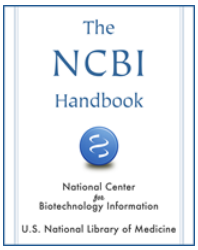
U.S. National Library of Medicine
National Center for Biotechnology Information

NLM Citation: Madden T. The BLAST Sequence Analysis Tool. 2002 Oct 9 [Updated 2003 Aug 13]. In: McEntyre J, Ostell J, editors. The NCBI Handbook [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2002-. Chapter 16.
Bookshelf URL: https://www.ncbi.nlm.nih.gov/books/



Chapter 16. The BLAST Sequence Analysis Tool

Tom Madden

Created: October 9, 2002; Updated: August 13, 2003.

Summary

The comparison of nucleotide or protein sequences from the same or different organisms is a very powerful tool in molecular biology. By finding similarities between sequences, scientists can infer the function of newly sequenced genes, predict new members of gene families, and explore evolutionary relationships. Now that whole genomes are being sequenced, sequence similarity searching can be used to predict the location and function of protein-coding and transcription-regulation regions in genomic DNA.

Basic Local Alignment Search Tool (BLAST) (1, 2) is the tool most frequently used for calculating sequence similarity. BLAST comes in variations for use with different query sequences against different databases. All BLAST applications, as well as information on which BLAST program to use and other help documentation, are listed on the BLAST homepage. This chapter will focus more on how BLAST works, its output, and how both the output and program itself can be further manipulated or customized, rather than on how to use BLAST or interpret BLAST results.

Introduction

The way most people use BLAST is to input a nucleotide or protein sequence as a query against all (or a subset of) the public sequence databases, pasting the sequence into the textbox on one of the BLAST Web pages. This sends the query over the Internet, the search is performed on the NCBI databases and servers, and the results are posted back to the person's browser in the chosen display format. However, many biotech companies, genome scientists, and bioinformatics personnel may want to use "stand-alone" BLAST to query their own, local databases or want to customize BLAST in some way to make it better suit their needs. Stand-alone BLAST comes in two forms: the executables that can be run from the command line; or the Standalone WWW BLAST Server, which allows users to set up their own in-house versions of the BLAST Web pages.

There are many different variations of BLAST available to use for different sequence comparisons, e.g., a DNA query to a DNA database, a protein query to a protein database, and a DNA query, translated in all six reading frames, to a protein sequence database. Other adaptations of BLAST, such as PSI-BLAST (for iterative protein sequence similarity searches using a position-specific score matrix) and RPS-BLAST (for searching for protein domains in the Conserved Domains Database, Chapter 3) perform comparisons against sequence profiles.

This chapter will first describe the BLAST architecture—how it works at the NCBI site—and then go on to describe the various BLAST outputs. The best known of these outputs is the default display from BLAST Web pages, the so-called "traditional report". As well as obtaining BLAST results in the traditional report, results can also be delivered in structured output, such as a hit table (see below), XML, or ASN.1. The optimal choice of output format depends upon the application. The final part of the chapter discusses stand-alone BLAST and

describes possibilities for customization. There are many interfaces to BLAST that are often not exploited by users but can lead to more efficient and robust applications.

How BLAST Works: The Basics

The BLAST algorithm is a heuristic program, which means that it relies on some smart shortcuts to perform the search faster. BLAST performs "local" alignments. Most proteins are modular in nature, with functional domains often being repeated within the same protein as well as across different proteins from different species. The BLAST algorithm is tuned to find these domains or shorter stretches of sequence similarity. The local alignment approach also means that a mRNA can be aligned with a piece of genomic DNA, as is frequently required in genome assembly and analysis. If instead BLAST started out by attempting to align two sequences over their entire lengths (known as a global alignment), fewer similarities would be detected, especially with respect to domains and motifs.

When a query is submitted via one of the BLAST Web pages, the sequence, plus any other input information such as the database to be searched, word size, expect value, and so on, are fed to the algorithm on the BLAST server. BLAST works by first making a look-up table of all the “words” (short subsequences, which for proteins the default is three letters) and “neighboring words”, i.e., similar words in the query sequence. The sequence database is then scanned for these “hot spots”. When a match is identified, it is used to initiate gap-free and gapped extensions of the “word”.

