UNIVERSITY OF

1. Introduction

This manual gives an overview of FIESTA, a utility for simple assembly and annotation of EST data using ungapped alignment and BLAST homology searches or for searching EST libraries for query proteins of interest. General details about Command-line options can be found in the RJE Appendices document included with this download. Details of command-line options specific to FIESTA can be found in the distributed readme.txt and readme.html files.

Like the software itself, this manual is a 'work in progress' to some degree. If the version you are now reading does not make sense, then it may be worth checking the website to see if a more recent version is available, as indicated by the Version section of the manual. Check the readme on the website for up-to-date options etc. In particular, default values for options are subject to change and should be

FIESTA has two assembly and annotation pipelines: a protein pipeline based loosely on BUDAPEST and a DNA pipeline for "true" EST assembly.

1.3.3. EST library assembly/annotation

In addition to the main functions, parts of the main FIESTA assembly/annotation pipeline can be run as standalone functions. ESTs can be converted to Reading Frame (RF) translations with est2rf=T:

1. Identify orientation using 5' poly-T or 3' poly-A.
    a. Where poly-AT tail exists, remove, translate in 3 forward RF and truncate at terminal stop codon.
    b. Where no poly-AT tail exists, translate in all six RF.
2. BLAST translations vs. search database with complexity filter on.
    a. If EST has BLAST hits, retain RF translations with desired e-value or better.
    b. If no BLAST hits, retain all RF translations.

Alternatively, translated RFs or other unannotated protein sequences can be given crude BLAST-based annotations using searchdb=FILE sequences with blastann=T. Note that these are simply the top BLAST hit and better annotation would be achieved using HAQESAC (or MultiHAQ for many sequences).

1.4. Getting Help

Much of the information here is also contained in the documentation of the Python modules themselves. A full list of command-line parameters can be printed to screen using the **help** option, with short descriptions for each one:

```
python fiesta.py help
```

General details about Command-line options can be found in the PEAT Appendices document included with this download. Details of command-line options specific to FIESTA can be found in the distributed readme.txt and readme.html files.

If still stuck, then please e-mail me (r.edwards@southampton.ac.uk) whatever question you have. If it is the results of an error message, then please send me that and/or the log file (see Chapter 2) too.

1.4.1. Something Missing?

As much as possible, the important parts of the software are described in detail in this manual. If something is not covered, it is generally not very important and/or still under development, and can therefore be safely ignored. If, however, curiosity gets the better of you, and/or you think that something important is missing (or badly explained), please contact me.

1.5. Citing FIESTA

FIESTA is part of a manuscript in preparation (Jones *et al.* in prep.). Until published, please cite the FIESTA Website.

1.6. Availability and Local Installation

FIESTA is distributed as a number of open source Python modules. It should therefore work on any system with Python installed without any extra setup required. If you do not have Python, you can download it free from www.python.org at http://www.python.org/download/. The modules are written in Python 2.5. The Python website has good information about how to download and install Python but if you have any problems, please get in touch and I will help if I can.

All the required files should have been provided in the download zip file. Details can be found at http://www.southampton.ac.uk/~re1u06/software/ and the accompanying PEAT Appendices document. The Python Modules are open source and may be changed if desired, although please give me credit for any useful bits you pillage. I cannot accept any responsibility if you make changes and the program stops working, however!

Note that the organisation of the modules and the complexity of some of the classes is due to the fact that most of them are designed to be used in a number of different tools. As a result, not all the options listed in the __doc__() (`help`) will be of relevance. If you want some help understanding the way the modules and classes are set up so you can edit them, just contact me.

2. Fundamentals

2.1. Running FIESTA

2.1.1. The Basics

If you have python installed on your system, you should be able to run FIESTA directly from the command line in the form:

```
python fiesta.py seqin=FILENAME
```

To run FIESTA DNA assembly (3.2) with default settings, no other commands are needed. Otherwise, see 2.1.2 and the relevant sections of this manual.

IMPORTANT: If filenames contain spaces, they should be enclosed in double quotes: *data="example file"*. That said, it is recommended that files do not contain spaces as function cannot be guaranteed if they do.

