Date: 2011-03-31

JCVI Prokaryotic Metagenomics Annotation Pipeline (JPMAP)

Process Documentation

Data Snapshot Directory

File	Date	Description
ALL_LIB.HMM	12/09/2010	HMMER3 model library collection (contains TIGRFAM and PFAM models)
hmm3.db	12/10/2010	Sqlite database with HMMER3 model meta-information stored in two tables (hmm3 and hmm_go_link). Information stored includes Iso-Type, EC, GO, cut offs, etc.. **Used in steps 2.2 and 2.3.**
synonyms.tab	12/21/2010	Tab-delimited file with synonym pairs. Column 1 non-preferred name; Column 2 preferred name. The file is used to replace non-preferred with preferred names in the final annotation process. Names have been exported from the Protein Naming Utility http://www.jcvi.org/pn-utility/web/download/download_full_matches.php **Used in step 6.3.**
Uniref100.db	11/30/2010	Sqlite database with selected UniRef100 information stored in the annotation table. Column 1 accession; Column 2 key; Column 3 value. Keys that are stored are gene symbol (GN), GO, EC, CAZY, reviewed state (Swissprot/TrEMBL). **Used in step 3.3.**
Uniref100.tab	11/30/2010	Tab delimited file with comprehensive UniRef 100 cross-reference information. Column 1: accession; Column 2: key; Column 3: value. A subset of cross references described in http://www.uniprot.org/docs/dbxref are available
Uniref100.fasta	11/30/2010	Used as subject database for the blastp searches. **Used in step 3.1** This file contains all UniRef100 entries in FASTA format. The definition line in the FASTA format includes cluster specific information such as cluster name, number of members and and common taxonomy and also the ID of the representative protein. The format is as follows: >UniqueIdentifier ClusterName n=Members Tax=Taxon RepID=RepresentativeMember where: UniqueIdentifier is the primary accession number of the UniRef cluster. ClusterName is the name of the UniRef cluster. Members is the number of UniRef cluster members. Taxon is the scientific name of the lowest common taxon shared by all UniRef cluster members. RepresentativeMember is the entry name of the representative member of the UniRef cluster. For example: >UniRef100_P99999 Cytochrome c n=5 Tax=Hominidae RepID=CYC_HUMAN

Step 1 Split Sequences

Split sequences for parallel searching (Steps 2-6)

executable	`split_multifasta.pl`
input	`fasta file`
output	`multiple split fasta files`
command	`split_multifasta.pl --input_file=input.fasta --output_dir=/tmp --output_list=/tmp/split.list --output_file_prefix='split_' --seqs_per_file=50000 --compress_output=0`

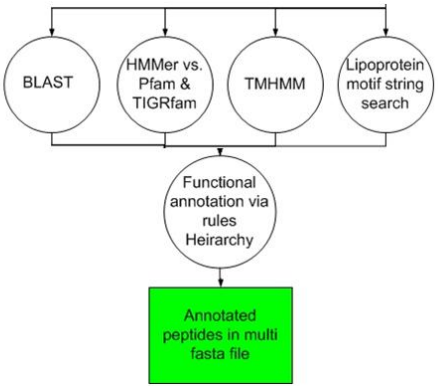

Figure 1 Overview (Steps 2-5 are executed in parallel)

Step 2 HMMER3 Component (Pfam & TIGRfam)

Step2.1 Run HMMER3 search

executable	hmmscan
version	HMMER 3.0
data dependencies	formatted ALL_LIB.HMM
input	split fasta file
output	HMMER3 raw output (hmmer3.outr_0)
command	hmmscan --cut_tc -Z 15908 -o <tmp-dir>/hmmer3.outr_0 --tblout <tmp-dir>/hmmer3.SeqHits.tblr_0 --domtblout <tmp-dir>/hmmer3. DomainHits.tblr_0 <HMM3-db-dir> <input-file>

Step 2.2 Parse HMMER3 results; generate tab delimited file (JCVI HTAB)

Parses output files generated by hmmscan. Uses sqlite database (hmm3.db) to fetch HMM meta-information (HMM Iso-Type, cut offs, etc.).

