

# Quality metrics for benchmarking sequences comparison tools

Erwan Drezen, Dominique Lavenier

► **To cite this version:**

Erwan Drezen, Dominique Lavenier. Quality metrics for benchmarking sequences comparison tools. Sérgio Campos. 9th Brazilian Symposium on Bioinformatics, BSB 2014, Oct 2014, Belo Horizonte, Brazil. Springer, 8826, pp.144-153, Advances in Bioinformatics and Computational Biology Lecture Notes in Computer Science. <[http://link.springer.com/chapter/10.1007/978-3-319-12418-6\\_18](http://link.springer.com/chapter/10.1007/978-3-319-12418-6_18)>. <hal-01088595>

**HAL Id: hal-01088595**

**<https://hal.archives-ouvertes.fr/hal-01088595>**

Submitted on 28 Nov 2014

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

## METHODOLOGY

# Quality metrics for benchmarking sequences comparison tools

Erwan Drezen\*  
and Dominique Lavenier

\*Correspondence:  
erwan.drezen@inria.fr  
INRIA/IRISA/Genscale, Campus  
de Beaulieu, Rennes, France  
Full list of author information is  
available at the end of the article

## Abstract

**Background:** Comparing sequences is a daily task in bioinformatics and many software try to fulfill this need by proposing fast execution times and accurate results. Introducing a new software in this field requires to compare it to recognized tools with the help of well defined metrics.

**Results:** A set of quality metrics is proposed that enables a systematic approach for comparing alignment tools. These metrics have been implemented in a dedicated software, allowing to produce textual and graphical benchmark artifacts.

**Keywords:** sequence; alignment; benchmark

## 1 Introduction

In Bioinformatics, the task of comparing genomic sequences is ubiquitous, leading to various software proposals over the years. With the NGS revolution, this task is becoming more and more time consuming as the size of NGS data truly explodes.

Software have tried to cope with this situation in different ways. First, by using known algorithms with optimized hardware usage. Second, by proposing new heuristics to reduce the searching space. The historically sequence comparison tools are:

- SSEARCH [1] that proposes an implementation of the Smith-Waterman algorithm with an efficient usage of multicore / SSE architecture; it provides an "exact" search alignment algorithm and is therefore slow, and consequently inadapted for large requests.
- BLAST [2] and FASTA [1] that both propose a seed based heuristics to speed-up the computation with a small loss of quality w.r.t. exact algorithms.

The trade-off between speed and quality is the key point. As a consequence, to evaluate new Alignment Search Tools (AST for short), benchmarking has to focus on time and quality metrics. Comparing execution times is straightforward. On the other hand, comparing quality is more challenging since it requires to define a precise quality metric.

The paper presents a methodology for comparing the results of AST. We explicitly target intensive sequence comparison for which there is an increasing demand due to high throughput sequencing opportunities. A quality metric is defined and applied to a set of software and benchmarks.

## 2 Benchmarking AST

### 2.1 Configuration

To fairly benchmark AST, many configuration aspects must be taken into consideration:

- 1 For a given request, AST have to be configured for providing similars results. This is far from obvious since they generally propose a large set of parameters. In some cases, they can have hidden configuration.
- 2 Some AST implicitly modified the input banks. For instance, low complexity region [3], [4] can be systematically masked. These transformations may have significant influence on the results compared to other tools that are not applying these kinds of pre-processing.
- 3 AST often provide a statistical model telling whether an alignment has to be kept or not (for instance Karlin/Altschul model of Blast). Thus, AST with different models are likely to produce different alignments.

These points have a strong impact on the alignment list generated by the AST. Consequently, they need to be included in the quality metric itself. As a side effect, the AST command lines have to be explicitly described as part of the benchmark results.

In the following, we note  $T^{(0)}$  (or  $T$  for short) an AST with its default configuration. Specific configurations of  $T$  will be noted as  $T^{(i)}$  for  $i \geq 1$ .

