FARAO – Flexible All-Round Annotation Organizer – Documentation

This is a guide on how to install and use FARAO. The software is written in Perl, is aimed for Unix-like platforms, and should work on nearly all Linux-based systems, as well as MacOS X.

Contents of this manual

1. Detailed installation instructions

Perl needs to be installed on the computer. Most Unix-based systems including Linux and MacOS X have Perl pre-installed. You can check this by opening a command line terminal and type "perl -v". In case Perl is not installed, you have to download (http://www.perl.org) and compile the program.

Go to http://microbiology.se/software/FARAO in order to download the FARAO package. Download it to your preferred directory. Unpack the downloaded tarball with "unzip FARAO_1.0b.zip". A new directory will be created. Enter the directory, and type "chmod -R 755 *". Then type copy all the files into your preferred bin-directory, e.g. "cp -r * /home/user/bin". You may need superuser privileges to complete this action. If FARAO is successfully installed you should see its help message when typing the command "add_annotation --help".

2. Usage and commands

setup_annotation_db

Usage: `setup_annotation_db <New annotation file> <Input file in FASTA format>`

This tool sets up a database file to add annotations to. The command takes two arguments, the name of the new annotation database file, and a file containing sequences in FASTA format that will be associated with the database. This command is the same for setting up *both* annotation databases and read coverage databases (see below). Note that these database files must be kept separately – it is not possible to add both annotation and coverage data to the same database file.

add_annotation

Usage: `add_annotation -a <File to add annotations to> -f <Format of input file> -i <Input file> -t <Annotation type> [-m <Database mapping file (optional)>]`

This command adds annotations based on an output file (usually from some other bioinformatics tool) to the annotation database specified. The command requires four arguments; the name of the annotation database file [-a], the format of the input file (which must be compatible with a FARAO parser, see below) [-f], the path to the input file [-i], and the "annotation type" handle – a free-text string which will later be used to refer to this specific set of annotations in the database [-t]. In addition, a mapping of sequence IDs to e.g. gene names can be specified using the "-m" option. The mapping file format is described in another section of the FARAO documentation.

Main options:

-i {file}	File containing annotations to be added. Required option.
-a {file}	Annotation database file setup using setup_annotation_db. Required option.
-f {string}	Format of the annotation file to be added (use --formats for a complete list of available input formats). Required option.
-t {string}.	Type of prediction to be added. Free text. Will be used when querying the database for information. Can for example be the name of the reference database. Required option.
-m {file}	Mapping file containing sequence IDs and strings that will be assigned to each ID.

Parser options:

-p {string}	Additional options to be supplied to the parser. Use the --formats option to get information of which parsers that utilize this option.
--strange_ID_parsing	If this option is supplied, the software will try to interpret strange FASTA IDs more elaborately.

Information options:

-h	Displays basic usage help.
--help	Displays the help message, complete with all options.

--formats	Displays a list of available input formats and some brief information on those.
--license	Displays licensing information.

get_annotations

Usage: `get_annotations -i <Annotation file> -o <Output type>`

This command outputs annotations stored in the annotation database specified, for all sequences matching the filtration criteria. The command requires one argument; the name of the annotation database file [-i]. There are several filtration options, as well as options for creating PNG and EPS image representations of the selected database sequences.

Main options:

-i {file}	Annotation database file. Required option.

Filtration options:

-s {sequence ID or file}	A specific sequence ID, or a file containing sequence IDs, to look for among the annotated entries. If not specified, output will be created for all entries fulfilling the other filtration options.
-p {string}	Output only annotations of this type. Several types can be specified, separated by comma. Default is to output all annotations for each sequences.
-e {value}	E-value cutoff for a sequence to be included in the output. Default is no cutoff.
-c {value}	Identity cutoff for a sequence to be included in the output (in percent). Default is 0.0
-b {T or F}	Output only the best match at a given position in the database sequence. On (T) by default.
-m {value}	Minimal length of the database sequence to be included in the output. Default = 0
-n {value}	Minimal number of matches to a database sequence passing the filters to be included in the output. Default = 1
-l {value}	Minimal length of the match between the database sequence and the annotated feature. Default = 0
-r {start-stop}	Region to output of the sequence. Default is full-length.
--overlap {value}	Maximum allowed overlap to another feature before only the best feature is shown in the output. Default = 50
--matches {string}	Output database sequences that matches this specific sequence ID.
--select {string}	Only output sequences that match the specified string (for example a gene or query ID).
--select_type {string}	Only output sequences that match this type.
--select_method {AND, OR, XOR, NOT}	Select logical method for selecting using multiple strings. Possible methods are AND, OR, XOR and NOT. Default = AND.
--n_types {value}	Minimal number of types a database sequence must have annotations for to produce output. Default = 1

Output options:

-o list	Outputs a list of matches satisfying the selection options (default).
-o gff	Outputs a list of matches satisfying the selection options, in GFF format.
-o view	Outputs a semi-graphical display (in text format) of matches satisfying the selection options.
-o png	Output graphical representations of matches satisfying the selection options to a PNG files in the directory specified using the --images option. Will also output a list of matches (as for the "-o list" option).
-o eps	Output graphical representations of matches satisfying the selection options to an EPS files in the directory specified using the --images option. Will also output a list of matches (as for the "-o list" option).
-o ids	Output a list of database sequence IDs satisfying the selection options.
-o db	Output a new annotation database in the FARAO format of contigs satisfying the selection options.
-a {list of attributes}	A list of attributes to output for matches (comma-separated). Possible attributes: number,query,length,gc-content,start,end,accession,description,type,identity,evalue,overlap Short forms: n,q,len,gc,start,end,acc,desc,type,id,eval,over Default attributes: number,query,length,start,end,description,identity,evalue
--scale {value}	Scale of the output graph. Default = 0
--separate_types {T or F}	Output matches of different types on different lines. Only valid for the graphical output formats. On (T) by default.

Image options:
Note that these options only apply to PNG and EPS files.

--images {directory}	A directory to save PNG or EPS files in. If this directory does not exist, it is created.
--colors {string}	Allows specification of colors for any annotated property. Format: attribute:value:color;attribute:value:color Patterns can be used for values. Colors either in text or RGB (x,x,x) format. All attributes and values not specified will be set to black (RGB:0,0,0). Available colors specified as text: black, grey, red, green, blue, yellow, orange, violet, moss, cyan. Example: description:MobA:red;description:GyrA:blue;type:NCBI:cyan
--type_colors {string}	Allows specification of colors for different annotation types (as specified by the -p option). Essentially, this option is a subset of the --colors options specific for types. Format: type:color;type:color Example: Pfam:red;NR:green;ORF;grey For colors, see --colors option
--font {string}	Sets the font (EPS output only). Default = Helvetica.
--fontsize {string}	Sets the font size (EPS output only). Default = 9
--max_char	The maximum number of characters that will be outputted for each annotation.

--circular {T or F}	Creates circular (plasmid-like) plots. Works only for EPS output format. Off (F) by default.
--radius {value}	Radius of circle (for circular output). Default = 500
--margin {value}	Margin outside of circle (for circular output). Default = 100
--positions {T or F}	Shows position numbers. Can only be turned off for the EPS output format. On (T) by default.
--annotations {T or F}	Shows annotation texts. On (T) by default.
--annotate {string}	Output one annotation per entry of a certain type from of a list of other types (comma-separated). Example: ORF:PFAM,UNIPROT,GENBANK This will annotate all entries of types ORF with the information from matches in PFAM, UNIPROT and GENBANK, in that order of preference.

Information options:

-h	Displays basic usage help.
--help	Displays the help message, complete with all options.
--license	Displays licensing information.

remove_annotations

Usage: `remove_annotation <File to remove annotations from> <Annotation type> <Keyword to contain for removal (optional)>`

This tool removes annotation entries from the annotation database. The command takes two required arguments, the name of the annotation database file, and a type of annotations to be removed. Optionally, the user can also specify a keyword that annotations must contain to be removed from the database, to enable more specific removal of certain entries.

add_mapped_reads

Usage: `add_mapped_reads -a <File to add reads counts to> -f <Format of input file> -i <Input file> -t <Library>`

This command adds coverage information based on an output file (usually from some other bioinformatics tool) to the coverage database specified. The command requires four arguments; the name of the coverage database file [-a], the format of the input file (which must be compatible with a FARAO parser, see below) [-f], the path to the input file [-i], and the library handle – a free-text string which will later be used to refer to this specific library of reads in the database [-t].

Main options:

-i {file}	File containing output from a read mapper. Required option.
-a {file}	Coverage database file for mapped reads setup using setup_annotation_db. Required option.
-f {string}	Format of the annotation file to be added (use --formats for a complete list of available input formats). Required option.

-t {string}.	Name of the library to be added. Free text. Will be used when querying the database for information. Required option.

Filtration options:

-c {integer}	Minimum percent identity of a read to be added to the mapped reads database. Default = 0
-l {integer}	Minimum length of a read to be added to the mapped reads database. Default = 0

Parser options:

-p {string}	Additional options to be supplied to the parser. Use the --formats option to get information of which parsers that utilize this option.

Information options:

-h	Displays basic usage help.
--help	Displays the help message, complete with all options.
--formats	Displays a list of available input formats and some brief information on those.
--license	Displays licensing information.

get_coverage

Usage: `get_coverage -i <Read coverage file> -o <Output type>`

This command outputs read coverage data stored in the coverage database specified, for all sequences matching the filtration criteria. The command requires one argument; the name of the coverage database file [-i]. There are several filtration options, as well as options for creating PNG and EPS image representations of the selected database sequences.