executable	htab.pl
data dependencies	sqlite database hmm3.db
input	split HMMER3 raw files (hmmer3.outr_0)
output	split HTAB files (hmmer3.htab)
command	cat hmmer3.outr_0| perl htab.pl -d <snapshot-dir>/hmm3.db > hmm3.htab

Step 2.3 Parse JCVI HTAB

Performs HMM annotation lookups for common name, gene symbol, GO, and EC assignments from a sqlite database (hmm3.db). Classifies HMM hits based on HMM Iso-Types (10 classes, see box below).

executable	camera_parse_annotation_results_to_text_table.pl
data dependencies	sqlite database hmm3.db
input	JCVI HTAB (hmm3.htab)
output	JCVI HTAB parsed(hmm3.htab.parsed)
command	perl camera_parse_annotation_results_to_text_table.pl --input_file hmm3.htab --input_type HTAB --output_file hmm3.htab.parsed --work_dir <snapshot-dir>

HMM ISO-TYPES

```
if ($iso_type =~ /^(equivalog)$|^(PFAM_equivalog)$/) {
        $type .= 'Equivalog';
     } elsif ($iso_type =~ /^(hypoth_equivalog)$/) {
        $type .= 'HypotheticalEquivalog';
     } elsif ($iso_type =~ /^(exception)$/) {
        $type .= 'Exception';
     } elsif ($iso_type =~ /^(subfamily)$/) {
        $type .= 'Subfamily';
    } elsif ($iso_type =~ /^(superfamily)$/) {
        $type .= 'Superfamily';
     } elsif ($iso_type =~ /^(equivalog_domain)$|^(PFAM_equivalog_domain)$/) {
        $type .= 'EquivalogDomain';
     } elsif ($iso_type =~ /^(hypoth_equivalog_domain)$/) {
        $type .= 'HypotheticalEquivalogDomain';
     } elsif ($iso_type =~ /^(subfamily_domain)$/) {
        $type .= 'SubfamilyDomain';
     } elsif ($iso_type =~ /^(domain)$/) {
        $type .= 'Domain';
     } elsif ($iso_type =~ /^(PFAM)$/) {
        $type .= 'Uncategorized';
     } else {
        $type = '';
     }
```

PFAM HMM custom mapping (provided by Dan Haft)

```
if($hmm3Result->{hmm_acc} =~ /^PF/) {
          if($hmm3Result->{iso_type} =~ /^Domain$/) {
                    $hmm3Result->{iso_type} = 'domain';
          }
          elsif($hmm3Result->{iso_type} =~ /^Motif$/) {
                    $hmm3Result->{iso_type} = 'domain';
          }
          elsif($hmm3Result->{iso_type} =~ /^Family$/) {
                    $hmm3Result->{iso_type} = 'PFAM';
          }
}
```

Step 3 BLAST Component

Step 3.1 Run BlastP

Run blastp on individual fasta split files and generate JCVI BTAB format from blast XML output (-m 7 option)

executable	blastall
input	split fast file
output	Blast results in XML format
command	blastall -v 20 -b 20 -X 15 -e 1e-5 -M BLOSUM62 -J F -K 10 -f 11 -Z 25.0 -W 3 -U F -I F -E -1 -y 7.0 -G -1 -A 40 -Y 0.0 -F "T" -g T -p blastp -z 1702432768 -m 7'

Step 3.2 Convert XML files to JCVI tab delimited blast result files (BTAB)

executable	blast_xml_to_btab.pl
input	XML formatted blastp results

output	Tab-delimited blastp results (BTAB)
command	blast_xml_to_btab.pl < blastp.xml > blastp.btab

Step 3.3 Parse JCVI BTAB

Perform UniRef100 defline lookups (sqlite database uniref.db) for gene symbol, GO, EC, CAZY, and reviewed status (Swissprot or TrEMBL entry). Classifies blastp hits based on sequence coverage and identity (5 classes, see box below).

executable	camera_parse_annotation_results_to_text_table.pl
data dependencies	sqlite database uniref.db
input	JCVI BTAB (blastp.btab)
output	JCVI BTAB parsed
command	perl camera_parse_annotation_results_to_text_table.pl --input_file blastp.btab --input_type BTAB --output_file blastp.btab.parsed --work_dir <snapshot-dir>

