



Global Alignment from BLAST Local AlignMent

Richard J. Edwards & Norman E. Davey © 2006.

Contents

1. Introduction.....	3
1.1. Version.....	3
1.2. Copyright, License and Warranty	3
1.3. Using this Manual	3
1.4. Why use GABLAM?	3
1.5. Getting Help.....	4
1.5.1. Something Missing?	4
1.6. Citing GABLAM	4
1.7. Availability and Local Installation	4
1.7.1. Programs Used by GABLAM	4
1.7.2. Setting up the INI File	5
2. Fundamentals	6
2.1. Running GABLAM	6
2.1.1. The Basics	6
2.1.2. Interactivity and Verbosity settings	6
2.1.3. Forking.....	6
2.1.4. Other Options	6
2.2. Input	6
2.2.1. Input sequences	6
2.2.2. Input BLAST Results File.....	7
2.3. Output	7
2.3.1. Detailed Results	7
2.3.2. Summary Results	8
2.3.3. Additional Optional Fasta Output	8
2.3.4. Additional Distance Matrix Output	8
2.3.5. Log Files.....	8
2.4. Commandline Options	9
3. The GABLAM Algorithm	10
3.1. Overview of GABLAM.....	10
3.2. GABLAM Options	11
3.2.1. Search Options.....	11
3.2.2. BLAST Options.....	11
3.2.3. Additional ALIGN Global Identity	11
4. GABLAM and Genomics	12
5. Appendices.....	13
5.1. Appendix I: Troubleshooting.....	13
5.2. Appendix II: References.....	13

Figures

Figure 1. Overview of the GABLAM (Global Alignment from BLAST Local AlignMent) method.	10
-------------------------------------------------------------------------------------------------	----

Tables

Table 2.1. Main GABLAM output files.	7
--------------------------------------------------	----------

1. Introduction

This manual gives an overview of **GABLAM**, a utility for processing proteomics data with an intensity component, where there is an interest in enrichment in one sample over another. **GABLAM** 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 **RJE Python Appendices** document included with this download. Details of command-line options specific to **GABLAM** can be found in the distributed [readme.html](#) file and online at bioware.soton.ac.uk.

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 may have been added over the past few weeks and not all 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 **GABLAM version 2.5**.

The manual was last edited on 05 December 2011.

1.2. Copyright, License and Warranty

GABLAM is Copyright © 2006 Richard J. Edwards.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

The GNU General Public License should have been supplied with the **GABLAM** program and is also available at www.gnu.org.

1.3. 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**.

1.4. Why use GABLAM?

Fast, accurate sequence similarity statistics are useful for a number of biological applications. BLAST scores and e-values are quick to generate but are sensitive to protein length and biases introduced by low complexity regions and multiple homologous regions within a protein. Furthermore, these scores are hard to intuitively translate into the extent of relatedness of two proteins in terms of sequence identity (or similarity) or coverage. Pairwise alignment, on the other hand, is slower, and prone to error for low similarity comparisons or proteins with domain rearrangements. **GABLAM** (Global Alignment from BLAST Local AlignMent) is a fast and simple algorithm that converts BLAST local alignments into useful percentage identity, similarity and coverage statistics for both query and hit proteins. These statistics are output in simple delimited tables for easy parsing or uploading into other

applications. **GABLAM** will also run the ALIGN pairwise alignment program for direct comparison, if desired.