### 2.2 Input request

To compare AST, a minimal set of common parameters needs to be specified. The following table lists the parameters we selected:

**Table 1 Request parameters**

$r_1$ ) query bank	$r_6$ ) reward cost
$r_2$ ) subject bank	$r_7$ ) penalty cost
$r_3$ ) e-value	$r_8$ ) substitution matrix
$r_4$ ) open gap cost	$r_9$ ) low complexity filtering
$r_5$ ) extend gap cost	$r_{10}$ ) number of cores

Parameters  $r_1$  to  $r_9$  have a direct impact on the output alignments. The  $r_{10}$  parameter has an impact on the execution time. The execution time can be used to analyze the scalability of AST as a function of the number of cores.

A request  $R$  can now be defined by a set of  $r_i$ . If not all  $r_i$  are specified, it is supposed that default values are used instead. A relation of equivalence can also be defined between two requests  $x$  and  $y$ :

$$x \sim y \iff \forall i \in [1..10]; r_i(x) = r_i(y)$$

Here is an example of a request  $R_1$  :

$$R_1 = \begin{cases} r_1 = \text{"swissprot"} \\ r_2 = \text{"yeats"} \\ r_3 = 1e - 5 \end{cases}$$

### 2.3 Output data

Comparing AST qualities implies that AST generates the same nature of information. We choose the information provided by the Blast tabular output format; for each alignment, it provides 12 properties  $p_i$  as shown in table 2. Table 3 is an illustration of a Blast output.

**Table 2 Alignment properties**

$p_1$ ) query sequence id	$p_7$ ) start offset in query
$p_2$ ) subject sequence id	$p_8$ ) end offset in query
$p_3$ ) percentage of identity	$p_9$ ) start offset in subject
$p_4$ ) alignment length	$p_{10}$ ) end offset in subject
$p_5$ ) number of mismatches	$p_{11}$ ) e-value
$p_6$ ) number of gap openings	$p_{12}$ ) bitscore

Relying only on this set of properties means that only AST able to provide this information would be candidate. Fortunately, many AST can be parametrized to generate Blast-like tabular output format.

**Table 3 Example of Blast tabular output**

$p_1$	$p_2$	$p_3$	$p_4$	$p_5$	$p_6$	$p_7$	$p_8$	$p_9$	$p_{10}$	$p_{11}$	$p_{12}$
SPO13535	SPP02309	39.34	305	181	4	29	332	17	302	7.5e-62	231.6
SPO13535	SPQ91G55	34.93	209	134	2	42	250	46	252	1.5e-31	130.4
SPP02400	SPQ196T6	30.18	603	388	3	90	663	49	647	1.6e-76	282.4

## 3 Alignment Quality Metric

### 3.1 Definitions

Before specifying an alignment quality metric, we introduce the following definitions:

- $I$  is a set of integers  $i \in [1..12]$ .  $I$  specifies a subset of properties  $p_i$  defined in table 2
- For a given set  $I$ , an alignment  $a$  is an object having properties  $p_i$  for  $i \in I$
- Two alignments  $x$  and  $y$  are equivalent if their properties have the same values:

$$x \sim y \iff \forall i \in I; p_i(x) = p_i(y)$$

- an alignment set  $A$  contains alignments defined for a specific set  $I$

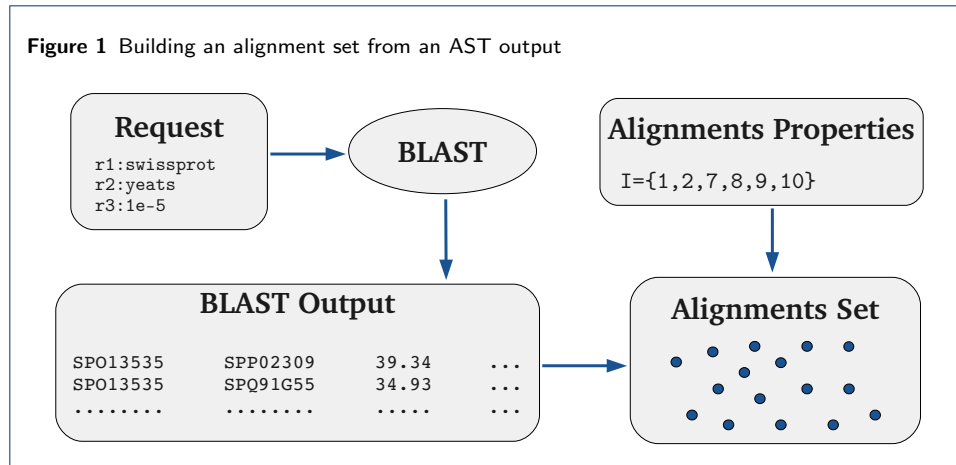
An alignment set may contain equivalent items. In such a case, items equivalent to some alignment  $a$  are grouped into an equivalence class  $\tilde{a}$ .

$$\tilde{a} = \{x \in A; a \sim x\}$$

We define  $\tilde{A}$  as the set built from  $A$  by keeping only one representant of each equivalence class  $\tilde{a}$ .

