GenomeThreader Gene Prediction Software

Manual

Gordon Gremme
E-mail: gordon@gremme.org

February 7, 2018

Contents

List of Tables

To the Impatient Reader

Seeing this manual, one could think:

53 pages of documentation, you must be joking! [1]

We try hard to fully document *GenomeThreader*, helping to put the user in a position where he can understand and use the various tools and options of *GenomeThreader*.

Nevertheless, if you want to start using *GenomeThreader* as fast as possible (without reading the whole documentation upfront), you can do so by reading the installation guidelines given in Section 2 and the tutorial given in Section 13 first and coming back to the other sections on demand later.

1 Introduction

This document describes how to use *GenomeThreader*, a software tool to compute gene structure predictions. The gene structure predictions are calculated using a similarity-based approach where additional cDNA/EST and/or protein sequences are used to predict gene structures via spliced alignments.

The algorithms, the phases, and the software engineering of *GenomeThreader*[2] are described in [Gre12] and [GBSK05]. More details on the core Dynamic Programming (DP) algorithms used to computed spliced alignments via cDNAs/ESTs are given in [UZB00]. The DP algorithms used to compute spliced alignments via protein sequences are described in [UB00]. Here are the most important features of *GenomeThreader*.

Intron Cutout Technique

The core DP algorithms have been extended by the *intron cutout technique* [GBSK05] which allows to apply *GenomeThreader* to organisms with long introns. This overcomes the time and space limitations the algorithms described in [UZB00] and [UB00] have when applied to genomic sequences containing long introns.

Incremental Updates

With the help of *incremental updates* a lot of duplicated computations can be avoided, when the used cDNA/EST and/or protein databases have been updated. See Section 4 for details.

[1] The idea for this paragraph was shamelessly stolen from the nice user guide of *HMMER* (`http://hmmer.wustl.edu/`).

[2] The name was suggested by Volker Brendel, because the method can be seen as a spliced "threading" of ESTs with a genomic template, allowing gene structure predictions to be computed.

Highly Parameterized

GenomeThreader is highly parameterized. That is, you can set many of the internal parameters via command line option to adjust the program to your personal gene prediction needs.

1.1 The parts of *GenomeThreader*

When referring to *GenomeThreader* we mean the collection of gene prediction tools with this name. Whereas `gth` (in typewriter font) denotes the most important tool in this collection which computes the gene structure predictions. Besides `gth`, there are the following tools available:

1. `gthconsensus` computes consensus spliced alignments using intermediate files.
2. `gthsplit` splits intermediate files.
3. `gthgetseq` gets FASTA sequences from intermediate files.
4. `gthfilestat` show statistics of spliced alignments contained in intermediate files.
5. `gthbssmfileinfo` prints information about BSSM [3].
6. `gthbssmtrain` trains a BSSM.
7. `gthbssmbuild` builds a BSSM file.
8. `gthclean.sh` removes all indices.

1.2 Structure of the Manual

In Section 3 it is shown how to use the various options of `gth` to perform gene predictions and in Section 4 it is described how to use `gthconsensus` to process intermediate files produced by `gth`. In the following three sections some tools are explained which are helpful in handling intermediate files. In Section 8 the small tool `gthbssmfileinfo` is introduced. In Section 9 the BSSM training tool `gthbssmtrain` is documented. In Section 10 the tool `gthbssmbuild` to build BSSMs is explained. In Section 11 the shell script `gthclean.sh` is described. In Section 12 it is explained how to construct the indices used by *GenomeThreader*. If you are new to *GenomeThreader* and want to use the program as fast as possible skip this sections in the first run and go directly to the tutorial given in Section 13. At the end of the manual you can find the acknowledgments, the recent changes, the references, and the index.

[3] BSSM stands for *Bayesian Splice Site Model*

2 Installation

Extracting the *GenomeThreader* distribution for your platform gives you a directory named after your distribution. For your convenience you should extend your `PATH` variable by its `bin` subdirectory.

To be able to use all features of *GenomeThreader* you have to make sure that the subdirectories `bssm` and `gthdata` remain in the same directory as the executables or you have to set two environment variables, namely `BSSMDIR` and `GTHDATADIR`, pointing to the subdirectories `bssm` and `gthdata`.

If one uses the `csh` or the `tcsh` shell, the definition of the environment variables could look like this:

```
$ setenv BSSMDIR "$HOME/gth-1.6.1-Linux_i686-32bit/bin/bssm"
$ setenv GTHDATADIR "$HOME/gth-1.6.1-Linux_i686-32bit/bin/gthdata"
```

For the `bash` or the `sh` the definitions could look like:

```
$ export BSSMDIR="$HOME/gth-1.6.1-Linux_i686-32bit/bin/bssm"
$ export GTHDATADIR="$HOME/gth-1.6.1-Linux_i686-32bit/bin/gthdata"
```

One can also specify more then one directory. In this case, they have to be separated by a colon in the according environment variable definition.

To disable file locking in *GenomeThreader* (not recommended), set the environment variable `GTHNOFLOCK` to any value. In `csh` or `tcsh`, this would look like this:

```
$ setenv GTHNOFLOCK "yes"
```

In `bash` or `sh` like this:

```
$ export GTHNOFLOCK="yes"
```

The `GTHNOFLOCK` environment variable should only be used, if one experiences problems with file locking. This may happen if a Network File System (NFS) is used.

3 `gth`: Computing Gene Predictions

`gth` is called as follows:

`gth` [*options*] `-genomic` *genseqfiles* `-cdna` *cdnafiles* `-protein` *proteinfiles*

Here *genseqfiles* denotes the input files containing the genomic template, *cdnafiles* denotes the input files containing cDNAs/ESTs sequences, and *proteinfiles* the input files containing protein sequences. `-cdna` and `-protein` do not have to be used simultaneously. For the input files indices have to be constructed, either by `mkvtree` or automatically, as described in Section 12. If an error occurs during the computation of *GenomeThreader*, the program exits with error code 1. Otherwise, the exit code is 0. All available *options* are explained below. They have been divided into different categories for clarity. An overview of the option categories with a short one-line description of each option is given in Table 1.

Table 1: Overview of the `gth`-Options sorted by Categories

Input Options	
`-genomic`	specify input files containing genomic sequences
`-cdna`	specify input files containing cDNA/EST sequences
`-protein`	specify input files containing protein sequences
Parameter Files	
`-species`	specify species to select splice site model
`-bssm`	read bssm parameter from file
`-scorematrix`	read amino acid substitution scoring matrix from file
`-translationtable`	set the codon translation table
Strand Direction	
`-f`	analyze only forward strand of genomic sequences
`-r`	analyze only reverse strand of genomic sequences
`-cdnaforward`	align only forward strand of cDNAs
Genomic Sequence Positions	
`-frompos`	analyze genomic sequence from this position
`-topos`	analyze genomic sequence to this position
`-width`	analyze only this width of genomic sequence
Output	
`-v`	be verbose
`-xmlout`	show output in XML format
`-gff3out`	show output in GFF3 format
`-md5ids`	show MD5 fingerprints as sequence IDs
`-o`	redirect output to specified file
`-gzip`	gzip compressed output file
`-bzip2`	bzip2 compressed output file
`-force`	force writing to output file
`-skipalignmentout`	skip output of spliced alignments
`-mincutoffs`	show full spliced alignments
`-showintronmaxlen`	set the maximum length of a fully shown intron
`-minorflen`	set the minimum length of an ORF to be shown
`-startcodon`	require than an ORF must begin with a start codon
`-finalstopcodon`	require that the final ORF must end with a stop codon
`-showseqnums`	show sequence numbers in output
`-pglgentemplate`	show genomic template in PGL lines
`-gs2out`	output in old GeneSeqer2 format
Data Preprocessing	
`-maskpolyatails`	mask poly(A) tails in cDNA/EST files
`-proteinsmap`	specify smap file used for protein files
`-noautoindex`	do not create indices automatically
`-createindicesonly`	stop program flow after the indices have been created
`-skipindexcheck`	skip index check (in preprocessing phase)

	Similarity Filter
-minmatchlen	specify minimum match length (cDNA matching)
-seedlength	specify the seed length (cDNA matching)
-exdrop	specify the Xdrop value for edit distance
-prminmatchlen	specify minimum match length (protein matches)
-prseedlength	specify seed length (protein matching)
-prhdist	specify Hamming distance (protein matching)
-online	run the similarity filter online
-inverse	invert query and index in vmatch call
-exact	use exact matches
-gcmaxgapwidth	set the maximum gap width for global chains
-gcmincoverage	set the minimum coverage of global chains
-paralogs	compute paralogous genes (different chaining procedure)
-enrichchains	enrich genomic sequence part of global chains with additional matches
	Intron Cutout Technique
-intron cutout	enable the intron cutout technique
-fastdp	use jump table to increase speed of DP calculation
-autointroncutout	set the automatic intron cutout matrix size
-icinitialdelta	set the initial delta used for intron cutouts
-iciterations	set the number of intron cutout iterations
-icdeltaincrease	set the delta increase during every iteration
-icminremintronlen	set the minimum remaining intron length
	U12-type Intron Model
-nou12intronmodel	disable the U12-type intron model
-u12donorprob	set the probability for perfect U12-type donor
-u12donorprob1mism	set the prob. for U12-type donor w. 1 mismatch
	Basic DP Algorithm
-probies	set the initial exon state probability
-probdelgen	set the genomic sequence deletion probability
-identityweight	set the pairs of identical characters weight
-mismatchweight	set the weight for mismatching characters
-undetcharweight	set the weight for undetermined characters
-deletionweight	set the weight for deletions
	Short Exon/Intron Parameters
-dpminexonlen	set the minimum exon length for the DP
-dpminintronlen	set the minimum intron length for the DP
-shortexonpenal	set the short exon penalty
-shortintronpenal	set the short intron penalty
	Special Parameters DP Algorithm
-wzerotransition	set the zero transition weights window size
-wdecreasedoutput	set the decreased output weights window size
	Processing of “raw” spliced alignments
-leadcutoffsmode	set the cutoffs mode for leading bases
-termcutoffsmode	set the cutoffs mode for terminal bases
-cutoffsminexonlen	set the cutoffs minimum exon length
-scoreminexonlen	set the score minimum exon length
	Advanced Similarity Filter Option
-minaveragessp	set the minimum average splice site prob.
	Spliced Alignment Filter
-minalignmentscore	set the minimum alignment score
-maxalignmentscore	set the maximum alignment score
-mincoverage	set the minimum coverage
-maxcoverage	set the maximum coverage
-intermediate	stop after calc. of spliced alignments
-sortags	sort alternative gene structures

-sortagswf	set the weight factor for the sorting of AGSs
-exondistri	show the exon length distribution
-introndistri	show the intron length distribution
-refseqcovdistri	show the reference sequence coverage distribution
-first	set the maximum number of spliced alignments
-help	show basic options and exit
-help+	show all options and exit
-version	display version information and exit

3.1 Input Options

-genomic *genseqfiles*

genseqfiles denotes the input files containing the genomic sequences, for which the gene prediction is to be computed.

-cdna *cdnafiles*

cdnafiles denotes the input files containing the cDNAs/ESTs which are spliced aligned to the genomic sequences.

-protein *proteinfiles*

proteinfiles denotes the input files containing the protein sequences which are spliced aligned to the genomic sequences.

The option -genomic is mandatory. Furthermore, at least one of the options -cdna and -protein has to be used.