1.5. 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 gablam.py help
```

General details about Command-line options can be found in the **RJE Python Appendices** document included with this download. Details of command-line options specific to **GABLAM** can be found in the distributed **readme.html** file and online at bioware.soton.ac.uk.

If still stuck, then please e-mail me (seqsuite@gmail.com) whatever question you have. If it is the results of an error message, then please send me that and the log file (see **Chapter 2**) too.

1.5.1. 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.

1.6. Citing GABLAM

GABLAM was published as part of the SLiMDisc paper. Please cite:

- **Davey NE, Shields DC & Edwards RJ (2006):** SLiMDisc: short, linear motif discovery, correcting for common evolutionary descent. *Nucleic Acids Res.* **34(12):**3546-54.

1.7. Availability and Local Installation

GABLAM is distributed as a number of open source Python modules as part of the SeqSuite package. It should 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 2.7. 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 downloaded zip file. Details can be found at <http://bioware.soton.ac.uk/> and the accompanying **RJE Python 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.

1.7.1. Programs Used by GABLAM

In addition to the python modules listed above, **GABLAM** makes use of the following published programs. These are freely available for downloading and installing. It is recommended that the user downloads and installs these programs according to the instructions given on the appropriate website.

ALIGN: This is part of the Fasta package (Pearson 1994; Pearson 2000) and can be downloaded from the University of Virginia: <http://fasta.bioch.virginia.edu/>. Make sure that align is part of the download. For some reason it seems to have been dropped from later packages. You may need to install an earlier package first (e.g. 2.1) and then a later package. **NB:** ALIGN is optional and the main **GABLAM** functionality does not need it.

BLAST: BLAST (Altschul et al. 1990) is freely available for download from NCBI at: <http://www.ncbi.nlm.nih.gov/blast/download.shtml>.

R. If using the PNG image generation options, a working copy of R will need to be installed. Note that R can be a bit annoying and different installations may need slight tweaks to the R code itself. Please contact the author if this is the case for you.

1.7.2. Setting up the INI File

It is recommended that a **gablam.ini** file is made and placed in the same directory as the **gablam.py** program. This file should contain the paths to the above programs:

```
blastpath=PATH
```

```
fastapath=PATH
```

```
rpath=PATH
```

Note that these are just paths to the programs. See the included **gablam.ini** file for an example. If running in windows, it is also advisable to add the **win32=T** command to the ***.ini** file.

NB. For **PATH** variables, directories should be separated by a forward slash (/). If paths contain spaces, they should be enclosed in double quotes: **path="example path"**. It is recommended that paths do not contain spaces as function cannot be guaranteed if they do.

2. Fundamentals

2.1. Running GABLAM

2.1.1. The Basics

If you have python installed on your system (see 1.7), you should be able to run **GABLAM** directly from the command line in the form:

```
python gablam.py seqin=FILENAME
```

For the example provided in the distribution, enter the data/ directory and run:

```
python ../tools/gablam.py seqin=example.fas
```

If the search database file is different to the input sequence file, use the **searchdb=FILE** option:

```
python gablam.py seqin=query_file searchdb=search_file
```

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

2.1.2. Interactivity and Verbosity settings

By default, **GABLAM** will run through to completion without any user-interaction if given all the options it needs. For more interaction with the program as it runs, use the argument '**i=1**'

```
python ../tools/gablam.py seqin=example.fas i=1
```

Both the level of interactivity and the amount printed to screen can be altered, using the interactivity [**i=x**] and verbosity [**v=x**] command-line options, respectively, where **x** is the level from none (-1) to lots (2+). Although in theory **i=-1** and **v=-1** will ask for nothing and show nothing, there is a good chance that some print statements will have escaped in these early versions of the program. There is also the possibility that accessory programs may print things to the screen beyond the control of **GABLAM**.

Please report any irritations and suggestions for changes to what is printed at different verbosity levels.

2.1.3. Forking

If using a multiple processor machine in UNIX, **GABLAM** can fork out multiple processes to increase processing speed using the **forks=x** option. Each query sequence is processed by a different fork. See the [RJE Python Appendices](#) document for details.

2.1.4. Other Options

At first, you will probably want to run the program with its default parameters. If you want to change them, there are a number of parameters that can be set by the user and other options. These are described in the relevant sections and summarised in the readme. These may be given after the run command, as above, or loaded from one or more *.ini files (see the [RJE Python Appendices](#) document for details).

2.2. Input

2.2.1. Input sequences

The main input for **GABLAM** is two fasta files of DNA or protein sequences:

1. The query sequences (**seqin=FILE**)
2. The search database (**orthdb=FILE**)

To get the most out of the program, one of the set fasta formats from common sequence databases should be used. The included manual for the [rje_seq.py](#) sequence manipulation module has more

details on formats and reformatting/filtering of input sequences. If **orthdb=FILE** is not specified, the query sequence file will also be used as the search database.

Note that if both files are not protein sequences, the **blastp=X** option will need to be set (**blastx** for DNA vs protein; **tblastn** for protein vs DNA).

2.2.2. Input BLAST Results File

Instead of input sequences, **GABLAM** can process a pre-existing BLAST results file. In this case, **ALIGN** cannot be called using **globid=T**. This can be slow and memory intensive for large files and cannot use forking.

2.3. Output

As of **version 2.3**, **GABLAM** outputs four delimited text files (By default, these are tab delimited (***.tdt**) but this can be changed with the **delimit=X** option. If **mysql=T** is used, field headings will be all lower case. These can be converted into MySQL build statements using **rje_mysql.py**.

Table 2.1). Unless over-riden using **basefile=X**, these will all be named **X.***, where **X** takes the form **seqin.vs.searchdb**.

By default, these are tab delimited (***.tdt**) but this can be changed with the **delimit=X** option. If **mysql=T** is used, field headings will be all lower case. These can be converted into MySQL build statements using **rje_mysql.py**.

Table 2.1. Main **GABLAM** output files.

File	Description	Option ¹	Section ²
.gablam.	Detailed summary of pairwise Query-Hit BLAST results.	fullres=T/F	2.3.1
.hitsum.	Simple one-line summary of BLAST hits per query.	hitsum=T/F	2.3.2
.local.	Detailed information for individual local alignments for each query-hit pair. This can get very large for large searches.	local=T/F	TBA
.dismat.	All-by-all distance matrix based on GABLAM results as defined by diskey=X option [default: Qry_OrderedAlnID]. Only generated in seqin and searchdb are the same file.	dismat=T/F	2.3.4

1. Command-line option to switch on/off output file.

2. Manual section with more details of contents.

2.3.1. Detailed Results

For all pair-wise comparisons:

- **Qry** = Query Short Name (or AccNum)
- **Hit** = Hit Short Name
- **Rank** = Rank of that Hit vs Query (based on Score)
