Chapter 3

MetaCAT — Metagenome Cluster Analysis Tool

3.1 Introduction

Much of what we know in biology today is the result of careful studies of cultivated pure cultures over decades. Yet today it is apparent that natural microbial communities look nothing like microbes cultivated *in vitro*. Microbial communities in nature can be vastly complex assemblies, often containing hundreds of species of bacteria or more, with many forming intricate associations. One example is the higher termite hindgut, which contains a complex microbial community that specialized in lignocellulose degradation [1]. A similar complex microbial community resides in the human gut, with every human harboring about 150 bacterial species, with most bacteria and genes shared among humans [2]. Similar levels of complexities were found in many other environments, from marine to soil to deep sea "whale fall" carcasses [3,4]. Thus, communities are far more complex that any single organism in a pure culture. However, traditional microbiological techniques are limited in their capability to study these communities since it has been estimated that >99% of microbes in nature cannot be cultured *in vitro* [5]. As a result, the field of metagenomics came to the fore. Metagenomics is the study of genetic material recovered directly from environmental samples [6,7]. The field is also referred to as environmental genomics, or community genomics. With the advent of next-generation sequencing, metagenomics enables direct extraction, cloning and sequencing of DNA from their natural environment with protocols optimized to capture unexplored microbial

diversity [8,9]. Recently, the viral fraction of microbial communities has come into the spot light, revealing the vast complexity of viral diversity on Earth [10,11,12].

Today, researchers can, at a reasonable cost and effort, obtain a metagenome of the environment they are interested in. Thus, currently the problem has shifted from sampling the diversity in a given environment to developing the bioinformatic tools to analyze this complexity and the enormous amounts of data generated by such studies. Current metagenome analysis tools such as MEGAN [13], CAMERA [14], and MG-RAST [15] focus on the annotation and classification of the gene fragments present in metagenomes. While these tools are essential, they do not provide an annotation-independent census of the various genes present in a metagenome. Thus, there is no available tool that we are aware of that can produce a ranked list of the abundance of all genes in a metagenome automatically, grouping genes based on some similarity criterion without relying on annotation. Such a census can be useful in comparing relative abundances of genes in a given community in a way that does not rely on annotation. Current methods of achieving this goal involve searching for keywords and summing hits manually. Such methods are both not rigorous and depend on the quality of annotation. Furthermore, annotation-based methods do not group genes based on a similarity criterion (indicating homology), thus potentially significantly biasing results. Another use for an annotation-independent census of a metagenome is to collect a diversity of genes for primer design, especially for genes that exhibit significant diversity such as viral genes (see Chapter 2).

Here we present a new tool called MetaCAT (metagenome cluster analysis tool) that is designed to calculate an “abundance spectrum” of known genes present in a given

metagenome. This spectrum can be used to quickly ascertain the “major players” in terms of genes present/genes being expressed in the given environment.

The process of annotation of a metagenome involves BLASTing each metagenome gene object against a database of “known reference genes” and searching for the best hit. MetaCAT generates a census by reversing this procedure and BLASTing a database of “known reference genes” against the metagenome, counting the number of hits for each reference gene. The “known reference genes” that MetaCAT uses as a reference when constructing its spectrum can be in principle all currently known genes or a particular subset of these genes, such as all known viral genes, all known mammalian genes, all known plastid genes, and so on. Although there is no restriction on the library of known genes that MetaCAT can use, MetaCAT was designed with NCBI’s RefSeq database in mind. The RefSeq database is intended to be a “stable, consistent, comprehensive, non-redundant database of genomes” [16] and can be downloaded in full or for certain major taxonomic or other logical groups. MetaCAT, in addition to specifying which known genes are present in the metagenome and their abundances, also gives additional information regarding the known genes. This additional information includes a detailed description of the genes and a description of the lineage of the source organism in which these genes appeared. In the cases of viruses for example, such information can be useful in obtaining a quick snapshot of the classes of viruses that may be abundant in the given environment (e.g., tailed phages versus filamentous phages).

The “abundance spectrum” for the metagenome is calculated by inverting the normal BLAST process. In this scheme every known reference gene is BLASTed against the metagenome (instead of vice versa) and the number of significant hits in the metagenome is counted. Thus in principle every gene in the reference library is given a score which is the number of significant hits that that gene received in the metagenome. This list is, in general, very long but can be significantly compressed with little loss of information by noticing that the reference library often contains many reference genes that yield similar “signatures” in the metagenome, making this list partly redundant. Furthermore, many genes in the reference library yield only tenuous homologies and can be discarded by placing an E value threshold for the best alignment of a given reference gene. MetaCAT therefore compresses the reference gene library for a given metagenome by taking advantage of both these factors, thus generating a spectrum containing a tractable list of genes.

3.2 The MetaCAT algorithm

3.2.1 Overview

The objective of MetaCAT, loosely speaking, is the following: (1) given a metagenome, find in an annotation-independent manner all of the clusters of homologous genes within the metagenome, and (2) for each cluster that was found report the number of members (i.e., gene objects) within the cluster and in addition report the best mach to this cluster among all genes present in reference database, such as the RefSeq database. Fig. 3.1 illustrates the end goal of a MetaCAT analysis of a metagenome.

Typically when one executes a BLAST analysis one BLASTs an unknown gene, such as a gene object from a metagenome, against a reference database of known genes, such as the

RefSeq database (Fig. 3.2A). This procedure results in a list of known genes that pass a certain E value cutoff. If one is interested in mapping a metagenome, this process is repeated for every gene object in the metagenome. The key idea behind MetaCAT is that instead of BLASTing every gene object in the metagenome, MetaCAT BLASTs every known gene in reference database against the metagenome (Fig. 3.2B).

