UNIVERSITY OF Southampton
School of Biological Sciences

GFESSA: Genome-Free EST SuperSAGE Analysis

Richard J. Edwards © 2011.

Contents

Tables

1. Introduction

This manual gives an overview of GFESSA, a utility for processing SuperSAGE data against transcriptome (EST assembly) Fasta files. GFESSA is not currently designed with another user in mind but feel free to try it if you like. If anything is missing or needs clarification, please contact me. The fundamentals are covered in Chapter 2, Fundamentals, including input and output details. Later sections give more details on how the methods work and statistics are generated. 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 GFESSA 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. Furthermore, recent changes may not have found their way into the manual yet. 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 checked in the readme.

Good luck.

Rich Edwards, 2011.

1.1. Version

This manual is designed to accompany GFESSA version 1.0.

The manual was last edited on 19 August 2011.

1.2. Using this Manual

As much as possible, I shall try to make a clear distinction between explanatory text (this) and text to be typed at the command-prompt etc. Command prompt text will be `written in Courier New` to make the distinction clearer. Program options, also called 'command-line parameters', will be **`written in bold Courier New`** (and coloured **`red`** for fixed portions or **`dark`** for user-defined portions, such as file names etc.). Command-line examples will be given in (green) *`italicised Courier New`*. Optional parameters will (if I remember) be `[`in square brackets`]`. Names of files will be marked in coloured normal text.

1.3. Why use GFESSA?

This program is for the automated processing, mapping and identification-by-homology for SuperSAGE tag data for organisms without genome sequences, relying predominantly on EST libraries etc. Although designed for genome-free analysis, there is no reason why transcriptome data from genome projects cannot be used in the pipeline.

GFESSA aims to take care of the following main issues:

1. Removal of unreliable tag identification/quantification based on limited count numbers.

2. Converting raw count values into enrichment in one condition versus another.

3. Calculating mean quantification for genes based on all the tags mapping to the same sequence.

4. The redundancy of EST libraries, by mapping tags to multiple sequences where necessary and clustering sequences on shared tags.

The final output is a list of the sequences identified by the SAGE experiment along with enrichment data and clustering based on shared tags.

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 gfessa.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 GFESSA 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 GFESSA

Until published, please cite the GFESSA Website.

1.6. Availability and Local Installation

GFESSA is distributed as a number of open source Python modules as part of the PEAT (Protein Evolution Analysis Toolkit) package. 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://bioinformatics.ucd.ie/shields/software/peat/ 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 GFESSA

2.1.1. The Basics

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

```
python gfessa.py tagfile=FILENAME seqin=FILENAME
```

To run with default settings, no other commands are needed. Otherwise, see the relevant sections of this manual.

IMPORTANT: If parameter settings contain spaces, they should be enclosed in double quotes: *data=“example file”*. It is recommended that files do not contain spaces as function cannot be guaranteed if they do.

2.1.2. Options

Command-line options are suggested in the following sections. General details about Command-line options can be found in the PEAT Appendices document included with this download. These may be given after the run command, as above, or loaded from one or more *.ini files (see PEAT Appendices for details).

A full list of commandline options can be found in the distributed readme file or by running:

```
python gfessa.py help
```

The main GFESSA commandline options are summarised in Table 2.1.

Table 2.1. Main commandline options for GFESSA.