- **Score** = BLAST Score (one-line)
- **E-Value** = BLAST E-value
- **QryLen** = Length of Query Sequence
- **HitLen** = Length of Hit Sequence
- **Qry_AlnLen** = Total length of local BLAST alignment fragments in Query (Unordered)
- **Qry_AlnID** = Number of Identical residues of Query aligned against Hit in local BLAST alignments (Unordered)
- **Qry_AlnSim** = Number of Similar residues of Query aligned against Hit in local BLAST alignments (Unordered)
- **Qry_OrderedAlnLen** = Total length of local BLAST alignment fragments in Query (Ordered)

- Qry_OrderedAlnID = Number of Identical residues of Query aligned against Hit in local BLAST alignments (Ordered)
- Qry_OrderedAlnSim = Number of Similar residues of Query aligned against Hit in local BLAST alignments (Ordered)
- Hit_AlnLen = Total length of local BLAST alignment fragments in Hit (Unordered)
- Hit_AlnID = Number of Identical residues of Hit aligned against Query in local BLAST alignments (Unordered)
- Hit_AlnSim = Number of Similar residues of Hit aligned against Query in local BLAST alignments (Unordered)
- Hit_OrderedAlnLen = Total length of local BLAST alignment fragments in Hit (Ordered)
- Hit_OrderedAlnID = Number of Identical residues of Hit aligned against Query in local BLAST alignments (Ordered)
- Hit_OrderedAlnSim = Number of Similar residues of Hit aligned against Query in local BLAST alignments (Unordered)
- ALIGN_ID = Number of Identical residues as determined by pairwise ALIGN (if **globid=T**)
- ALIGN_Len = Length of pairwise ALIGN (if **globid=T**)

If the **alnstats=F** option is used, **GABLAM** stats will not be produced and output will be limited to rank, score e-value and sequence lengths.

2.3.2. Summary Results

For each query protein:

- Qry = Query Short Name (or AccNum)
- Hits = Number of non-self Hits
- MaxScore = Max non-self BLAST Score (one-line)
- E-Value = BLAST E-value for max non-self score

2.3.3. Additional Optional Fasta Output

In addition to the standard output above, **GABLAM** can also output a fasta file per input sequence of all the BLAST hits for that query. This is turned on with the **fasout=T/F** option. The directory for these files, BLASTFAS/ by default, can be changed with the **fasdir=PATH** command.

If **combinedfas=T** then a combined file (**BASEFILE.fas**) will also be produced containing a non-redundant set of all the hits sequences from the individual Query searches.

By default, full length hit sequences will be output. This can be undesirable for long search sequences, such as genomic DNA. If **fragfas=T** then only the regions covered by the local alignments will be output. Note that this can cause issues if a gene has introns, as each exon will be output as a separate sequence.

2.3.4. Additional Distance Matrix Output

If an all-by-all distance matrix is output then a couple of additional outputs will also be produced:

1. A PNG visualisation of summary distance matrix (***.png**). This only really works for small numbers of input sequences (<100). This can be switched off with **dispng=F**. PNG visualisation needs R to be installed.
2. A Newick Standard Format text tree of the distance matrix. This will be named ***.nsf** but can be switch to produce ***.nwk** for ease of use with MEGA using **nsf2nwk=T**. This can be switched off with **disnsf=F**.

2.3.5. Log Files

The **GABLAM** log file records information that may help subsequent interpretation of results or identify problems. Probably it's most useful content is any error messages generated. By default the

log file is `gablam.log` but this can be changed with the `log=FILE` option. Logs will be appended unless the `newlog` option is used. (See the [RJE Python Appendices](#) document for details.)

2.4. Commandline Options

Important commandline options are given in the appropriate sections. A full list of commandline options can be found in the [readme](#) file, online at bioware.soton.ac.uk or by running:

```
python gablam.py help
```

3. The GABLAM Algorithm

3.1. Overview of GABLAM

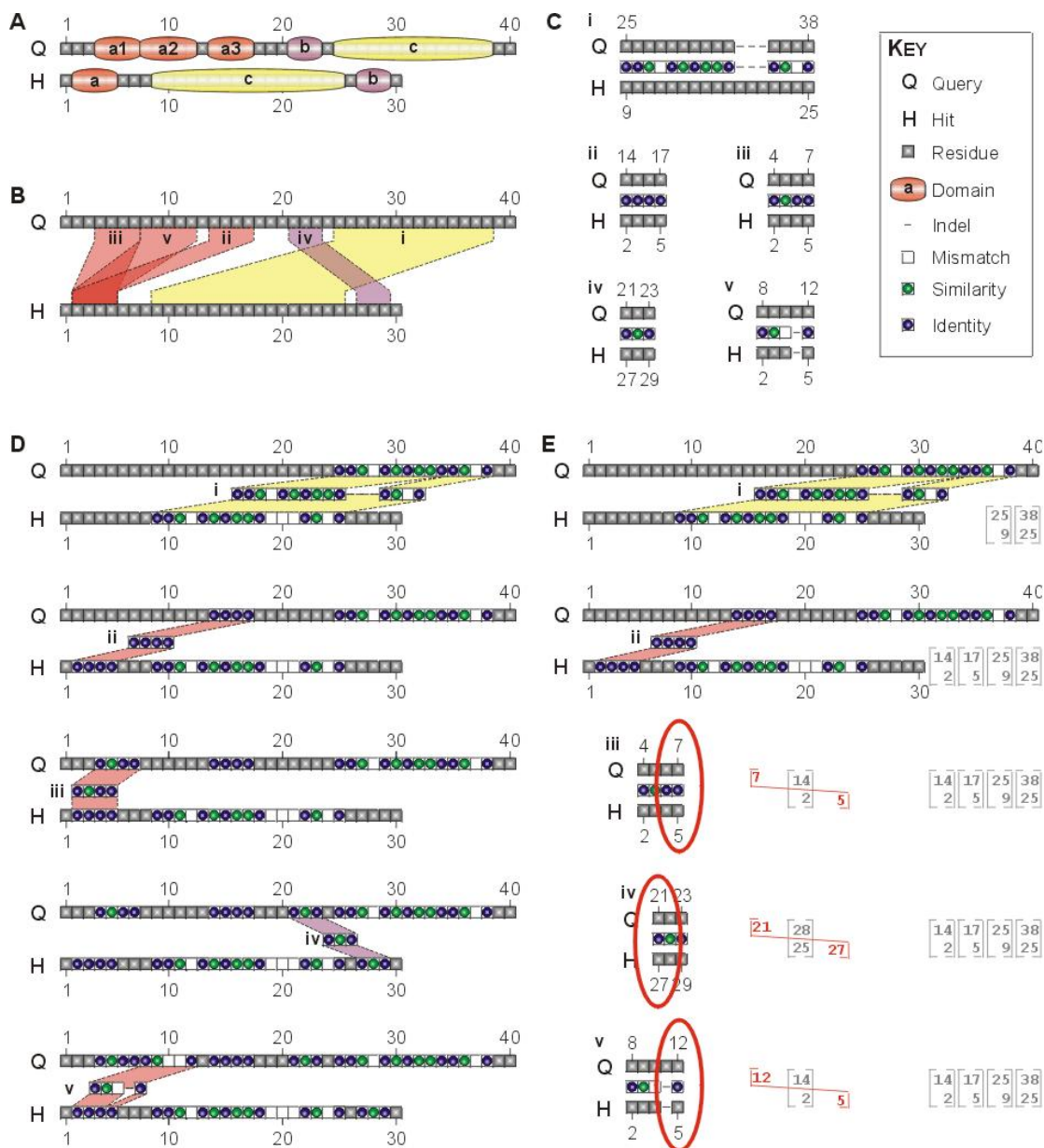


Figure 1. Overview of the GABLAM (Global Alignment from BLAST Local Alignment) method.

(A) Each Query-Hit pair from the BLAST search is compared in a pair-wise fashion. For simplicity, this example has used artificially short sequences and “Domains”, which for this purpose are regions of local homology. (B) BLAST detects five regions of local homology, labelled i-v in order of score, producing (C) five local alignments between regions of query and hit. (D) In the basic GABLAM algorithm, each local alignment is taken in order of BLAST score and mismatches, similarities and identities transposed on to the Query and Hit, assuming that residue has not already been assigned a better status. E.g. when alignment iii is mapped onto the Hit, the similar residue is not transposed at position 3 as an identity is already present. (E) In the ordered algorithm (GABLAMO), each alignment is again considered in order of BLAST score. This time, however, pairs of Query and Hit positions are also stored. If, as in iii, an alignment would split an existing pair of Query-Hit positions, it is judged to be in conflict with the linear ordering of previous alignments and is not added.

3.2. GABLAM Options

3.2.1. Search Options

GABLAM has the following basic search options:

Command	Description	Default
alnstats=T/F	Whether to output full GABLAM alignment stats or limit to BLAST one-line results stats (blastb=0)	True
selfhit=T/F	Whether to include self hits in the output	True
qryacc=T/F	Whether to use the Accession Number rather than the short name for the Query. Will not work when reading pre-computed BLAST results.	True
fullres=T/F	Whether to output the full results table	True
mysql=T/F	Whether to output lower case column headers for a MySQL table build	False
startfrom=X	Accession number to start from.	None
append=T/F	Whether to append to output file or not.	False
fasout=T/F	Output a fasta file per input sequence "ACCNUM.DBASE.fas".	False
fasdir=PATH	Directory in which to save fasta files.	BLASTFAS/

3.2.2. BLAST Options

In addition, the following commands control the BLAST options used:

Command	Description	Default
blastpath=X	Path for blast files. *Use forward slashes.*	c:/bioware/blast/
blaste=X	E-Value cut-off for BLAST searches (BLAST -e X).	1e-4
blastv=X	Number of one-line hits per query (BLAST -v X)	1000
blastb=X	Number of hit alignments per query (BLAST -b X)	1000
blastf=T/F	Complexity Filter (BLAST -F X)	False

3.2.3. Additional ALIGN Global Identity

In addition to GABLAM statistics, GABLAM can use ALIGN (Pearson 2000) to calculate pairwise global alignment statistics with the **globid=T** option. By default, this will be for all pairs of BLAST hits but can be restricted to the top ranked hits using **rankaln=X**, or hits below a given e-value using **evalaln=X**. Alternatively, global percentage identities can be calculated with ALIGN until a hit drops below a percentage identity threshold set by the **alnrcut=X** option.

4. GABLAM and Genomics

For many high-throughput genome-scale analyses, it is useful to have an accurate measure of global (whole length) sequence similarity between pairs of sequences. One of the most obvious applications is for the identification of orthologous proteins, which can be difficult due to the occurrence of gene families that give rise to the presence of multiple homologues in the two (or more) genomes being compared. True orthologue identification requires the use of phylogenetic information, in which outgroup sequences are used to identify the closest clustering homologues in the different species. Phylogenetic analysis can also identify "in-paralogues" - lineage-specific gene duplications - which can hinder compilation of orthologues as they are both true orthologues to the same protein in a different species. For high-throughput analyses, this can be impractical. Identifying suitable outgroups is not a trivial task and phylogenetic inference has its own weaknesses and biases. As a result, pairwise sequence comparisons are routinely used in its place.