2.1.2. Running specific FIESTA pipelines

To run candidate gene discovery on a single EST dataset:

```
python fiesta.py querydb=FILENAME seqin=FILENAME
```

To run candidate gene discovery on a single dataset:

```
python fiesta.py querydb=FILENAME batch=LIST
```

To assemble and annotate an EST library using a specific protein dataset for annotation:

```
python fiesta.py seqin=FILENAME searchdb=FILENAME annotate=T
```

To execute BLAST-based EST to RF translation/annotation:

```
python fiesta.py seqin=FILENAME est2rf=T
```

To run candidate gene discovery on a single dataset:

```
python fiesta.py seqin=FILENAME blastann=T
```

2.1.3. Options

Command-line options are suggested in the following sections. General details about Command-line options can be found in the RJE Appendices document included with this download. Details of command-line options specific to FIESTA can be found in the distributed readme.txt and readme.html files. These may be given after the run command, as above, or loaded from one or more *.ini files (see RJE Appendices for details).

2.1.4. Running in Windows

If running in Windows, you can just double-click the fiesta.py file. It is recommended to use the `win32=T` option. (Place this command in a file called fiesta.ini.)

2.1.5. Additional programs

FIESTA needs NCBI BLAST installed to run and identified with the `blastpath=PATH` option. (See RJE Appendices for details.)

2.2. Input

As its name implies, FIESTA takes fasta-formatted ttGdCk1‘PPq

- **batch=LIST** : List of EST libraries to search for candidate genes. (Overrides **seqin=FILE**.)

2.3. Output

The main FIESTA output is a series of fasta files and a tab delimited file that maps the sequence identifiers from various parts of the output. The candidate gene discovery algorithm will also output alignments, trees and possibly additional HAQESAC output as outlined in Section 3.1.

2.4. Commandline Options

Commandline options are given in the appropriate sections. The main options are listed in Table 2.1, Table 2.2 and Table 2.3. A full list of commandline options can be found in the readme file or by running:

```
python fiesta.py help
```

Table 2.1. General commandline options.

Option	Description	Default
seqin=FILE	EST file to be processed	None
blastopt=FILE	File containing additional BLAST options for run, e.g. -B F	None
gnspacc=T/F	Convert sequences into gene_SPECIES__AccNum format wherever possible.	False
spcode=X	Species code for EST sequences	None
species=X	Species for EST sequences	None
newacc=X	New base for sequence accession numbers	“” or spcode
searchdb=FILE	Fasta file for GABLAM search of EST translations	None

Table 2.2. Candidate gene discovery options.

Option	Description	Default
batch=LIST	List of EST libraries to search (will use seqin if none given)	
querydb=FILE	File of query sequences to search for in EST library	None
qtype=X	Sequence "Type" to be used with NewAcc for annotation of translations	Hit
assembly=T/F	Assemble EST sequences prior to search	False
haqesac=T/F	HAQESAC analysis of identified EST translations	True
multihaq=T/F	Whether to run HAQESAC in two-phases	True

Table 2.3. EST assembly and annotation software.

Option	Description	Default
minorf=X	Min length of ORFs to be considered	20
minpolyat=X	Min length of poly-AT to be considered a poly AT	10

3. FIESTA Details

This chapter gives more details on the inner workings of FIESTA.

3.1. Candidate Gene Discovery

The primary standalone function of FIESTA is the identification of novel candidate proteins from EST libraries given a list of query proteins. Multiple proteins (**querydb=FILE**) can be searched for in multiple EST libraries (**batch=LIST**) or just a single library (**seqin=FILE**). If multiple libraries are used, then these should already be annotated with species information for correct processing.