Option	Default	Description
INPUT OPTIONS		
tagfile=FILE	[None]	File containing SuperSAGE tags and counts
expconvert=FILE	[None]	File containing 'Header', 'Experiment' conversion data
experiments=LIST	[]	List of (converted) experiment names to use
seqin=FILE	[None]	File containing EST/cDNA data to search for tags within
tagindex=FILE	[*.tag.index]	File containing possible tags and sequence names from **seqin** file
tagmap=FILE	[None]	Tag to sequence mapping file to over-ride auto-generated file based on **seqin** and **mismatch**
tagstart=X	['GATC']	Sequence starting tags
PROCESSING OPTIONS		
mintag=X	[6]	Minimum number of counts for a tag to be included (summed replicates)
minenrtag=X	[15]	Minimum number of counts for a tag to be retained for enrichment etc. (summed replicates)
enrcut=X	[2.5]	Minimum mean fold change between experiments
pwenr=X	[1.0]	Minimum fold change between pairwise experiment comparisons
expand=T/F	[True]	Whether to expand from enriched TAGs to unenriched TAGs through shared sequence hits
mismatch=X	[-1]	No. mismatches to allow. -1 = Exact matching w/o BLAST
bestmatch=T/F	[True]	Whether to stop looking for more mismatches once hits of a given stringency found
normalise=X	[ppm]	Method for normalising tag counts within replicate (None/ppm)
OUTPUT OPTIONS		
basefile=X	[TAG file basename]	"Base" name for all results files, e.g. X.gfessa.tdt

2.1.3. Running in Windows

If running in Windows, you can just double-click the gfessa.py file and use the menu prompts to navigate through the program. It is recommended to use the `win32=T` option. (Place this command in a file called gfessa.ini.) **Note:** The menu system may not yet be implemented.

2.2. Input

GFESSA takes the following input files:

1. A file of tag counts from the SAGE experiment (`tagfile=X`). This should have one Tag field (default 'Tag sequence' but can be modified with `tagfield=X`) and the rest of the fields should be different tag count experiments.
2. [Optional] experiment conversion file, set by `expconvert=FILE`. This should contain two fields:
    a. Header. This corresponds to the field headers for the Tag input file above.
    b. Experiment. This contains the new name for the corresponding field.
3. A file of DNA sequences in FASTA format (typically ESTs) for comparison (`seqin=X`).

2.3. Output

There are four main levels of output from a GFESSA run, some of which can be read back in on subsequent runs to save time:

- SuperSAGE Input files:
    - Tag file *.Tag.tdt. This contains counts for each Tag in each experiment, summed counts over all experiments, and a count of the number of different experiments in which the tags are seen.
- Sequence Input files:
    - Tag index file *.X.tag.index, where X is the tag start. This contains all possible Tags in the sequence file and the corresponding sequence IDs. This is used for the BLAST-free method only (mismatch<0).
- Intermediate combined sequence/SAGE files:
- Final GFESSA summary output:
    - *.gfessa.hits.fas. A fasta formatted sequence file of all retained hits.
    - *.gfessa.tags.tdt. A final output for each retained tag, complete with the sequences and sequence/tag clusters onto which it maps.
    - *.gfessa.hits.tdt. A final output for each retained sequence hit, complete with mean enrichment and the tags/clusters onto which it maps.
    - *.gfessa.clusters.tdt. A final summary file containing an entry for each sequence/tag cluster, identifying the tags and sequences that form each cluster.

Explanations of the output fields are given in Table 2.2 and Table 2.3.

Table 2.2. Main fields for GFESSA output files.

Field	Description
Tag	SAGE tag sequence
E1 - En	Tag counts for each experiment (Headers are experiment identifiers then replicate numbers). May be normalised.
Count	Summed tag count over all experiments
ECount	Number of different experiments in which tag is seen
Seq	List of sequences from seqin mapping on to tag.
SeqN	Number of sequences from seqin mapping on to tag
A/B	Ratio of mean normalised counts in condition A / condition B
minabsA/B	Smallest pairwise ratio of normalised count in condition A / condition B
Cluster	Sequence-Tag cluster identifier
CSeqN	Number of sequences in cluster
CTagN	Number of tags in cluster
CSeq	List of sequences in cluster
CTag	List of Tags in cluster
C|A/B	Geometric mean of mean enrichment ratios for all sequences in cluster

Table 2.3. Specific fields for the primary GFESSA output file *.gfessa.hits.tdt.