The names of the input files are separated by white spaces. We support the following formats for the input files. They are recognized according to the first non-white space symbol in the file.

multiple FASTA format If the file begins with the symbol >, then this file is considered to be a file in multiple FASTA format (i.e. it contains one or more sequences). Each line starting with the symbol > contains the *description* of the sequence following it.

multiple EMBL/SWISSPROT format If the file begins with the string ID, then this file is considered to be a file in multiple EMBL format (i.e. containing one or more sequences, each in EMBL-format). The information contained in the ID and DE-lines is taken as the *description* of the corresponding sequence. The EMBL format is identical to the SWISSPROT format (w.r.t. the information we need to extract from such entries). So one can also use files in multiple SWISSPROT format as input.

multiple GENBANK format If the file begins with the string LOCUS, then this file is considered to be a file in multiple GENBANK format (i.e. containing one or more entries in GENBANK-format). The information contained in the LOCUS and the DEFINITION-lines is taken as the *description* of the corresponding sequence.

plain format If the file does not begin with the symbol > or the strings `ID` or `LOCUS`, then the file is taken verbatim. That is, the entire file is considered to be the input sequence (white spaces are *not* ignored).

3.2 Parameter File Options

If no BSSM parameter file is specified by one of the following two options, the generic splice site model is used. BSSM stands for *Bayesian Splice Site Model*.

`-species` *speciesname*
: Use precomputed BSSM parameter file for species with name *speciesname*, according to Table 3. `gth` searches for the file in the directory where it is run and in the directory specified by the environment variable `BSSMDIR`. How to set `BSSMDIR` is explained in Section 2.

`-bssm` *paramfileprefix*
: Load the BSSM parameter file *paramfileprefix*`.bssm`. If *paramfileprefix*`.bssm` is not in the current directory, the directories given by `BSSMDIR` are searched for *paramfile*`.bssm`.

`-scorematrix` *scorematrixname*
: Use the amino acid substitution matrix given in the file *scorematrixname* for spliced alignments with protein sequences. The default score matrix file is `BLOSUM62`. `gth` searches for the file *scorematrixname* in the directory where it is run and in the directory specified by the environment variable `GTHDATADIR`. How one can set `GTHDATADIR` is explained in Section 2.

`-translationtable` t
: Set the codon translation table t used in the matching, DP, and output phase. t must be a number in the range $[1, 23]$ except for 7, 8, 17, 18, 19 and 20. Table 2 gives the possible numbers and their names. The codon translation tables were taken from the website `ftp://ftp.ncbi.nih.gov/entrez/misc/data/gc.prt`.

Option `-species` and option `-bssm` exclude each other. If two excluding options are used together, an error is thrown.

All BSSM parameter files have the extension `.bssm`.

It is highly recommended to use an adequate BSSM parameter file via the option `-species` or `-bssm`, if one is available. Because it is most likely that using a BSSM parameter file for the species under consideration (or a cognate one) will yield in better gene predictions. If no such file is available, feel free to contact us.

If possible, one should prefer `-species` over `-bssm`, because in this case additional internal parameters can be adjusted for the given species.

1	Standard
2	Vertebrate Mitochondrial
3	Yeast Mitochondrial
4	Mold Mitochondrial; Protozoan Mitochondrial; Coelenterate Mitochondrial; Mycoplasma; Spiroplasma
5	Invertebrate Mitochondrial
6	Ciliate Nuclear; Dasycladacean Nuclear; Hexamita Nuclear
9	Echinoderm Mitochondrial
10	Euplotid Nuclear
11	Bacterial
12	Alternative Yeast Nuclear
13	Ascidian Mitochondrial
14	Flatworm Mitochondrial
15	Blepharisma Macronuclear
16	Chlorophycean Mitochondrial
21	Trematode Mitochondrial
22	Scenedesmus Obliquus Mitochondrial
23	Thraustochytrium Mitochondrial

Table 2: The possible codon translation table numbers and table names.

#	Species	Filename	Class	GT	GC	AG
1	*Homo sapiens*	`human`	2	$\surd$	–	$\surd$
2	*Mus musculus*	`mouse`	2	$\surd$	–	$\surd$
3	*Rattus norvegicus*	`rat`	7	$\surd$	–	$\surd$
4	*Gallus gallus*	`chicken`	7	$\surd$	–	$\surd$
5	*Drosophila*	`drosophila`	7	$\surd$	–	$\surd$
6	*Caenorhabditis elegans*	`nematode`	7	$\surd$	–	$\surd$
7	*Schizosaccharomyces pombe*	`fission_yeast`	7	$\surd$	–	$\surd$
8	*Aspergillus*	`aspergillus`	7	$\surd$	–	$\surd$
9	*Arabidopsis thaliana*	`arabidopsis`	7	$\surd$	$\surd$	$\surd$
10	*Zea mays*	`maize`	7	$\surd$	$\surd$	$\surd$
11	*Medicago truncatula*	`medicago`	7	$\surd$	–	$\surd$
12	*Oryza sativa*	`rice`	7	$\surd$	$\surd$	$\surd$

Table 3: Available BSSM parameter files: *GenomeThreader* comes with BSSM parameter files for twelve different species. All parameter files have the extension `.bssm`. *Class* denotes the Bayesian classification model: Either a Bayesian two classification model or a Bayesian seven classification model was used for calculation of the parameter (as described in [BXZ04]). A $\surd$ in the GT, GC, and AG columns means that the corresponding BSSM file contains a model for GT donor, GC donor, or AG acceptor sites, respectively.

3.3 Strand Direction Options

`-f`
: Analyze only the forward strand of the genomic sequences.

`-r`
: Analyze only the reverse strand of the genomic sequences.

`-cdnaforward`
: Align only forward strand of cDNAs.

Option `-f` and option `-r` exclude each other. If neither `-f` nor `-r` is used, both strands of the genomic sequences are analyzed.

3.4 Genomic Sequence Positions Options

Positions in the genomic sequence begin with 1. The following options are available to specify a region of the genomic sequence for which the gene prediction should be computed. They are only applicable if exactly one genomic sequence is given.

`-frompos` i
: Specify first position i of a genomic region to analyze. The parameter i must be a positive integer.

`-topos` j
: Analyze genomic sequence up to (and including) position j, whereas j must be a positive integer.

`-width` w
: Analyze only width w of genomic sequence, whereas w must be a positive integer.

If option `-topos` or `-width` are used, the option `-frompos` is required and the other way round, that is, `-topos` and `-width` exclude each other. If the parameters do not specify a substring of the genomic sequence, then an error is thrown.

Furthermore, if option `-frompos` is used the option `-inverse` is set automatically. This may lead to a large memory consumption which can be avoided by putting the desired part of the genomic sequence in a separate file and running `gth` without the options `-frompos` and `-inverse`.

3.5 Output Options

The following options concern the output of *GenomeThreader*:

`-v`
: Be verbose, that is, give reports about the different steps as well as the resource requirements of the computation.

`-xmlout`
: Shows the output in XML format. This can be useful, if one wants to post-process the output files.

`-gff3out`
: Shows the output in GFF3 format. Either the spliced alignments (if option `-intermediate` is used) or the consensus spliced alignments (if option `-skipalignmentout` is used) are shown.

`-md5ids`
: Show MD5 fingerprints as sequence IDs. This makes subsequent mapping of the annotation to the actual sequences less error-prone.

`-o` *outputfile*
: Redirect output to specified file *outputfile*.

`-gzip`
: Compress the output file given by `-o` with `gzip`.

`-bzip2`
: Compress the output file given by `-o` with `bzip2`.

`-force`
: Forces writing to the output file *outputfile* given by `-o`. By default, writing to *outputfile* is only performed if the file does not exist already.

`-skipalignmentout`
: Skip the output of spliced alignments.

`-mincutoffs`
: The complete spliced alignments are shown. That is, on either side of the alignment only insertions and introns are cut off. This option has the same effect as using `-leadcutoffsmode MINIMAL -termcutoffsmode MINIMAL`.

`-showintronmaxlen` *maxintronlen*
: Sets the maximal length of an intron to be shown completely. If an intron is larger than *maxintronlen*, it is shown in an abbreviated form. Set to 0 to show all introns completely regardless of their lengths.

`-minorflength` *o*

Sets the minimum length of an open reading frame to be shown. Thereby, o denotes the number of amino acids which must be an integer value greater 0. The default value is 64.

`-startcodon`

Require than an ORF must begin with a start codon.

`-finalstopcodon`

Require that the final ORF must end with a stop codon.

`-showseqnums`

Show the sequence numbers in output. That is, in the output lines describing the sequences used in a spliced alignment an additional tag is added giving the number of the sequence in the corresponding sequence file. Sequences are numbered from 0 on. This may be useful when *GenomeThreader* output is postprocessed.

`-pglgentemplate`

Show genomic template in PGL lines. The default is `yes`. Switch off with `-pglgentemplate no` for backward compatibility.

`-gs2out`

Output is shown in the format of the program *GeneSeqer2*, which is a predecessor of *GenomeThreader*. We do not recommend to use this option. It is available for compatibility. For example, all wildcards (S, Y, W, R, K, V, B, D, H and M) of the input sequences are replaced by the wildcard N.

If the options `-o` and `-v` are invoked together: The additional output produced by the option `-v` is not redirected to the file *outputfile* given by the option `-o`. This allows to save the results of the computation in a file while watching its progress on stdout.

3.6 Data Preprocessing

This options affect the preprocessing of the input data, for a detailed discussion see Section 12. This is done by calling `mkvtree`, which is part of the *Vmatch* package (see `http://vmatch.de/`), internally.

`-maskpolyatails`

When this option is used, all poly(A) tails and poly(T) heads in cDNA/EST reference sequences are masked automatically. To be able to predict gene structures correctly it is very important that poly(A) tails are masked!

`-proteinsmap` smapfile

Specifies the smap file *smapfile* used for the (automatic) index construction of protein files. That is, internally `mkvtree` is called with option `-smap` *smapfile* (see the *Vmatch* manual for details). If this option is not used, `mkvtree` is called internally with its option `-protein`.

`-noautoindex`

This option disables the automatic construction of the necessary indices. That is, it is assumed that they have been created manually beforehand using `mkvtree`. See Section 12 for details.

If you use `gth` with option `-createindicesonly` (instead of `mkvtree`) to construct the indices, use the option `-skipindexcheck` and not this option.

`-createindicesonly`

This option stops the program flow after the indices have been created (that is, after the preprocessing phase). This is useful if one wants to let multiple instances of *GenomeThreader* run on the same data set simultaneously without interfering each other during the index construction.

`-skipindexcheck`

This option skips the index checks. That is, it is not checked anymore if an index exists or is corrupted. This is useful if one lets multiple instances of *GenomeThreader* run on indices created with `-createindicesonly`. This option speeds up the preprocessing phase, especially if one uses multiple indices over a network file system. To ensure the correct functioning of this option, one has to use the same type of input data (that is, using `-cdna` and/or `-protein` in a similar way) and use `-maskpolyatails`, `-online`, and `-inverse` similarly in both index construction and splice alignment computation. If you want to use this option together with `-frompos`, pass option `-inverse` to the corresponding `-createindicesonly` call.

The options `-maskpolyatails` and `-noautoindex` as well as `-proteinsmap` and `-noautoindex` exclude each other. Furthermore, the option `-createindicesonly` excludes using the option `-noautoindex` or `-skipindexcheck` (and the other way around).

3.7 Options of the Similarity Filter

The following options are used to compute the similarity regions, that is, the regions in the genomic sequence which are similar to the cDNAs/ESTs and/or proteins. This is done by calling *Vmatch* (`http://vmatch.de/`) internally.

Vmatch matches a sequence file called *query* against a persistent index named *subject* of another sequence file. When matching a cDNA/EST file against a genomic

file, the default is to use the cDNAs/ESTs as query and the genomic sequences as subject. For proteins it is the other way round.

`-minmatchlen` ℓ
: Specify the length value ℓ for the initial matches used in the similarity filter for cDNA/EST matching. The default value is 20.

`-seedlength` m
: Set the length m of the exact seeds used for cDNA/EST matching. m must be a positive integer. The default value is 18.