Candidate Gene Discovery is executed as follows:

1. Optionally (**assembly=T**), each EST library is pre-processed using the DNA assembly pipeline (3.2).
2. A BLAST search is conducted of the proteins from querydb against each (assembled) EST library. If the species of the hit sequences cannot be established, they are assigned using **spcode=X** and/or **species=X**. For the sequence "type" used in sequence accession number generation, the **qtype=X** variable is used. (See 3.6.)
    a. Hits are then converted to annotated RF translations (3.5) using the querydb as the source of BLAST-based annotation (and truncation).
    b. RF translations from step 3 are then assembled in consensus protein sequences (3.2.3). This eliminates the requirement to pre-assemble to EST libraries, which saves a lot of time.
    c. Assembled translations are re-annotated using the querydb as the source of BLAST-based annotation (3.3). This produces a file *.X.fas for each EST library, where X is set by **qtype=X**.
3. A crude "quick and dirty" alignment and tree (see rje_seq and rje_tree for options) is then generated for each candidate protein by BLASTing it against the original query protein database and all the candidate sequence files produced from searches against EST libraries. These are output into ALN/ and TREE/ directories.
4. Each candidate protein is then re-annotated using alignments and trees generated from processing the sequences of step 3 plus BLAST searches of an additional protein database, set by **searchdb=FILE**. This is performed using the MultiHAQ implementation of HAQESAC. (See HAQESAC manual for details.) This is to determine whether candidates are in fact different members of the same gene family to the candidates. If **haqesac=F** then the actual HAQESAC run will not be performed but a batch file will be made allowing it to be run later. If **multihaq=F** then HAQESAC will not run in the two-phase MultiHAQ mode. (See multihaq.py documentation and HAQESAC manual for more details.)

3.2. FIESTA EST assembly pipeline (FIESTA DNA)

The default settings for this pipeline are based on old TIGR assembly rules over 95%+ identity over 40+nt at each end. By default, each EST is taken in turn and BLASTed against the full EST database. BLAST hits are then used to build up consensus sequences of overlapping ESTs. The next EST is then used for BLAST and the process continues. After a certain number of ESTs are clustered and removed, the BLAST-database is reduced speed up subsequent BLASTs.

The pipeline proceeds as follows:

1. Any sequences with too few defined (non-N) positions are removed. This is defined as the product **minid=X** (the min % identity of overlap) and **minaln=X** (the min overlap for clustering). Other RJE_SEQ filtering options (except non-redundancy) will also be applied if selected. (See RJE_SEQ manual for details.)
2. BLAST-based consensus generation is performed according to the selected assembly mode (**assmode=X**) and output to *.cons.fas:

   a. "One-Query" [Default/**assmode=onequery**]: Each query is taken in turn, in length order, and appropriate BLAST hits are assembled into a consensus before moving on to the next sequence. This is the most efficient mode for anything but small EST libraries. (See 3.2.1)
   b. "GABLAM" [**assmode=gablam**]: Each query is used to generate a large cluster of ESTs that share any BLAST similarity with each other. The whole cluster is then assembled into consensus sequences. This mode can cause problems with large EST libraries as very large clusters can be assembled through a few "sticky" sequences with many BLAST homologues. (See 3.2.2)
   c. "No GABLAM" [**assmode=nogab**]: The BLAST step is ignored and all sequences are converted into consensus sequences via pairwise assembly. (See 3.2.3)
3. Consensus ID mappings for original ESTs are output to *.cons.tdt.
4. If **annotate=T** (default=F) then the consensus ESTs will be annotated using the EST annotation pipeline (3.4).

3.2.1. Query-based consensus generation

This method is an iterative BLAST (GABLAM {Edwards, 2006 #57}) and consensus generation method. Each sequence is taken in turn, longest first, and used as the query for consensus generation:

1. Initially just the query sequence is considered. This is BLASTed against the EST database using GABLAM. By default this is accelerated by performing an ungapped BLAST. (Gapped BLAST can be switched on using **gapblast=T** but it should be noted that the EST assembly does not allow indels.)
2. Each BLAST hit is considered and rejected if there are insufficient identical residues in the local alignments. This is defined as the product **minid=X** (the min % identity of overlap) and **minaln=X** (the min overlap for clustering).
3. Remaining hits are then compared to the query in the orientation identified by BLAST and assembled into a temporary consensus using pairwise consensus assembly (3.2.3).
4. Any sequences not part of the consensus cluster generated in (3) are discarded and steps 1-3 repeated using the new consensus sequence as the query. This continues to cycle, potentially growing the consensus incrementally, until no more sequences are added to the consensus cluster in step 3.
5. The final consensus sequence is then generated from the consensus cluster.

Following consensus generation, all sequences that formed the consensus cluster are removed as future queries and the next query sequence is then processed. If removing consensus cluster sequences causes the EST library to have shrunk by 200+ sequences (**resave=X**) since it was last saved, the reduced library is resaved and formatted for BLAST searching. This speeds up assembly considerably by removing members of large gene families that have already been compared and not clustered with remaining family members.

3.2.2. Full GABLAM consensus generation

NB. This mode is not recommended and exists predominantly for historical reasons. The updated One-Query method seems to be faster and is less prone to memory issues etc.

This method is an iterative BLAST (GABLAM {Edwards, 2006 #57}) and consensus generation method similar to that outlined above (3.2.1). Each sequence is taken in turn, longest first, and used as the initial query for consensus generation:

1. This initial is BLASTed against the EST database using GABLAM. By default this is accelerated by performing an ungapped BLAST. (Gapped BLAST can be switched on using gapblast=T but it should be noted that the EST assembly does not allow indels.)
2. Each BLAST hit is considered and rejected if there are insufficient identical residues in the local alignments. This is defined as the product **minid=X** (the min % identity of overlap) and **minaln=X** (the min overlap for clustering).
3. Remaining hits are then used as queries themselves and steps 1-2 repeated (unless already used as a query in earlier cycles) until all BLAST hits of BLAST hits are identified.

4. The entire BLAST cluster is assembled into multiple consensus sequences using pairwise consensus assembly (3.2.3).

Following consensus generation, all sequences that formed the BLAST cluster are removed as future queries and the next query sequence is then processed. If removing consensus cluster sequences causes the EST library to have shrunk by 200+ sequences (`resave=X`) since it was last saved, the reduced library is resaved and formatted for BLAST searching.

3.2.3. Pairwise consensus assembly

Pairwise sequence assembly in FIESTA makes use of two main concepts: (1) a working "reference" sequence against which other sequences are compared, and (2) a "consensus cluster" of sequences to be combined into the final consensus. This assembly takes a set of input sequence, which may have been pre-processed by one of the BLAST-based methods (above).

Each input sequence is taken in turn, longest first, and used as the initial query for consensus generation:

1. The query is compared to each other sequence in a pairwise fashion. For DNA sequences, both forward and reverse complemented sequences are compared unless the orientation has already been established with BLAST (see 3.2.1) (unless `gabrev=F`). In order to be combined, the two sequences must share a region of at least 40nt or 20aa (`minaln=X`) at 95%+ sequence identity (`minid=X`). This region may wholly or partially overlap the shorter sequence but all overlapping positions are considered without any indels allowed, so a frameshift indel can potentially prevent the algorithm from clustering sequences correctly. Essentially this achieved by sliding one sequence relative to the other until a matching overlap is found.
2. If a suitable overlapping region is identified the hit sequence is added to the consensus cluster. If the new sequence is not entirely contained within the current reference sequence, the reference is extended using the appropriate part of the new sequence.
3. If sequences have been added to the consensus cluster, Steps 1 & 2 are repeated with the new reference sequence. This continues until no new sequences are added to the consensus because either (a) all input sequences are in the consensus cluster, or (b) none of the remaining sequences have the required overlap with the reference sequence.
4. The sequences in the consensus cluster are combined into a consensus sequence using the alignments produced during step 2. At each position, the most frequent (non-N) nucleotide or (non-X) amino acid is used for the consensus. In the case of a tie, the nucleotide/residue from first appropriate sequence added to the consensus cluster is used.
5. If consensus generation is part of the "one query" pipeline (3.2.1), the consensus sequence and consensus cluster are returned. If not, all remaining sequences are fed back into the cycle and steps 1-4 repeated until no more sequences are left.

NB. If combining proteins sequences and the minimum ORF length (`minorf=X`) is smaller than the minimum alignment length (`minaln=X`), then the smaller value is used.

3.3. BLAST-Annotation of protein sequences

This is a very simple BLAST-based annotation process. Each input protein sequence is taken in turn and BLAST searched against the search database (`searchdb=FILE`). If any hits are found, the best hit is used to annotate the sequence as "Similar (e=X) to Y", where X is the e-value of the BLAST hit and Y is the name of the sequence hit. If no BLAST hits are made, the sequence is annotated as "No BLAST hit (e<X) to Y", where X is the BLAST e-value threshold and Y is the name of the search database.

If `truncnt=T` then the query protein will be truncated to the N-proximal methionine if the following conditions are met:

1. The BLAST hit sequence starts with a methionine.
2. The query protein has an N-proximal methionine in the equivalent position to, or more N-terminal than, 1.

3.4. EST Annotation

EST annotation in FIESTA first converts raw EST sequences to reading frame (RF) translations, using poly-A tails and BLAST searches to identify the correct RF(s) (3.5). If **est2rf=T** then annotation will stop here. Otherwise, RF translations will be combined into consensus sequences using the same "One-Query" BLAST-based approach as used for DNA sequences (3.2.1). Consensus generation occurs in three phases:

1. First, sequences sharing BLAST hits during the EST to translated RF conversion are used to make consensus sequences directly using pairwise consensus assembly (3.2.3).
2. Sequences with and without BLAST hits are independently combined using the "One-Query" assembly.
3. Consensus sequences from 2 are combined using the "One-Query" assembly.

This three-stage approach enables different types of translated RF consensus sequences (with respect to BLAST homology) to be identified if desired.

Finally, if **annotate=T**, BLAST-based annotation of protein sequences (3.3) will be executed on the final consensus sequences. This ensures that BLAST e-values match the final consensus sequences.

NB. This protein-based consensus generation is largely obsolete since DNA-based consensus generation was added. The recommended analysis would now be to assemble the ESTs as DNA (see 3.2) and annotate using the EST to RF conversion.

3.5. EST to Reading Frame conversion

ESTs are translated into protein reading frames, using terminal poly-A or poly-T repeats to identify strand orientation where possible. If an EST has a 3' terminal poly-A repeat of 10+ nucleotides (or **minpolyat=X**), that EST is determined to be in the forwards orientation and only forward reading frames are considered. Alternatively, if an EST has a 5' poly-T repeat of 10+ nucleotides (or **minpolyat=X**), only the reverse-complement reading frames are used. In all other cases, all six RFs are used. Where poly-A or poly-T repeats are identified, reading frames are further truncated to the stop codon (if any) nearest the end of the translation, as these ESTs definitely include parts of the 3' UTR. (The assumption is made that no stop codons are introduced by sequencing errors.) Any RF that lacks an ORF with at least 20 non-X amino acids are removed (**minorf=X**).

3.5.1. BLAST-based identification of correct RF translations

All EST RF translations are BLASTed against the protein database (BLASTP e < 10^{-4}, complexity filter on) and hit mapped onto the translations. If one or more translations of a given EST have BLAST hits, any RFs *without* BLAST hits are removed. If no RFs have BLAST hits, all RFs are retained unless **bestorf=T**, in which case only the longest open reading frame (ORF) is used.

Any RF translations with BLAST hits are reduced to those ORFs that have partial overlap with 1+ BLAST hit alignments. If any part of an ORF is involved in a BLAST local alignment, the entire ORF is retained, *i.e.* it is not truncated at the extremities of the BLAST alignment.

Example. A translation has 3 ORFs (*i.e.* protein sequences separated by stop codons). BLAST hits have local alignments to the first ORF and the first half of the second ORF. The translation would be reduced in length by removing the third ORF, which had no BLAST hits.

3.6. Sequence Naming Conventions

During assembly and annotation, RF translations and consensus sequences will be given new names. This is controlled by four commandline options:

- **gnspacc=T/F** : Convert sequences into gene_SPECIES__AccNum format wherever possible. [False]
- **spcode=X** : Species code for EST sequences [None]
- **species=X** : Species for EST sequences [None]
- **newacc=X** : New base for sequence accession numbers ["" or spcode]

This enables the user to specify the species (or species code), allowing sequence species to be recognised for correct processing by HAQESAC, GOPHER etc. If a species is given but no species code, the species code will be generated using the first three letters of the genus and first two of the species, *e.g*. *Emiliania huxleyi* become "EMIHU".