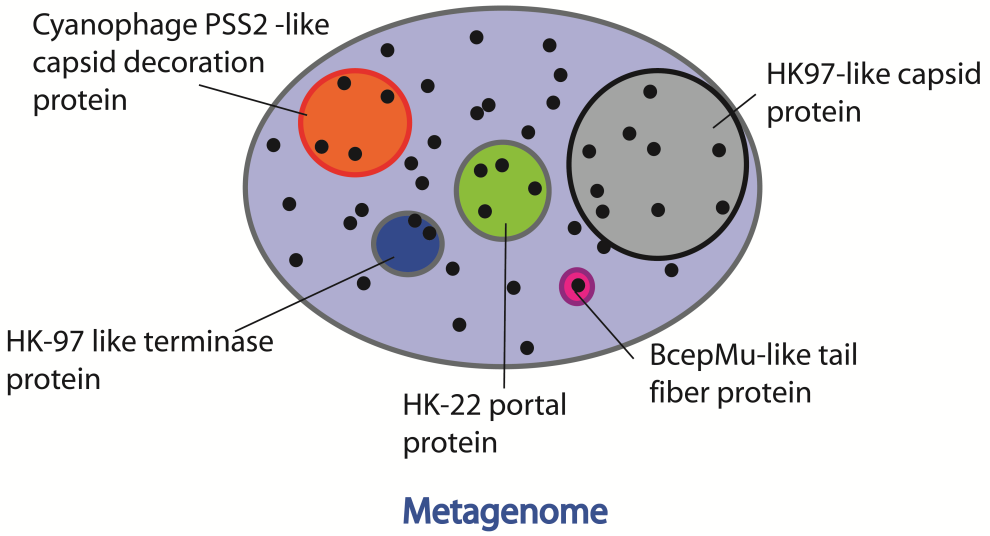

Figure 3.1 Ideal clustering of gene objects in a metagenome. Each dot represents a gene object in a metagenome, with the entire metagenome depicted by the blue oval. Similar genes are grouped into clusters (circles of different colors) and each cluster is represented by a single gene from a known reference database. In this schematic description the distance between dots is interpreted in an abstract manner and does not correspond to a rigorous metric.

We refer to this "inverse" BLAST analysis as an iBLAST transformation. Each known gene that is iBLASTed results in a list of metagenome gene objects that pass a certain E value threshold. This list is referred to as the "coverage" of the particular known gene in the metagenome. The number of gene objects in this list is interpreted as the abundance of the particular known gene in the metagenome. This process is repeated for every known gene in the reference database. The result is a rather long table that describes for every gene in the reference database its abundance in the metagenome.

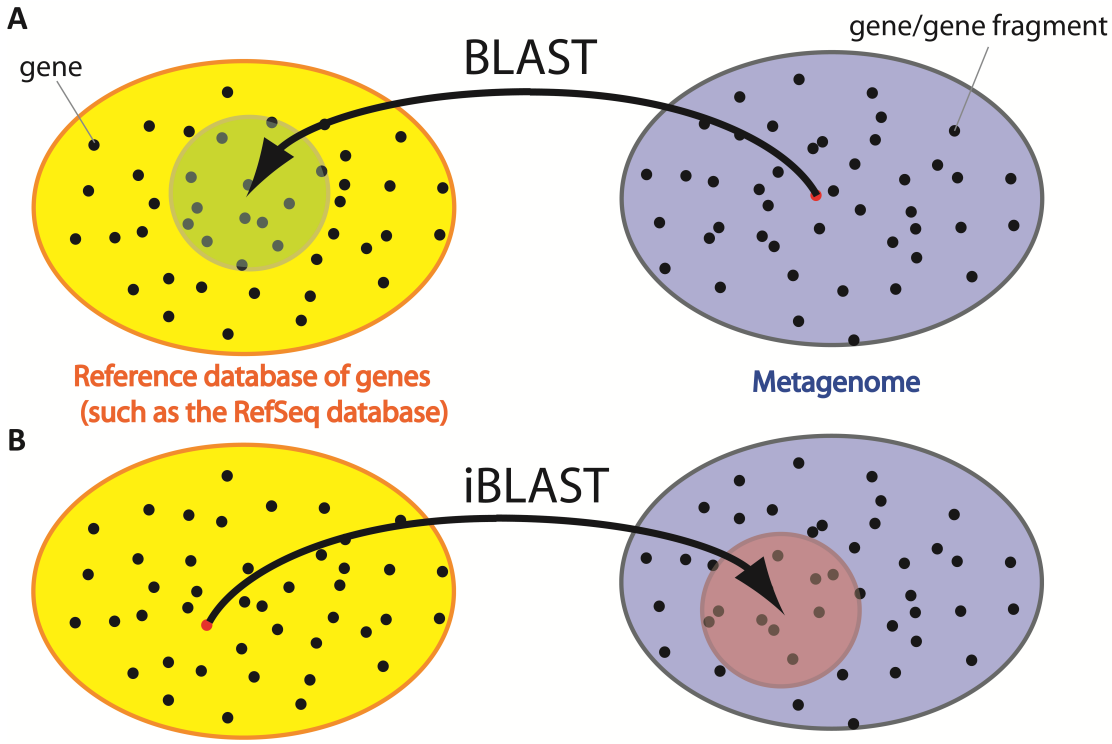

Figure 3.2 An illustration of a BLAST and an iBLAST analysis. A. Typically when performing a BLAST analysis a novel gene object from the metagenome (blue oval) is BLASTed against a reference database of known genes, such as the RefSeq database (yellow oval). The result of the BLAST analysis is a list of "hits" that pass a certain E value threshold. **B.** In an inverse BLAST analysis ("iBLAST" for short) a gene from a known database is BLASTed against the metagenome. The corresponding list of "hits" that pass a certain E value threshold is defined as the "coverage" or "abundance" of the particular known gene in the metagenome.

The list of known genes can be quite long. For example, there are currently approximately 80,000 known viral genes in the RefSeq viral database. MetaCAT compresses this list in two stages. The first filtering stage to impose an E value thresholds for the iBLAST: MetaCAT rejects a gene from the reference database if the *lowest* E value is not low enough. In this way known genes that are remote homologs of every metagenome gene object are automatically discarded from further analysis. The second filtering step has to do with removing redundancy from the reference database. Many genes in this reference database can be close homologs. Close homologs can correspond to very similar clusters of gene objects in the metagenome (Fig. 3.3). Ideally MetaCAT would report and reject all of the close homologs in the reference database and report a single gene, whose E value is the lowest with respect to the metagenome gene objects. MetaCAT identifies related genes in the reference database (red dots in yellow oval in Fig. 3.3) by comparing the overlap of the metagenome gene object (overlapping red circles in the blue oval in Fig. 3.3). If the overlap exceeds a certain threshold the genes in the reference database are said to be "related". Once all related genes are found, only one gene is chosen to represent the group. Though the resulting list of genes from the reference library may still have residual redundancy (see below), this step significantly removes a great deal of redundancy from this database. Note that declaring that two known genes are related by comparing their overlap in the metagenome database is more general that comparing the homology of the genes directly, since in the latter case the reduction is performed independently of the metagenome, and therefore there is information loss. An illustration of a final clustering of genes by MetaCAT is given in Fig. 3.4.

