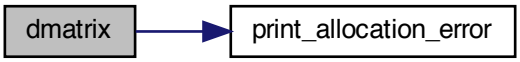
- double ∗∗ dmatrix (int num_row, int num_col)
- double ∗ dvector (int nh)
- int ∗∗ imatrix (int num_row, int num_col)
- int ∗ ivector (int nh)
- void free_dvector (double ∗v)
- void free_dmatrix (double ∗∗m, int num_row)
- void free_ivector (int ∗v)
- void free_imatrix (int ∗∗m, int num_row)
- int tr2int (char ∗nt)
- int nt2int (char nt)
- int nt2int_rc (char nt)
- int trinucleotide (char a, char b, char c)
- void get_protein (char ∗dna, char ∗protein, int strand, int whole_genome)
- void print_usage ()
- void print_allocation_error (const char ∗format,...)

4.7.1 Documentatie van functies

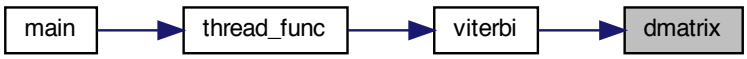
4.7.1.1 dmatrix()

```
double** dmatrix (
            int num_row,
            int num_col )
```

Makes an matrix with datatype double. Elements are double pointers en matrix is a double double pointer (∗∗pointer). Exits when allocation fails. Hier is de call graaf voor deze functie:



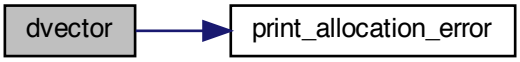
Hier is de caller graaf voor deze functie:



4.7.1.2 dvector()

```
double* dvector (
            int nh )
```

Makes an vector (array) with datatype double. Elements are doubles en vector is a double pointer. Exits when allocation fails. Hier is de call graaf voor deze functie:



4.7.1.3 free_dmatrix()

```
void free_dmatrix (
            double ** m,
            int num_row )
```

Frees the memory allocation of an matrix with datatype double. Hier is de caller graaf voor deze functie:

4.7.1.4 free_dvector()

```
void free_dvector (
            double * v )
```

Frees the memory allocation of an vector with datatype double. Hier is de caller graaf voor deze functie:

4.7.1.5 free_imatrix()

```
void free_imatrix (
            int ** m,
            int num_row )
```

Frees the memory allocation of an matrix with datatype int. Hier is de caller graaf voor deze functie:

4.7.1.6 free_ivector()

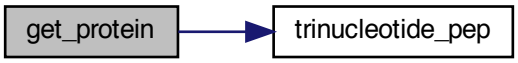
```
void free_ivector (
            int * v )
```

Frees the memory allocation of an vector with datatype int. Hier is de caller graaf voor deze functie:

4.7.1.7 get_protein()

```
void get_protein (
            char * dna,
            char * protein,
            int strand,
            int whole_genome )
```

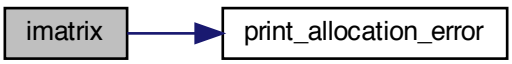
Get a protein of dna if Whole_genome equals to zero, then we want a short read and stop early. Hier is de call graaf voor deze functie:



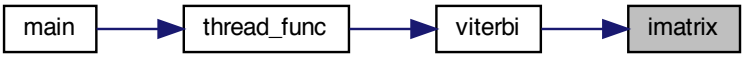
4.7.1.8 imatrix()

```
int** imatrix (
            int num_row,
            int num_col )
```

Makes an matrix with datatype int. Elements are int pointers en matrix is a double int pointer. Exits when allocation fails. Hier is de call graaf voor deze functie:



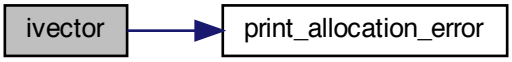
Hier is de caller graaf voor deze functie:



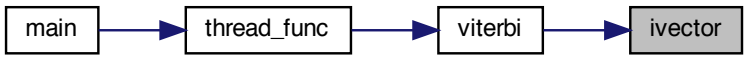
4.7.1.9 ivector()

```
int* ivector (
            int nh )
```

Makes an vector array) with datatype int. Elements are ints en vector is a int pointer. Exits when allocation fails.
Hier is de call graaf voor deze functie:



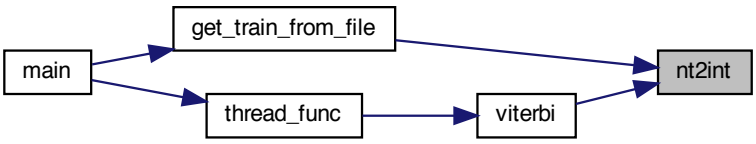
Hier is de caller graaf voor deze functie:



4.7.1.10 nt2int()

```
int nt2int (
            char nt )
```

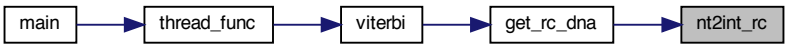
Hier is de caller graaf voor deze functie:



4.7.1.11 nt2int_rc()

```
int nt2int_rc (
            char nt )
```

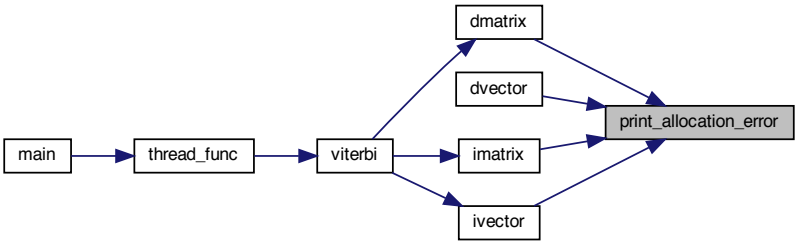
Hier is de caller graaf voor deze functie:



4.7.1.12 print_allocation_error()

```
void print_allocation_error (
            const char * format,
             ...  )
```

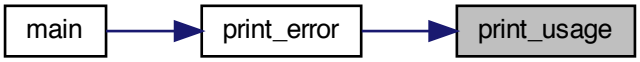
Custom error function to print allocation errors. Mostly called from matrix or vector functions. Hier is de caller graaf voor deze functie:



4.7.1.13 print_usage()

```
void print_usage ( )
```

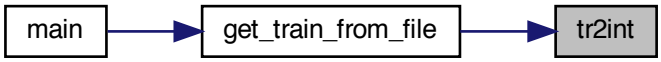
Print how the program should be used. called mainly on help or error. Hier is de caller graaf voor deze functie:



4.7.1.14 tr2int()

```
int tr2int (
            char * tr )
```

Converts a given transition to int. Use for example as indexing. switch case not possible due the fact that strings are not constonant. Hier is de caller graaf voor deze functie:



4.7.1.15 trinucleotide()

```
int trinucleotide (
            char a,
            char b,
            char c )
```

Hier is de caller graaf voor deze functie: