Glimmer-MG Release Notes
Version 0.1

David R. Kelley, Art L. Delcher

17 May 2011

1 Introduction

This document describes Version 0.1 of the GLIMMER-MG gene-finding software for metagenomic sequences. Users discovering problems or errors are encouraged to report them to `dakelley@umiacs.umd.edu`.

GLIMMER is a collection of programs for identifying genes in microbial DNA sequences. The system works by creating a variable-length Markov model from a training set of genes and then using that model to attempt to identify all genes in a given DNA sequence. The three versions of GLIMMER are described in [SDKW98], [DHK+99], and [DBPS07].

GLIMMER-MG is released as OSI Certified Open Source Software under the Artistic License. The license is contained in the file, `LICENSE`, in the distribution.

2 Installation

GLIMMER-MG software was written for the Linux software environment. The following instructions assume a Linux system. They also work under Mac OSX.

To install GLIMMER-MG, download the compressed tarfile `glimmer-mg-1.0.1.tar.gz` from the website. Then uncompress the file by typing

```
tar xzf glimmer-mg.1.0.1.tar.gz
```

A directory named `glimmer-mg` should result. In that directory, is a subdirectory named `src`. Within the `src` subdirectory type

```
make
```

(or alternately `gmake`). This will compile the GLIMMER-MG programs and put the executable files in the directory `glimmer-mg/bin`. These files can be copied or moved to whatever directory is convenient to the user.

3 Running Glimmer-MG

GLIMMER-MG can be run in a few different modes, depending on the characteristics of the sequences provided, e.g., whether they represent accurate contigs or reads from the Illumina or 454 technologies. Then within each mode, GLIMMER-MG can be run with varying amounts of classification/clustering preprocessing.

3.1 Preprocessing options

The GLIMMER-MG pipeline can be run a number of different ways, including a Python script to run the full pipeline and options to classify/cluster the reads separately and predict genes using those results.

3.1.1 Running from scratch

The Python script `glimmer-mg.py` runs the full GLIMMER-MG pipeline.

```
glimmer-mg.py seqs.fasta
```

This script will classify the sequences with PHYMM [BS09], make initial predictions for all sequences, cluster the sequences with SCIMM [KS10], and make final predictions within each cluster.

3.1.2 Running from scratch with no retraining

By passing the option `--iter 0` to `glimmer-mg.py`, the clustering and re-training steps will be skipped. That is, the sequences will be classified with PHYMM and predictions made.

```
glimmer-mg.py --iter 0 seqs.fasta
```

3.1.3 Classification separate

Some users may prefer to run the computationally intensive classification of sequences separately (e.g. on a computer cluster). `glimmer-mg.py` can be made to expect this by using the `--raw` option, which specifies that the raw PHYMM output file already exists in the current working directory.

```
glimmer-mg.py --raw seqs.fasta
```

3.1.4 Clustering separate

In a similar vein, clustering can be performed separately before making gene predictions by specifying the `--clust` option, which specifies that SCIMM or PHYSCIMM output is in the current working directory.

```
glimmer-mg.py --clust seqs.fasta
```

3.2 Sequence modes

Depending on the error characteristics of the input sequences, GLIMMER-MG has three modes that best handle certain types of data. In each case, the options described can be passed to the C++ binary `glimmer-mg` or the Python script to manage the entire pipeline `glimmer-mg.py`.

3.2.1 Accurate contigs

If you believe that the sequences on which you would like to find genes are very accurate, than run GLIMMER-MG in the default mode with no additional options.

3.2.2 454 indels

If the input sequences are reads or contigs from the 454 or Ion Torrent technologies and indel errors are expected to exist, GLIMMER-MG can be made to predict such errors and account for them in its gene predictions by using *indel* mode.

```
glimmer-mg.py --indels 454.fasta
```

3.2.3 Illumina substitution errors

If the input sequences are reads or contigs from the Illumina technology and substitution errors are expected to exist, GLIMMER-MG can be made to predict such errors and account for them in its gene predictions by using *substitution* mode.

```
glimmer-mg.py --sub illumina.fasta
```

4 Sample Run Directory

A directory containing a sample run of GLIMMER3 is provided. This directory, named `sample_run` contains the genome sequence for *Treponema pallidum* (file `tpall.fna`) and a list of annotated genes for it (file `tpall.nh`), both downloaded from GenBank. The files whose names begin `from-scratch` are the result of running the script

```
g3-from-scratch.csh tpall.fna from-scratch
```

The files whose names begin `from-training` are the result of running the script

```
g3-from-training.csh tpall.fna tpall.nh from-training
```

The files whose names begin `iterated` are the result of running the script

```
g3-iterated.csh tpall.fna iterated
```

Users will need to modify the path directories at the top of these scripts to be able to run them (see Section **??** above).

5 Notes on Glimmer-MG programs

5.1 `glimmer-mg` Program

This is the main program that makes gene preditions.

5.1.1 `glimmer-mg` Parameters & Options

The invocation for `glimmer-mg` is:

`glimmer-mg` [*options*] *sequence* *tag*

where *sequence* is the name of the file containing the DNA sequence(s) to be analyzed and *tag* is a prefix used to name the output files.

options can be the following:

`-b` *filename* or `--rbs_pwm` *filename*

Read a position weight matrix (PWM) from *filename* to identify the ribosome binding site to help choose start sites. The format of this file is indicated by the following example:

```
6
a     212     309      43      36     452     138
c      55      58       0      19      48      26
g     247     141     501     523       5     365
t      64      70      34       0      73      49
```

The first line is the number of positions in the pattern, *i.e.*, the number of columns in the matrix (not counting the first column of labels). The column values are the relative frequencies of nucleotides at each position.

`-c` *filename* or `--class` *filename*

Read sequence classifications from *filename*. Each line of *filename* should specify a sequence fasta header followed by a series of GenBank reference genome classifications. The classifications should name the reference genome's Phymm directory (corresponding to the GenBank strain name), followed by the character '|', followed by the genomic sequence's unique identifier. It is our expectation that users will not create these files, but they generate them with `glimmer-mg.py`. For example:

```
read1     Mycobacterium_leprae_Br4923|NC_011896 Corynebacterium_glutamicum_ATCC_13032|NC_003450
read2     Rhodopseudomonas_palustris_BisA53|NC_008435 Rhodopseudomonas_palustris_HaA2|NC_007778
...
```

`-f` *filename* or `--features` *filename*

filename specifies counts for features such as gene length, start codon usage, adjacent gene orientations, and adjacent gene distances for coding and noncoding ORFs. GLIMMER-MG will convert these counts into probability models for use in log-likelihood computations for the features. It is our expecation that users will not create these files, but generate them with `train_features.py` (via `glimmer-mg.py`). For an example of the input format, see the `sample_run` directory.

`-g` *n* or `--gene_len` *n*

Set the minimum gene length to *n* nucleotides. This does not include the bases in the stop codon.

`-h` or `--help`

Print the usage message.

`-i` or `--indel`

Predict genes in "indel-mode" where gene predictions may shift the coding frame, implicitly predicting an insertion or deletion in the sequence. If quality values are provided with the `-q` option, positions at which frame shifts should be considered will be identifed by their low quality. If no quality values are provided, frame shifts will be considered at long homopolymer runs, assuming that the sequences were generated by the 454 or Ion torrent technologies.

`-m` *filename* or `--icm` *filename*

filename contains an ICM trained on coding sequence to be used for ORF scoring rather than ICM's obtained via classification results.

`-o` n or `--max_olap` n

Set the maximum overlap length to n. Overlaps of this many or fewer bases are allowed between genes. The new dynamic programming algorithm should *never* output genes that overlap by more than this many bases.

`-q` *filename* or `--quality` *filename*

filename contains quality values in fasta format matching up with the sequences fasta file. The quality values are used in "indel-mode" and "substitution-mode" to identify low quality positions in the sequences where errors are most likely.

`-r` or `--circular`

Assume a circular rather than linear genome, *i.e.*, there may be genes that "wraparound" between the beginning and end of the sequence.

`-s` or `--sub`

Predict genes in "substitution-mode" where gene predictions may pass through stop codons, implicitly predicting a sequencing error that mutated a standard codon to a stop codon. If quality values are provided with the `-q` option, they will be used to compute a log-likelihood penalty for passing through a given stop codon. Otherwise, we assume all quality values are Phred 30.

`-u` n or `--fudge` n

n specifices a "fudge factor" to be added to the log-likeihood score for every ORF. The "fudge factor" acts as a means to tune the sensitivity versus specificity of the predictions.

`-z` n or `--trans_table` n

Use Genbank translation table number n to specify stop codons.

`-Z` *codon-list* or `--stop_codons` *codon-list*

Specify stop codons as a comma-separated list. Sample format: `-Z tag,tga,taa`. The default stop codons are `tag`, `tga` and `taa`.

5.1.2 `glimmer3` Output Formats

`.predict` File

This file has the final gene predictions. It's format is the fasta-header line of the sequence followed by one line per gene. Here is a sample of the beginning of such a file:

```
>gms:3447|cmr:632 chromosome 1 {Mycobacterium smegmatis MC2}
orf00001      499     1692  +1    13.14
orf00004     1721     2614  +2    14.20
orf00006     2624     3778  +2    10.35
orf00009     3775     4359  +1     9.34
```

The columns are:

Column 1 The identifier of the predicted gene. The numeric portion matches the number in the `ID` column of the `.detail` file.

Column 2 The start position of the gene.

Column 3 The end position of the gene. This is the last base of the stop codon, *i.e.*, it includes the stop codon.

Column 4 The reading frame.

Column 5 The per-base "raw" score of the gene. This is slightly different from the value in the `.detail` file, because it includes adjustments for the PWM and start-codon frequency.

6 Notes on Glimmer programs

6.1 `build-icm` Program

This program constructs an interpolated context model (ICM) from an input set of sequences.

6.1.1 `build-icm` Parameters & Options

The format for invoking `build-icm` is:

`build-icm` [*options*] *output-file* `<` *input-file*

Sequences are reads from standard input, the ICM is built and written to *output-file*. If *output-file* is "-", then the output will be sent to standard output. Since input comes from standard input, one also can "pipe" the input into this program, *e.g.*,

```
cat abc.in | build-icm xyz.icm
```

or even type in the input directly.

Possible *options* are:

`-d` *num* or `--depth` *num*

Set the depth of the ICM to *num*. The depth is the maximum number of positions in the context window that will be used to determine the probability of the predicted position. The default value is 7.

`-F` or `--no_stops`

Do not use any input strings with in-frame stop codons. Stop codons are determined by either the `-z` or `-Z` option.

`-h` or `--help`

Print the usage message.

`-p` *num* or `--period` *num*

Set the period of the ICM to *num*. The period is the number of different submodels for different positions in the text in a cyclic pattern. *E.g.*, if the period is 3, the first submodel will determine positions $1, 4, 7, \ldots$; the second submodel will determine positions $2, 5, 8, \ldots$; and the third submodel will determine positions $3, 6, 9, \ldots$. For a non-periodic model, use a value of 1. The default value is 3.

`-r` or `--reverse`

Use the reverse of the input strings to build the ICM. Note that this is merely the reverse and *NOT* the reverse-complement. In other words, the model is built in the backwards direction.

`-t` or `--text`

Output the model in a text format. This is for informational/debugging purposes only—the `glimmer3` program cannot read models in this form.

The format of the output is a header line containing the parameters of the model, followed by individual probability lines. The entries on each probability line are:

Column	Description
1	ID number
2	Context pattern
3	Mutual information
4	Probability of A
5	Probability of C
6	Probability of G
7	Probability of T

The context pattern is divided into codons by the vertical lines (this option assumes the default 3-periodic model). The "?" represents the position being predicted. Letters represent specific values in their respective positions in the context window. The asterisk indicates the position that has maximum mutual information with the predicted position.

`-v` *num* or `--verbose` *num*

Set the verbose level to *num*. This controls extra debugging output—the higher the value the more output.

`-w` *num* or `--width` *num*

Set the width of the ICM to *num*. The width includes the predicted position. The default value is 12.

`-z` *n* or `--trans_table` *n*

Use Genbank translation table number *n* to specify stop codons.

`-Z` *codon-list* or `--stop_codons` *codon-list*

Specify stop codons as a comma-separated list. Sample format: `-Z tag,tga,taa`. The default stop codons are `tag`, `tga` and `taa`.

6.2 `glimmer3` Program

This is the main program that makes gene preditions.

6.2.1 `glimmer3` Parameters & Options

The invocation for `glimmer3` is:

`glimmer3` [*options*] *sequence icm tag*

where *sequence* is the name of the file containing the DNA sequence(s) to be analyzed and *icm* is the name of the file containing the ICM model produced by `build-icm`. *tag* is a prefix used to name the two output files: *tag* `.detail` and *tag* `.predict`.

options can be the following:

`-A` *codon-list* or `--start_codons` *codon-list*

Specify start codons as a comma-separated list. Sample format: `-A atg,gtg`. The default start codons are `atg`, `gtg` and `ttg`. Use the `-P` option to specify the relative proportions of use. If `-P` is not used, then the proportions will be equal.

`-b` *filename* or `--rbs_pwm` *filename*

Read a position weight matrix (PWM) from *filename* to identify the ribosome binding site to help choose start sites. The format of this file is indicated by the following example:

```
6
a     212     309      43      36     452     138
c      55      58       0      19      48      26
g     247     141     501     523       5     365
t      64      70      34       0      73      49
```

The first line is the number of positions in the pattern, *i.e.*, the number of columns in the matrix (not counting the first column of labels). The column values are the relative frequencies of nucleotides at each position.

`-C` *p* or `--gc_percent` *p*

Use *p* as the GC percentage of the independent model, *i.e.*, the model of intergenic sequence. Note: *p* should be a percentage, *e.g.*, `-C 45.2`

If this option is not specified, the GC percentage will be counted from the input file.

`-E` *filename* or `--entropy` *filename*

Read entropy profiles from *filename* . The format is one header line, then 20 lines of 3 columns each, which is the format produced by the program `entropy-profile` with the `-b` option. The columns are amino acid, positive entropy, and negative entropy, respectively. Rows must be in alphabetical order by amino acid code letter. This currently does not affect GLIMMER3 predictions, but is used in the `long-orfs` program. If the option is specified, the entropy-distance ratio for each potential gene is printed as the last column of the `.detail` file. If *filename* is "`#`", then a set of default entropy profiles, constructed from a wide range of species, is used.

`-f` or `--first_codon`

Use the first possible codon in an orf as the start codon for initial scoring purposes. Otherwise, the highest-scoring codon will be used. This only affects the start positions in the `.detail` file. The final start predictions in the `.predict` file are always based on the scoring functions.

`-g` *n* or `--gene_len` *n*

Set the minimum gene length to *n* nucleotides. This does not include the bases in the stop codon.

`-h` or `--help`

Print the usage message.

`-i` *filename* or `--ignore` *filename*

filename specifies regions of bases that are off limits, so that no bases within that area will be examined. The format for entries in this file is one line per region, with the start and end positions of the region specified as the first two fields on the line. The rest of the line is regarded as comments. Additionally, any line beginning with a `#` is regarded as a comment. *E.g.*, the following file:

```
   1001     1600   Comment here
# The region can be specified high-low as well as low-high
   5600     5001
```

would ignore bases 1001 . . . 1600 and 5001 . . . 5600 in the input sequence. This option should not be used with multi-sequence input files.

`-l` or `--linear`

Assume a linear rather than circular genome, *i.e.*, there will be no genes that "wraparound" between the beginning and end of the sequence.

`-L` *filename* or `--orf_coords` *filename*

filename specifies a list of orfs that should be scored separately, with no attempt to resolve overlaps or determine start codons. The format of the list is one orf per line, with entries separated by white space. The first entry is an identifier for the orf. It can be an arbitrary string without spaces. The next two entries are the start and end positions of the orf, respectively, (coordinates counting from 1), including the stop codon. The fourth entry is the reading frame. This is used only to determine the direction of the orf in cases of circular genomes where the orf might "wrap around" the end of the input sequence. If positive the orf is presumed to be on the positive DNA strand; otherwise, on the negative strand. Any further entries on the line are ignored.

The output with this option goes both to the `.predict` file and to the `.detail` file.

`-M` or `--separate_genes`

sequence-file is a multifasta file of separate genes to be scored separately, with no overlap rules. Each sequence is assumed to be in $5'$ to $3'$ order and to include the stop codon.

`-o` n or `--max_olap` n

Set the maximum overlap length to n. Overlaps of this many or fewer bases are allowed between genes. The new dynamic programming algorithm should <u>*never*</u> output genes that overlap by more than this many bases.

`-P` *number-list* or `--start_probs` *number-list*

Specify the probability of different start codons (same number and order as in `-A` option). If no `-A` option is given, then there should be 3 values: for `atg`, `gtg` and `ttg`, in that order. Sample format: `-P 0.6,0.35,0.05`. If `-A` is specified without `-P`, then each start codon is equally likely (which is very unusual).

`-q` n or `--ignore_score_len` n

Consider any gene n or more bases long as a potential gene, regardless of its in-frame score. Without this option, this value is calculated automatically to be the length such that the expected number of orfs this long or longer in a random sequence of a million bases is one.

`-r` or `--no_indep`

Don't use the independent probability score column at all. Using this option will produce more short gene predictions.

`-t` n or `--threshold` n

Set the threshold score for consideration as a gene to n. If the in-frame score $\geq n$, then the region is given a number and considered a potential gene. Note this is the integer score in the column labelled "InFrm" in the `.detail` file, not the decimal score in the column labelled "Raw".

`-X` or `--extend`

Also score orfs that extend off the end of the sequence(s). This option presumes that the sequence(s) is linear and not circular. Reported positions off the end of the sequence are the nearest positions in the correct reading frame. Note that this ignores any partial codons at the ends of a sequence. Suppose, for example, that a sequence is 998bp long and an orf in reading frame +1 starts at position 601 and extends off the end of the sequence. Then the end of that gene/orf will be reported at position 999, as if the stop codon were in positions 997...999. This is true even if the last two characters of the sequence are, say, `cc` and cannot possibly be part of a stop codon.

Any scores associated with orfs that extend past the end of a sequence are computed using only complete codons contained in the sequence.

`-z` n or `--trans_table` n

Use Genbank translation table number n to specify stop codons.

`-Z` *codon-list* or `--stop_codons` *codon-list*

Specify stop codons as a comma-separated list. Sample format: `-Z tag,tga,taa`. The default stop codons are `tag`, `tga` and `taa`.

6.2.2 `glimmer3` Output Formats

`.detail` File

The `.detail` file begins with the command that invoked the program and a list of the parameters used by the program. Here is a sample:

```
Command:  /fs/szgenefinding/Glimmer3/bin/glimmer3 -o 50 -g 110 -t 30 -b iterated.motif -P
0.603,0.338,0.059 tpall.fna iterated.icm iterated

Sequence file = tpall.fna
Number of sequences = 1
ICM model file = iterated.icm
Excluded regions file = none
List of orfs file = none
Input is NOT separate orfs
Independent (non-coding) scores are used
Circular genome = true
Truncated orfs = false
Minimum gene length = 110 bp
Maximum overlap bases = 50
Threshold score = 30
Use first start codon = false
Start codons = atg,gtg,ttg
Start probs = 0.603,0.338,0.059
```

```
Stop codons = taa,tag,tga
GC percentage = 52.8%
Ignore score on orfs longer than 799
```

Following that, for each sequence in the input file the fasta-header line is echoed and followed by a list of orfs that were long enough for `glimmer3` to score. Here is a sample of the beginning of such a section:

```
>gi|15638995|ref|NC_000919.1| Treponema pallidum subsp. pallidum str. Nichols, complete ge
nome
Sequence length = 1138011
```

		Start			Length		Scores							
ID	Frame	of Orf	of Gene	Stop	of Orf	of Gene	Raw	InFrm	F1	F2	F3	R1	R2	R3	NC
	+2	17	20	139	120	117	-4.94	0	99	0	-	0	-	-	0
	+2	140	242	361	219	117	0.99	0	87	0	-	12	-	-	0
	-1	435	417	148	285	267	5.48	2	97	-	-	2	-	-	0
	+2	668	668	790	120	120	2.89	0	99	0	-	-	-	-	0
	-3	899	839	717	180	120	-0.86	1	95	-	-	-	-	1	3
	-1	936	933	808	126	123	0.38	13	78	-	-	13	-	-	8
	-3	1124	1109	918	204	189	-1.32	0	99	-	-	-	-	0	0
0001	+1	4	4	1398	1392	1392	6.61	99	99	-	-	-	-	-	0
	-2	1750	1720	1457	291	261	-0.92	8	-	-	-	-	8	-	91
	-2	1957	1945	1751	204	192	-1.47	1	-	-	70	-	1	-	27
	-3	2078	2063	1908	168	153	-1.88	4	-	-	20	-	-	4	75
	-2	2308	2293	2174	132	117	-0.38	5	-	-	85	-	5	-	9
0002	+3	1542	1641	2756	1212	1113	3.20	99	-	-	99	-	-	-	0
	-3	2807	2774	2616	189	156	-2.08	3	0	-	-	-	-	3	96

Below is a description of the columns. All positions are counted from the beginning of the sequence with the first base being position 1.

`ID` An identification number for a potential gene. Only orfs whose in-frame (`InFrm`) score is above the threshold score (set by the `-t` option) or are longer than the ignore-score length have an entry in this column.

`Frame` The reading frame of the orf—positive for forward strand, negative for reverse strand. It is determined by the position of the leftmost base of the stop codon:

frame $+1$ if the stop begins in position $1, 4, 7, \ldots$;

frame $+2$ if the stop begins in position $2, 5, 8, \ldots$;

frame $+3$ if the stop begins in position $3, 5, 9, \ldots$;

frame -1 if the stop begins in position $3, 5, 9, \ldots$ (so the leftmost base is position $1, 4, 7, \ldots$);

frame -2 if the stop begins in position $4, 7, 10, \ldots$ (left base position $2, 5, 8, \ldots$);

frame -3 if the stop begins in position $5, 8, 11, \ldots$ (left base position $3, 6, 9 \ldots$).

Note that if the genome length is not a multiple of 3, for genes that wrap around the end of the sequence the same rules applied to the start codon position will not yield the same reading frame.

`Start` The positions of the first base of the orf and the first base of the start codon of the gene. Note that the gene start may be different for the same orf in the `.predict` file.

`Stop` Position of the last base of the stop codon.

`Length` Number of bases in the orf and in the gene. It does *NOT* include the bases of the stop codon.

`Raw` Score This is 100 times the per-base log-odds ratio of the in-frame coding ICM score to the independent (*i.e.*, non-coding) model score. It gives a rough quantification to how well an orf scores that can be compared between any two orfs.

`InFrm` Score The normalized (to the range $0 \ldots 99$) score of the gene in its reading frame. This is just the appropriate-frame value among the next six scores.

Frame Scores The normalized (to the range $0 \ldots 99$) score of the gene in each reading frame. A "`-`" indicates the presence of a stop codon in that reading frame. The normalization compares only scores without stop codons and the independent (non-coding) `NC` score. If the orf is sufficiently long, *i.e.*, longer than the value stated in "`Ignore score on orfs longer than...`", the score is not used.

`NC` Score The normalized independent (*i.e.*, non-coding or intergenic) model score. This model is adjusted for the fact that the orf, by definition, has no in-frame stop codons.

`EDR` Score An additional column of scores is produced if the `-E` option is specified. This is the entropy-distance ratio, *i.e.*, the ratio of the distance of the amino-acid distribution from a positive model to the distance from a negative model. Scores below 1.0 are more likely to be genes; scores above 1.0 less likely to be genes. It is not currently used in the scoring process.

`.predict` File

This file has the final gene predictions. It's format is the fasta-header line of the sequence followed by one line per gene. Here is a sample of the beginning of such a file:

```
>gms:3447|cmr:632 chromosome 1 {Mycobacterium smegmatis MC2}
orf00001      499     1692  +1    13.14
orf00004     1721     2614  +2    14.20
orf00006     2624     3778  +2    10.35
orf00009     3775     4359  +1     9.34
```

The columns are:

Column 1 The identifier of the predicted gene. The numeric portion matches the number in the `ID` column of the `.detail` file.

Column 2 The start position of the gene.

Column 3 The end position of the gene. This is the last base of the stop codon, *i.e.*, it includes the stop codon.

Column 4 The reading frame.

Column 5 The per-base "raw" score of the gene. This is slightly different from the value in the `.detail` file, because it includes adjustments for the PWM and start-codon frequency.

6.3 `long-orfs` Program

This program identifies long, non-overlapping open reading frames (orfs) in a DNA sequence file. These orfs are very likely to contain genes, and can be used as a set of training sequences for the `build-icm` program. More specifically, among all orfs longer than a minimum length ℓ, those that do not overlap any others are output. The start codon used for each orf is the first possible one. The program, by default, automatically determines the value ℓ that maximizes the number of orfs that are output. With the `-t` option, the initial set of candidate orfs also can be filtered using entropy distance, which generally produces a larger, more accurate training set, particularly for high-GC-content genomes. Entropy distance is described in [OZWS04].

6.3.1 `long-orfs` Parameters & Options

The format for invoking `long-orfs` is:

`long-orfs` [*options*] *sequence* *output*

where *sequence* is the name of the file containing the DNA sequence to be analyzed and *output* is the name of the output file of coordinates. *sequence* may contain only one sequence. If *output* is "`-`", then the output is directed to standard output.

Possible *options* are:

`-A` *codon-list* or `--start_codons` *codon-list*

Specify allowable start codons as a comma-separated list. Sample format: `-A atg,gtg`. The default start codons are `atg`, `gtg` and `ttg`.

`-E` *filename* or `--entropy` *filename*

Read entropy profiles from *filename*. The format is one header line, then 20 lines of 3 columns each, which is the format produced by the program `entropy-profile` with the `-b` option. The columns are amino acid, positive entropy, and negative entropy, respectively. Rows must be in alphabetical order by amino acid code letter.

The entropy profiles are used only if the `-t` option is specified.

`-f` or `--fixed`

Do *NOT* automatically calculate the minimum gene length that maximizes the number or length of output regions, but instead use either the value specified by the `-g` option or else the default, which is 90.

`-g` *n* or `--min_len` *n*

Set the minimum gene length to *n* nucleotides. This does not include the bases in the stop codon.

`-h` or `--help`

Print the usage message.

`-i` *filename* or `--ignore` *filename*

filename specifies regions of bases that are off limits, so that no bases within that area will be examined. The format for entries in this file is described above for the same option in the `glimmer3` program.

`-l` or `--linear`

Assume a linear rather than circular genome, *i.e.*, there will be no "wraparound" genes with part at the beginning of the sequence and the rest at the end of the sequence.

`-L` or `--length_opt`

Find and use as the minimum gene length the value that maximizes the total *length* of non-overlapping genes, instead of the default behaviour, which is to maximize the total *number* of non-overlapping genes.

`-n` or `--no_header`

Do not include the program-settings header information in the output file. With this option, the output file will contain only the coordinates of the selected orfs.

`-o` *n* or `--max_olap` *n*

Set the maximum overlap length to *n*. Overlaps of this many or fewer bases between genes are not regarded as overlaps.

`-t` *x* or `--cutoff` *x*

Only genes with an entropy distance score less than *x* will be considered. This cutoff is made before any subsequent steps in the algorithm.

`-w` or `--without_stops`

Do *NOT* include the stop codon in the region described by the output coordinates. By default it is included.

`-z` *n* or `--trans_table` *n*

Use Genbank translation table number *n* to specify stop codons.

`-Z` *codon-list* or `--stop_codons` *codon-list*

Specify allowable stop codons as a comma-separated list. Sample format: `-Z tag,tga`. The default stop codons are `tag`, `tga` and `taa`.

7 Versions

7.1 Version 0.10

- Initial release.

References

[BS09] A. Brady and S.L. Salzberg. Phymm and phymmbl: metagenomic phylogenetic classification with interpolated markov models. *Nature Methods*, 6(9):673–676, 2009.

[DBPS07] A.L. Delcher, K.A. Bratke, E.C. Powers, and S.L. Salzberg. Identifying bacterial genes and endosymbiont dna with glimmer. *Bioinformatics*, 23(6):673, 2007.

[DHK+99] A. Delcher, D. Harmon, S. Kasif, O. White, and S. Salzberg. Improved microbial gene identification with GLIMMER. *Nucl. Acids Res.*, 27(23):4636–4641, 1999.

[KS10] D.R. Kelley and S.L. Salzberg. Clustering metagenomic sequences with interpolated markov models. *BMC Bioinformatics*, 11(1):544, 2010.

[OZWS04] Z. Ouyang, H. Zhu, J. Wang, and S.Z. She. Multivariate entropy distance method for prokaryotic gene identification. *J. Bioinformatics & Comp. Biol.*, 2(2):353–373, 2004.

[SDKW98] S. Salzberg, A. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models. *Nucl. Acids Res.*, 26(2):544–548, 1998.