Main options:

-i {file}	Read coverage database file. Required option.

Filtration options:

-s {sequence ID or file}	A specific sequence ID, or a file containing sequence IDs, to look for among the entries in the database. If not specified, output will be created for all entries fulfilling the other filtration options.
-l {string} --lib {string}	Output only coverage for this library (several can be comma-separated). Default is for all libraries in the database.
-c {value}	Coverage cutoff for a sequence to be included in the output (in coverage per base pair. Default is 0.0
-p {value}	Percent of length covered for a sequence to be included in the output. Default is 0.0
-m {value}	Minimal length of the database sequence to be included in the output. Default = 0
-n {value}	Minimal number of mapped reads to a query sequence to produce output. Default = 1

-r {start-stop}	Region to output of the sequence. Default is full-length.
--matches {string}	Output database sequences that matches this specific sequence ID.
--select {string}	Only output sequences that match the specified string (for example a library or query ID).
--uneven {value}	Only output sequences with a max coverage <value> times larger than the average coverage (e.g. 2). Not used by default.
--n_lib {value}	Minimal number of libraries a database entries must have coverage in to produce output. Default = 1

Output options:

-o list	Outputs a list of matches satisfying the selection options (default).
-o table	Outputs a table of counts satisfying the selection options across different libraries (useable for analysis in e.g. edgeR)
-o view	Outputs a semi-graphical display (in text format) of matches satisfying the selection options.
-o png	Output graphical representations of matches satisfying the selection options to a PNG files in the directory specified using the --images option. Will also output a list of matches (as for the "-o list" option).
-o eps	Output graphical representations of matches satisfying the selection options to an EPS files in the directory specified using the --images option. Will also output a list of matches (as for the "-o list" option).
-o db	Output a new coverage database in the FARAO format of contigs satisfying the selection options.
--scale {value}	Scale of the output graph. Default = 0
--table {string}	The type of values to be outputted in the table (if option '-o table' is used). Can be: total, mean, max, coverage, length. Default = total.

Image options:
Note that these options only apply to PNG and EPS files.

--images {directory}	A directory to save PNG or EPS files in. If this directory does not exist, it is created.
--lib_colors {string}	Allows specification of colors for different libraries. Format: library:color;libarary:color Example: LIBRARY_A:red;LIBRARY_B:green;LIBRARY_C;grey Colors can be specified as text or comma-seprated RGB values (such as 0,0,0 for black). Available colors specified as text: black, grey, red, green, blue, yellow, orange, violet, moss, cyan.
--yscale {value}	Scale of the Y-axis in the output graph. Default = 1
--ymax {value}	Maximum length of Y-axis in the output graph. This option overrides the --yscale parameter. A value of zero means that there is no specified max value for the Y-axis length. Default = 0

Information options:

-h	Displays basic usage help.
--help	Displays the help message, complete with all options.

--license	Displays licensing information.

remove_mapped_reads

Usage: `remove_mapped_reads <File to remove read coverage information from> <Library>`

This tool removes coverage information for certain libraries from the coverage database. The command takes two required arguments, the name of the read coverage database file, and the name of the library to be removed.

estimate_coverage

Usage: `estimate_coverage -i <Read coverage database> -a <get_annotations output file>`

This command outputs read coverage data stored in the coverage database specified, for specific features specified in a supplied output file from get_annotations. The command requires two arguments; the name of the coverage database file [-i] and the output file from get_annotations to read features from [-a]. Note that the output file from get_annotations must be created using the "-a all" option for this command to work properly! It is also worth mentioning that running estimate_coverage can take very long time for large datasets.

Main options:

-i {file}	Read coverage database file. Required option.
-a {file}	A file containing output from get_annotations (with option '-a all') of features to estimate coverage for. Required option.

Filtration options:

-l {string} --lib {string}	Output only coverage for this library (several can be comma-separated). Default is for all libraries in the database.
-m {string}	Method for estimating coverage, can be: mean, median, max, min, bases. Default is mean.
-r {start-stop}	Region to output of the sequence. Default is full-length.
-c {integer or interval}	Number of bases at start and end of the contig to ignore when estimating coverage. If an integer, that number of bases will be ignored at each end, and if the contig is shorter than 2x the integer, the software will return N/A. If an interval, the lower number will be used as the integer above, but if the length of the contig permits it, the number of bases ignored will be increased up to the higher number of the interval. Default is 50-100
-d	Use description rather than feature name.

Information options:

-h	Displays basic usage help.
--help	Displays the help message, complete with all options.
--license	Displays licensing information.

annotation_db_to_mysql

Usage: `annotation_db_to_mysql -a <Database file to convert> -d <MySQL database> -u <MySQL username> -o <New database file> [-p <MySQL password>]`

This command converts a specified annotation database in FARAO format to MySQL format. It requires four arguments; the name of the annotation database file [-a], the address to the MySQL database (for example "farao:localhost" if the MySQL server is running on the same computer) [-d], the MySQL user name [-u] and the name of the new database file to be created [-o]. Usually, a MySQL password must also be supplied using the "-p" option. The conversion process requires a MySQL server to be running, and the DBD and DBI Perl libraries to be installed.