BLAST does not search GenBank flatfiles (or any subset of GenBank flatfiles) directly. Rather, sequences are made into BLAST databases. Each entry is split, and two files are formed, one containing just the header information and one containing just the sequence information. These are the data that the algorithm uses. If BLAST is to be run in “stand-alone” mode, the data file could consist of local, private data, downloaded NCBI BLAST databases, or a combination of the two.

After the algorithm has looked up all possible "words" from the query sequence and extended them maximally, it assembles the best alignment for each query–sequence pair and writes this information to an SeqAlign data structure (in ASN.1 ; also used by Sequin, see Chapter 12). The SeqAlign structure in itself does not contain the sequence information; rather, it refers to the sequences in the BLAST database (Figure 1).

The BLAST Formatter, which sits on the BLAST server, can use the information in the SeqAlign to retrieve the similar sequences found and display them in a variety of ways. Thus, once a query has been completed, the results can be reformatted without having to re-execute the search. This is possible because of the QBLAST system.

BLAST Scores and Statistics

Once BLAST has found a similar sequence to the query in the database, it is helpful to have some idea of whether the alignment is “good” and whether it portrays a possible biological relationship, or whether the similarity observed is attributable to chance alone. BLAST uses statistical theory to produce a bit score and expect value (E-value) for each alignment pair (query to hit).

The bit score gives an indication of how good the alignment is; the higher the score, the better the alignment. In general terms, this score is calculated from a formula that takes into account the alignment of similar or identical residues, as well as any gaps introduced to align the sequences. A key element in this calculation is the “substitution matrix ”, which assigns a score for aligning any possible pair of residues. The BLOSUM62 matrix is the default for most BLAST programs, the exceptions being blastn and MegaBLAST (programs that perform nucleotide–nucleotide comparisons and hence do not use protein-specific matrices). Bit scores are normalized,

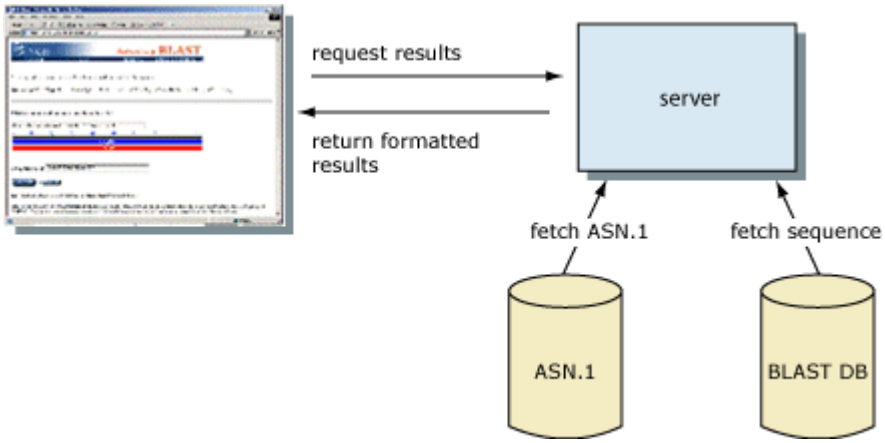


Figure 1. How the BLAST results Web pages are assembled. The QBLAST system located on the BLAST server executes the search, writing information about the sequence alignment in ASN.1. The results can then be formatted by fetching the ASN.1 (*fetch ASN.1*) and fetching the sequences (*fetch sequence*) from the BLAST databases. Because the execution of the search algorithm is decoupled from the formatting, the results can be delivered in a variety of formats without re-running the search.

which means that the bit scores from different alignments can be compared, even if different scoring matrices have been used.

The E-value gives an indication of the statistical significance of a given pairwise alignment and reflects the size of the database and the scoring system used. The lower the E-value, the more significant the hit. A sequence alignment that has an E-value of 0.05 means that this similarity has a 5 in 100 (1 in 20) chance of occurring by chance alone. Although a statistician might consider this to be significant, it still may not represent a biologically meaningful result, and analysis of the alignments (see below) is required to determine "biological" significance.