`-exdrop` x
: Specify the $Xdrop$-score x when extending a seed in both directions allowing for matches, mismatches, insertions, and deletions. The argument x must be a positive integer smaller or equal to 255. Matches are scored 2, mismatches are scored -1, and indels are scored -2. The default value is 2. The extension procedure is further explained in the *Vmatch* manual which can be found at `http://vmatch.de/`. The minimum length of the seeds is specified by the argument to option `-seedlength`.

`-prminmatchlen` ℓ
: Specify the length value ℓ for the initial matches used in the similarity filter for protein matching. The default value is 24.

`-prseedlength` m
: Set the length m of the exact seeds used for protein matching. m must be a positive integer. The default value is 10.

`-prhdist` h
: Set the maximum Hamming distance h a protein match is allowed to have. The default value is 4.

`-online`
: Run online algorithms to compute initial matches in the similarity filter. In this case the complete index is not needed, except for the original and the transformed input sequences plus the descriptions. Therefore, this option is more space efficient. However, the online algorithms usually run not as fast as the indexed based algorithms.

`-inverse`
: This option only affects cDNA/EST files. Invert subject and query, i.e., use the genomic files as query and the cDNA/EST files as subject.

`-exact`
: Use exact matches in the similarity filter. If this option is invoked it is not possible to use the options `-seedlength` and `-exdrop`, because it would make no sense.

`-gcmaxgapwidth` *gw*
: Set the maximum gap width *gw* for global chains. This width is approximately the same as the maximum intron length in the studies organism. The default is 1000000 which might be a good guess for human. It is very important to set this parameter appropriately!

`-gcmincoverage` *mc*
: Set the minimum coverage a global chain must achieve to be kept. That is, at least this proportion of the according cDNA/EST/protein sequence must be covered by the global chain. Thereby, *mc* is an integer denoting the minimum coverage in percent. The default is 50. This option has a great influences on the number of computed spliced alignments!

`-paralogs`
: By default, the chaining returns all global chains with maximal score (usually 1), if their coverage is higher than the argument *mc* to option `-gcmincoverage`. If this option is used, all non-overlapping global chains with a coverage higher than *mc* are reported. This can be helpful to detect paralogous genes. The total run time usually increases, because more spliced alignments need to be computed.

`-enrichchains`
: Enrich genomic sequence part of global chains with additional matches.

Using option `-inverse` without using option `-online` can lead to a large main memory consumption.

3.8 Intron Cutout Technique Options

In this section the options used for the intron cutout technique are described.

`-intron cutout`
: Enable the intron cutout technique.

`-fastdp`
: Use jump table to increase speed of DP calculation.

`-autointroncutout` *s*
: Enables the automatic intron cutout technique. That is, the intron cutout technique is only used if the Dynamic Programming matrix in the "normal" DP call would be larger than *s* megabytes. Increasing *s* increases the computation time and decreases the probability of wrong gene predictions.

`-icinitialdelta` *d*
: Set the initial delta *d* used for intron cutouts. The parameter *d* must be a positive integer. The default value is 50.

`-iciterations` *i*
: Set the number of iterations *i* the initial delta *d* is increased by the increase delta *di*. Thereby, the first iteration counts, too. That is, in the default setting of 2 *d* is increased once by *di*, resulting in (up to) two iterations. The incrementation is only triggered if the intron cutout technique did not succeed.

`-icdeltaincrease` *di*
: Set the increment *di* for the delta used in the intron cutout technique.

`-icminremintronlen` *r*
: Set the minimum remaining intron length *r* for an intron to be cut out. The parameter *r* must be a positive integer. The default value is 10.

The options `-intron cutout` and `-autointroncutout` exclude each other.

3.9 Advanced Options

This section describes the advanced options. To understand their semantics, a basic understanding of the underlying spliced alignment algorithms is required. For the initial use of *GenomeThreader*, this section can safely be skipped.

3.9.1 Options for U12-type introns

GenomeThreader has a model for U12-type introns built in. That is, donor sites which match the consensus /[AG]TATCCTT (where / denotes the exon end and [AG] indicates A or G) [ZB03] perfectly or with one mismatch get a high probability. The mismatch is only allowed in the last 6 bases of the consensus. See [ZB03] for details on U12-type introns and additional pointers to the literature.

`-nou12intronmodel`
: If this option is used, the U12-type intron model is disabled.

`-u12donorprob` p
: Sets the probability p for perfectly matching U12-type donor sites. The default value is 0.99.

`-u12donorprob1mism` p
: Sets the probability p for U12-type donor sites with exactly one mismatch. The default value is 0.9.

In both cases, p must be a positive floating point value smaller or equal than 1.0.

3.9.2 Basic DP Algorithm Options

With these options all DP parameters can be changed.

`-probies` τ_{e_1}
: Sets the probability that the initial state is an exon state. τ_{e_1} must be a positive floating point value smaller or equal than 1.0. The default value is 0.5.

`-probdelgen` $P_{\Delta g}$
: Sets the probability of a nucleotide deletion in the genomic sequence. τ_{e_1} must be a positive floating point value smaller or equal than 1.0. The default value is 0.03.

`-identityweight` σ
: Sets the weight for pairs of identical characters. σ must be a floating point value. The default value is 2.0.

`-mismatchweight` μ
: Sets the weight for mismatching characters. μ must be a floating point value. The default value is -2.0.

`-undetcharweight` ν
: Sets the weight for alignment positions involving undetermined characters, i.e., involving character N. ν must be a floating point value. The default value is 0.0.

`-deletionweight` δ
: Sets the weight for deletions. δ must be a floating point value. The default value is -5.0.

3.9.3 Short Exon/Intron Parameters

The following options allow to change the parameters for short exons and introns. If an exon is shorter than the minimum exon length ξ, then a penalty for short exons χ is *subtracted* from the actual weight. Analog, if an intron is shorter than the minimum intron length η, then a penalty for short introns ψ is *subtracted* from the actual weight.

`-dpminexonlength` ξ
: Sets the minimum exon length. ξ must be an integer value greater 0. The default value is 5.

`-dpminintronlength` η
: Sets the minimum intron length. η must be an integer value greater 0. The default value is 50.

`-shortexonpenal` χ
: Sets the short exon penalty. χ must be a floating point value greater or equal 0.0. The default value is 100.0.

`-shortintronpenal` ψ
: Sets the short intron penalty. ψ must be a floating point value greater or equal 0.0. The default value is 100.0.

3.9.4 Special Parameters for the DP Algorithm

With these options, the more special DP Parameter can be changed.

`-wzerotransition` ϑ
: Sets the window size for zero transition weights. ϑ must be an integer value greater or equal to 0. The default value is 80.

`-wdecreasedoutput` ω
: Sets the window size for decreased output weights. ω must be an integer value greater or equal to 0. The default value is 80.

3.9.5 Options for Processing of "raw" Spliced Alignments

The following options affect the processing of "raw" spliced alignments after the dynamic programming.

`-leadcutoffsmode` *leadingmode*
: Set the mode for determination of the leading cutoffs. The mandatory argument *leadingmode* can be `RELAXED`, `STRICT`, or `MINIMAL`. The default is `RELAXED`.

`-termcutoffsmode` *terminalmode*
: Set the mode for determination of the terminal cutoffs. The mandatory argument *terminalmode* can be `RELAXED`, `STRICT`, or `MINIMAL`. The default is `STRICT`.

`-cutoffsminexonlen` κ
: Set the minimum length κ an exon must have, such that it is not cut off when the corresponding cut off mode is `STRICT`. The default is 5.

`-scoreminexonlen` λ
: Set the minimum length λ an exon must have, such that it is included in the computation of the overall similarity score of a spliced alignment. The default is 50.

3.9.6 Spliced Alignment Filter

The spliced alignment filter controls which spliced alignments are stored in the internal set of spliced alignment. That is, only these spliced alignments are shown or written to an output file and are used to compute consensus spliced alignments.

When a spliced alignment is computed, its *coverage* is also determined. The coverage of a spliced alignment is the maximum of the the *genomic coverage* and the *reference coverage*. Thereby, the genomic coverage denotes the cumulative exon length divided by the length of the genomic sequence part used for the dynamic programming. The reference coverage denotes the cumulative exon length divided by the length of the reference sequence. If the genomic coverage was maximal a `G` in shown in the output[4]. If the reference coverage was maximal a `C` (for cDNA/EST based spliced alignments) or `P` (for protein based spliced alignments) is shown in the output.

`-minalignmentscore` s
: A spliced alignment must at least an alignment score of s.

`-maxalignmentscore` s
: A spliced alignment is not allowed to have an alignment score higher than s.

`-mincoverage` c
: A spliced alignment must at least a coverage of c.

`-maxcoverage` c
: A spliced alignment is not allowed to have a coverage higher than c.

By default all computed spliced alignments are used.

3.9.7 Advanced Similarity Filter Option

`-minaveragessp` ζ
: Sets the minimum average splice site probability. ζ must be a floating point value in the interval $[0.0, 1.0]$. The default value is 0.5.

`-duplicatecheck` mode
: Set the criterion used to check for spliced alignment duplicates. The mandatory argument *mode* can be `none`, `id`, `desc`, `seq`, or `both`. The default is `both`.

[4] In the textual output as last part of the `MATCH` line and in the final XML output as the `high_type` attribute.

With the option `-duplicatecheck` the "sameness" criterion of the duplicate check is set. The duplicate check is designed to prevent duplicate alignments (that is, alignments from the "same" cDNA/EST or protein sequence to the same genomic region on the same strand of a particular genomic sequence). Duplicate alignments can occur for two reasons:

1. The chaining algorithm (which chains the matches before the actual spliced alignment computation) produced two overlapping chains in the same genomic region which leads to the same spliced alignments. This is a rather rare event, but it can happen from time to time. For this reason alone, a duplicate check is needed, if we don't want to see spliced alignment duplications.

2. The same cDNA/EST or protein sequence was fed to GenomeThreader more than once. The duplicate check should prevent that it appears in the output multiple times and the "sameness" criterion used to compare the sequences depends on the *mode* supplied to `-duplicatecheck`.

The different duplicate check modes work as follows:

`none`: No duplicate check is done whatsoever. Only useful for testing purposes.

`id`: The duplicate check is based on the "ID" of the cDNA/EST or protein sequence. The "ID" is parsed from the sequence description as follows: leading '`gi|`', '`SQ;`', '`(gi|`' , '`ref|`' are dropped and the part until the first '`:`', '`|`', ' ' or '`\t`' is stored. This can cause problems if the description is empty or starts with a prefix that is not recognized. This was the behavior up to (and including) version 1.4.6.

`desc`: In contrast to the `ID` mode, the complete description is used to compare sequences. This can cause problem if the description is empty or ambiguous.

`seq`: The actual sequences is used to determine, if two cDNA/EST or protein sequences are the same. This would lead to the exclusion of cDNA/EST or protein sequences where the actual sequence equals but the descriptions differ (for example, equal ESTs sequenced independently from another and therefore with different descriptions).

`both`: This mode combines the `desc` and `seq` modes. That is, a cDNA/EST or protein sequences is only considered equal to another one, if the complete descriptions and the actual sequences are equal. This is the default (from version 1.4.7 onwards).

3.9.8 Interrupt Option

With the following option it is possible to perform the incremental updates which are described in Section 4.

`-intermediate`
: If this option is invoked, the dataflow of `gth` is interrupted after the output of the spliced alignments.

This option implies option `-xmlout` since the intermediate results are stored in an XML format. You should save the intermediate results using option `-o` instead of redirecting them to a file, because this allows for an internal check which ensures that the intermediate output reflects the spliced alignments stored in main memory. Do not process the intermediate XML output yourself, use the "normal" XML output instead!

3.9.9 Options for Postprocessing of Predicted Gene Locations

With the following options it is possible to postprocess the *predicted gene locations* (PGLs), also referred to as *consensus spliced alignments*.

