1. Introduction

This manual gives an overview of BUDAPEST, a utility for processing proteomics data with an intensity component, where there is an interest in enrichment in one sample over another. BUDAPEST 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 BUDAPEST 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. Options are added from time to time and not all of them may 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, 2008.

1.1. Version

This manual is designed to accompany BUDAPEST version 1.8.

The manual was last edited on 19 January 2010.

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 BUDAPEST?

Proteomic analysis of EST data presents a bioinformatics challenge that is absent from standard protein-sequence based identification. EST sequences are translated in all six Reading Frames (RF), most of which will not be biologically relevant. In addition to increasing the search space for the MS search engines, there is also the added challenge of removing redundancy from results (due to the inherent redundancy of the EST database), removing spurious identifications (due to the translation of incorrect reading frames), and identifying the true protein hits through homology to known proteins.

BUDAPEST (Bioinformatics Utility for Data Analysis of Proteomics using ESTs) aims to overcome some of these problems by post-processing results to remove redundancy and assign putative homology-based identifications to translated RFs that have been "hit" during a MASCOT search of MS data against an EST database. BUDAPEST utilises an EST annotation and consensus generation program, FIESTA (Fasta Input EST Analysis) for consensus generation and HAQESAC (Homologue Alignment Quality, Establishment of Subfamilies and Ancestor Construction) for additional annotation of ESTs. Details can be found in Chapter 3.

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 budapest.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 BUDAPEST 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 BUDAPEST

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

1.6. Availability and Local Installation

BUDAPEST 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 BUDAPEST

2.1.1. The Basics

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

```
python budapest.py mascot=FILENAME seqin=FILENAME searchdb=FILENAME
```

To run with default settings, no other commands are needed. Otherwise, see 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. 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. Details of command-line options specific to BUDAPEST 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 PEAT Appendices for details).

2.1.3. Running in Windows

If running in Windows, you can just double-click the budapest.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 budapest.ini.) Note: The menu system may not yet be implemented and BUDAPEST has not yet been tested in Windows.

2.2. Input

BUDAPEST takes three main files as input:

- **A MASCOT results file, specified by `mascot=FILENAME`.**
- **The EST sequences used (or, at least, hit by) the MASCOT search, in fasta format, specified by `seqin=FILENAME`.**
- **A protein database for BLAST-based annotation in fasta format, specified by `searchdb=FILENAME`.**

2.3. Output

BUDAPEST produces the following main output files, where X is set by basefile=X (see also 3.5):

- **X.budapest.tdt = main output table of results (see 3.5.1), which:**
  - **Maps ESTs onto final consensus sequences**
  - **Summarises peptide counts for hits and consensus sequences**
- **X.budapest.fas = BLAST-annotated clustered consensus EST translations using FIESTA**
  - **Sequences are provided in lower case with peptides in upper case**
- **X.summary.txt = summary of results from BUDAPEST pipeline, which:**
  - **lists the Hits and RFs removed at various stages of the analysis.**
  - **highlights possible redundancy.**
  - **summarises putative homology-based annotation of hits.**
- **X.details.txt = full details of processing for each original MASCOT hit (see 3.5.2).**

Additional information can also be obtained from the additional sequence files:

- **X.est.fas = subset of EST sequences from EST database that have 1+ hits in MASCOT results.**
- **X.translations.fas = fasta format of translated RF Hits that are retained after BUDAPEST cleanup.**
- **X.fiesta.fas = BLAST-annotated consensus EST translations using FIESTA (pre min. peptide filtering)**
- **X.peptides.tdt = full list of peptide sequences used in identifications, detailing which consensus sequences were identified by that peptide and whether it is unique to a single consensus/cluster, or common to several.**
- **X_HAQESAC/X.* = HAQESAC results files for annotating translated ESTs (`haqesac=T` only)**
- **X_seqfiles/X.cluster*.fas = fasta files of translations and BLAST hits in NR clusters (`clusterfas=T` only)**

Lastly, reformatted MASCOT files are produced, named after the original input file (Y):

- **Y.mascot.txt = header information from the MASCOT file.**
- **Y.mascot.csv = the delimited data portion of the MASCOT file.**

The usual log file is also produced,

2.4. Commandline Options

Commandline options are given in the appropriate sections. A full list of commandline options can be found in the readme file or by running:

```
python budapest.py help
```

The most important commands are given in Table 1.

Table 1. BUDAPEST Commandline Options.