Main options:

-a {file}	Annotation database file setup using setup_annotation_db to convert. Required option.
-d {string}	Address to the MySQL database to add the converted database to. This must include the database name, the host, and optionally the port. For example: 'farao:localhost'. Required option.
-t {string}.	MySQL table name to use for the database. The default is the same as the output file (-o option).
-u {string}	MySQL username. Required option.
-p {string}	MySQL password corresponding to the given username. Can be empty.
-o {file}	New database file to write to. Required option.

Information options:

-h	Displays basic usage help.
--help	Displays the help message, complete with all options.
--license	Displays licensing information.

coverage_db_to_mysql

Usage: `coverage_db_to_mysql -a <Database file to convert> -d <MySQL database> -u <MySQL username> -o <New database file> [-p <MySQL password>]`

This command converts a specified coverage database in FARAO format to MySQL format. It requires four arguments; the name of the coverage database file [-a], the address to the MySQL database (for example "farao:localhost" if the MySQL server is running on the same computer) [-d], the MySQL user name [-u] and the name of the new database file to be created [-o]. Usually, a MySQL password must also be supplied using the "-p" option. The conversion process requires a MySQL server to be running, and the DBD and DBI Perl libraries to be installed.

Main options:

-a {file}	Coverage database file setup using setup_annotation_db to convert. Required option.
-d {string}	Address to the MySQL database to add the converted database to. This must include the database name, the host, and optionally the port. For example: 'farao:localhost'. Required option.
-t {string}.	MySQL table name to use for the database. The default is the same as the output file (-o option).
-u {string}	MySQL username. Required option.
-p {string}	MySQL password corresponding to the given username. Can be empty.
-o {file}	New database file to write to. Required option.

Information options:

-h	Displays basic usage help.
--help	Displays the help message, complete with all options.
--license	Displays licensing information.

3. FARAO Parsers

FARAO comes bundled with a range of parsers for different file formats, but is also designed to be extremely adaptable using custom parsers (see section 6 of this manual). This section will list the parsers bundled with the main FARAO package. More parsers are available as additional downloads from http://microbiology.se/software/farao/parsers

New parsers can be added simply by adding the parser file to the directory "add_annotation_parsers" (for annotation parsers) or the "add_mapped_reads_parsers" directory (for read coverage parsers). Note that these two directories must be located in the same directory as the add_annotation and add_mapped_reads executables for FARAO to find them.

Annotations parsers:

blastn	BLAST output assumed to be in tabular format (blastall option -m 8)
blastp	BLAST output assumed to be in tabular format (blastall option -m 8). This parser assumes direct translation of sequence data into protein sequences using transeq (part of the EMBOSS package)
blastx	BLAST output assumed to be in tabular format (blastall option -m 8)
blastp-prodigal	BLAST is assumed to have been run on gene predictions made by the Prodigal software. BLAST output assumed to be in tabular format (blastall option -m 8). This parser require the FASTA file generated by Prodigal to be supplied as an additional argument (-p).
blat	BLAT output assumed to be in tabular, BLAST-like, format.
fasta	Parses sequences in the FASTA format and adds the sequence as an annotation to the database.
gff	Parser for the GFF format.

hmmsearch	HMMER hmmsearch output assumed to be in normal format (NOT tabular!). An additional option to the parser (-p) can be "description", which will use the descriptions of the HMM-profiles instead of the profile names.
hmmsearch-prodigal	HMMER is assumed to have been run on gene predictions made by Prodigal. HMMER hmmsearch output assumed to be in normal format (NOT tabular!). An additional option to the parser (-p) can be "description", which will use the descriptions of the HMM-profiles instead of the profile names.
prodigal	This parser assumes FASTA output from Prodigal.
rast	This parser reads output from the RAST genome annotation pipeline. Output is assumed to be in tabular format.
text	Parses custom text input from a file in tabulated format: contig_ID <tab> range <tab> text Range should be specified as, e.g., 23-837

Read coverage parsers:

blastn	BLAST output assumed to be in tabular format (blastall option -m 8).
blat	BLAT output assumed to be in tabular, BLAST-like, format.
bowtie2	SAM or BAM format is assumed. Samtools is required and assumed to be installed.
bam	BAM format parser. Samtools is required and assumed to be installed.
razers3	Tabular RazerS3 output assumed
sam	SAM format parser.
text	Custom text input from a file in tabulated format: contig_ID <tab> read_ID <tab> range <tab> percent_identity Range should be specified as, e.g., 23-837. The identity column can be omitted.
vmatch	Parser for vmatch. Output assumed to be in the default vmatch output format.