Taking the example of table 3, for  $I = \{1\}$ , we have the alignment set  $A = \{a1, a2, a3\}$  and the set of equivalent alignments  $\tilde{A} = \{\tilde{a}_1, \tilde{a}_2\}$  with:

$$\begin{cases} p_1(a1) = \text{"SPO13535"} \\ p_1(a2) = \text{"SPO13535"} \\ p_1(a3) = \text{"SPP02400"} \end{cases} \quad \begin{cases} p_1(\tilde{a}_1) = \text{"SPO13535"} \\ p_1(\tilde{a}_2) = \text{"SPP02400"} \end{cases}$$



### 3.2 From AST output to alignment set

With these definitions, an AST output can be formally transformed into an alignment set  $A$  where only some properties defined by the set  $I$  are kept. An alignment set can thus be defined as  $A := \varphi(T, R, I)$  with  $T$  the AST,  $R$  the request and  $I$  the set of properties.

Figure 1 illustrates the transformation of a Blast output.

Note also that the same output generated by a tool will be viewed differently for two distinct sets  $I_1$  and  $I_2$ . Therefore several quality metrics based on different  $I$  sets can be defined to provide different benchmark points of view.

### 3.3 Comparing AST

For a request  $R$  (defined by a set of  $r_i$  from table 1) and a given set  $I$ , we note:

$A_1 = \varphi(T_1, R, I)$ , alignment set found by AST  $T_1$  for request  $R$

$A_2 = \varphi(T_2, R, I)$ , alignment set found by AST  $T_2$  for request  $R$

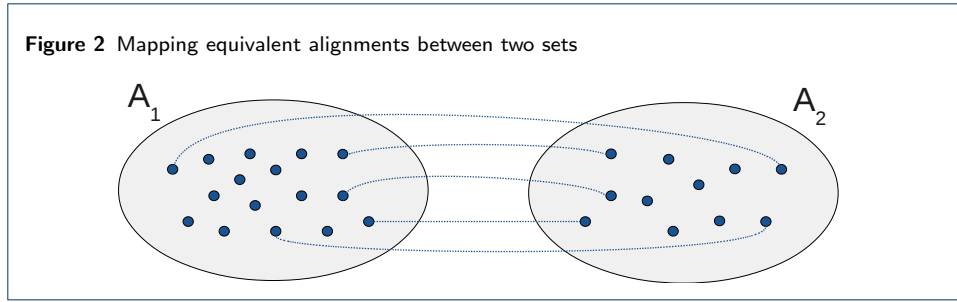
Comparing  $T_1$  to  $T_2$  consists in finding a set  $M^{(I)}$  such as:

$$M^{(I)} = \left\{ \begin{array}{l} (x, y) \in A_1 \times A_2; \quad x \sim y \\ \forall (x', y') \in A_1 \times A_2; \quad \left\{ \begin{array}{l} x \sim y' \Rightarrow y' = y \\ x' \sim y \Rightarrow x' = x \end{array} \right. \end{array} \right.$$

The intuitive idea of this definition is to map an item from one set to at most one item of the other set. Note that some alignments in one set may not be connected to an alignment in the other set as shown figure 2.