Option	Description	Default
	Input Options	
mascot=FILE	Name of MASCOT csv file	[None]
seqin=FILE	Name of EST fasta file used for search	[None]
searchdb=FILE	Fasta file for GABLAM search of EST translations	[None]
partial=T/F	Whether partial EST data is acceptable (True) or all MASCOT hits must be found (False)	[True]
	Processing Options	
minpolyat=X	Min length of poly-A/T to be counted. (-1 = ignore all)	[10]
minorf=X	Min length of ORFs to be considered	[10]
topblast=X	Report the top X BLAST results	[10]
minaln=X	Min length of shared region for FIESTA consensus assembly	[20]
minid=X	Min identity of shared region for FIESTA consensus assembly	[95.0]
minpep=X	Minimum number of different peptides mapped to final translation/consensus	[0]
	Sequence Formatting Options	
newacc=X	New base for sequence accession numbers	['BUD']
gnspacc=T/F	Convert sequences into gene_SPECIES__AccNum format wherever possible.	[True]
spcode=X	Species code for EST sequences	[None]
	Output Options	
basefile=X	"Base" name for all results files, e.g. X.budapest.tdt	[MASCOT file basename]
hitdata=LIST	List of hit data to add to main budapest table	[prot_mass,prot_pi]
seqcluster=T/F	Perform additional sequence (BLAST/GABLAM) clustering	[True]
clusterfas=T/F	Generate fasta files of translations and BLAST hits in NR clusters	[False]
clustertree=LIST	List of formats for cluster tree output (3+ seqs only)	[text,nsf,png]
fiestacons=T/F	Use FIESTA to auto-construct consensi from BUDAPEST RF translations	[True]
haqesac=T/F	HAQESAC analysis of identified EST translations	[True]
blastcut=X	Reduced the number of sequences in HAQESAC runs to X (0 = no reduction)	[50]
multihaq=T/F	Whether to run HAQESAC in two-phases with second, manual phase	[False]
cleanhaq=T/F	Delete excessive HAQESAC results files	[True]

3. BUDAPEST Details

This chapter gives more details on the inner workings of BUDAPEST. An overview of the analysis is as follows (Figure 1, details follow):

1. **Split the input MASCOT file into the "header" and "delimited" portions, saving as *.mascot.txt and *.mascot.csv, respectively.**
2. **Read in results from *.mascot.csv, assigning peptides to EST "Hits".**
3. **Extract EST Hits from the EST databases and translate Reading Frames (RF) (see 3.1).**
4. **BLAST (Altschul, et al., 1990) RF translations against protein database and remove incorrect RFs (see 3.2).**
5. **For each EST identified by MASCOT, peptides derived from MASCOT searches of MS/MS data ("MASCOT peptides") are mapped onto the corrected RF translations. Any peptides not mapping onto retained translations are removed. (These peptides will match to the raw EST six-RF translation but not the processed translations.) Any ESTs without any supporting peptides that map to RF translations are removed.**
6. **Any translations that have no supporting MASCOT peptide are removed, leaving for each EST identified with MASCOT only those RF translations that have survived the Translation/BLAST filtering *and* have 1+ identified peptides within them. If `rfhits=T` (the default) then all future processing is done on RF translations (hereon called "Hits"). Otherwise, a "Hit" constitutes an EST as a whole, considering all remaining translations together (where an EST has retained 2+ RFs).**
7. **Redundancy is identified within the MASCOT identifications by identifying those Hits for which all identified peptides were also identified in another Hit (see 3.3). Hits are designated "Unique" if none of their peptides are found within another Hit.**
8. **Hits are clustered based on sharing at least one peptide in common.**
9. **Unless `seqcluster=F`, Hits are clustered (see 3.4) according to:**
    a. **Sharing a common top 10 BLAST hit (4)**
    b. **Sharing BLAST-detectable sequence homology.**

    If `clusterfas=T` (default), sequences for each cluster are output along with the top ten BLAST hits for each translation in the cluster, and aligned using CLUSTALW or MUSCLE.
10. **Translations are output for the Non-redundant MASCOT hits into *.translations.fas. These sequences are annotated with "Similar to..." the best BLAST hit, where appropriate. Peptides are marked in upper case on an otherwise lower case sequence.**
11. **Unless `fiestacons=F`, FIESTA (Fasta Input EST Analysis) is used to generate consensus sequences from the EST translations (3.4.2). FIESTA will try to stick RF translations together by overlapping their ends and looking for a match. To be assembled, a region at least `minaln` long must align with `minid` identity.**
12. **Clustered consensus sequences are then further processed by HAQESAC in conjunction with the BLAST search database to produce multiple sequence alignments and phylogenetic trees that can inform more accurate assignment of protein identities.**
13. **A summary of the processed hits is output to *.summary.txt, while a detailed breakdown for each original MASCOT hit, including its top 10 BLAST hits, is given in *.details.txt. (The number of top BLAST hits reported is controlled by `topblast=X`.)**