4. Database mapping files and custom input to FARAO parsers

Database mapping files

Most parsers for the add_annotations FARAO command will accept a database mapping file through the "-m" option, allowing sequence identifiers (or e.g. HMM profile identifiers) to be connected to gene names or descriptions. These files should be organized as plain text files in a two-column tab-separated fashion, with the first column containing the sequence IDs of the entries in the database and the second column containing the corresponding gene names or descriptions. Such a database mapping file could, for example, look like this:

```
gi|489223532|ref|WP_003131952.1| 30S ribosomal protein S18
gi|15674171|ref|NP_268346.1|     30S ribosomal protein S18
gi|116513137|ref|YP_812044.1|    30S ribosomal protein S18
gi|125625229|ref|YP_001033712.1| 30S ribosomal protein S18
```

Additional columns containing more information could be added after those, but these will be ignored by the FARAO parser. A simple way to create a database mapping file from a FASTA file is to simply use a combination of grep, cut and sed to extract the first part of the FASTA header, like this:

```
grep "^>" db.fasta | cut -f 2 -d ">" | sed "s/ /\t/" > db.mapping.txt
```

Custom input to FARAO parsers

Some FARAO parsers also accept or require additional arguments to be able to function correctly. Such arguments are submitted to the parser using the "-p" option. Note that the add_annotations and add_mapped_reads commands only take one single instance of the "-p" argument, and thus any parser requiring several arguments need to parse those through a string without spaces for FARAO to handle the communication between the parser and the main command correctly.

5. Output

get_annotations

The get_annotations command can produce output in the form of a list of sequence IDs, a list of annotations matching the given criteria in FARAO or GFF format, a text-based semi-graphical overview, or graphical output in the PNG or EPS formats. The output formats are briefly described below.

List output in FARAO format

The list output is the standard output format for FARAO. It consists of a tab-separated list, with one annotation entry per line, which can be customized by the user through the "-a" option. Possible attributes to output are: number, sequence ID, sequence length, GC-content, start of annotated feature, end of annotated feature, feature ID, feature description, type of annotation, percent identity, E-value, length of overlap between sequence and feature. By default, the following attributes are written to the output table: number, query ID, length, start, end, description, identity, E-value. If the "-a all" option is used, all fields will be written to output. This is required for use of the estimate_coverage command. The attributes are described below.

Attribute	**Description**
Number	The number of the annotation (recreated for each contig and for each time the command is called).
Sequence ID	The sequence ID of the annotated sequence (often referred to as the contig).
Sequence length	The length of the annotated sequence (contig).
GC-content	The GC-content of the annotated sequence (contig). [Optional]

Start	The start of the annotated feature, i.e. most often the best match in that database in this particular region of the contig.
End	The end of the annotated feature.
Feature ID	The identifier of the annotated feature. Often the sequence ID of the database sequence, but this can also be other kinds of database identifiers. [Optional]
Description	A string describing the annotated feature. Sometimes this is the same as the feature ID. If a database mapping file was used when adding information to the annotation database, this field will contain that information, which could e.g. be gene or species names.
Type	The type of this annotation. Often corresponding to the database the match was derived from. [Optional]
Identity	The percent identity between the annotated sequence (contig) and the database match, if applicable.
E-value	The E-value of the match between the annotated sequence (contig) and the database entry, if applicable.
Overlap length	The length of the overlap between the annotated sequence (contig) and the database feature.

List output in GFF format

The GFF format is read by several other bioinformatics packages. FARAO outputs a simple form of the version 2 of the format. More information about this specification can be found here: https://www.sanger.ac.uk/resources/software/gff/spec.html

GFF output is tab-separated, with the following fields: sequence_ID, match_type, region, start, end, E-value, strand, frame, attributes. FARAO then adds attributes as follows: ID=sequence_ID.x; Name=feature_name; Note=description; Identity=percent_identity; E-value=e_value; GC=gc_content; Length=sequence_length. This scheme is further detailed in the table below. This is an example of a line in a GFF-file generated by FARAO:

```
contig-23401000007  PLASMID      region 258    314    2e-11  +      0
ID=contig_23401000007.5;Name=gi|386076436|ref|NC_017553.1|;Note=gi|386076436
|ref|NC_017553.1|;Identity=91.23;E-value=2e-11;GC=58.58;Length=850;
```

Attribute	**Description**
Sequence ID	The sequence ID of the annotated sequence (often referred to as the contig).
Match type	The type of this annotation. Often corresponding to the database the match was derived from.
Region	Will always be "region" in GFF output from FARAO.
Start	The start of the annotated feature, i.e. most often the best match in that database in this particular region of the contig.
End	The end of the annotated feature.
E-value	The E-value of the match between the annotated sequence (contig) and the database entry, if applicable.
Strand	Strand on which the feature is located. "+" for the forward strand, and "-" for the reverse strand.