`-sortags`
: If this option is invoked, the alternative gene structures (AGSs) of every PGL are sorted according to the so-called *overall score*. Every AGS has an exon score for every contained exon and a donor and a acceptor site probability for every contained intron (if any). The overall score o of an AGS is computed as follows:

  - If the AGS consists of exactly one exon, the overall score simply equals the exon score.
  - Otherwise, the average exon score e and the average spliced site probability s is calculated. Then

    $$o = \frac{w \cdot e + s}{w + 1.0},$$

    whereas w is the *weight factor* (see option below).

`-sortagswf` *wf*
: Set the weight factor *wf* for the calculation of the overall score (see option above). *wf* must be a floating point value larger than 0.0. If *wf* is set to a value larger than 1.0, the average exon score gets more weight in the calculation of the overall score. If *wf* is set to a value smaller than 1.0, the average splice site probability gets more weight. The default value is 1.0.

If option `-sortagswf` is used, option `-sortags` is required.

3.9.10 Statistical Options

In this section the options are described which yield in the output of additional statistical information at the end of a program run.

`-exondistri`
: Show the exon length distribution at the end of the `gth` output.

`-introndistri`
: Show the intron length distribution at the end of the `gth` output. This option might be useful to get a feeling for a good setting of the option `-gcmaxgapwidth`.

`-refseqcovdistri`
: Show the reference sequence coverage distribution at the end of the `gth` output. This option might be useful to get a feeling for a good setting of the option `-gcmincoverage`.

3.9.11 Miscellaneous Options

`-first` n
: The positive integer n specifies the maximum number of computed spliced alignments per genomic DNA input. The default value is 0, which leads to the computation of all sensible spliced alignments.

`-help`
: Show a summary of the basic options. That is, only the most common options. Afterwards terminate.

`-help+`
: Show a summary of all options and terminate.

`-version`
: Show version number and built-date of *GenomeThreader*. Afterwards terminate.

If one of the last three options is used, `gth` terminates with exit code 0 after showing the corresponding output.

4 `gthconsensus`: Incremental Updates

With the help of `gthconsensus` one can perform so-called *incremental updates* of cDNA/EST and protein databases[5]:

[5] To simplify the explanation, in the following we mention only cDNA/EST databases, but everything said also applies to protein databases.

In a typical application of `gth` one uses a cDNA/EST database to annotate a genomic sequence. As a result of the ongoing sequencing efforts it is very likely that a few weeks after such an annotation the used cDNA/EST database is not up-to-date anymore, because new suitable cDNAs/ESTs are available. The conventional approach was to rerun the whole annotation process using an updated database. This has the drawback that one repeats a lot of spliced alignment calculations of the cDNAs/ESTs which have already been in the database before the update while computing typically only a few new spliced alignments of the added cDNAs/ESTs.

This observation leads to the idea to split the annotation process into two parts, such that one can reuse the already computed spliced alignments (that is, to perform incremental updates):

1. In one part one performs the computations which are independent of each other only once and stores the intermediate results.
2. In the other part the stored results are used to perform the calculations which depend on one another.

The first part refers to the calculation of spliced alignments (or predicted gene structures) and is realized in `gth` by the option `-intermediate` (see Section 3.9.8). If these results are stored in a file, we call it an *intermediate file*.

The second part refers to the calculation of *consensus* spliced alignments (or predicted gene locations) and is realized by the program `gthconsensus`, which processes a set of intermediate files. In the following section the application of `gthconsensus` is explained. An example session using `gth -intermediate` and `gthconsensus` is shown in Section 13.

4.1 The Options of `gthconsensus`

`gthconsensus` is called as follows:

`gthconsensus` [*options*] *intermediate_files*

Here *intermediate_files* denotes a list of one or more files containing XML intermediate output produced by `gth` invocations using the option `-intermediate`. The *options* available for `gthconsensus` are a subset of the options available for `gth`. To find out which options are included in the subset try:

`gthconsensus -help`

Basically all options are included into `gthconsensus` which make sense for the purpose of it. If you think a useful option is missing in `gthconsensus`, feel free to contact the author.

5 `gthsplit`: Split Intermediate Files

With the help of `gthsplit` one can split *GenomeThreader* output files containing intermediate results into multiple sets according to different criteria. `gthsplit` is called as follows:

`gthsplit` [*options*] *intermediate_files*

If no intermediate file is given as input, stdin is used instead. This allows to use `gthsplit` in a UNIX pipe.

`gthsplit` offers the following options:

`-alignmentscore`
: If this option is used, the input files are split according to the overall alignment score (scr).

`-coverage`
: If this option is used, the input files are split according to the coverage (cov).

`-range` $\Re$
: Set the percentage range $\Re$ used to create the sets. Each set contains the spliced alignments where the corresponding percentage (i.e., the alignment score or the coverage) is greater or equal then the lower set bound and lower then the higher bound. Spliced alignments with a percentage of 100% go into the last set. $\Re$ must divide 100 without rest. The default range is 5.

Furthermore, the options `-v`, `-gzip`, `-bzip2`, `-force`, and `-help` are available and have the same semantic as in `gth`.

The spliced alignment filter options described in Section 3.9.6 can also be used to reduce the set of spliced alignment accordingly before splitting it.

5.1 Applying `gthsplit`

The following examples show how to use `gthsplit` and its output:

```
$ gthsplit -alignmentscore -v -gzip U89959.inter.gz
$ process all intermediate output files
$ process file 1/1: U89959.inter.gz
$ split file created: U89959.inter.scr70-75.gz (size=1)
$ split file created: U89959.inter.scr80-85.gz (size=11)
$ split file created: U89959.inter.scr85-90.gz (size=18)
$ split file created: U89959.inter.scr90-95.gz (size=25)
$ split file created: U89959.inter.scr95-100.gz (size=144)
```

5.2 The Script `gthsplit2dim.sh`

With the shell script `gthsplit2dim.sh` on can split up an intermediate file in both dimensions (i.e., along the alignment score and the coverage) at the same time. It offers the following options:

`-r` ℜ
: Same as `-range` when `gthsplit` is used.

`-f`
: Same as `-force` when `gthsplit` is used.

`-v`
: Same as `-v` when `gthsplit` is used.

`-g`
: Same as `-gzip` when `gthsplit` is used.

`-b`
: Same as `-bzip2` when `gthsplit` is used.

5.3 Applying `gthsplit2dim.sh`

The following example shows how to use `gthsplit2dim.sh` and its output:

```
$ gthsplit2dim.sh -r 10 -v -g ceres_full.inter.gz
$ split according to alignment score
$ process all intermediate output files
$ process file 1/1: ceres_full.inter.gz
$ split file created: ceres_full.inter.scr70-80.gz (size=30)
$ split file created: ceres_full.inter.scr80-90.gz (size=102)
$ split file created: ceres_full.inter.scr90-100.gz (size=1310)
$ split according to coverage
```

6 `gthgetseq`: Get FASTA Sequences

With the help of `gthgetseq` one can get the used FASTA sequences from *GenomeThreader* output files containing intermediate results. That is, all sequences which are represented in the intermediate files are printed in FASTA format on stdout. Strictly speaking, the sequences are not extracted from the intermediate file, but from the corresponding input sequence files. `gthgetseq` is called as follows:

`gthsplit` [*options*] `-getcdna` | `-getprotein` | `-getgenomic` *intermediate_files*

If no intermediate file is given as input, stdin is used instead. This allows to use `gthgetseq` in a UNIX pipe. `gthgetseq` offers the following options:

`-getcdna`
: Get cDNA/EST sequences used in the spliced alignments contained in the given intermediate files.

`-getcdnacomp`
: Get cDNA/EST sequences *not* used in the spliced alignments contained in the given intermediate files. That is, the *complement* of `-getcdna`.

`-getprotein`
: Get protein sequences used in the spliced alignments contained in the given intermediate files.

`-getproteincomp`
: Get protein sequences *not* used in the spliced alignments contained in the given intermediate files. That is, the *complement* of `-getprotein`.

`-getgenomic`
: Get genomic sequences used in the spliced alignments contained in the given intermediate files.

`-getgenomiccomp`
: Get genomic sequences *not* used in the spliced alignments contained in the given intermediate files. That is, the *complement* of `-getgenomic`.

Furthermore, the options `-gzip`, `-bzip2`, and `-help` are available and have the same semantic as in `gth`.

The spliced alignment filter options described in Section 3.9.6 can also be used to extract sequences only from spliced alignments which pass the filter.

At least one of the options `-getcdna`, `-getcdnacomp`, `-getprotein`, `-getproteincomp`, `-getgenomic`, `-getgenomiccomp` is mandatory.

6.1 Applying `gthgetseq`

Lets assume we have an intermediate file called `ceres_full.inter.gz`. To get all cDNAs which led to a spliced alignment with a maximum alignment score of 0.75, we would issue the following command:

```
$ gthgetseq -getcdna -gzip -maxalignmentscore 0.75 ceres_full.inter.gz
>7894
accactacaaccaccgcaacaaccaccaaaaaccctctcaaagaaatttctttttttttct
tactttcttggtttgtcaaatatggtcagccatccaatggagaaagctgcaaatggtgcg
tctgcgttggaaacgcagacgggtgagttagatcagccggaacggcttcgtaagatcata
tcggtgtcttccattgccgccggtgtacagttcggttgggctttacagttatctctgttg
actccttacgtgcagctactcggaatcccacataaatgggcttctctgatttggctctgt
ggtccaatctccggtatgcttgttcagcctatcgtcggttaccacagtgaccgttgcacc
tcaagattcggccgtcgtcgtcccttcatcgtcgctggagctggtttagtcaccgttgct
```

```
gttttccttatcggttacgctgccgatataggtcacagcatgggcgatcagcttgacaaa
ccgccgaaaacgcgagccatagcgatattcgctctcgggttttggattcttgacgtggct
aacaacaccttacaaggaccctgcagagctttcttggctgatttatcagcagggaacgct
aagaaaacgcgaaccgcaaacgcgtttttctcgtttttcatggcggttggaaacgttttg
ggttacgctgcgggatcttacagaaatctctacaaagttgtgcctttcacgatgactgag
tcatgcgatctctactgcgcaaacctcaaaacgtgttttttcctatccataacgcttctc
ctcatagtcactttcgtatctctctgttacgtgaaggagaagccatggacgccagagcca
acagccgatggaaaagcctccaacgttccgtttttcggagaaatcttcggagctttcaag
gaactaaaaagacccatgtggatgcttcttatagtcactgcactaaactggatcgcttgg
ttccctttccttctcttcgacactgattggatgggccgtgaggtgtacggaggaaactca
gacgcaaccgcaaccgcagcctctaagaagctttacaacgacggagtcagagctggtgct
ttggggcttatgcttaacgctattgttcttggtttcatgtctcttggtgttgaatggatt
ggtcggaaattgggaggagctaaaaggctttggggtattgttaacttcatcctcgccatt
tgcttggccatgacggttgtggttacgaaacaagctgagaatcaccgacgagatcacggc
ggcgctaaaacaggtccacctggtaacgtcacagctggtgctttaactctcttcgccatc
ctcggtatcccccaagccattacgtttagcattccttttgcactagcttccatattttca
accaattccggtgccggccaaggactttccctaggtgttctgaatctagccattgtcgtc
cctcagatggtaatatctgtgggaggtggaccattcgacgaactattcggtggtggaaac
attccagcatttgtgttaggagcgattgcggcagcggtaagtggtgtattggcgttgacg
gtgttgccttcaccgcctccggatgctcctgccttcaaagctactatgggatttcattga
attttagcagtggttgtttggctctctttctctcataaaacagtagtgttgtgcaaatcc
tacataaagaaaaaagaaaaggaaattaaactcattgggttggtttgtattttacctaaa
cccacgaagttccttttttcttttgtaactcaatttaaatttggagtatattttacttttt
tgc
```

To store the output in a file `cdna`, one can redirect the output like this:

```
$ gthgetseq -getcdna -gzip -maxalignmentscore 0.75 ceres_full.inter.gz > cdna
```

7 `gthfilestat`: Show Statistics

`gthfilestat` shows statistics about spliced alignments in *GenomeThreader* output files containing intermediate results. This might be helpful in getting an overview of a set of intermediate files, before they are processed further, for example, with `gthsplit`.