The MetaCAT algorithm is described in detail in the following Section, and a guide for how to use this tool and interpret results is presented in Section 3.3. Installation instructions are given in Section 3.4. An example of a MetaCAT run is given in Chapter 2, Table 2.2.

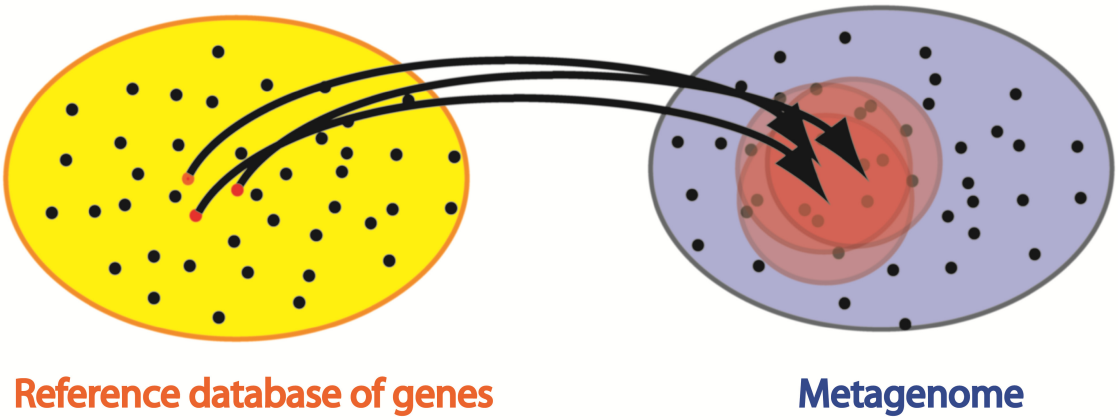

Figure 3.3 Coverage overlap in a metagenome. Similar genes in the reference database (e.g., closely related homologous genes) can have an overlapping coverage in the metagenome. MetaCAT can identify this overlap and will consequently report only one of the reference genes (the one with the lowest E value).

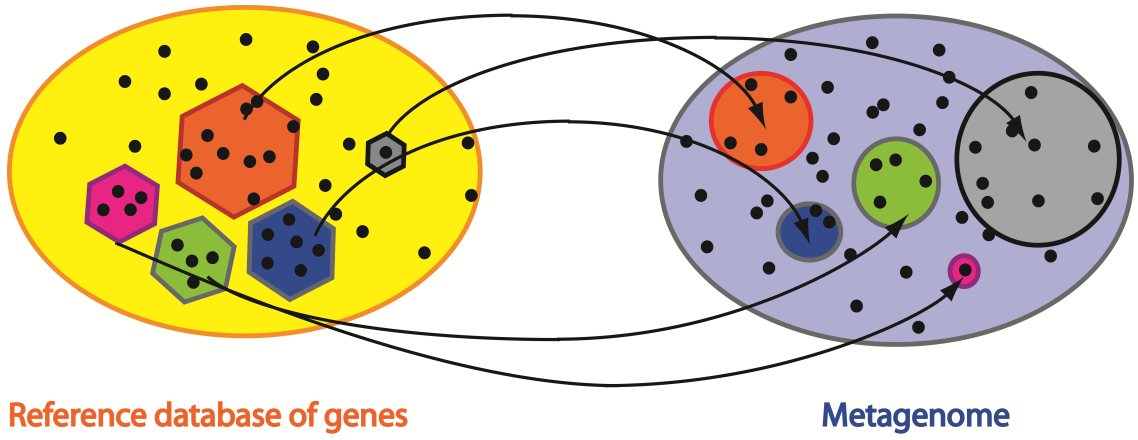

Figure 3.4 Illustration of a final MetaCAT analysis. At the end of the analysis, MetaCAT has identified all known genes that result in overlapping coverage, reporting one representative for each such group. In this manner the apparent redundancy of the reference database is significantly compressed (though the current algorithm used by MetaCAT does not remove this redundancy totally — see discussion below).

3.2.2 The MetaCAT algorithm in detail

MetaCAT analyzes a metagenome through the following sequence of steps:

1. BLAST analysis. The amino acid sequence of every known gene from a reference database K_i ($i=1..N$) — which can be for example a RefSeq database [16] — is BLASTed using NCBI's blastp (v2.2.22+) [17] against the amino acid sequences of all gene objects in the metagenome M_j ($j=1..M$). All BLAST hits must be lower than a maximal E value threshold of 10^{-3}. All alignment information is stored by blastp in a table. This step involves NXM alignments.

2. Extracting best E value scores and abundances of known reference genes in metagenome. MetaCAT reads the resulting BLAST table and for each known reference gene (KRG) finds the lowest E value score $E_i^{\min}$ and the number of different metagenome gene objects (MGO) n_i ($i=1..N$) that were equal or lower than this E value score. n_i is said to be the *footprint* of K_i in the metagenome. Each KRG, K_i, is therefore homologous to n_i MGOs. The list of n_i elements will hereafter be designated as the *signature* of K_i in the metagenome. A *footprint* is therefore defined as the number of elements in a *signature*.

3. E value filtering of the known reference gene database. We wish to keep only the KRGs that yielded reasonable alignments to MGOs, since we do not want to be concerned with KRGs yielding tenuous similarities. We therefore discard all KRGs whose best E value exceeded a maximal E value threshold (with a default threshold of 10^{-7}), i.e., we

require that $E_i^{\min} \le E_{th} = 10^{-7}$. This filtering step will reduce the number of KRGs from *N* to *N'*.

4. Clustering known reference genes. Two KRGs with similar *signatures* are said to be *related* (the measure of similarity will be defined below). Once we identify for given KRG all of the other KRGs to which it is *related*, instead of declaring the given KRG, MetaCAT will declare out of all the *related* KRGs, the KRG with the lowest E value. This filtering method is conservative in the sense that every KRG that is omitted from the final list of declared reference genes is represented by a different KRG that has a very similar metagenome *signature* but has a better E value score, thus there is essentially no loss of information. The outcome of this process is reduction of the number of reference genes that MetaCAT reports by one if the reported (or "declared") KRG is different from the given KRG.