3.1. EST Translation into Reading Frames

ESTs are translated into protein in all possible 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, 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, 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 lost through by sequencing errors.) Any RF that lacks an ORF with at least 10 non-X amino acids is removed.

These settings can be altered using `minpolyat=X`, which sets the minimum length of ploy-A/T to be counted (-1 = ignore all; default = 10), and `minorf=X`, which sets the minimum non-X length for an ORF to be included (default = 10).

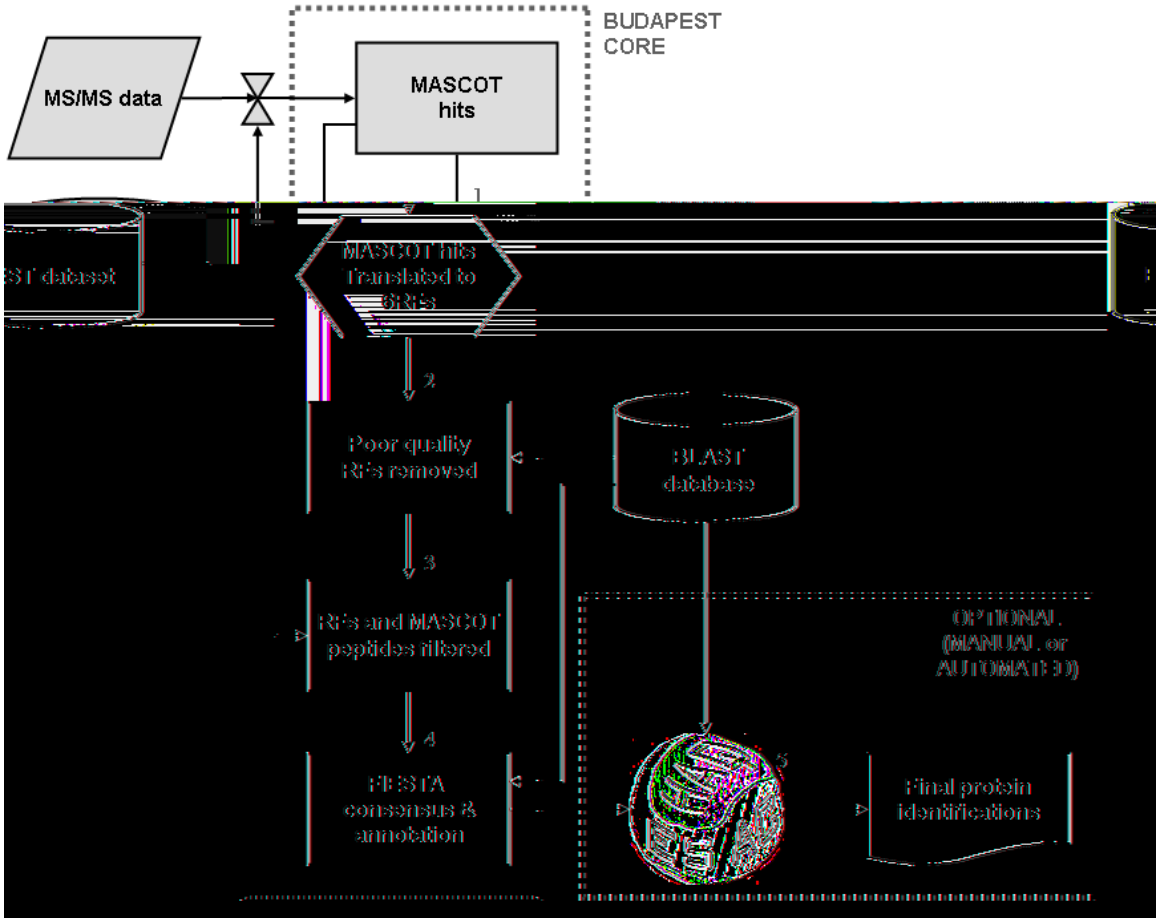

Figure 1. Schematic Summary BUDAPEST workflow.