One common approach is to use BLAST (Altschul et al. 1990) scores or e-values. For example, orthologues are often identified using the "Mutual Best Hit" (MBH) approach. Under this model, human sequence A and mouse sequence B are only considered orthologues if A has the best score when B is BLASTed against the human genome, and B has the best score when A is BLASTed against the mouse genome. MBH has severe flaws, however. In-paralogues in one species will be missed, with only one identified as an orthologue, while in-paralogues in both the species being compared may disrupt the MBH assignment totally. Using BLAST for this purpose introduces even more problems. BLAST scores are notoriously sensitivity to sequence length and multi-domain proteins. If multiple regions of one protein are homologous to the same region of the other then the BLAST score will be artificially amplified. At the other end of the scale, unrelated proteins will generate BLAST hits if they both contain similar low complexity regions, such as poly-glutamine repeats or leucine-rich regions. This can be avoided using the Complexity Filter but at the risk of artificially decreasing the BLAST scores of closely related sequences with low complexity regions, which are of most interest for orthologue assignment.

To avoid reducing the scores associated with highly similar sequences, one alternative is to leave the Complexity Filter off and use a stricter BLAST score or e-value cut-off to filter out weak homologues. The appropriate threshold can be hard to determine, however, as neither BLAST score nor e-value translate into the degree of similarity in a particularly intuitive fashion. A more intuitive measure is the percentage sequence identity or similarity between two sequences, which can be calculated using pairwise sequence alignment programs, such as ALIGN (Pearson 2000). The disadvantage is that performing multiple pairwise sequence alignments is slow compared to a single BLAST search, which can have serious run-time issues with large genomic analyses. Furthermore, pairwise sequence alignment has well documented drawbacks (Rosenberg 2005). The most obvious is that the program will enforce an alignment, whether the sequences have reasonable similarity or not. As a result, when default settings are used, even random sequences will return a percentage identity of 20% or more for the shorter sequence. Furthermore, if one is interested in the overall similarity of two sequences incorporating domain duplications or rearrangements then pairwise alignment will fail.

We have developed an algorithm, Global Alignment from BLAST Local AlignMent (GABLAM) that returns a set of informative and intuitive pairwise sequence similarity statistics using the results from a basic BLAST search. Unlike ALIGN (Pearson 2000), GABLAM will not return an actual alignment. Instead, GABLAM is ideal when comparing a number of sequences and determining the relative levels of similarity without using phylogenetic inference. GABLAM will compile local sequences alignments generated by BLAST and return both ordered and unordered sequence coverage, identity and similarity independently for query and hit sequences (**Figure 1**). These similarity measures could then be used to generate a distance matrix for much faster phylogeny reconstruction than using Multiple Sequence Alignment, or for selecting the sequences deemed appropriate to align fully.

5. Appendices

5.1. Appendix I: Troubleshooting

Currently, this is a small section as I have not had enough feedback to have FAQs, or anything like that. Here is a list of things that I think MAY cause problems to the unwary:

- Giving file names with spaces or including spaces in paths to programs etc.
- Incorrect formatting of input files.
- I'm not sure when but there is a possibility of problems if running in Windows without the `win32=T` option.

5.2. Appendix II: References

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. J Mol Biol 215:403-10.

Pearson WR (1994) Using the FASTA program to search protein and DNA sequence databases. Methods Mol Biol 24:307-31.

Pearson WR (2000) Flexible sequence similarity searching with the FASTA3 program package. Methods Mol Biol 132:185-219.

Rosenberg MS (2005) Evolutionary distance estimation and fidelity of pair wise sequence alignment. BMC Bioinformatics. 6:102