More specifically: for each remaining KRG, K_i (*i=1..N'*), MetaCAT finds all *related* KRGs $\{K_i\}$. Two KRGs are said to be *related* if their corresponding signatures in the metagenome share $P_{th} = 50\%$ of their elements, i.e., if L_i is the number of MGOs homologous to K_i, and L_j is the number of MGOs homologous to K_j, then K_i and K_j are said to be related if and only if $P_{th} \le 100 \cdot \min\left(L_i \cap L_j / L_i, L_i \cap L_j / L_j\right)$. The stringent *min* function ensures that the overlap between MGOs is normalized by the length of the longer list of MGOs. This prevents relating two KRGs in situations for example where one signature list is very short and is included in a second, very long signature list

(which would yield 100% if the *max* function was used). This stringent definition of overlap ensures that all related genes have roughly similar footprints in the metagenome.

Note that the group $\{K_i\}$ will always include K_i since K_i is always related to itself by definition. Each KRG of the remaining KRGs (*i=1..N'*) is then said to "declare" one element of the group $\{K_i\}$ to represent this group. The element that is chosen to represent the group is the KRG with the lowest E value. In case more than one element has the same E value then the following criteria are tested sequentially: highest percent identity, highest number of identical amino acids, highest percent of gene length aligned. If all measures are equal (that can happen if the KRGs have identical amino acid sequences) then to prevent dependence on the order of the genes in the reference library, the KRGs are sorted by their FASTA gene name and the first one is selected.

This clustering step involves *N'* X *N'* comparisons. The group of KRGs that are "declared" K_i (*i=1..N'')* are the final list of reference genes reported by MetaCAT that are said to be found in the metagenome. If two or more KRGs declare the same gene, then that gene appears only once in the final output (this is the compression stage). The total number of KRG declared by MetaCAT will therefore satisfy $N''<N'<N$ and typically $N''<<N$. The abundance of each declared KRG K_i in the metagenome is then simply its *footprint* n_i .

Predictably of the algorithm to changing E value thresholds

The algorithm behaves in a predictable fashion when changing E_{th}. For example, if E_{th} is increased from E_1 to, say, E_2, more KRGs are declared in the final list, however these additional KRGs will simply add to the previous declared list when $E_{th} = E_1$. This is because while increasing E_{th} may add additional KRGs to the list of *related* genes $\{K_i\}$ of a given KRG, these additional *related* genes will (by definition) have lower best E value scores, and therefore will not be declared. Thus, increasing the E value threshold can only expand the set of declared KRGs.

3.3 Future directions

Redundancy in the final list of declared reference genes

Note that the final list of declared KRGs can still be redundant. That is, two declared KRGs can still be related, i.e., share $> P_{th}$ of their signature elements. This can happen for example if K_i declares K_j, but K_j, which happens to be related also to K_k, declares K_k. K_j and K_k are thus both declared, yet they are related. Therefore there may still be redundancy present in the final list of declared MGOs that needs to be removed.

We have just seen that how MetaCAT generated a list of *N''* KRGs that can still be redundant, i.e., some KRGs can still have overlapping signatures. We will therefore repeat the clustering algorithm (step 4) until all redundancy is removed. As long as there is redundancy left the clustering will continue to remove nodes. The algorithm will therefore cease when all redundancy is removed (see proof below).

The iterative clustering algorithm discards KRGs at each step, however the KRGs that are discarded have an overlapping signature with one of the remaining KRGs, and therefore in principle no information is lost by this compression.

When does the compression algorithm cease to remove KRGs? The compression algorithm will cease when every node declares itself, a state where by definition there are no more related KRGs and thus no more redundancy. In other words, all redundancy is removed when every node is a local minimum of E value. To see this we note the following: if there are any related KRGs in the final list of declared KRGs (Fig. 3.5) then some node Y will not be at a local E value minimum and will have to declare a node which isn't itself (because if for example node X is a local E value minimum and declares itself, then any node connected to X, like Y, will certainly not declare itself since the E value of X is lower). There are two possibilities at this point. If no other node declares Y then Y will not be declared and the algorithm has compressed the list of KRGs by 1 (Y is not declared). The second possibility is that some other node Z with a higher E value than Y will declare Y. In this case the question would be, is there another node with a higher E value than Z that declares Z. If Z is not declared by any other node than the list has been compressed by 1 (Z is not declared). If there is a node with a higher E value that declares Z we can continue the chain, each time increasing the E value, however at some point we will reach a node which has the highest E value (i.e., a maximum) and therefore will never be declared, resulting in compression of the KRG list. Therefore, as long as there is connectivity in the KRG list, the KRG list will be compressed in the current iteration.

Compression will cease once all connectively is eliminated, that is when each KRG is at local minimum of E values and therefore declares itself.

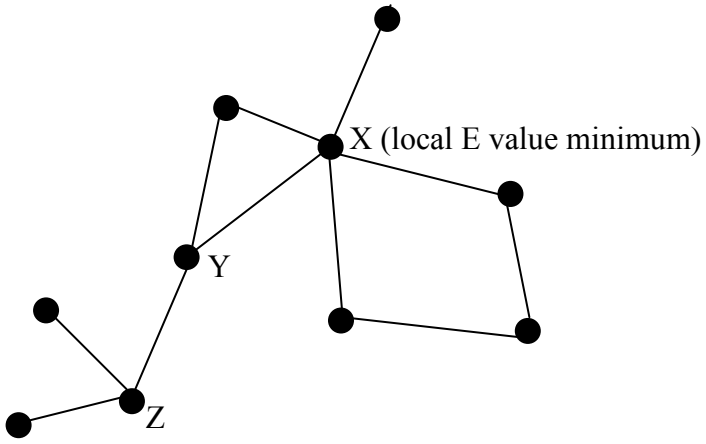

Figure 3.5 Example of list of connected KRGs at one of the clustering iterations

Sample abundance versus metagenome abundance

MetaCAT currently is written to work on the amino acid sequences of metagenome gene objects. These gene objects are the result of reads that have been assembled and translated. The true abundance of a gene object in nature is proportional to the number of reads found in the database and not the number of times this gene object appears in the metagenome. The reason for this discrepancy is that in principle, each gene object should appear only once in the metagenome, since the assembler should automatically remove identical gene objects. For this reason it would be advisable to incorporate into MetaCAT information regarding the abundance of reads. One way to accomplish this would be to use available programs to map reads to gene objects, and then count the number of hits per gene object. For the cases of viruses our prediction is that assembler collapse of contigs did not introduce significant bias to abundances because the viral genes tend to naturally mutate

(being part of quasi-species). These mutated gene objects are most likely not collapsed by the assembler, especially given horizontal gene transfer that can affect neighboring genes. This appeared to be the situation in the case of the higher termite metagenome. Terminase alleles in the metagenome were quite divergent (see for example Fig. 2.2). Nevertheless it would be interesting to compare MetaCAT's report of abundances with the abundances corrected for assembler bias.