The estimation of "how close two sets  $A_1$  and  $A_2$  are" can be done by considering the cardinal  $|M^{(I)}|$  which represents the number of common alignments between  $A_1$  and  $A_2$ . We define three numbers  $N_c^{(I)}$ ,  $N_1^{(I)}$  and  $N_2^{(I)}$ :

- $N_c^{(I)} := |M^{(I)}|$ , number of common alignments between  $A_1$  and  $A_2$
- $N_1^{(I)} := |A_1| - |M^{(I)}|$ , number of alignments specific to  $A_1$
- $N_2^{(I)} := |A_2| - |M^{(I)}|$ , number of alignments specific to  $A_2$



Finally, the alignment quality metric for a given  $I$  by the  $Q^{(I)}$  function is defined as:

$$A_1 \times A_2 \xrightarrow{Q^{(I)}} (N_c^{(I)}, N_1^{(I)}, N_2^{(I)})$$

With the following properties:

- 1  $Q^{(I)}(A, A) = (|A|, 0, 0)$
- 2  $Q^{(I)}(A_1, A_2) = (x, y, z) \implies Q^{(I)}(A_2, A_1) = (x, z, y)$

Note that this definition of the alignment quality metric is relative: it only gives information for a specific couple of alignment sets. It does not provide absolute assessment for one AST. The consequence is that one of the AST ( $T_1, T_2$ ) needs to be chosen as a reference when comparing many AST together.

The percentage of alignments found by  $T_2$  in the alignments of  $T_1$  can also be defined as:

$$\alpha_2^{(I)} = \frac{N_c^{(I)}}{N_c^{(I)} + N_1^{(I)}}$$

Taking figure 2 as example, we have  $Q^{(I)}(A_1, A_2) = (5, 11, 6)$ .  $T_2$  is able to find  $\alpha_2^{(I)} = \frac{5}{5+11} = 31.25\%$  of the alignments found by  $T_1$ . Similarly,  $T_1$  finds  $\alpha_1^{(I)} = \frac{5}{5+6} = 45.45\%$  of the alignments found by  $T_2$ .

Comparing  $N$  AST  $T_i$  is simply done by computing  $Q^{(I)}(A_i, A_j)$  for all  $(i, j) \in [1..N]^2$ . Note that computation is only required for  $i \geq j$  because of the second property of the quality definition.

## 4 Results

### 4.1 List of benchmarked tools and quality metrics

The evaluation of our methodology has been done with the following tools :

$$T_1 := \text{SSEARCH} \quad T_2 := \text{BLAST} \quad T_3 := \text{PLAST} \quad T_4 := \text{UBLAST}$$

In this set of AST, SSEARCH is the only exact tool, the other ones are based on heuristics. Thus, SSEARCH is taken as the reference. In that way, it can be seen how close heuristics based tools are from an exact reference.

Heuristics based tools are configured by default to get the best trade-offs between speed and quality. Of course, these tools can also be configured by setting specific

parameters in order to get a better quality or a better speed. Table 4 gives several configurations used for the benchmark.

**Table 4** Tools specific configurations

Name	Configuration	Purpose
BLASTN	blast -task blastn	nucl/nucl requests
MEGABLAST	blast -task megablast	nucl/nucl requests
BLASTQ	blast -max-target-seqs 25000	quality improvement
UBLASTQ	ublast -accel 1	quality improvement
PLASTS	plast -seeds-use-ratio 0.01	speed improvement

Three different quality metrics are considered:

- $I_{query} = \{1\}$  with  $\tilde{A}$ , the equivalent class alignment set; this metric compares the number of matched queries between tools
- $I_{hit} = \{1, 2\}$  with  $\tilde{A}$ , the equivalent class alignment set; this metric compares the number of matched couples [subject,query] between tools
- $I_{align} = \{1, 2, 7, 8, 9, 10\}$  with  $A$ , the alignment set; this metric compares the number of alignments sharing the same [subject,query] identifiers and the same [subject,query] boundaries.

These 3 metrics go from the worst to the best precision. As a matter of fact, the corresponding sets  $I$  provide more and more selective relations of equivalence between alignments. As explained later, two AST may be close for  $I_{query}$  and even for  $I_{hit}$  but may have significant differences for  $I_{align}$ ; this is shown by small value for  $N_c^{(I_{align})}$  and big values for  $N_1^{(I_{align})}$  and  $N_2^{(I_{align})}$ .

## 4.2 Results

In the following tables, we dump  $\alpha_2^{(I)}$ , the percentage of alignments found by  $T_2$  in the alignment set of  $T_1$ .