New sequence accession numbers will use newacc as the base and will have an additional code appended depending on the FIESTA pipeline and stage at which they were created (Table 3.1). If no newacc is given, the species code will be used.

Example. If newacc=EG and species="Emiliania huxleyi", the accession of the first DNA consensus sequence would be EGCONS001 and the species code would be EMIHU. If gnspacc=T then the sequence name will be cons_EMIHU__EGCONS001. If no newacc was given, the accession number would be EMIHUCONS001. If renamed as part of the Candidate gene discovery (3.1) and qtype=CAND, the accession number would be EMIHUCAND001.

Table 3.1. Candidate gene discovery options.

Sequence Code	FIESTA Pipeline/stage
qtype=X	Candidate gene discovery.
TRANS	Annotated EST translations.
TOP	ESTs clustered on Top BLAST hits.
HIT	Consensus sequences with BLAST hits.
NON	Consensus sequences without BLAST hits.
CONS	DNA consensus sequence.

4. Appendices

4.1. Troubleshooting & FAQ

There are currently no specific Troubleshooting issues arising with FIESTA. Please see general items in the PEAT Appendices document and contact me if you experience any problems not covered.

4.2. Abbreviations and Glossary

The following terms and abbreviations are used in this manual:

- **GO.** Gene Ontology. (See www.geneontology.org.)
- **PPI**. Protein-protein interaction.

4.3. References

- Amanchy R, Periaswamy B, Mathivanan S, Reddy R, Tattikota SG & Pandey A (2007). "A curated compendium of phosphorylation motifs." *Nat Biotechnol.* **25**(3): 285-6.
- Balla S, Thapar V, Verma S, Luong T, Faghri T, Huang CH, Rajasekaran S, del Campo JJ, Shinn JH, Mohler WA, Maciejewski MW, Gryk MR, Piccirillo B, Schiller SR & Schiller MR (2006). "Minimotif miner: A tool for investigating protein function." *Nat Methods.* **3**(3): 175-7.
- Davey NE, Shields DC & Edwards RJ (2006). "Slimdisc: Short, linear motif discovery, correcting for common evolutionary descent." *Nucleic Acids Res.* **34**(12): 3546-54.
- Davey NE, Edwards RJ & Shields DC (2007). "The slimdisc server: Short, linear motif discovery in proteins." *Nucleic Acids Res* **35**(Web Server issue): W455-9.
- Edwards RJ, Davey NE & Shields DC (2007). "Slimfinder: A probabilistic method for identifying over-represented, convergently evolved, short linear motifs in proteins." *PLoS ONE* **2**(10): e967.
- Neduva V, Linding R, Su-Angrand I, Stark A, de Masi F, Gibson TJ, Lewis J, Serrano L & Russell RB (2005). "Systematic discovery of new recognition peptides mediating protein interaction networks." *PLoS Biol.* **3**(12): e405.
- Neduva V & Russell RB (2005). "Linear motifs: Evolutionary interaction switches." *FEBS Lett.* **579**(15): 3342-5 Epub 2005 Apr 18.
- Neduva V & Russell RB (2006). "Dilimot: Discovery of linear motifs in proteins." *Nucleic Acids Res.* **34**(Web Server issue): W350-5.
- Puntervoll P, Linding R, Gemund C, Chabanis-Davidson S, Mattingsdal M, Cameron S, Martin DM, Ausiello G, Brannetti B, Costantini A, Ferre F, Maselli V, Via A, Cesareni G, Diella F, Superti-Furga G, Wyrwicz L, Ramu C, McGuigan C, Gudavalli R, Letunic I, Bork P, Rychlewski L, Kuster B, Helmer-Citterich M, Hunter WN, Aasland R & Gibson TJ (2003). "Elm server: A new resource for investigating short functional sites in modular eukaryotic proteins." *Nucleic Acids Res* **31**(13): 3625-30.