Frame	Frame in which the feature occurs. Always zero in FARAO GFF output.	
Attributes	GFF attributes associated with the features. FARAO assigns attributes as follows:	
	ID	The contig ID, with ".x" appended, where x is the number of the feature on each contig.
	Name	The identifier of the annotated feature (feature ID). Often the sequence ID of the database entry.
	Note	A string describing the annotated feature. Sometimes this is the same as the feature ID. If a database mapping file was used when adding information to the annotation database, this field will contain that information, which could e.g. be gene or species names.
	Identity	The percent identity between the annotated sequence (contig) and the database match, if applicable.
	E-value	The E-value of the match between the annotated sequence (contig) and the database entry, if applicable
	GC	The GC-content of the annotated sequence (contig).
	Length	The length of the overlap between the annotated sequence (contig) and the database feature.

Semi-graphical text output format

FARAO can generate a semi-graphical output indented to be used as an overview readable directly in the Unix terminal or as a text file. This output is accompanied by a list of features in FARAO list format (see above) for each contig, positioned below the "graphical" representation of the contig. This is a somewhat edited example of the semi-graphical format generated by FARAO:

```
contig-23401000007   0 ========================================== 850
PRODIGAL_PREDICTION           1:Predicted gene>
PFAM-A                         3:cobW
PLASMID                        2:gi|317053820|ref|NC_01483>4:gi|31>5:gi|38

#  Query               Length Start End Description    Identity E-value
..........................................................................

1  contig-23401000007  850    75    848 Predicted gene
2  contig-23401000007  850    82    142 gi|317053820|  86.89    1e-06
3  contig-23401000007  850    90    705 cobW                    7.2e-55
4  contig-23401000007  850    248   304 gi|317053820|  91.23    2e-11
5  contig-23401000007  850    258   314 gi|386076436|  91.23    2e-11
..........................................................................
```

PNG image output

FARAO generates PNG output using the GD package. In addition to the generated images (which will be placed in the directory supplied to the --images option), FARAO also writes the normal list output to the terminal, which can be customized as normal. The options for manipulating colors and other features of the images are described in the command-line options for "get_annotations" in section 2 above. Note that PNG output requires the Perl package GD (http://search.cpan.org/~lds/GD-2.56/lib/GD.pm) to be installed.

EPS vector graphics output

FARAO generates EPS output using the PostScript::Simple package. In addition to the generated images (which will be placed in the directory supplied to the --images option), FARAO also writes the normal list output to the terminal, which can be customized as normal. The options for manipulating colors and other features of the images are described in the command-line options for "get_annotations" in section 2 above. Note that EPS output requires PostScript::Simple (http://search.cpan.org/~mcnewton/PostScript-Simple-0.09/) to be installed.

FARAO database format

FARAO can also generate output in the form of a FARAO database of sequences satisfying the selection options. This is useful to create sub-databases of, for example, only one specific type of predictions or only the best hits for each prediction type. It is also highly useful for converting a database stored in MySQL to FARAO format. Remember that only the best hits for each prediction type in a given sequence range get saved to the new database unless the "-b" option is turned off ("-b F").

get_coverage

The get_coverage command can produce output in the form of a list of annotations matching the given criteria in FARAO format, a text-based semi-graphical overview, or graphical output in the PNG or EPS formats. The output formats are briefly described below.

List output in FARAO format

The list output is the standard output format for FARAO. It consists of a tab-separated list, with one entry per line, representing a contig/read library combination. For each entry, the following information is included: sequence ID, sequence length, GC-content, average coverage, total mapped reads, and maximum number of mapped reads in one position. The attributes are described below.

Attribute	Description
Sequence ID	The sequence ID of the annotated sequence (often referred to as the contig).
Sequence length	The length of the annotated sequence (contig).
GC-content	The GC-content of the annotated sequence (contig). [Optional]
Average coverage	The average coverage over the entire contig, in mapped reads per basepair.
% covered	The proportion of the contig that has been covered by reads from this sequence library.
Total mapped reads	The total number of reads mapped to this contig from this sequence library.
Max reads	The maximal number of reads mapped to one single position on the contig.