Two ratios of execution time between  $T_1$  and  $T_2$  can be defined:

- 1  $SU_{total}$  : speedup for the total execution time of a request, including bank preparation when needed (use of *makeblastdb* for instance)
- 2  $SU_{AST}$  : speedup for the execution time of the sequences comparison only (exclude *makeblastdb* execution time for instance)

**Table 5** [Escherichia Coli vs. uniprot\_sprot, eval=1e-3, nbcores=16]

$T_1$	$T_2$	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	$SU_{total}$	$SU_{AST}$
<i>ssearch</i>	<i>blast</i>	55.5	58.1	99.8	5.48	5.80
<i>ssearch</i>	<i>plast</i>	93.9	97.1	99.8	26.22	32.54
<i>ssearch</i>	<i>ublast</i>	29.0	64.4	99.8	64.41	146.60
<i>ssearch</i>	<i>blastQ</i>	94.2	97.4	99.8	4.90	5.08
<i>ssearch</i>	<i>plastS</i>	73.2	77.1	99.4	52.07	84.43
<i>ssearch</i>	<i>ublastQ</i>	30.4	69.1	99.8	40.61	62.63

**Table 6** [Chitinophaga Pinensis vs. uniprot\_sprot, eval=1e-3, nbcores=16]

$T_1$	$T_2$	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	$SU_{total}$	$SU_{AST}$
<i>ssearch</i>	<i>blast</i>	61.4	64.2	98.7	5.33	5.45
<i>ssearch</i>	<i>plast</i>	92.0	95.9	97.9	29.23	33.26
<i>ssearch</i>	<i>ublast</i>	19.9	52.4	78.9	89.48	157.96

Tables 5 and 6 show a protein/protein comparison between swissprot [5] and two bacterias [6],[7] from which several comments can be done:

- Heuristics based tools have good speedups, especially UBLAST
- AST quality are very close to SSEARCH for  $I^{(query)}$  (except UBLAST, Table 6)
- With default configuration, BLAST and UBLAST have poor results for  $I^{(hit)}$  and  $I^{(align)}$ ; only PLAST manages to recover most of the SSEARCH results
- For two similar requests, PLAST has the smallest discrepancies for the 3 metrics, followed by BLAST and then by UBLAST

**Table 7** [uniprot\_sprot\_40000b vs. uniprot\_sprot\_40000, evalue=1e-30, nbcores=16]

$T_1$	$T_2$	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	$SU_{total}$	$SU_{AST}$
<i>ssearch</i>	<i>blast</i>	86.5	89.9	97.1	8.40	8.46
<i>ssearch</i>	<i>plast</i>	88.9	92.4	96.3	44.62	46.48
<i>ssearch</i>	<i>ublast</i>	6.4	19.9	22.8	375.46	591.03

Table 7 compares a subset of 40000 sequences from swissprot to another subset of 40000 sequences from swissprot. Here, BLAST and PLAST have similar values for the 3 metrics and are close to the reference; speedups are interesting, especially for PLAST. For this request, UBLAST is still very fast but has poor values for the 3 metrics. Other tools configurations give similar quality results.

**Table 8** [SRR142736 vs. uniprot\_sprot, evalue=1e-3, nbcores=16]

$T_1$	$T_2$	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	$SU_{total}$	$SU_{AST}$
<i>blastQ</i>	<i>blast</i>	73.8	72.3	100.0	1.00	1.00
<i>blastQ</i>	<i>plast</i>	77.7	95.4	98.1	6.70	7.10
<i>blastQ</i>	<i>ublast</i>	19.8	64.9	82.0	40.30	68.91

**Table 9** [TARA sample vs. uniprot\_sprot, evalue=1e-3, nbcores=16]

$T_1$	$T_2$	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	$SU_{total}$	$SU_{AST}$
<i>blastQ</i>	<i>blast</i>	82.1	80.8	100.0	1.00	1.00
<i>blastQ</i>	<i>plast</i>	80.8	94.7	83.1	6.50	6.68
<i>blastQ</i>	<i>ublast</i>	17.2	59.2	29.9	71.10	105.50

Table 8 shows a genomic query and table 9 a meta-genomic query [8], with swissprot as subject bank. SSEARCH can't be used as reference here so BLASTQ is used instead. BLAST and PLAST are similar for  $I^{(align)}$ , with better results for PLAST on  $I^{(hit)}$  and better results for BLAST on  $I^{(query)}$ . UBLAST is still the fastest tool with poor quality, in particular in the meta-genomic case.