`gthfilestat` [*options*] *intermediate_files*

If no intermediate file is given as input, stdin is used instead. This allows to use `gthfilestat` in a UNIX pipe.

Besides the options `-v` and `-help`, the spliced alignment filter options described in Section 3.9.6 are available. The semantic is the same as in `gth`.

7.1 Applying `gthfilestat`

The following example shows how to use `gthfilestat` and its output:

```
$ gthfilestat ceres_sub.inter.gz
spliced alignment alignment score distribution:

spliced alignment coverage distribution:

memory statistics:
5 spliced alignments have been stored
5 predicted gene locations have been stored

date finished: 2008-11-18 11:32:05
```

8 gthbssmfileinfo: BSSM File Information

With gthbssmfileinfo one can print out information about BSSM files. It is called as follows:

gthbssmfileinfo *bssm_file*

gthbssmfileinfo uses the directories specified by the environment variable BSSMDIR to look for the specified *bssm_file*. For example, printing information about the human BSSM file works like this:

```
$ gthbssmfileinfo human
$ the specified BSSM parameter file contains the following models:
$ GT donor sites   = True (two-class)
$ GC donor sites   = False
$ AG acceptor sites= True (two-class)
```

9 gthbssmtrain: Train BSSMs

gthbssmtrain is called as follows:

gthbssmtrain [*options*] -seqfile(s)|-regionmapping *arg GFF3_file*

-outdir *dir*
: Specify the name of the output directory to which the training files are written. The default is training_data.

-gcdonor *arg*
: Extract training data for GC donor sites. The default is yes.

-filtertype *type*
: Set type of features used for filtering (usually exon or CDS). The default is exon.

`-goodexoncount` *n*
: Set the minimum number *n* of good exons a feature must have to be in the training data. The default is 1.

`-cutoff` *s*
: Set the minimum score *s* an exon must have to count towards the "good exon count" (exons without a score count as good). The default is 1.0.

`-extracttype` *type*
: Set type of features to to be extracted as exons (usually `exon` or `CDS`). The default is `CDS`.

`-seqfile` *file*
: Set the sequence file from which to extract the features.

`-seqfiles` *file ...*
: Set the sequence files from which to extract the features. The list of sequence files can be terminated with `--`.

`-matchdesc` *arg*
: Match the sequence descriptions for the desired sequence IDs. The default is `no`.

`-usedesc` *arg*
: Use sequence descriptions to map the sequence IDs (in the GFF3 file) to actual sequence entries. If a description contains a sequence range position (for example, `III:1000001..2000000`), the first part is used as sequence ID ('`III`') and the first range as offset ('`1000001`'). The default is `no`.

`-regionmapping` *file*
: Set file containing sequence-region to sequence file mapping.

`-seed` *s*
: Set seed for random generator manually. 0 generates a seed from the current time and the process id. The default is 0.

One of the options `-seqfile`, `-seqfiles`, or `-regionmapping` is mandatory. The mandatory argument *GFF3_file* must contain the annotation used to train the BSSM in GFF3 format.

9.1 Applying `gthbssmtrain`

The following example shows how to use `gthbssmtrain`.

At first we have to create the annotation data which can be used to train a BSSM. For this purpose, `gth` or `gthconsensus` is called with the options `-gff3out` and

-skipalignmentout. Furthermore, the option -md5ids should be used to make the subsequent sequence ID mapping of the GFF3 file used in gthbssmtrain easier and less error prone. The annotation should be of high quality (that is, all gene predicted gene structures seem to be correct) and contain at least a couple of hundred splice sites. A gth call could look like this (we store the result in a file called arab.gff3):

```
$ gth -genomic U89959_genomic.fas -cdna U89959_ests.fas -gff3out -skipalignmentout -md5ids
```

Afterwards we can train the BSSM with gthbssmtrain using the following command. Thereby, we supply the genomic sequence file used in the annotation to the option -seqfile. If more than one genomic sequence file was used, the option -seqfiles can be used. Because we used -md5ids when we produced the GFF3 file, the mapping from sequence ID works automatically:

```
$ gthbssmtrain -seqfile U89959_genomic.fas arab.gff3
gt-ag:  94.33% (n=133)
gc-ag:   0.71% (n=1)
```

The output of gthbssmtrain shows you how many gt-ag and gc-ag splice sites have been processed. If this number looks wrong, probably something is wrong with the sequence ID mapping. Because we did not set option -outdir, the training data is stored in the default output directory training_data.

We can now use gthbssmbuild (which is described in detail in the next Section) to build the actual BSSM file arab.bssm. We use only the options -gtdonor and -agacceptor, because there were not enough donor sites that it would make sense to build the GC donor model into the BSSM file with -gcdonor:

```
$ gthbssmbuild -gtdonor -agacceptor -datapath training_data -bssmfile arab.bssm
```

We now have a new BSSM file named arab.bssm which can be used in subsequent gth or gthconsensus calls with the option -bssm (without the .bssm suffix), like this:

```
$ gth -bssm arab -genomic U89959_genomic.fas -cdna U89959_ests.fas
```

10 gthbssmbuild: Build BSSM files

With the help of gthbssmbuild one can build a BSSM file from a directory tree containing the training data. The training data is usually generated with gthbssmtrain which is described in the previous section.

gthbssmbuild is called as follows:

gthsplit [*options*] -databasedir *dir* -bssmfile *file*

`-bssmfile` *file*
: Specify the name of the BSSM file *file* to store parameters in.

`-datapath` *dir*
: Specify root of species-specific training data directory tree *dir*.

`-gtdonor`
: Train the `GT` donor model.

`-gcdonor`
: Train the `GC` donor model.

`-agacceptor`
: Train the `AG` acceptor model.

Furthermore, the options `-gzip` and `-help` are available and have the same semantic as in `gth`.

The options `-bssmfile` and `-datapath` are mandatory. Furthermore, at least one of the options `-gtdonor`, `-gcdonor`, and `-agacceptor` is required.

10.1 The BSSM training data directory

A BSSM training data directory has up to three subdirectories named `GT_donor`, `GC_donor`, and `AG_acceptor` which contain the training data files for the corresponding models. Each such directory must contain seven files named `F0`, `F1`, `F2`, `Fi`, `T0`, `T1`, and `T2` (with an additional suffix `.gz` if they are compressed). For example, the directory tree containing the compressed training data for *rice* looks like this:

```
$ ls -R rice
AG_acceptor/ GC_donor/    GT_donor/

rice/AG_acceptor:
F0.gz  F1.gz  F2.gz  Fi.gz  T0.gz  T1.gz  T2.gz

rice/GC_donor:
F0.gz  F1.gz  F2.gz  Fi.gz  T0.gz  T1.gz  T2.gz

rice/GT_donor:
F0.gz  F1.gz  F2.gz  Fi.gz  T0.gz  T1.gz  T2.gz
```

11 `gthclean.sh`: Remove Indices

The script `gthclean.sh` removes all automatically and manually constructed indices in the directory where it is called. Furthermore, all files ending with `.polya`

and `.polya.info` created by using the option `-maskpolyatails` of `gth` are removed. Do not use the endings removed by the script, otherwise the corresponding files will be deleted when `gthclean.sh` is called! The script looks as follows:

12 Construction of the Indices

GenomeThreader uses an persistent index to determine which spliced alignments have to be computed during an initial matching and chaining phase called *similarity filter*.

If `gth` is called the first time on a set of input files (without using the option `-noautoindex`), these indices are constructed automatically. In subsequent calls it is recognized that these indices already exist and the construction phase is skipped. That is, the index construction is completely transparent to the user, the only difference is the reduced execution time when the index already exists.

If the option `-noautoindex` is used, it is assumed that the indices already exist. They have to be created by `mkvtree` beforehand. It is part of the *Vmatch* software suite, which is available on the website `http://vmatch.de`. On this website you can also find a manual for `mkvtree`. For DNA files, indices are constructed as follows (the file is called `dna.fasta`):

```
$ mkvtree -v -dna -allout -pl -db dna.fasta
```

And for protein files like this (the filename is `protein.fasta`):

```
$ mkvtree -v -protein -allout -pl -db protein.fasta
```

If you use DNA reference files, you have to construct the index for the genomic files, if option `-inverse` is not used. And for the reference files, if option `-inverse` is used. If you use protein reference files, the index has to be constructed for the reference files, no matter if option `-inverse` is used or not. You can mix both types of input files.

Input files can be compressed with `gzip`. In this case they must end with `.gz`.

13 Tutorial

In this section we give an tutorial on how to use *GenomeThreader* by presenting some typical uses of *GenomeThreader*. On our walk through the different examples, the most important features and options of *GenomeThreader* are introduced.

13.1 Mapping a Single EST on the *A. thaliana* Chromosome 1

We invoke `gth` in the simplest possible way, just using two mandatory options `-genomic` and `-cdna` to map the EST with gi number 19875482 against the first chromosome of *Arabidopsis thaliana* (gi 42592260). The EST is supplied as file `AU236313.gbk.gz` in GENBANK format and the chromosome in FASTA format (file `NC_003070.fna.gz`). Both files have been compressed with the program `gzip` beforehand, which is indicated by the ending `.gz`.

```
$ gth -genomic NC_003070.fna.gz -cdna AU236313.gbk.gz
$ GenomeThreader 1.2.3 (2008-11-15 02:38:08)
$ Date run: 2008-11-18 11:32:05
$ Arguments: -genomic NC_003070.fna.gz -cdna AU236313.gbk.gz
###############################################################################
EST Sequence: file=AU236313.gbk.gz, strand=+, description=AU236313 AU236313 RAFL14 Arabidopsis thali

    1  GCGTGTTGTT AAAGAGCTCA CTAGTTGGGT GATTTATTCA GAGGAGGATC CGGAAGCTCA
   61  ACAAAGATAT TACTATTGGT CTTATCCAGC GTGAGTTGCT TAGCCTAGCG GAGTACAATG
  121  TCCACATGGC GAAGCATCTT GATGGAGGGA GAAACAAGAC CGCAACTGAC TTTGCTATTT
  181  CTCTACTCCA ATCCTTGGTC ACTGAGGAGT CNAGTGTCAT TTNNTAG

Genomic Template: file=NC_003070.fna.gz, strand=+, from=382240, to=383395, description=gi|42592260|r

Predicted gene structure:

 Exon  1   382540   382567 (  28 n);  cDNA      1     28 (  28 n); score: 1.000
  Intron  1   382568   382802 ( 235 n); Pd: 0.050 (s:    0), Pa: 0.050 (s: 0.95)
 Exon  2   382803   382929 ( 127 n);  cDNA     29    156 ( 128 n); score: 0.980
  Intron  2   382930   383029 ( 100 n); Pd: 0.050 (s: 1.00), Pa: 0.050 (s: 1.00)
 Exon  3   383030   383100 (  71 n);  cDNA    157    227 (  71 n); score: 0.944

MATCH 42592260+ AU236313+ 0.967 226 0.996 C
PGS_42592260+_AU236313+ (382540  382567,382803  382929,383030  383100)

Alignment (genomic DNA sequence = upper lines):

GCGTGTTGTT AAAGAGCTCA CTAGTTGGGT ATGTTTACAA CCTTTTCAAG ATTTCACTTC      382599
|||||||||| |||||||||| ||||||||
GCGTGTTGTT AAAGAGCTCA CTAGTTGG.. .......... .......... ..........          28


GCTGATGTGC TTTGTTCAGT TACTTCTCCA TTAATTTGTC ACTATTTCTG TCAGAACACA      382659

.......... .......... .......... .......... .......... ..........          28


TAGGATAACA CATATCATAT AAGTGCTAGG TCGAGTCTGT TTCCTGTAGT TGGAGCCTAT      382719

.......... .......... .......... .......... .......... ..........          28


CCTCAACTGG TTATAGATAC TAGATTTGTT TCTTTGGTAT TTTTAGTTAT AATTAATTAT      382779

.......... .......... .......... .......... .......... ..........          28


CTTTCTTCAA ACTTTTGACA CAGGTGATTT ATTCAGAGGA GGAT-CGGAA GCTCAACAAA      382838
                         ||||||| |||||||||| |||| ||||| ||||||||||
.......... .......... ...GTGATTT ATTCAGAGGA GGATCCGGAA GCTCAACAAA          65
```

```
GATATTACTA TTGGTCTTAT CCAGCGTGAG TTGCTTAGCC TAGCGGAGTA CAATGTCCAC      382898
|||||||||| |||||||||| |||||||||| |||||||||| |||||||||| ||||||||||
GATATTACTA TTGGTCTTAT CCAGCGTGAG TTGCTTAGCC TAGCGGAGTA CAATGTCCAC         125


ATGGCGAAGC ATCTTGATGG AGGGAGAAAC AGTATGCTGA ATTGCTTAAC CTTTGTTGAT      382958
|||||||||| |||||||||| |||||||||| |
ATGGCGAAGC ATCTTGATGG AGGGAGAAAC A......... .......... ..........         156


GTCCTTGTGT GGTAACATTC TTTTTTTGTT TTGGCTAGTG AACTTGTTTT AACAAGTTGT      383018

.......... .......... .......... .......... .......... ..........         156


TTGTTACGCA GAGACCGCAA CTGACTTTGC TATTTCTCTA CTCCAATCCT TGGTCACTGA      383078
           ||||||||| |||||||||| |||||||||| |||||||||| ||||||||||
.......... .AGACCGCAA CTGACTTTGC TATTTCTCTA CTCCAATCCT TGGTCACTGA         205


GGAGTCGAGT GTCATTTCAG AG                                               383100
|||||| ||| |||||||    ||
GGAGTCNAGT GTCATTTNNT AG                                                  227
```