(1) ESTs identified by the initial Mascot search were translated in all possible reading frames (RFs), using terminal poly-A or poly-T repeats to identify strand orientation where possible. (2) Poor quality translations, predicted to be from an incorrect reading frame on the basis of a BLASTP homology search (e < 10-4) against a protein database (e.g. UniProt), are removed. (3) RF translations without any supporting Mascot peptides and any Mascot peptides matching excluded RFs are removed. (4) The remaining RF translations are combined into consensus sequences based on overlapping regions of 20+ amino acids at 95%+ sequence identity and annotated using similarity (if any) to known proteins as established by a second BLASTP search. (5) Optionally, phylogenetic analysis of consensus sequences using HAQESAC can be used to improve sequence annotation and construct the final list of protein identifications.

3.2. 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 hits mapped onto the translations.

If one or more RF translations for a given EST have BLAST hits, any translations from the same EST *without* **BLAST hits are removed. If no translations have BLAST hits, all translations are retained.**

Any RF translations with BLAST hits are reduced to those Open Reading Frames (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.**

3.3. MASCOT Hit redundancy identification

Redundancy is determined in terms of the remaining "good" peptides that map to 1+ retained RF translations. If all the peptides for a given Hit are found in another Hit, they are considered to be redundant versions of each other and will be marked "Redundant". If a Hit has at least one unique peptide it will be marked "Non-redundant". If all peptides are unique, a Hit is labelled "Unique". The main BUDAPEST output also indicates which hits cluster together based on shared peptide sequences, which can further help identify any redundancy that might have been missed by sequence-based clustering (3.4).

3.4. Sequence based clustering of Hits

To distinguish true redundancy from hits to multiple protein isoforms and/or different members of a gene family, clustering based on shared peptide (3.3) is insufficient. Instead, it is important to align the relevant protein sequences and manually verify whether shared peptides reflect short regions of identity or are due to multiple ESTs encoding the same protein sequence.

To this end, BUDAPEST also sequences hits based on shared homology. Homology is identified both directly, using BLAST to search RF translations against each other, and indirectly, by identifying translations that share BLAST hits (3.2). These clusters, indicated in the main BUDAPEST output table (3.5.1), are output as FASTA files into a `*_seqfiles/` directory (see also 3.5.3).

3.4.1. Manual post-analysis of sequence clusters

Due to the difficulty of aligning partially overlapping sequences and distinguishing polymorphisms or splice variation from sequencing errors (point mutations and frameshifts) in an automated fashion, the recommend strategy for post-processing the sequence clusters produced by BUDAPEST is manual inspection of the sequence alignments using software such as the freely available BioEdit. Using any available information from the aligned BLAST-detected homologues, it should be possible to identify and remove and putative frameshifts (from sequencing errors) and sub-cluster the sequences into distinct isoforms. These sub-clusters can then be combined in consensus sequences for further analysis.

To further place these consensus sequences into the context of their protein families, an analysis package that helps generate quality alignments and phylogenetic trees using a combination of automated rules and manual intervention, such as HAQESAC (Edwards, 2004), can be used. Alternatively, a faster but less accurate approach would be to use automated orthologues prediction, such as GOPHER (Edwards, 2006).

3.4.2. Automatic consensus generation using FIESTA

Instead of the recommended manual analysis of BUDAPEST sequence alignments, retained RF translations can be assembled into consensus sequences in an automated fashion using FIESTA (Edwards, 2009). This requires the following additional commandline options:

- **`minid=X` : Min identity of shared region [95.0]**
- **`minaln=X` : Min length of shared region [20]**
- **`newacc=X` : New base for sequence accession numbers ['']** *E.g. "EHUX"*

FIESTA will try to stick translations together by overlapping their ends and looking for a match. To be assembled, a region at least `minaln` long must align with `minid` identity. More details can be found in the FIESTA manual.

3.5. BUDAPEST Output

BUDAPEST generates three main types of output: a summary results table, summary text files, and sequence files in FASTA format. These are described in more detail below.

3.5.1. BUDAPEST results table

The main results information from BUDAPEST is output into a delimited results file, *.budapest.tdt. This file has a number of output columns (Table 2) plus selected columns from the original MASCOT input file, given with the command `hitdata=LIST` (default prot_mass & prot_pi).

Table 2. Main BUDAPEST output columns.