3.4 Software operation

3.4.1 First-time run on a metagenome

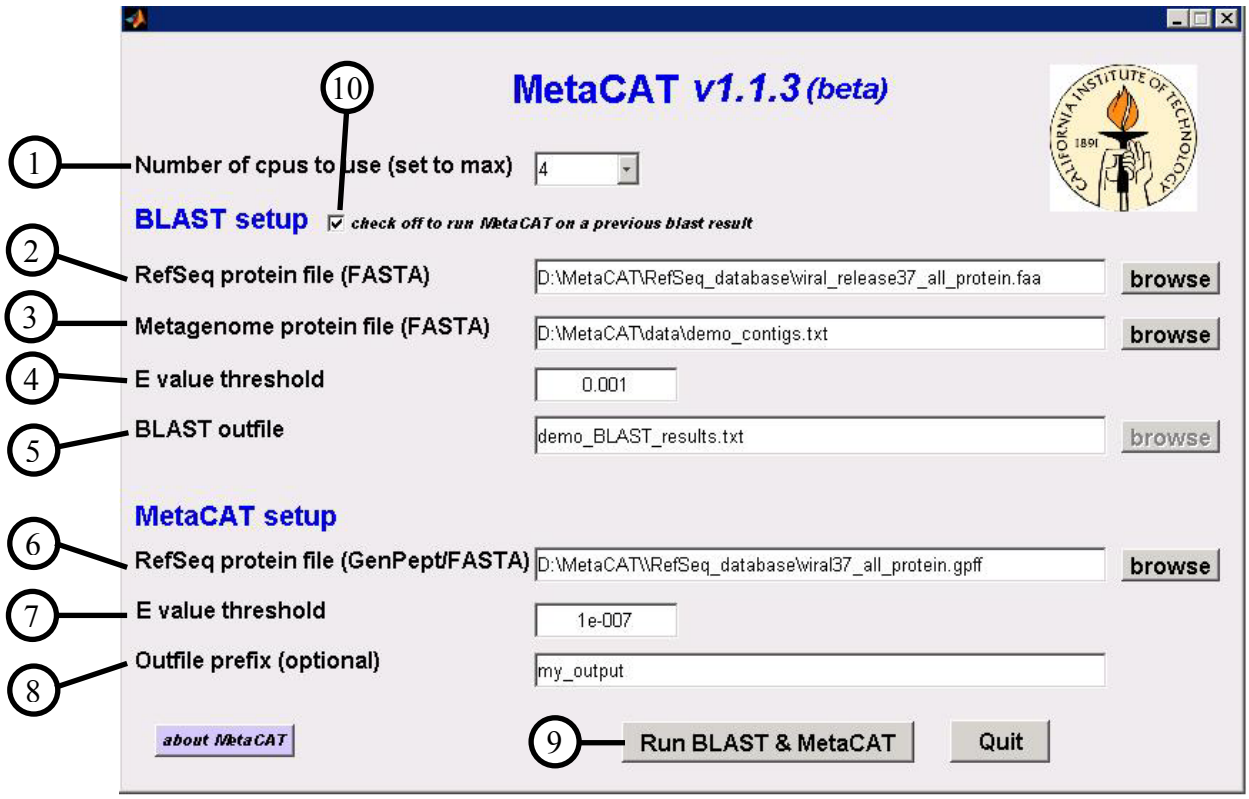

Figure 3.6. Meta main interface.

Computing resources parameters:

If the pull-down menu is activated this means you have the Matlab Parallel Processing Toolbox installed. The default number is set to the number of processors on your computer. It is recommended you utilize all cpus for faster execution, however you can use the pull-down menu to restrict the number of cpus used.

BLAST input parameters:

1. Enter the protein FASTA file for your known reference database. Any FASTA file can be used, however we recommend using NCBI's RefSeq database. This database is a comprehensive, non-redundant database curated by NCBI. Each RefSeq FASTA file issued by NCBI is accompanied by a corresponding GenPept file that includes comprehensive information about the gene and its origin. This GenPept file can be parsed by MetaCAT (see description of output below). For demonstration purposes the "`RefSeq_database`" folder includes the file: "`viral_release37_all_protein.faa`" which is release 37 (Sep. 2009) of all RefSeq viral genes, spanning 2386 distinct species.
2. Enter the FASTA protein file for the metagenome of interest. For demonstration purposes the file "`demo_contigs.txt`" is provided in the "`data`" folder.
3. Enter the E value threshold for BLAST (the default is 0.001).
4. Enter a name for the BLAST output file that will be generated in the "`output`" folder.

MetaCAT input parameters:

5. Enter the GenPept file corresponding to FASTA file entered in option #2. If this file is not available use the FASTA file entered in option #2. For demonstration purposes the "`RefSeq_database`" folder contains the GenPept file "`viral_release37_all_protein.gpff`" that corresponds to the RefSeq FASTA file "`viral_release37_all_protein.faa`".
6. Enter the E value threshold for the MetaCAT algorithm (default is 1e-7).
7. Enter a string that will be included in all MetaCAT output files for your reference.
8. Click "Run BLAST & MetaCAT" to run MetaCAT.

3.4.2 Output files generated

MetaCAT generated files

At the end of MetaCAT's run six output files will be generated to the '`output`' folder:

- The BLAST output file
- `<blast file name><MetaCAT output string>_MetaCAT_output0_params.txt`
- `<blast file name><MetaCAT output string>_MetaCAT_output1_AllGenes.txt`
- `<blast file name><MetaCAT output string>_MetaCAT_output2_AllGenesFilt.txt`
- `<blast file name><MetaCAT output string>_MetaCAT_output3_RelatedGenes.txt`
- `<blast file name><MetaCAT output string>_MetaCAT_output4_ShortTable.txt`

These output files can be opened in EXCEL. The "Filter" option in EXCEL can be used to filter these output files. The last file, '…Output4_ShortTable', is the final output of MetaCAT. See the Section 3.6 for a description of output files 0, 1, 2, 3 and 4.