**Table 10** [SRR027344 vs. nt 2000000 seqs, evalue=1e-30, nbcores=16]

$T_1$	$T_2$	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	$SU_{total}$	$SU_{AST}$
<i>blastQ</i>	<i>blast</i>	64.8	64.6	100.0	1.03	1.03
<i>blastQ</i>	<i>megablast</i>	61.1	61.0	96.6	5.17	12.48
<i>blastQ</i>	<i>plast</i>	94.2	94.2	96.8	5.38	14.07
<i>blastQ</i>	<i>ublast</i>	90.1	94.9	95.6	0.63	0.65

Table 10 is a nucleotide/nucleotide request, comparing a set of reads to 2,000,000 sequences from the nt [9] database. Here, PLAST and UBLAST provide the best quality metrics w.r.t. BLASTQ, with good speedups for PLAST.

In brief, we summarize this benchmark with the following observations:

- As expected, heuristics based tools are fast at the expense of quality; for instance, UBLAST is very fast but has often poor quality for the chosen metrics



- Surprisingly, BLAST with default configuration has sometimes poor quality for some metrics; a special configuration like BLASTQ is needed for recovering full quality
- PLAST is a good trade-off between speed and quality; it is faster than BLAST with similar quality values.

## 5 Conclusion

In this paper, we have defined a set of quality metrics for comparing alignment search tools (AST). Comparing AST is important to understand how the AST behave and what should be expected from the alignments they produce. In particular, these metrics allow the distance between exact and heuristics based tool to be precisely evaluated.

These metrics have been tested on standard AST. The benchmark results provides some interesting hints that can be used to match a specific tool with user needs; for instance, PLAST is a good trade-off in terms of speed/quality, and UBLAST is interesting if speed is crucial and if quality loss is acceptable (particularly for alignment and hit metrics).

Other quality metrics are currently under investigation to better capture user needs according to the application domains where AST are involved. A next step is also to gather in a public database many experimentatons and offer to the scientific community a way to select the best AST according to their needs. Other AST will also be added to extend the scope of this methodology. Finally, a web interface [10] is under development and already provides a graphical representation of the metrics.

## References

1. Pearson, Lipman: Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 2444–2448 (1988)
2. Altschul, Gish, Miller, Myers, Lipman: Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410 (1990)
3. Wootton, J.C., Federhen, S.: Statistics of local complexity in amino acid sequences and sequence databases. *Computers and Chemistry* **17**, 149–163 (1993)
4. Morgulis, Gertz, Schaffer, Agarwala: Fast and Symmetric DUST Implementation to Mask Low-Complexity DNA Sequences
5. Swissprot Bank.  
[ftp://ftp.uniprot.org/pub/databases/uniprot/current\\_release/knowledgebase/complete/uniprot\\_sprot.fasta.gz](ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complete/uniprot_sprot.fasta.gz)
6. Proteome of Escherichia Coli. [ftp://ftp.ncbi.nlm.nih.gov/genbank/genomes/Bacteria/Escherichia\\_coli\\_K\\_12\\_-substr\\_DH10B\\_uid20079/CP000948.faa](ftp://ftp.ncbi.nlm.nih.gov/genbank/genomes/Bacteria/Escherichia_coli_K_12_-substr_DH10B_uid20079/CP000948.faa)
7. Proteome of Chitinophaga Pinensis. [ftp://ftp.ncbi.nlm.nih.gov/genbank/genomes/Bacteria/Chitinophaga\\_-pinensis.DSM.2588\\_uid27951/CP001699.faa](ftp://ftp.ncbi.nlm.nih.gov/genbank/genomes/Bacteria/Chitinophaga_-pinensis.DSM.2588_uid27951/CP001699.faa)
8. Environmental Samples from TARA.  
<ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/ERR/ERR550/ERR550538/ERR550538.sra>
9. Nt Database. <ftp://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/nt.gz>
10. Web Interface for Metrics Display. <http://irisa.lavenier.net/cast/ISCB.LA.2014/>