Coverage table output

The table option outputs the coverage for each contig in different read libraries as a matrix, with just the requested coverage number in each entry. These numbers can be the total number of mapped reads, the mean coverage, the maximum coverage of any base in the contig, the percent of the contig covered by reads from each library, or the length of the contig. This can be specified by the "--table" option (see command-line options for "get_coverage" in section 2 above). This is an example of how such a table can look like, in this case showing the percent of the contig length covered (option "--table coverage"):

```
Contig                           set10  set11  set12  set13  set14  set15
set10_167_contig-3745000003      80.32  26.97  48.68  57.20  79.81  90.56
set15_121_contig-184000005       97.57  33.25  49.21  93.23  99.86  98.83
set14_129_contig-106000009       97.38  25.83  12.51  83.03  98.89  93.94
set14_130_contig-1779000002      95.06  33.02  15.21  91.26  98.16  100
```

Semi-graphical text output format

FARAO can generate a semi-graphical output of the coverage plots indented to be used as an overview readable directly in the Unix terminal or as a text file. This output is accompanied by the usual entry for that contig/sequence library combination in FARAO list format (see above) for each contig, positioned below the "graphical" representation of the coverage. This is a somewhat edited example of the semi-graphical format generated by FARAO:

```
set12_228_contig-9932000008   0 ============================ 560
set1                             --------------------------
 3                                 -----------------------
 6                                  ---------------------
 9                                   ----  ------- -----
 12                                  ----  -- ----  ----
 15                                   --      ----   -
 18                                   --      ----
 21                                   --       --
 24                                            --
 27                                            --
 30                                            -
 33
 36

Contig                       Length  GC    Library  Avg. Coverage   % covered
Total mapped reads   Max reads
………………………………………………………………………………………………………………………………………………
set12_228_contig-9932000008  560     28.75 set1    11.66607         91.96
99                   38
……………………………………………………………………………………………………………………………………
```

PNG image output

FARAO generates PNG output using the GD package. In addition to the generated images (which will be placed in the directory supplied to the --images option), FARAO also writes the normal list output to the terminal, which can be customized as normal. The options for manipulating colors and other features of the images are described in the command-line options for "get_coverage" in section 2 above. Note that PNG output requires the Perl package GD (http://search.cpan.org/~lds/GD-2.56/lib/GD.pm) to be installed.

EPS vector graphics output

FARAO generates EPS output using the PostScript::Simple package. In addition to the generated images (which will be placed in the directory supplied to the --images option), FARAO also writes the normal list output to the terminal, which can be customized as normal. The options for manipulating colors and other features of the images are described in the command-line options for "get_coverage" in section 2 above. Note that EPS output requires PostScript::Simple (http://search.cpan.org/~mcnewton/PostScript-Simple-0.09/) to be installed.

FARAO database format

FARAO can also generate output in the form of a FARAO coverage database, containing only sequences satisfying the selection options. This is useful to create sub-databases of coverage of, for example, only reads from a certain library. It is also highly useful for converting a coverage database stored in MySQL to FARAO format.

estimate_coverage

The output from estimate_coverage is divided into two sections. The first section is a copy of the files containing annotation features supplied to the "-a" option, but with an additional column added, containing the requested coverage information for each feature in the file. By default, this column will contain the mean coverage over the feature, but using the "-m" option this can be specified to instead contain the median coverage ("median"), maximum coverage across the feature ("max"), the minimum coverage across the feature ("min") or the covered number of bases ("bases").

The second part of the output follows after a separation line beginning with a "+" and followed by several "=" characters. Each line in this section begins with a "+", making it easy to separate the two parts of the output using e.g. the grep utility. This section contains the coverage information *per feature* regardless of which contig it was on, making it easy to calculate the coverage of e.g. a certain Pfam domain, as in the example below. The table contains the following columns: a plus sign, the identifier or description of the feature, the number of such features encountered on the contigs, the total sum of the (mean – or whatever else specified by the "-m" option) coverage across all reads, and the average coverage divided across the identified features (that is, the average coverage per feature). Note that it most often makes more sense to calculate the total sum rather than the average. Below, an example of this second section is shown, showing the coverage of a few Pfam protein families:

```
+=======================================================
+       Feature         Counts  Total coverage          Average coverage
+       AMP-binding     1       3.82357790601814        3.82357790601814
+       Cation_efflux   1       9.01935483870968        9.01935483870968
+       DDE_Tnp_ISL3    1       5.20512820512821        5.20512820512821
+       DDE_Tnp_Tn3     1       21.991452991453         21.991452991453
+       DUF4158         1       15.5515151515152        15.5515151515152
+       FAD-oxidase_C   1       5.08729139922978        5.08729139922978
+       FAD_binding_4   1       5.08796296296296        5.08796296296296
+       Fic             2       71.469696969697         35.7348484848485
+       Gly_radical     1       3.19444444444444        3.19444444444444
```

6. Custom FARAO parsers

FARAO is written to be easily extended to accept additional file formats, through the use of custom parsers. As described in section 3, several additional parsers can be downloaded from http://microbiology.se/software/farao/parsers

New parsers can be added simply by adding the parser file to the directory "add_annotation_parsers" (for annotation parsers) or the "add_mapped_reads_parsers" directory (for read coverage parsers). Note that these two directories must be located in the same directory as the add_annotation and add_mapped_reads executables for FARAO to find them.

New parsers can be written in *any* programming language, but need to adhere to the following criteria to be functional:

1) The parser must be self-contained, that is it must be able to be executed on its own, using only the path to it, without interpreter information.
2) The parser must be executable.
3) The parser must accept command-line arguments and write its output to standard output.