Column	Description
hit	Reading Frame Hit
est	Parent EST gi number
rf	Reading Frame
trans	Accession No. + RF used during FIESTA Consensus generation
type	UNIQUE/REDUNDANT/NON-REDUNDANT based on shared peptides
pepcluster	Cluster ID based on crude shared-peptide clustering
seqcluster	Cluster ID based on crude shared BLAST hit clustering
prot_mass	Carried over from MASCOT
prot_pi	Carried over from MASCOT
pepseq	Peptide sequences
bad_pep	No. of peptides lost from EST (no good RF hit)
pep_num	Total no. of peptides assigned to RFs from EST
nonrf_pep	No. of peptides assigned to different RF of same EST
rf_pep	No. of peptides assigned to this RF of EST
pepscore	(rf_pep x rf_pep) / pep_num → Above 1 is good.
bud	FIESTA consensus ID
consensus	Clustered BUDAPEST consensus ID
description	Description of consensus based on best BLAST hit
conspep	No. of peptides assigned to consensus
uniqpep	No. of peptides unique to consensus
cluspep	No. of peptides shared within but unique to consensus cluster
commpep	No. of peptides shared between different clusters
exact	No. of peptides with exact match to consensus sequence
coverage	Coverage of consensus sequence by exact matches

3.5.2. BUDAPEST details text files

The *.details.txt file gives full details of processing for each MASCOT hit, including:

- **Redundancy classification**
- **All peptide sequences identified by MASCOT and which are:**
  - **"good" (in a retained RF translation)**
  - **"bad" (wrong RF or outside ORFs with BLAST homology)**
- **Top ten BLAST hits (E-value and description)**
- **How many RFs the EST was initially translated into based on presence (or not) of 5' poly-T or 3' poly-A.**
  - **Which of these RFs were removed due to lack of ORFs of required length**
- **Number of BLAST hits for each RF translation**
  - **Which ORFs were covered by local BLAST alignments and whether RF translations truncated**
  - **RF annotation based on strongest BLAST hit**

  - Which RFs are removed due to lack of BLAST hits
- Which peptides are found in which retained RF translation
- Which other translations shared peptides with this translation
  - Whether translation is deemed redundant based on shared peptides
- Which other translations shared BLAST hits with this translation
- FIESTA consensus sequence identifiers and annotation.

3.5.3. File naming convention

All output files are named after the "root" of the original MASCOT file given with the `mascot=FILE` command. This is indicated by a * in the appropriate sections of this manual. This can be over-ridden with the `basefile=X` option.

E.g. **In the case of `mascot=mydata.txt`, all output would start with mydata (*i.e.* mydata.budapest.tdt, mydata.translations.fas *etc.*)**

3.6. Post-BUDAPEST HAQESAC Analysis

Details available on request.

3.7. iTRAQ Data

BUDAPEST 1.8 can perform some limited compilation and summary of iTRAQ results overlaid on standard EST processing. Details available on request.

4. Appendices

4.1. Troubleshooting & FAQ

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

The most common foreseeable problems will be related to the format of the MASCOT results file and the naming convention of EST sequences interfering with other pattern recognition and processing. If errors indicate that a format problem may be to blame, please contact the author so that the required modifications can be made to the program.

4.2. Abbreviations and Glossary

The following terms and abbreviations are used in this manual:

- **EST. Expressed Sequence Tag. Product of high-throughput sequencing of expressed gene transcripts, often including untranslated regions (UTRs).**
- **Mascot Peptide. Peptide derived from MASCOT searches of MS/MS data.**
- **ORF. Open Reading Frame. Stretch of translated DNA without any Stop Codons.**
- **RF. Reading Frame used to translate DNA in protein *in silico*.**
- **UTR. Untranslated Region of a gene, i.e. it is expressed as mRNA but not part of the coding sequence of the protein.**

4.3. References

Altschul SF, Gish W, Miller W, Myers EW and Lipman DJ (1990). Basic local alignment search tool. J Mol Biol, 215: 403-410.

Edwards RJ (2004). HAQESAC: Homologue Alignment Quality, Establishment of Subfamilies and Ancestor Construction. http://www.bioinformatics.rcsi/~redwards/haqesac/

Edwards RJ (2006). GOPHER: Generation of Orthologous Proteins using High-throughput Evolutionary Relationships. http://bioinformatics.ucd.ie/shields/software/gopher/

Edwards RJ (2009). FIESTA: Fasta Input EST Assembly. RJE Software. University of Southampton, http://www.southampton.ac.uk/~re1u06/software/fiesta/