BLAST Categories

```
if ($pct_id >= 35 && $pct_cov >= 80) {
        if($isReviewed) {
                return "UnirefBLASTP::Reviewed";
        }
        else {
                return "UnirefBLASTP::HighConfidence";
        }
} elsif ($pct_id < 35 && $pct_cov >= 80) {
        return "UnirefBLASTP::Putative";
} elsif ($pct_id >= 35 && $pct_cov < 80) {
        return "UnirefBLASTP::ConservedDomain";
}
else {
        return "UnirefBLASTP::LowConfidence";
}
```

Step 4 Lipoprotein Motif Search

Step 4.1 Run lipoprotein motif search

Scans for membrane lipoprotein lipid attachment sites on amino acid sequence. Uses PROSITE motif (^.{0,6}[KR]).{0,18}[^DERK][^DERK][^DERK][^DERK][^DERK][^DERK][LIVMFWSTAG][LIVMFWSTAG][LIVMFWSTAGCQY][AGS]C).

executable	lipoprotein_motif.pl
input	split fast file
output	BSML formatted file
command	lipoprotein_motif.pl --input split1.fasta --output lipoprotein_out.bsml --gzip_output 0 --id_repository workflow/project_id_repository --is_mycoplasm 0

Step 4.2 Parse lipoprotein motif results

executable	camera_parse_annotation_results_to_text_table.pl
input	BSML formatted file (lipoprotein_out.bsml)
output	BSML parsed file (lipoprotein_out.bsml.parsed)
command	camera_parse_annotation_results_to_text_table.pl --input_file lipoprotein_out.bsml --input_type LipoproteinMotifBSML --output_file lipoprotein_out.bsml.parsed /peptide.fasta.q1_q10_1532122841942589727.bsml.parsed --work_dir /tmp

Step 5 TMHMM Search

Scans proteins for trans-membrane domains.

Step 5.1 Run TMHMM

executable	tmhmm
version	2.0
input	split fast file
output	tmhmm_out.raw
command	tmhmm split1.fasta > tmhmm_out.raw

Step 5.2 Parse TMHMM results

executable	tmhmm2bsml.pl
input	TMHMM raw file (tmhmm_out.raw)
output	BSML formatted file (tmhmm_out.bsml)
command	tmhmm2bsml.pl --input tmhmm_out.raw --output tmhmm_out.bsml --fasta_input split1.fasta --compress_bsml_output 0 --id_repository workflow/project_id_repository

Step 5.3 Parse TMHMM BSML

executable	camera_parse_annotation_results_to_text_table.pl
input	TMHMM BSML file (tmhmm.bsml)
output	parsed BSML formatted file (tmhmm_out.bsml.parsed)
command	camera_parse_annotation_results_to_text_table.pl --input_file tmhmm_out.bsml --input_type TMHMMBSML --output_file tmhmm_out.bsml.parsed --work_dir /tmp

Step 5.4 Set Default Names

executable	camera_parse_annotation_results_to_text_table.pl
input	split fast file
output	split fasta parsed
command	camera_parse_annotation_results_to_text_table.pl --input_file split.fasta --input_type Hypothetical --output_file split_fasta.parsed --work_dir /tmp

Step 6 Annotation Rules

The final annotation for each peptide is being derived based on all previously collected evidences. How evidences are being used to assign the various annotation data types (common name, gene symbol, EC, GO, Tigr Role) is based on a evidence rules hierarchy in lib/CAMERA/AnnotationRules/PredictedProtein.pm.

Step 6.1 Concatenate parsed results obtained in steps 2-5

executable	cat
input	all parsed files
output	out.cat.sorted
command	cat *.sorted > out.cat

Step 6.2 Sort the concatenated file

executable	sort
input	concatenated results (out.cat)
output	sorted concatenated results (out.cat.sorted)
command	sort --key=1,1 -T /tmp -S 1G -d -o out.cat.sorted out.cat

Step 6.3 Generate tab delimited annotation file (final output)

executable	camera_annotate_from_sorted_table.pl
data dependencies	tab-delimited synonyms.tab file
input	sorted concatenated files (out.cat.sorted)
output	tab delimited annotation results (annotation.tab)
command	perl camera_annotate_from_sorted_table.pl --input out.cat.sorted --synonyms <snapshot-dir>/synonyms.tab --output out.cat.tmp > annotation.tab