Predicted gene locations (1):

```
PGL   1 (+ strand):      382540     383100
AGS-1 (382540  382567,382803  382929,383030  383100)
SCR   (e 1.000  d 0.050 a 0.050,e 0.980  d 0.050 a 0.050,e 0.944)

  Exon  1   382540   382567 (  28 n); score: 1.000
    Intron  1   382568   382802 ( 235 n);          Pd: 0.050  Pa: 0.050
  Exon  2   382803   382929 ( 127 n); score: 0.980
    Intron  2   382930   383029 ( 100 n);          Pd: 0.050  Pa: 0.050
  Exon  3   383030   383100 (  71 n); score: 0.944

  PGS (382540  382567,382803  382929,383030  383100) AU236313+

3-phase translation of AGS-1 (+strand):

          .        .        .       :   .        .        .
  382540  GCGTGTTGTTAAAGAGCTCACTAGTTGG : GTGATTTATTCAGAGGAGGATCGGAAGCTCAA
           A  C  C  *  R  A  H  *  L   : G  D  L  F  R  G  G  S  E  A  Q
            R  V  V  K  E  L  T  S  W  :  V  I  Y  S  E  E  D  R  K  L  N
             V  L  L  K  S  S  L  V  G :   *  F  I  Q  R  R  I  G  S  S


          .        .        .        .        .        .
  382835  CAAAGATATTACTATTGGTCTTATCCAGCGTGAGTTGCTTAGCCTAGCGGAGTACAATGT
           Q  R  Y  Y  Y  W  S  Y  P  A  *  V  A  *  P  S  G  V  Q  C
            K  D  I  T  I  G  L  I  Q  R  E  L  L  S  L  A  E  Y  N  V
          T  K  I  L  L  L  V  L  S  S  V  S  C  L  A  *  R  S  T  M


          .        .        .        .    :     .        .
  382895  CCACATGGCGAAGCATCTTGATGGAGGGAGAAACA : AGACCGCAACTGACTTTGCTATTTC
```

```
          P  H  G  E  A  S  *  W  R  E  K  Q  :     D  R  N  *  L  C  Y  F
           H  M  A  K  H  L  D  G  G  R  N     :  K  T  A  T  D  F  A  I  S
         S  T  W  R  S  I  L  M  E  G  E  T  :  R  P  Q  L  T  L  L  F


        .        .        .        .        .
  383055  TCTACTCCAATCCTTGGTCACTGAGGAGTCGAGTGTCATTTCAGAG
          S  T  P  I  L  G  H  *  G  V  E  C  H  F  R
           L  L  Q  S  L  V  T  E  E  S  S  V  I  S  E
         L  Y  S  N  P  W  S  L  R  S  R  V  S  F  Q


Maximal non-overlapping open reading frames (>= 64 codons)

>42592260+_PGL-1_AGS-1_PPS_1 (382541  382567,382803  382929,383030  383100)
(frame '1';   225 bp,   75 residues)
       1  RVVKELTSWV IYSEEDRKLN KDITIGLIQR ELLSLAEYNV HMAKHLDGGR NKTATDFAIS
      61  LLQSLVTEES SVISE



$ general statistics:
$ 1 chain has been computed
$
$ memory statistics:
$ 1 spliced alignments have been stored
$ 1 predicted gene locations have been stored
$ 0 megabytes was the average size of the backtrace matrix
$ 2 backtrace matrices have been allocated
$
$ date finished: 2008-11-18 11:32:07
```

In the first line (starting with symbol `$`) of all examples, the user input consisting of the called program plus the arguments are shown. Output lines starting with the symbol `$` give you useful information about the job — which parameters have been used is shown in the beginning and various statistical information is output at the end. In our example call an additional warning was issued after the parameter section complaining about the missing usage of the option `-species` which specify a so-called BSSM[6] parameter file. These files contain statistical information which makes it possible to "get the splice sites right" more easily (this is an oversimplification, [BXZ04] provides a good starting point to the theory behind this). So we execute the job again with the correct splice site file and the option `-v` which gives us additional information about the run:

```
$ gth -species arabidopsis -genomic NC_003070.fna.gz -cdna AU236313.gbk.gz -v
$ GenomeThreader 1.2.3 (2008-11-15 02:38:08)
$ Date run: 2008-11-18 11:32:07
$ Arguments: -species arabidopsis -genomic NC_003070.fna.gz -cdna AU236313.gbk.gz -v
$ make sure all necessary indices exist
$ make sure the necessary indices of all genomic input files exist
$ check the following file for index:
$ NC_003070.fna.gz
$ index exists
```