The first point can generally be accomplished by either compiling the parser into a standalone executable, or by adding a "`#!/path/to/interpreter`" as the first line of the parsing script (supported by the vast majority of scripting language interpreters, such as Perl, Python and Bash).

The second point is a matter of giving the parser file the correct permissions. Generally, this is achieved by executing a command like "`chmod 744 parser_file`" on the parser file.

The third point is important since the main FARAO commands will call the parser using the following form:

```
/path/to/farao/add_annotation_parsers/parse_annotations_format      input_file
additional_options
```

They will then wait for output from the command, sent to standard output in the following format, in the formats described below.

We strongly encourage users to submit functional and tested parsers to the FARAO parser database, by attaching them to an e-mail containing a brief parser description, and sending it to farao@microbiology.se

Annotation parsers (add_annotation)

The add_annotation command expects the parser to return information to standard output in the following format:

```
contig_ID     <>100-250::feature name::feature description::::98.4::1e-45<>
```

This format is specified as follows. First on each line is the sequence ID of the entry in the annotation database (often referred to as the contig ID), followed by a tab. The "<>" symbols indicate the start of each feature annotation. Currently, only one single annotation is allowed for each line returned from the parser, but this might be extended in future versions of FARAO. Backwards compatibility with this system will, however, be ensured in case of such changes. Within each feature annotation, the "::" symbols are used to separate fields. Consequently, feature names or feature descriptions cannot contain the substrings "::" or "<>". The starting "<>" symbols are followed by an interval specifying the base pair range the new feature covers. The next item (after the "::" separator) is the name of the feature, followed by a feature description (which can be left blank). The fourth field is left blank, and is reserved for future development of the parser format (this field will be ignored by the add_annotation commands). The second to last field should contain the percent identity of the feature (a number between 0 and 100), is applicable. This field can be left blank. The last field contains the E-value for the feature, and each line is then terminated by another "<>" before the newline character ("\n").

Read coverage parsers (add_mapped_reads)

The add_mapped_reads command expects the parser to return information to standard output in a similar, but slightly different, format:

```
contig_ID    <>100-250::read_ID:::::98.4::1e-45<>
```

This format is specified as above, but the second item (separated by "::") of each read entry (one per line) should be the read ID, and *both* the third and fourth fields remain unused (and should be left blank).

Recommended practice for parsers

There are a few recommendations and tips for writing FARAO parsers.

- To allow the main programs to display a list of installed parsers and some brief description of them, every line beginning with "# +" in the parser code will be interpreted as information lines by FARAO, and displayed when add_annotation or add_mapped_reads is executed with the "--formats" option.
- The parser can call other software, e.g. for reading the input file (see the BAM parser for an example of this using Samtools).
- Bear in mind that a parser can always exit with an error message to standard error and return *no output* to standard output, which will leave the annotation or coverage database untouched.
- Although any number of additional arguments in theory can be supplied to the parser scripts, the use of more than one additional option is strongly discouraged. This is because the user then has to supply a complete set of options separated by spaces, correctly quoted, when executing the FARAO command. This greatly increases the probability for mistakes.
- The simplest parsers only read one-item-per-line input and generate one-item-per-line output. For an example of a more complicated parser, see the HMMER hmmsearch parser supplied with FARAO.

Parser examples

There are parser examples written in Perl, Python and Bash bundled with the FARAO package. These example files are located in the "parser_examples" directory. These can be used as a starting point for building complete parsers for any output format.

7. MySQL compatibility

FARAO supports MySQL through the DBI and DBD Perl modules. Note that all databases need to be setup as regular FARAO databases using setup_annotation_db. They can then be converted to MySQL format using the annotation_db_to_mysql and coverage_db_to_mysql commands. This will create a file representing the MySQL database, which will contain the information FARAO needs to connect to the MySQL server. This file is essential for the functionality of FARAO and cannot be removed. Databases stored in MySQL format can be dumped to the regular FARAO format using the following commands:

For annotation databases:

```
get_annotations -i <Database in MySQL format> -o db -b F > farao_database
```

For coverage databases:

```
get_coverage -i <Database in MySQL format> -o db > farao_database
```

Our internal testing suggests that for smaller databases (less than 10000 sequences), the regular FARAO format for databases is usually somewhat faster to work with. For really large, databases (several million sequences), MySQL is faster for the vast majority of tasks. In the range between, performance is highly dependent on what kind of task the user wants to accomplish. However, we have also noticed that the MySQL implementation of add_mapped_reads is particularly slow, and there is probably room for future improvement of this code.

It is worth mentioning that storing the databases in MySQL introduces more dependencies on DBD, DBI and MySQL versions, and that all such combinations cannot be fully supported by the developers. Also, use of the MySQL format reduces the portability of databases between computers and users.

8. License information