Field	Description
Seq	Hit sequence name (*.hits.tdt).
Desc	Description of hit sequence (from seqin)
Tags	List of tags mapping on to sequence
E1 - En	Mean normalised tag counts for all tags in sequence.
A/B	Geometric mean of enrichment ratios for individual tags

3. GFESSA Details

This chapter gives more details on the inner workings of GFESSA. The program is split into three main phases, which will be expanded in the following sections:

1. Setup. Reading and reformatting input files.
2. Tag Mapping. Mapping tags from the SAGE experiments onto the ESTs.
3. Tag Normalisation. Converting raw counts into figures that are comparable between experiments.
4. Tag Enrichment. Calculating enrichment values for tags in one experiment versus another.
5. Processing of mapped ESTs. Tags are optionally filtered according to enrichment criteria and then enrichment values are calculated for each hit sequence. Sequences are also clustered based on shared tag mapping.

3.1. GFESSA Setup

The setup phase sets the names of the output files and reads in the input files. Additional processing of the two main input files – the tag count data and the sequence data – is also executed and intermediate outputs produced that can be used for subsequent analyses on the same data. These files are independent of each other, so the tag file output can be used for analyses on different sequences, and the sequence tag index (if produced, see below) can be used for analyses on different tag count data.

3.1.1. Tag File Processing

The produces a tag file (*.Tag.tdt) that contains counts for each tag in each experiment, summed counts over all experiments, and a count of the number of different experiments in which the tags are seen. A file of tag counts from the SAGE experiment (`tagfile=X`) is read in and processed. This should have one Tag field (default 'Tag sequence' but can be modified with `tagfield=X`) and the rest of the fields should be different tag count experiments. An optional experiment conversion file (`expconvert=FILE`) can be used to convert field headings for enrichment calculations (see 2.2 and 3.4). Unless a subset is defined with `experiments=LIST`, all experiments read at this stage will be used for later processing.

The output file is generated with summed counts over all experiments ('Count' field) and the number of different experiments with 1+ tag counts ('ECount') field. Note that if a reduced number of experiments are being used for analysis, these numbers will be recalculated prior to filtering and enrichment.

If this file already exists, it will be read in and used (unless `force=T`).

3.1.2. Generation of Tag Index for Input Sequence File

If no mismatched tags are desired, it is advisable to switch off the BLAST-based matching of tags and sequences in favour of an index-based method (`mismatch=-1`). Note that this mode needs a defined tag starting sequence (`tagstart=X`), default 'CATG' – this sequence is used to identify which sequences map onto which tags.

Each sequence is examined in turn and both forward and reverse-complemented tag start sequences identified (if any). To be included, tags must:

- Start with the sequence specified by `tagstart=X` ['CATG']
- Have a length equal to the tag length specified by `taglen=X` [26 nt]
- Contain no undefined 'n' nucleotides

Each tag matching these requirements will be output to the X.Y.Z.tag.tdt file, where X is the root of the sequence filename, Y is the tag start sequence and Z is the tag length. Each tag is output with a list of the sequence names matching that tag.

If this file already exists, it will be read in and used (unless `force=T`).

3.2. Mapping Tags to Sequences

The next stage is to map the SAGE tags from the Input onto the sequences. If a tag index file was made for BLAST-free mapping during setup (3.1.2) then this will be used for mapping. Otherwise, a BLAST-based approach will be used (3.2.2). Initially, all tags are mapped on to their respective sequences (3.2.1). These mappings are then reduced to the subset meeting the mintag criteria in the normalisation phase, next (3.3).

The product of index-based tag mapping is a file *.TagMap.tdt that contains the following fields:

- Tag. The tag sequence.
- Experiment fields containing raw tag counts.
- Count and ECount fields (see 3.1.1).
- Seq. A list of the sequence names mapping onto that tag.
- SeqN. The number of sequences mapping onto that tag.

If BLAST-based tag mapping (with possible mismatches) is used, the file will be named *.X.TagMap.tdt, where X is the maximum number of mismatches allowed. In this case, each mismatch level will have a SeqX and SeqNX field, with the names and counts of sequences mapping onto that tag with that number of mismatches, in addition to Seq and SeqN, which list the full sets.

These intermediate files can be re-loaded in future runs to save time.

3.2.1. Mapping SuperSAGE tags onto EST/cDNA sequences

If BLAST-free index mapping has been used (see above), mapping the observed tags onto the sequences is a fairly simple case of joining the two processed input files through the tag sequence. Any tags found in the experimental data that were not identified from the sequence file are given empty values. Any potential tags identified in the sequence file but not returned by any experiments are ignored.

3.2.2. BLAST-based mapping of tags onto sequences

If BLAST-based mapping of the tags is used, the input table is expanded and populated using a BLAST search for which the full list of tags is used as a query and the sequence file is used as the search database. Note that this BLAST search is controlled using the search e-value (`blaste=X`) but the results are controlled by the mismatch value (`mismatch=X`) and so GFESSA will report the maximum number of mismatches being returned using the value given. This can be used to adjust the e-value for future runs. In addition, the BLAST results are saved and can be re-used for future runs with a different mismatch threshold.

The BLAST performed is an ungapped blastn without complexity or composition filters. GFESSA reads in the results and compares each tag query with the sequences hit. The number of mismatches is calculated and, if below the cut-off, the sequence is added to the appropriate mismatch field. This therefore produces a file that has all tags matched to all sequences up to a certain mismatch threshold, with the number of mismatches recorded. (If `bestmatch=True` (default) then only sequences with the (joint) fewest mismatches will be analysed for each Tag. If not, all sequences will be used. This is performed during the final processing phase.)

3.3. Normalised Tag Evidence Counts

Because tag data is in the form of raw counts, it might not be entirely comparable between different experiments. GFESSA therefore features an optional normalisation set. (Switch of using `normalise=none`.) At this stage, the minimum tag count filter (`mintag=X`) is imposed to remove possible noise introduced by sequencing artefacts. By default, this is 6, meaning that each tag must have been seen at least six times in total across all input experiments.

Default normalisation is in the form of "ppm" or "parts per million", meaning that all tag counts will be converted into a count per million tags. Tags not meeting the mintage cut-off are first removed, and then the total tag count T is calculated for each experiment. Individual tag counts, n_t, are normalised within each experiment i by the total tag count for that experiment: n_t / T_i

3.4. Tag Enrichment

Next, the normalised values are used for calculating enrichment of individual tags. This is only performed on tags with counts exceeding the minimum enrichment tag count (minenrtag), which is 15 by default. Only tags meeting this criterion are used for further analysis.

Note. Even though the tags rejected at this stage are not used for enrichment per se, they still contribute to the overall tag count for normalisation and so the mintag=X is still important. If no normalisation is performed, however, there is no gain from having a more relaxed mintag=X setting. In other words, mintag=X is there to exclude sequencing artefacts, while minenrtag=X is there to try and exclude the use of noisy tag data for enrichment.

Enrichment is calculated by comparing (normalised) tag counts between different "conditions". This assumes the following experimental setup and results formatting:

- There are at least two different "conditions", e.g. A and B.
- Each condition has one or more replicates in the input.
- Replicates are named by condition and number, e.g. A1, A2, B1, B2.

The expconvert=FILE option can be used to convert field headers into the required format, while experiments=LIST can be used to limit analysis to a subset of all experiments. GFESSA will extract conditions and replicates using this name formatting, so it is important to get it right for enrichment calculations to work.

Enrichment is calculated for all possible pairs of conditions. E.g. for conditions A, B and C, comparisons will be made of A/B, A/C and B/C.

Enrichment for each tag is calculated as the mean value for condition 1 divided by the mean value for condition 2. As zero values are not allowed – they could give infinite enrichment – a minimum mean occurrence value is imposed for each condition. This is arbitrarily calculated for *all* conditions as the minimum observed tag count in *any* experiment divided by the minimum enrichment cut-off (enrcut=X). If enrcut is less than 2.0, then the minimum value is half the minimum observed tag count. This way, non-missing values will always be enriched compared to missing values.

In addition to calculating mean enrichment, the *minimum* enrichment is also calculated. This is the smallest individual experiment count for the enriched condition divided by the largest individual experiment count for the other condition (or the arbitrary minimum value if no counts were observed).