[6] BSSM stands for *Bayesian Splice Site Model*

```
$ make sure the necessary indices off all reference input files exist
$ check the following file for index:
$ AU236313.gbk.gz
$ index exists
$ invoking similarity filter
$ compute direct matches
$ call vmatch to compute matches
# args=-d -v -l 20 -seedlength 18 -exdrop 2 -q AU236313.gbk.gz.dna /Users/gordon/work/genometools/do
$ file=NC_003070.fna.gz 30867397 30432563
$ databaselength=30432562
$ alphabet "aAcCgGtTuUnsywrkvbdhmNSYWRKVBDHM" (size 32) mapped to "acgtn" (size 5)
$ NC_003070.fna.gz.dna.tis read
$ NC_003070.fna.gz.dna.suf read
$ NC_003070.fna.gz.dna.lcp read
$ NC_003070.fna.gz.dna.llv read
$ NC_003070.fna.gz.dna.sti1 read
$ NC_003070.fna.gz.dna.bck read
$ file=AU236313.gbk.gz 2488 227
$ databaselength=226
$ alphabet "aAcCgGtTuUnsywrkvbdhmNSYWRKVBDHM" (size 32) mapped to "acgtn" (size 5)
$ AU236313.gbk.gz.dna.tis read
$ AU236313.gbk.gz.dna.des read
$ AU236313.gbk.gz.dna.sds read
$ AU236313.gbk.gz.dna.ssp read
# matches are reported in the following way
# l(S) n(S) r(S) t l(Q) n(Q) r(Q) d e s i
# where:
# l = length
# n = sequence number
# r = relative position
# t = type (D=direct, P=palindromic)
# d = distance value (negative=hamming distance, 0=exact, positive=edit distance)
# e = E-value
# s = score value (negative=hamming score, positive=edit score)
# i = percent identity
# (S) = in Subject
# (Q) = in Query
$ find direct substring matches against query (maximal exact matches)
$ overall space peak: main=58.05 MB (2.00 bytes/symbol), secondary=214.19 MB (7.38 bytes/symbol)
$ d=+, compute chains for bucket 1/1 (matches in bucket=4)
$ sort global chains according to reference sequence coverage
$ calculate spliced alignment for every chain
$ d=+, compute spliced alignment, genseq=+, chain=1/1, refseq=+
$ compute palindromic matches
$ call vmatch to compute matches
# args=-p -v -l 20 -seedlength 18 -exdrop 2 -q AU236313.gbk.gz.dna /Users/gordon/work/genometools/do
$ file=NC_003070.fna.gz 30867397 30432563
$ databaselength=30432562
$ alphabet "aAcCgGtTuUnsywrkvbdhmNSYWRKVBDHM" (size 32) mapped to "acgtn" (size 5)
$ NC_003070.fna.gz.dna.tis read
$ NC_003070.fna.gz.dna.suf read
$ NC_003070.fna.gz.dna.lcp read
$ NC_003070.fna.gz.dna.llv read
$ NC_003070.fna.gz.dna.sti1 read
$ NC_003070.fna.gz.dna.bck read
$ file=AU236313.gbk.gz 2488 227
$ databaselength=226
$ alphabet "aAcCgGtTuUnsywrkvbdhmNSYWRKVBDHM" (size 32) mapped to "acgtn" (size 5)
$ AU236313.gbk.gz.dna.tis read
$ AU236313.gbk.gz.dna.des read
$ AU236313.gbk.gz.dna.sds read
$ AU236313.gbk.gz.dna.ssp read
```

```
# matches are reported in the following way
# l(S) n(S) r(S) t l(Q) n(Q) r(Q) d e s i
# where:
# l = length
# n = sequence number
# r = relative position
# t = type (D=direct, P=palindromic)
# d = distance value (negative=hamming distance, 0=exact, positive=edit distance)
# e = E-value
# s = score value (negative=hamming score, positive=edit score)
# i = percent identity
# (S) = in Subject
# (Q) = in Query
$ find palindromic substring matches against query
$ overall space peak: main=58.06 MB (2.00 bytes/symbol), secondary=214.19 MB (7.38 bytes/symbol)
$ calculate spliced alignment for every chain
$ output spliced alignments
********************************************************************************
EST Sequence: file=AU236313.gbk.gz, strand=+, description=AU236313 AU236313 RAFL14 Arabidopsis thali

    1  GCGTGTTGTT AAAGAGCTCA CTAGTTGGGT GATTTATTCA GAGGAGGATC CGGAAGCTCA
   61  ACAAAGATAT TACTATTGGT CTTATCCAGC GTGAGTTGCT TAGCCTAGCG GAGTACAATG
  121  TCCACATGGC GAAGCATCTT GATGGAGGGA GAAACAAGAC CGCAACTGAC TTTGCTATTT
  181  CTCTACTCCA ATCCTTGGTC ACTGAGGAGT CNAGTGTCAT TTNNTAG

Genomic Template: file=NC_003070.fna.gz, strand=+, from=382240, to=383395, description=gi|42592260|r

Predicted gene structure:

 Exon  1   382540   382567 (  28 n);  cDNA      1     28 (  28 n); score: 1.000
  Intron  1   382568   382802 ( 235 n); Pd: 0.985 (s:    0), Pa: 0.999 (s: 0.95)
 Exon  2   382803   382929 ( 127 n);  cDNA     29    156 ( 128 n); score: 0.980
  Intron  2   382930   383029 ( 100 n); Pd: 0.956 (s: 1.00), Pa: 0.983 (s: 1.00)
 Exon  3   383030   383100 (  71 n);  cDNA    157    227 (  71 n); score: 0.944

MATCH 42592260+ AU236313+ 0.967 226 0.996 C
PGS_42592260+_AU236313+ (382540  382567,382803  382929,383030  383100)

Alignment (genomic DNA sequence = upper lines):

GCGTGTTGTT AAAGAGCTCA CTAGTTGGGT ATGTTTACAA CCTTTTCAAG ATTTCACTTC      382599
|||||||||| |||||||||| ||||||||
GCGTGTTGTT AAAGAGCTCA CTAGTTGG.. .......... .......... ..........          28


GCTGATGTGC TTTGTTCAGT TACTTCTCCA TTAATTTGTC ACTATTTCTG TCAGAACACA      382659

.......... .......... .......... .......... .......... ..........          28


TAGGATAACA CATATCATAT AAGTGCTAGG TCGAGTCTGT TTCCTGTAGT TGGAGCCTAT      382719

.......... .......... .......... .......... .......... ..........          28


CCTCAACTGG TTATAGATAC TAGATTTGTT TCTTTGGTAT TTTTAGTTAT AATTAATTAT      382779

.......... .......... .......... .......... .......... ..........          28


CTTTCTTCAA ACTTTTGACA CAGGTGATTT ATTCAGAGGA GGAT-CGGAA GCTCAACAAA      382838
                         ||||||| |||||||||| |||| ||||| ||||||||||
```

```
.......... .......... ...GTGATTT ATTCAGAGGA GGATCCGGAA GCTCAACAAA          65


GATATTACTA TTGGTCTTAT CCAGCGTGAG TTGCTTAGCC TAGCGGAGTA CAATGTCCAC      382898
|||||||||| |||||||||| |||||||||| |||||||||| |||||||||| ||||||||||
GATATTACTA TTGGTCTTAT CCAGCGTGAG TTGCTTAGCC TAGCGGAGTA CAATGTCCAC         125


ATGGCGAAGC ATCTTGATGG AGGGAGAAAC AGTATGCTGA ATTGCTTAAC CTTTGTTGAT      382958
|||||||||| |||||||||| |||||||||| |
ATGGCGAAGC ATCTTGATGG AGGGAGAAAC A......... .......... ..........         156


GTCCTTGTGT GGTAACATTC TTTTTTTGTT TTGGCTAGTG AACTTGTTTT AACAAGTTGT      383018

.......... .......... .......... .......... .......... ..........         156


TTGTTACGCA GAGACCGCAA CTGACTTTGC TATTTCTCTA CTCCAATCCT TGGTCACTGA      383078
           ||||||||| |||||||||| |||||||||| |||||||||| ||||||||||
.......... .AGACCGCAA CTGACTTTGC TATTTCTCTA CTCCAATCCT TGGTCACTGA         205


GGAGTCGAGT GTCATTTCAG AG                                               383100
|||||| ||| |||||||    ||
GGAGTCNAGT GTCATTTNNT AG                                                  227


$ compute predicted gene locations
$ output predicted gene locations
--------------------------------------------------------------------------------

Predicted gene locations (1):


PGL   1 (+ strand):      382540     383100
AGS-1 (382540  382567,382803  382929,383030  383100)
SCR   (e 1.000  d 0.985 a 0.999,e 0.980  d 0.956 a 0.983,e 0.944)

  Exon  1   382540   382567 (  28 n); score: 1.000
    Intron  1   382568   382802 ( 235 n);          Pd: 0.985  Pa: 0.999
  Exon  2   382803   382929 ( 127 n); score: 0.980
    Intron  2   382930   383029 ( 100 n);          Pd: 0.956  Pa: 0.983
  Exon  3   383030   383100 (  71 n); score: 0.944

  PGS (382540  382567,382803  382929,383030  383100) AU236313+

3-phase translation of AGS-1 (+strand):

         .        .        .       :   .        .        .
  382540  GCGTGTTGTTAAAGAGCTCACTAGTTGG : GTGATTTATTCAGAGGAGGATCGGAAGCTCAA
           A  C  C  *  R  A  H  *  L   : G  D  L  F  R  G  G  S  E  A  Q
            R  V  V  K  E  L  T  S  W  :  V  I  Y  S  E  E  D  R  K  L  N
             V  L  L  K  S  S  L  V  G :   *  F  I  Q  R  R  I  G  S  S


         .        .        .        .        .        .
  382835  CAAAGATATTACTATTGGTCTTATCCAGCGTGAGTTGCTTAGCCTAGCGGAGTACAATGT
           Q  R  Y  Y  Y  W  S  Y  P  A  *  V  A  *  P  S  G  V  Q  C
            K  D  I  T  I  G  L  I  Q  R  E  L  L  S  L  A  E  Y  N  V
          T  K  I  L  L  L  V  L  S  S  V  S  C  L  A  *  R  S  T  M
```

```
          .         .         .         .    :    .         .
  382895  CCACATGGCGAAGCATCTTGATGGAGGGAGAAACA : AGACCGCAACTGACTTTGCTATTTC
           P  H  G  E  A  S  *  W  R  E  K  Q :   D  R  N  *  L  C  Y  F
            H  M  A  K  H  L  D  G  G  R  N   : K  T  A  T  D  F  A  I  S
          S  T  W  R  S  I  L  M  E  G  E  T  :  R  P  Q  L  T  L  L  F


          .         .         .         .         .
  383055  TCTACTCCAATCCTTGGTCACTGAGGAGTCGAGTGTCATTTCAGAG
           S  T  P  I  L  G  H  *  G  V  E  C  H  F  R
            L  L  Q  S  L  V  T  E  E  S  S  V  I  S  E
          L  Y  S  N  P  W  S  L  R  S  R  V  S  F  Q


Maximal non-overlapping open reading frames (>= 64 codons)

>42592260+_PGL-1_AGS-1_PPS_1 (382541  382567,382803  382929,383030  383100)
(frame '1';   225 bp,   75 residues)
       1  RVVKELTSWV IYSEEDRKLN KDITIGLIQR ELLSLAEYNV HMAKHLDGGR NKTATDFAIS
      61  LLQSLVTEES SVISE




$ general statistics:
$ 1 chain has been computed
$
$ memory statistics:
$ 1 spliced alignments have been stored
$ 1 predicted gene locations have been stored
$ 0 megabytes was the average size of the backtrace matrix
$ 1 backtrace matrix has been allocated
$
$ date finished: 2008-11-18 11:32:09
```

As one can see, the output grew quite a bit. In lines starting with the symbol # output from internal `vmatch` and `mkvtree` calls is shown. There is no `mkvtree` output in our example, because the necessary indices did already exist (see Section 12 for details). To combine the option `-v` with option `-o` which saves the output in the supplied output file is quite useful when performing larger tasks, because this gives us progress information on stdout while the actual output is saved in the output file. For example, to match 5000 full-length cDNAs against the first 200000 bases of the *A. thaliana* chromosome 1, we issue the following command (only the last 20 lines of progress information is shown):

```
$ gth -v -frompos 1 -topos 200000 -inverse -force -o arab_mapping.txt -cdna CeresTigr.gz -species ar
$ d=-, compute chains for bucket 368/377 (matches in bucket=1)
$ d=-, compute chains for bucket 369/377 (matches in bucket=1)
$ d=-, compute chains for bucket 370/377 (matches in bucket=1)
$ d=-, compute chains for bucket 371/377 (matches in bucket=1)
$ d=-, compute chains for bucket 372/377 (matches in bucket=1)
$ d=-, compute chains for bucket 373/377 (matches in bucket=9)
$ d=-, compute chains for bucket 374/377 (matches in bucket=1)
$ d=-, compute chains for bucket 375/377 (matches in bucket=2)
$ d=-, compute chains for bucket 376/377 (matches in bucket=2)
$ d=-, compute chains for bucket 377/377 (matches in bucket=1)
```

```
$ sort global chains according to reference sequence coverage
$ calculate spliced alignment for every chain
$ d=-, compute spliced alignment, genseq=-, chain=1/5, refseq=+
$ d=-, compute spliced alignment, genseq=-, chain=2/5, refseq=+
$ d=-, compute spliced alignment, genseq=-, chain=3/5, refseq=+
$ d=-, compute spliced alignment, genseq=-, chain=4/5, refseq=+
$ d=-, compute spliced alignment, genseq=-, chain=5/5, refseq=+
$ output spliced alignments
$ compute predicted gene locations
$ output predicted gene locations
```

After this run, the output can be found in the file `arab_mapping.txt`. To get better results, we would also add the option `-gcmaxgapwidth 5000`, limiting the maximal intron size in *A. thaliana* to 5000 bases.

13.2 Using the Intron Cutout Technique

Assume we want to map a single mRNA against human chromosome 21. The obvious call would not succeed (only last 3 lines of output shown):

```
$ gth -species human -genomic hs_ref_chr21.fa.gz -cdna NM_003253.gbk
$ important messages:
$ 1 matrix allocations failed
$ 1 undetermined spliced alignments
```

This is due to the fact that human genes can contain very long introns. Therefore, we have to use the option `-intron cutout`. Furthermore, we use the option `-introndistri` which gives us a distribution of the intron sizes (only this distribution is shown from the output):

```
$ gth -intron cutout -introndistri -species human -genomic hs_ref_chr21.fa.gz -cdna NM_003253.gbk
```

```
$ length distribution of all introns:
88: 1 (prob=0.0357,cumulative=0.0357)
304: 1 (prob=0.0357,cumulative=0.0714)
574: 1 (prob=0.0357,cumulative=0.1071)
1112: 1 (prob=0.0357,cumulative=0.1429)
2134: 1 (prob=0.0357,cumulative=0.1786)
2370: 1 (prob=0.0357,cumulative=0.2143)
3060: 1 (prob=0.0357,cumulative=0.2500)
3182: 1 (prob=0.0357,cumulative=0.2857)
3684: 1 (prob=0.0357,cumulative=0.3214)
4080: 1 (prob=0.0357,cumulative=0.3571)
4354: 1 (prob=0.0357,cumulative=0.3929)
4984: 1 (prob=0.0357,cumulative=0.4286)
5076: 1 (prob=0.0357,cumulative=0.4643)
5390: 1 (prob=0.0357,cumulative=0.5000)
5632: 1 (prob=0.0357,cumulative=0.5357)
5706: 1 (prob=0.0357,cumulative=0.5714)
6081: 1 (prob=0.0357,cumulative=0.6071)
```

```
7032: 1 (prob=0.0357,cumulative=0.6429)
7602: 1 (prob=0.0357,cumulative=0.6786)
8136: 1 (prob=0.0357,cumulative=0.7143)
9748: 1 (prob=0.0357,cumulative=0.7500)
10534: 1 (prob=0.0357,cumulative=0.7857)
13820: 1 (prob=0.0357,cumulative=0.8214)
17355: 1 (prob=0.0357,cumulative=0.8571)
```

13.3 Employing `gthconsensus`

Assume we want to perform *incremental updates*, i.e., we want to annotate a genomic sequence with a set of cDNAs/ESTs and/or proteins and update the annotation as soon as new sequences become available.

Because we do not want to redo the complete computation every time new sequences become available, we save the *intermediate results* containing the spliced alignments and only recompute the *consensus spliced alignments* using `gthconsensus`.