The final output of MetaCAT: the *output4_ShortTable file

This file lists all the RefSeq genes whose lowest E value was equal to or lower than the MetaCAT E value threshold, after removing *related* RefSeq genes. The list is sorted according to the number of metagenome gene objects (MGOs) that are homologous to the given RefSeq gene such that those with the largest number appear at the top of the list. Each line includes additional information about the given RefSeq gene and its alignment with the MGO that yielded the lowest E value. A detailed description of the fields included in this file is given in Section 3.6.4.

3.4.3 Subsequent runs of MetaCAT

The program can run in two modes — a BLAST analysis followed by MetaCAT analysis (the default mode) and just a MetaCAT analysis. The latter mode is useful for analyzing a previous BLAST run, for example with a different E value threshold. To toggle between the modes click the check box (option 10 in Fig. 3.6). This switch will inactivate all fields related to the BLAST run, and will allow the user to select a previous BLAST output file generated by MetaCAT.

3.5 Installation instructions

3.5.1 System requirements

1. Operating system: Windows (32bit/64bit), Linux (32 bit/64 bit), Mac OS (32bit/64bit)
2. Matlab 7.4 and higher
3. To enable parallel processing Matlab Parallel Processing Toolbox v4.2 is required.

For best performance:

The program is computationally cpu intensive and is capable of utilizing multiple cpus using Matlab's Parallel Processing Toolbox. For optimal performance we recommend computers with multiple fast processors. On a Dell Precision T3500 with Quad Core Xeon X5550 (eight 2.66 GHz processors) analysis of a metagenome of 80,000 gene objects with a reference viral library of 80,000 genes takes 3 hours.

3.5.2 Installation

Note: On Linux/Mac systems you may need to have root privileges. It is recommended that you install this software as a root/administrator for these operating systems.

Installation requires an internet connection.

1. Download the compressed sources for MetaCAT and extract locally .
2. Start Matlab.
3. Change directory to the `'bin'` folder of MetaCAT.
4. Run MetaCAT by typing `MetaCAT_EXE` in the Matlab command prompt.
5. Click "Automatic installation (recommended)".
6. The program automatically downloads and installs BLAST vs. 2.2.22+ from NCBI[1]. This process may take a few minutes. Once the installation of BLAST starts click "next" and accept all of the default entries.
7. Once the main interface loads you may start using MetaCAT.

[1] If you already have blast 2.2.22+ installed in the default MetaCAT directory this step is automatically skipped. Other blast versions are ignored by MetaCAT. If you have blast 2.2.22+ installed in a different directory MetaCAT gives you the option to select the folder in which you previously installed this program. Simply select the "Please proceed to manually install blast 2.2.22+" option in the first menu.

3.5.3 Troubleshooting

1. For Linux/Mac OS it is recommend you have root privileges.
2. If there is a problem downloading BLAST make sure your firewall does not block the ftp port. Alternatively you can download blast v2.2.22+ manually from the NCBI website (choosing the appropriate software version appropriate for your OS): ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.2.22/.

   Installation instructions for blast can be found here ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.2.22/user_manual.pdf.

   Once blast is installed, start MetaCAT, click "Locate sources on computer" and locate the "`bin`" folder of blast installation.
3. If there is a problem running blast there may be a previous installation of blast from another utility which is interfering with MetaCAT. If you are running Windows and Windows is not installed on the c:\ drive, check in the windows folder on your computer for a file called '`ncbi.ini`'. If this file exists, temporarily change its name.

3.5.4. Downloading and combining RefSeq files

The RefSeq database is continuously updated. The most up-to-date release of RefSeq files is available on the NCBI ftp server at ftp://ftp.ncbi.nih.gov/refseq/release/. You can select to download the entire database or subsets of this database for particular taxonomic or other logical groups. FASTA files have an *.faa extension and GenPept files have a *.gpff extension. Since the RefSeq databases are large, these databases have been separated into numerous smaller files to facilitate downloading from the web and handling. These files

can be downloaded in-bulk using one of the many free ftp software programs available on the web. Simply log on to the ftp site ftp://ftp.ncbi.nih.gov as an anonymous user and browse to the `refseq/release/` directory.

To combine all of the RefSeq files into one large FASTA/GenPept file one option is to use a MetaCAT utility:

1. Make sure MetaCAT's "`Combine_RefSeq_files`" folder *contains only one file:* `util_concat_all_files_in_folder.m`.
2. Copy all the files you wish to combine into one file (and only these files) into the above folder.
3. In Matlab change directories to the "`Combine_RefSeq_files`" folder.
4. Type `util_concat_all_files_in_folder` at the prompt.

This utility will combine all of the files in the given directory expect for the Matlab source into one file named "`combined_all`". For large files this may take some time. After the run is finished change the name of this file to whatever name you choose.

3.5.5 MetaCAT folders

The following folders are installed with MetaCAT:

`bin`	location of MetaCAT Matlab execution source MetaCAT_EXE.m
`msrc`	folder with MetaCAT source code
`blast`	folder where blast is installed
`data`	folder to store metagenome files
`RefSeq_database`	folder to store the RefSeq FASTA/GenPept files
`output`	folder to which all output files are written
`Combine_RefSeq_files`	folder that contains a Matlab source code for concatenating all the files in this folder (see §6)

3.5.6 Known bugs

NCBI blast 2.2.22+ appears to have an intrinsic bug where BLASTing a single FASTA record against a large database in some rare cases can add/miss some hits compared to the case where this record is embedded in a very large FASTA file. This can lead to very small differences in the BLAST output depending on the number of cpus used, as the parallelization requires splitting the RefSeq FASTA file into *n* smaller files, *n* being the number of cpus used as defined by the user. This bug appears to be very rare and doesn't affect results significantly, for example slightly affecting the number of hits (+/- 1 or 2) for 10 out of ~6500 records that passed the BLAST E value threshold.

3.6 Description of additional output files

Description of additional output files:

Output0_params

This file contains:

- Statistics on the run such as execution time, date and time of run, version of MetaCAT used, etc.
- The command line(s) used to run BLAST (if appropriate)
- A summary of the parameters/file names used to run MetaCAT
- The list of output files generated by this run.

Output1_AllGenes

Information parsed from the BLAST output file. The file lists all of the RefSeq genes that passed the BLAST E value threshold. Each RefSeq gene is followed by the list of all

metagenome gene objects (MGOs) that passed the BLAST E value threshold. The following additional information is provided for the MGO yielding the lowest E value:[2]

1. Index

Counter of RefSeq gene in table.