An enrichment file *.enr.tdt is currently produced but not re-read in future runs.

3.5. Processing mapped ESTs

Following enrichment calculation, additional processing is carried out (expanded below):

1. [Optional] tag enrichment filter to only keep results associated with tags enriched in one condition versus another.
2. Generation of sequence-centric rather than tag-centric results, including sequence enrichment based on all tags mapping to a given sequence.
3. Generation of sequence/tag clusters based on tags sharing sequences and sequences sharing tags. This helps to identify and deal with redundancy in the sequence data – a particular problem for EST libraries – as well as possible issues of tags that map onto multiple different sequences due to evolutionary conservation or relaxed mismatch options.

This final phase produces the main output files for GFESSA:

- *.hits.fas. A fasta formatted sequence file of all retained hits.
- *.tags.tdt. A final output for each retained tag, complete with the sequences and sequence/tag clusters onto which it maps.
- *.hits.tdt. A final output for each retained sequence hit, complete with mean enrichment and the tags/clusters onto which it maps.
- *.clusters.tdt. A final summary file containing an entry for each sequence/tag cluster, identifying the tags and sequences that form each cluster.

See output section (2.3) for details of file contents. Explanations of the output fields are given in Table 2.2 and Table 2.3.

3.5.1. Tag enrichment filter

Following tag enrichment, mapped tags undergo an optional additional tag enrichment filter. This reduces the output to only those hit sequences with one or more tags meeting the enrichment cut-offs set by `enrcut=X` and `pwenr=X`. Note that this applies only to tag enrichment and not final sequence enrichment, which is calculated on all tags in the next stage.

Two enrichment cut-offs are used:

- Mean enrichment, `enrcut=X`. This is based on the enrichment ratios generated from the mean of all experiments.
- Minimum pairwise enrichment, `pwenr=X`. This is based on the smallest difference between any pair of compared experiments.

E.g. If `enrcut=2.5` and `pwenr=1.0` then:

- mean of condition 1/ mean of condition 2>= 2.5x
- every condition 1 experiment > every condition 2 experiment

This filter is used to identify enriched tags, which are then used to identify the corresponding sequences. Because sequences can map onto multiple tags and tags can map onto multiple sequences, GFESSA then expands the set of tags and sequences to be considered for final sequence enrichment:

1. Generate list of enriched tags.
2. Generate list of sequences mapping onto enriched tags.
3. Take each sequence and add any non-enriched tags that also map onto that sequence. (Note that these will have met the `minenrtag=X` cut-off.)
4. Add any additional sequences also mapping on to any non-enriched tags added in step 4.
5. Repeat steps 3 and 4 until no new tags or sequences are added.

This expansion can be switched off by setting `expand=F`.

3.5.2. Generation of sequence enrichment

For each sequence, an enrichment value is calculated as the geometric mean enrichment of the individual tags mapping on to that sequence.

3.5.3. Generation of sequence/tag clusters

Sequence/tag clusters are generated by iteratively mapping tags onto a sequence and adding additional sequences shared by those tags and then additional tags mapping onto those sequences:

1. Take a sequence not yet in a cluster and start a new cluster.
2. Generate list of tags mapping onto the sequence in step 1.
3. Take each sequence mapping the tags from the previous step and add any sequences not yet in the cluster.
4. Take each sequence added in the previous step and add any tags that also map onto that sequence and are not yet in the cluster.
5. Repeat steps 3 and 4 until no new tags or sequences are added.
6. Repeat steps 1-5 until all sequences and tags are in a cluster.

Enrichment is calculated for clusters as the geometric mean of all the tags within each cluster.

4. Appendices

4.1. Troubleshooting & FAQ

There are currently no specific Troubleshooting issues arising with GFESSA. 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:

- **EST.** Expressed Sequence Tag.
- **SAGE**. Serial Analysis of Gene Expression.

4.3. References

GFESSA does not currently make use of any published literature. If using GFESSA, please cite the website.