For explanatory purposes, we use the same files as in the examples above. We produce the intermediate results of the mapping of 5000 cDNAs using the option `-intermediate` and store these in a file. The option `-intermediate` requires that we also use the option `-xmlout`, because the intermediate results are stored in an XML format. To save disk space we compress the output file on the fly using the option `-gzip`. The output is stored in the file `ceres.inter.gz`.

```
$ gth -intermediate -xmlout -gzip -o ceres.inter.gz -genomic NC_003070.fna.gz -cdna CeresTigr.gz -fr
```

To look at the results stored in the file `ceres.inter.gz`, we employ `gthconsensus` (with `-gzip`, because the file is compressed). The output was piped through `grep ^PGS` to shorten it:

```
$ gthconsensus ceres.inter.gz
PGS_42592260+_24737-(319898  320145,320252  320296,320380  320457,320769  321417,321745  321865,3219
PGS_42592260+_116833+ (345866  347527)
PGS_42592260-_1693+ (370967  370859,370751  370258)
PGS_42592260-_40042+ (389646  389410,389301  389226,389138  389050,388940  388851,388760  388716,388
PGS_42592260-_35733+ (404439  404186,403947  403771,403526  403195)
```

We can also print some information about the intermediate file with the help of `gthfilestat`:

```
$ gthfilestat ceres.inter.gz
spliced alignment alignment score distribution:

spliced alignment coverage distribution:

memory statistics:
5 spliced alignments have been stored
5 predicted gene locations have been stored

date finished: 2008-11-18 11:32:52
```

Now we map an additional EST and store the result in `new.inter.gz`.

```
$ gth -intermediate -xmlout -gzip -o new.inter.gz -genomic NC_003070.fna.gz -cdna AU236313.gbk.gz -f
```

Afterwards we combine the two results (the output was also shortened):

```
$ gthconsensus ceres.inter.gz new.inter.gz
PGS_42592260+_24737-(319898  320145,320252  320296,320380  320457,320769  321417,321745  321865,3219
PGS_42592260+_116833+ (345866  347527)
PGS_42592260-_1693+ (370967  370859,370751  370258)
PGS_42592260+_AU236313+ (382540  382567,382803  382929,383030  383100)
PGS_42592260-_40042+ (389646  389410,389301  389226,389138  389050,388940  388851,388760  388716,388
PGS_42592260-_35733+ (404439  404186,403947  403771,403526  403195)
```

14 Feedback

We would be glad to hear from you what you liked about *GenomeThreader*, what you did not, and what could be improved. You can send your remarks via email to `gordon@gremme.org`. This is of particular importance, if you find a bug which slipped through our test suite. As an incentive, we promise to fix it!

15 Acknowledgements

Stefan Kurtz, Volker Brendel, Sergej Friedmann, Alexander Sczyrba, Michael E. Sparks, Stefan Bienert, Daniel Lang, Robert Buels, Qunfeng Dong, Marko Skocibusic, Lieven Sterck, Matthew Wilkerson, Robert Schmieder, Jerome Gouzy, and Jer-Young Lin gave valuable hints on improving the manual and the programs described herein. Their help is much appreciated.

16 Recent Changes

The following new features have recently been integrated into the software described in this manual. From version `1.0.1` on the recent changes can be found in the `CHANGELOG` file.

***GenomeThreader* `1.0.0 (2007-06-20):`** Many internal changes due to switch to the GenomeTools library (see `http://genometools.org/` for details). For example, the new option parser shows the default value for each option. The deprecated options `-reference` and `-gcfilterthreshold` have finally been removed. Bug fixes.

***GenomeThreader* `0.9.60 (2007-03-08):`** The implementation of the consensus phase changed. Bug fixed.

***GenomeThreader* `0.9.59 (2007-02-08):`** The default value for option `-deletionweight` has been changed from -4.0 to -5.0.

***GenomeThreader* `0.9.58 (2006-10-20):`** Bug fix release.

***GenomeThreader* `0.9.57 (2006-09-19):`** The options `-createindicesonly` and `-skipindexcheck` have been added.

***GenomeThreader* `0.9.56 (2006-08-31):`** The options `-prminmatchlen`, `-prseedlength`, and `-prhdist` have been introduced. A bug concerning the output of protein alignments has been fixed.

***GenomeThreader* `0.9.55 (2006-08-07):`** A bug in the protein code has been fixed.

***GenomeThreader* `0.9.54 (2006-07-27):`** The XML output of *GenomeThreader* now reflects version 1.1 of the corresponding schema.

***GenomeThreader* `0.9.53 (2006-05-23):`** `gthbssmbuild` performs more sanity checks on the the training data.

***GenomeThreader* `0.9.52 (2006-05-22):`** The option `-replacewildcards` has been added to the tool `gthbssmbuild`.

***GenomeThreader* `0.9.51 (2006-05-16):`** The plain protein output has been improved. Bug fixes.

***GenomeThreader* `0.9.50 (2006-05-14):`** A minor problem concerning the XML output has been fixed.

***GenomeThreader* `0.9.49 (2006-05-12):`** The tool `gthbssmbuild` has been introduced.

***GenomeThreader* `0.9.48 (2006-04-12):`** Bug fix.

***GenomeThreader* `0.9.47 (2005-12-08):`** Bug fix.

***GenomeThreader* `0.9.46 (2005-11-16):`** Bug fixes.

***GenomeThreader* `0.9.45 (2005-11-16):`** Bug fixes. Time and space requirements of the consensus phase have been improved.

***GenomeThreader* `0.9.44 (2005-11-11):`** XML output adjusted.

***GenomeThreader* `0.9.43 (2005-11-09):`** Bug fixes.

***GenomeThreader* `0.9.42 (2005-10-11):`** Bug fixes.

***GenomeThreader* `0.9.41 (2005-09-21):`** Computation of paralogous genes has been made available via option `-paralogs`. `gthgetseq` has been extended by the options `-getcdnacomp`, `-getproteincomp`, and `-getgenomiccomp`. Bug fixes.

***GenomeThreader* `0.9.40 (2005-09-15):`** The option `-reference` has been replaced by the options `-cdna` and `-protein`. The tools `gthsplit`, `gthsplit2dim.sh`, `gthgetseq`, and `gthfilestat` have been introduced. A spliced alignment filter (see Section 3.9.6) has been built into some *GenomeThreader* tools. Bug fixes.

***GenomeThreader* `0.9.39:`** The small tool `gthbssmfileinfo` has been introduced. The options `-translationtable` and `-skipalignmentout` have been added.

***GenomeThreader* `0.9.38:`** Option `-showseqnums` added.

***GenomeThreader* `0.9.37:`** Option `-gcfilterthreshold` has been renamed to `-gcmincoverage`.

***GenomeThreader* `0.9.36:`** Rare matches. Some bug fixes.

***GenomeThreader* `0.9.35:`** Computation of paralogous genes has been introduced into the code.

***GenomeThreader* `0.9.34:`** The generic splice site model has been changed!

***GenomeThreader* `0.9.33`:** Bug involving option `-frompos` fixed. U12-type intron model has been added. Space consumption of cDNA backtrace matrix quartered.

***GenomeThreader* `0.9.32`:** Bug fixed.

***GenomeThreader* `0.9.31`:** More statistics.

***GenomeThreader* `0.9.30`:** Some XML output changes.

***GenomeThreader* `0.9.29`:** Minor extension to the XML output. Bug fixed.

***GenomeThreader* `0.9.28`:** Performance improved. Memory requirements reduced. Bugs fixed. Option `-gcmaxgapwidth` has been introduced.

***GenomeThreader* `0.9.27`:** XML output adjusted.

***GenomeThreader* `0.9.26`:** Speed optimization of the cDNA/EST DP.

***GenomeThreader* `0.9.25`:** Unnecessary environment variable dependency removed.

***GenomeThreader* `0.9.24`:** Internal error handling improved.

***GenomeThreader* `0.9.23`:** The XML output for protein spliced alignments has been implemented.

***GenomeThreader* `0.9.22`:** The options `-gzip` and `-bzip2` have been introduced.

***GenomeThreader* `0.9.21`:** First version of protein case ready.

***GenomeThreader* `0.9.20`:** Bug fixed.

***GenomeThreader* `0.9.19`:** The `gthconsensus` program has been introduced.

***GenomeThreader* `0.9.18`:** Option `-intermediate` has been implemented.

***GenomeThreader* `0.9.17`:** Option `-noautoindex` has been implemented.

***GenomeThreader* `0.9.16`:** Option `-xmlout` has been implemented.

***GenomeThreader* `0.9.15`:** Some improvements have been made in the MPI version.

***GenomeThreader* `0.9.14`:** The MPI version of *GenomeThreader* has been introduced. The MPI version is not publicly available yet.

***GenomeThreader* `0.9.13`:** The options `-iciterations` and `-icdeltaincrease` have been added.

***GenomeThreader* `0.9.12`:** Option `-inverse` has been added. Some bugs fixed.

***GenomeThreader* `0.9.11`:** Many additional options added.

***GenomeThreader* `0.9.10`:** Additional species have been added.

***GenomeThreader* `0.9.9`:** The options `-autointroncutout` and `-polyatailfilter` have been introduced.

***GenomeThreader* `0.9.8`:** The default for `-minmatchlen` has been increased to 20. Option `-inexact` has been removed, inexact matching is now the default. Exact matches can be computed via the new option `-exact`. The parameters of the inexact matching can be changed via the new options `-seedlength` and `-exdrop`. The default value for the Option `-gcfilterthreshold` has been increased to 50.

***GenomeThreader* `0.9.7`:** Option `-online` has been intronduced.

***GenomeThreader* `0.9.6`:** One can now specify multiple genomic and reference files. A few bugs have been fixed.

***GenomeThreader* `0.9.5`:** The genomic index can contain more than one sequence. The DP parameters are only computed for the considered genomic regions and not for the whole genomic sequence.

***GenomeThreader* `0.9.4`:** Option `-gcfilterthreshold` has been introduced. The default value for option `-icinitialdelta` has been changed from 30 to 50.

***GenomeThreader* `0.9.3`:** Segmentation fault fixed.

***GenomeThreader* `0.9.2`:** Some options used for the old similarity filter have been removed. Option `-minmatchlen` has been introduced. Small overlaps are now removed after the local chaining.

***GenomeThreader* `0.9.1`:** Some internal changes concerning error handling.

***GenomeThreader* `0.9`:** First version with new similarity filter, new chaining, and intron cutout technique.

***GenomeThreader* `0.8.11`:** Intron cutouts have been introduced in the DP.

***GenomeThreader* `0.8.10`:** The calculation of exon scores for alternative gene structures has been changed.

***GenomeThreader* `0.8.9`:** The exon scores for alternative gene structures are now shown.

***GenomeThreader* `0.8.8`:** The option `-minorflength` has been introduced.

***GenomeThreader* `0.8.7`:** The 3-phase translation of alternative gene structures and the output of maximal non-overlapping open reading frames have been introduced.

***GenomeThreader* `0.8.6`:** The following options have been introduced: `-leadcutoffsmode`, `-termcutoffsmode`, `-cutoffsminexonlen`, and `-scoreminexonlen`.

References

[BXZ04] V. Brendel, L. Xing, and W. Zhu. Gene Structure Prediction from Consensus Spliced Alignment of Multiple ESTs Matching the Same Genomic Locus. *Bioinformatics*, **20**(7):1157–1169, 2004.

[GBSK05] G. Gremme, V. Brendel, M.E. Sparks, and S. Kurtz. Engineering a Software Tool for Gene Structure Prediction in Higher Organisms. *Information and Software Technology*, **47**(15):965–978, 2005.

[Gre12] G. Gremme. *Computational Gene Structure Prediction*. PhD thesis, University of Hamburg, 2012.

[UB00] J. Usuka and V. Brendel. Gene Structure Prediction by Spliced Alignment of Genomic DNA with Protein Sequences: Increased Accuracy by Differential Splice Site Scoring. *J. Mol. Biol.*, **297**:1075–1085, 2000.

[UZB00] J. Usuka, W. Zhu, and V. Brendel. Optimal Spliced Alignment of Homologous cDNA to a Genomic DNA Template. *Bioinformatics*, **16**(3):203–211, 2000.

[ZB03] W. Zhu and V. Brendel. Identification, Characterization and Molecular Phylogeny of U12-dependent Introns in the *Arabidopsis thaliana* Genome. *Nucleic Acids Res.*, **31**(15):4561–4572, 2003.

Index