2. RefSeq gene

RefSeq gene identification as it appears in the RefSeq FASTA file definition line (extracted by BLAST).

3. RefSeq gene definition

RefSeq gene definition as it appears in the RefSeq FASTA file definition line (N/A for the AllGenes output file).

4. Metagenome gene object ID with lowest E value

Identification of MGO that yielded the lowest E value for the given RefSeq gene.

5. # of metagenome gene objects similar to this RefSeq gene

Number of MGOs that yielded an E value equal or lower than the BLAST E value threshold when aligned against the given RefSeq gene.

6. % identity

Percent identity of the given RefSeq gene and the MGO.

7. # of identical amino acids

Number of identical amino acids between the given RefSeq gene and the MGO.

8. E value

E value for the alignment between the given RefSeq gene and the MGO.

[2] To display the information for just the best MGOs use EXCEL and filter the first column by the string “table”.

9. Alignment length (amino acids)

Number of amino acids in the alignment.

10. RefSeq gene length (amino acids)

Number of amino acids in the RefSeq gene (N/A for the AllGenes output file).

11. % of RefSeq gene length aligned

The percent of the RefSeq gene length that appears in the alignment — i.e., the ratio of (9) and (10) times 100 (N/A for the AllGenes output file).

12. aa sequence

The amino acids sequence of the RefSeq gene (N/A for the AllGenes output file).

Output2_AllGenesFilt

Same as the 'Output1_AllGenes' file but showing only RefSeq genes whose lowest E value was equal to or lower than the MetaCAT E value threshold.

Output3_RelatedGenes

List of RefSeq genes whose lowest E value was equal to or lower than the MetaCAT E value threshold, after removing related RefSeq genes (i.e., the list of group representatives)[3]. Following each group representative is the list of related genes (i.e., group members). The list is sorted according to the number of MGOs homologous to the given RefSeq gene (highest number at the top of the list). The following additional information is given for every RefSeq gene:

1. Index

Counter of RefSeq gene in table.

[3] To display just this list in EXCEL, filter the first column by the string "table1".

2. RefSeq gene

RefSeq gene ID as it appears in the RefSeq FASTA file definition line (extracted by BLAST).

3. RefSeq gene definition

RefSeq gene "Definition" field as it appears in the GenPept file (or if a RefSeq FASTA file was supplied, the RefSeq gene definition as it appears in the FASTA definition line).

4. Min % of shared metagenome gene objects

The overlap between the MGO list of the given RefSeq gene and the MGO list of the group representative in units of percent (i.e., the number of MGOs shared between both RefSeq genes divided by the larger number of MGOs of both genes, in units of percent).

5. # of metagenome gene objects similar to this RefSeq gene

Number of MGOs that had an E value equal to or lower than the BLAST E value threshold when BLASTed against the given RefSeq gene.

6. % identity

Percent identity between the MGO with the lowest E value and the given RefSeq gene.

7. # of identical amino acids

Number of identical amino acids in the alignment of the MGO with the lowest E value and the given RefSeq gene.

8. E value

E value in the alignment of the MGO with the lowest E value and the given RefSeq gene.

9. Alignment length (amino acids)

Length of alignment in amino acids between the MGO with the lowest E value and the given RefSeq gene.

10. RefSeq gene length (amino acids)

Number of amino acids of the given RefSeq gene.

11. % of RefSeq gene length aligned

The percent of the RefSeq gene length that appears in the alignment — i.e., the ratio of (9) and (10) times 100.

12. GenPept Features

RefSeq gene Features field as it appears in the GenPept file.

Output4_ShortTable

This file is the main output of MetaCAT. This file contains the following fields:

1. Index

Counter of the RefSeq gene in the table.

2. RefSeq gene

RefSeq gene identification as it appears in the RefSeq FASTA file definition line (extracted by BLAST).

3. Metagenome gene object ID with lowest E value

Identification of the MGO that yielded the lowest E value for the given RefSeq gene.

4. # of metagenome gene objects similar to this RefSeq gene

Number of MGOs that had an E value equal to or lower than the BLAST E value threshold when the given RefSeq gene was BLASTed against the metagenome.

5. tot # of metagenome gene objects associated with this RefSeq gene group

Combined number of homologous MGOs of the given RefSeq gene and all its related RefSeq genes (i.e., group members).

6. # of related RefSeq genes

Number of RefSeq genes related to the given RefSeq gene, including the given RefSeq gene, i.e., number of group members including the group representative.

7. % identity

Percent identity between the MGO with the lowest E value and the given RefSeq gene.

8. # of identical amino acids

Number of identical amino acids in the alignment of the MGO with the lowest E value and the given RefSeq gene.

9. E value

E value for the alignment of the MGO with the lowest E value and the given RefSeq gene.

10. Alignment length (amino acids)

Length of alignment in amino acids between the MGO with the lowest E value and the given RefSeq gene.

11. RefSeq gene length (amino acids)

Number of amino acids for the given RefSeq gene.

12. % of RefSeq gene length aligned

The percent of the RefSeq gene length that appears in the alignment — i.e., the ratio of (10) and (11) times 100.

13. aa sequence

Amino acid sequence of the given RefSeq gene.

14. RefSeq gene definition

The “Definition” field for the given RefSeq gene as it appears in the GenPept file (or if a RefSeq FASTA file was supplied, the RefSeq gene definition as it appears in the FASTA definition line).

> **Definition field** — “Brief description of sequence; includes information such as source organism, gene name/protein name, or some description of the sequence's function (if the sequence is non-coding). If the sequence has a coding region (CDS), description may be followed by a completeness qualifier, such as ‘complete cds’.” [18]

15. GenPept GenBank division

GenBank division field for the given RefSeq gene as it appears in the GenPept file.

> **GenBank division field** — “The GenBank division to which a record belongs is indicated with a three-letter abbreviation. The GenBank database is divided into 18 divisions:
>
> 1. PRI — primate sequences
> 2. ROD — rodent sequences
> 3. MAM — other mammalian sequences
> 4. VRT — other vertebrate sequences
> 5. INV — invertebrate sequences
> 6. PLN — plant, fungal, and algal sequences
> 7. BCT — bacterial sequences
> 8. VRL — viral sequences
> 9. PHG — bacteriophage sequences
> 10. SYN — synthetic sequences
> 11. UNA — unannotated sequences
> 12. EST — EST sequences (expressed sequence tags)
> 13. PAT — patent sequences
> 14. STS — STS sequences (sequence tagged sites)

15. GSS — GSS sequences (genome survey sequences)
16. HTG — HTG sequences (high-throughput genomic sequences)
17. HTC — unfinished high-throughput cDNA sequencing
18. ENV — environmental sampling sequences"

16. GenPept molecule type

Molecule type field for the given RefSeq gene as it appears in the GenPept file.

> **Molecule type field** — "The type of molecule that was sequenced. Each GenBank record must contain contiguous sequence data from a single molecule type. The various molecule types are described in the Sequin documentation and can include genomic DNA, genomic RNA, precursor RNA, mRNA (cDNA), ribosomal RNA, transfer RNA, small nuclear RNA, and small cytoplasmic RNA." [18]

17. GenPept source

Source field for the given RefSeq gene as it appears in the GenPept file.

> **Source field** — "Free-format information including an abbreviated form of the organism name, sometimes followed by a molecule type." [18]

18. GenPept classification

Organism field for the given RefSeq gene as it appears in the GenPept file.

> **Organism filed** — "The formal scientific name for the source organism (genus and species, where appropriate) and its lineage, based on the phylogenetic classification scheme used in the NCBI Taxonomy Database. If the complete lineage of an organism is very long, an abbreviated lineage will be shown in the GenBank record and the complete lineage will be available in the Taxonomy Database." [18]

19. GenPept comments

Comments field for the given RefSeq gene as it appears in the GenPept file.

> **Comments field** — "A COMMENT identifying the RefSeq Status is provided for the majority of the RefSeq records. This comment may include information

about the RefSeq status, collaborating groups, and the GenBank records(s) from which the RefSeq is derived. The RefSeq COMMENT is not provided comprehensively in this release… Additional COMMENTS are provided for some records to provide information about the sequence function, notes about the aspects of curation, or comments describing transcript variants." [19]

20. GenPept Features

Features field for the given RefSeq gene as it appears in the GenPept file.

Features field — "Information about genes and gene products, as well as regions of biological significance reported in the sequence. These can include regions of the sequence that code for proteins and RNA molecules, as well as a number of other features.

Source: Mandatory feature in each record that summarizes the length of the sequence, scientific name of the source organism, and Taxon ID number. Can also include other information such as map location, strain, clone, tissue type, etc., if provided by submitter.

Taxon: A stable unique identification number for the taxon of the source oganism. A taxonomy ID number is assigned to each taxon (species, genus, family, etc.) in the NCBI Taxonomy Database.

CDS: "Coding sequence; region of nucleotides that corresponds with the sequence of amino acids in a protein (location includes start and stop codons)." [18]

Protein Names: Protein names may be provided by a collaborating group, may be based on the Gene Name, or for some records, the curation process may identify the preferred protein name based on that associated with a specific EC number or based on the literature.

Protein Products: Signal peptide and mature peptide annotation is provided by propagation from the GenBank submission that the RefSeq is based on, when provided by a collaborating group, or when determined by the curation process.

Domains: "Domains are computed by alignment to the NCBI Conserved Domain Database database for human, mouse, rat, zebrafish, nematode, and cow. The best hits are annotated on the RefSeq. For some records, additional functionally significant regions of the protein may be annotated by the curation staff. Domain annotation is not provided comprehensively at this time." [19]

3. 7 References

1. Warnecke F, Luginbühl P, Ivanova N, Ghassemian M, Richardson T, et al. (2007) Metagenomic and functional analysis of hindgut microbiota of a wood-feeding higher termite. Nature 450: 560-565.
2. Qin J, Li R, Raes J, Arumugam M, Burgdorf K, et al. (2010) A human gut microbial gene catalogue established by metagenomic sequencing. Nature 464: 59-65.
3. Venter JC, Remington K, Heidelberg JF, Halpern AL, Rusch D, et al. (2004) Environmental genome shotgun sequencing of the Sargasso Sea. Science 304: 66.
4. Tringe SG, Von Mering C, Kobayashi A, Salamov AA, Chen K, et al. (2005) Comparative metagenomics of microbial communities. Science 308: 554.
5. Hugenholtz P (2002) Exploring prokaryotic diversity in the genomic era. Genome Biol 3: reviews0003.
6. Handelsman J, Tiedje J, Alvarez-Cohen L, Ashburner M, Cann I, et al. (2007) The New Science of metagenomics: revealing the secrets of our microbial planet. National Academy of Sciences, Washington, D.C.
7. Riesenfeld CS, Schloss PD, Handelsman J (2004) Metagenomics: genomic analysis of microbial communities. Annual Review of Genetics 38: 525-552.
8. Singh J, Behal A, Singla N, Joshi A, Birbian N, et al. (2009) Metagenomics: Concept, methodology, ecological inference and recent advances. Biotechnology journal 4: 480-494.
9. Kunin V, Copeland A, Lapidus A, Mavromatis K, Hugenholtz P (2008) A bioinformatician's guide to metagenomics. Microbiology and Molecular Biology Reviews 72: 557.
10. Edwards R, Rohwer F (2005) Viral metagenomics. Nat Rev Microbiol 3: 504-510.
11. Dinsdale E, Edwards R, Hall D, Angly F, Breitbart M, et al. (2008) Functional metagenomic profiling of nine biomes. Nature 452: 629-632.

12. Kristensen D, Mushegian A, Dolja V, Koonin E (2009) New dimensions of the virus world discovered through metagenomics. Trends in Microbiology 18: 11-19.
13. Huson DH, Auch AF, Qi J, Schuster SC (2007) MEGAN analysis of metagenomic data. Genome Research 17: 377.
14. Sun S, Chen J, Li W, Altintas I, Lin A, et al. (2011) Community cyberinfrastructure for Advanced Microbial Ecology Research and Analysis: the CAMERA resource. Nucleic Acids Research 39: D546.
15. Meyer F, Paarmann D, D'Souza M, Olson R, Glass EM, et al. (2008) The metagenomics RAST server – a public resource for the automatic phylogenetic and functional analysis of metagenomes. BMC bioinformatics 9: 386.
16. Pruitt K, Tatusova T, Maglott D (2005) NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. Nucleic Acids Research 33: D501-D504.
17. Altschul S, Madden T, Schaffer A, Zhang J, Zhang Z, et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research 25: 3389.
18. http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html
19. RefSeq release notes ftp://ftp.ncbi.nih.gov/refseq/release/release-notes/