BLAST Output: 1. The Traditional Report

Most BLAST users are familiar with the so-called "traditional" BLAST report. The report consists of three major sections: (1) the header, which contains information about the query sequence, the database searched (Figure 2). On the Web, there is also a graphical overview (Figure 3); (2) the one-line descriptions of each database sequence found to match the query sequence; these provide a quick overview for browsing (Figure 4); (3) the alignments for each database sequence matched (Figure 5) (there may be more than one alignment for a database sequence it matches).

The traditional report is really designed for human readability, as opposed to being parsed by a program. For example, the one-line descriptions are useful for people to get a quick overview of their search results, but they are rarely complete descriptors because of limited space. Also, for convenience, there are several pieces of information that are displayed in both the one-line descriptions and alignments (for example, the E-values, scores, and descriptions); therefore, the person viewing the search output does not need to move back and forth between sections.

New features may be added to the report, e.g., the addition of links to Entrez Gene records (Chapter 19) from sequence hits, which result in a change of output format. These are easy for people to pick up on and take advantage of but can trip programs that parse this BLAST output.

By default, a maximum of 500 sequence matches are displayed, which can be changed on the advanced BLAST page with the **Alignments** option. Many components of the BLAST results display via the Internet and are hyperlinked to the same information at different places in the page, to additional information including help

documentation, and to the Entrez sequence records of matched sequences. These records provide more information about the sequence, including links to relevant research abstracts in PubMed.

BLAST Output: 2. The Hit Table

Although the traditional report is ideal for investigating the characteristics of one gene or protein, often scientists want to make a large number of BLAST runs for a specialized purpose and need only a subset of the information contained in the traditional BLAST report. Furthermore, in cases where the BLAST output will be processed further, it can be unreliable to parse the traditional report. The traditional report is merely a display format with no formal structure or rules, and improvements may be made at any time, changing the underlying HTML. The hit table format provides a simple and clean alternative (Figure 6).

The screening of many newly sequenced human Expressed Sequence Tags (ESTs) for contamination by the *Escherichia coli* cloning vector is a good example of when it is preferable to use the hit table output over the traditional report. In this case, a strict, high E-value threshold would be applied to differentiate between contaminating *E. coli* sequence and the human sequence. Those human ESTs that find very strong, near-exact *E.coli* sequence matches can be discarded without further examination. (Borderline cases may require further examination by a scientist.)

For these purposes, the hit table output is more useful than the traditional report; it contains only the information required in a more formal structure. The hit table output contains no sequences or definition lines, but for each sequence matched, it lists the sequence identifier, the start and stop points for stretches of sequence similarity (offset by one residue), the percent identity of the match, and the E-value.

BLAST Output: 3. Structured Output

There are drawbacks to parsing both the BLAST report and even the simpler hit table. There is no way to automatically check for truncated or otherwise corrupted output in cases when a large number of sequences are being screened. (This may happen if the disk is full, for example.) Also, there is no rigorous check for syntax changes in the output, such as the addition of new features, which can lead to erroneous parsing. Structured output allows for automatic and rigorous checks for syntax errors and changes. Both XML and ASN.1 are examples of structured output in which there are built-in checks for correct and complete syntax and structure. (In the case of XML, for example, this is ensured by the necessity for matching tags and the DTD.) For text reports, there is often no specification, but perhaps a (incomplete) description of the file is written afterward.

ASN.1 Is Used by the BLAST Server

As well as the hit table and traditional report shown in HTML, BLAST results can also be formatted in plain text, XML, and ASN.1 (Figure 7), and what's more, the format for a given BLAST result can be changed without re-executing the search.

A change in BLAST format without re-executing the search is possible because when a scientist looks at a Web page of BLAST results at NCBI, the HTML that makes that page has been created from ASN.1 (Figure 7). Although the formatted results are requested from the server, the information about the alignments is fetched from a disk in ASN.1, as are the corresponding sequences from the BLAST databases (see Figure 1). The formatter on the BLAST server then puts these results together as a BLAST report. The BLAST search itself has been uncoupled from the way the result is formatted, thus allowing different output formats from the same search. The strict internal validation of ASN.1 ensures that these output formats can always be produced reliably.

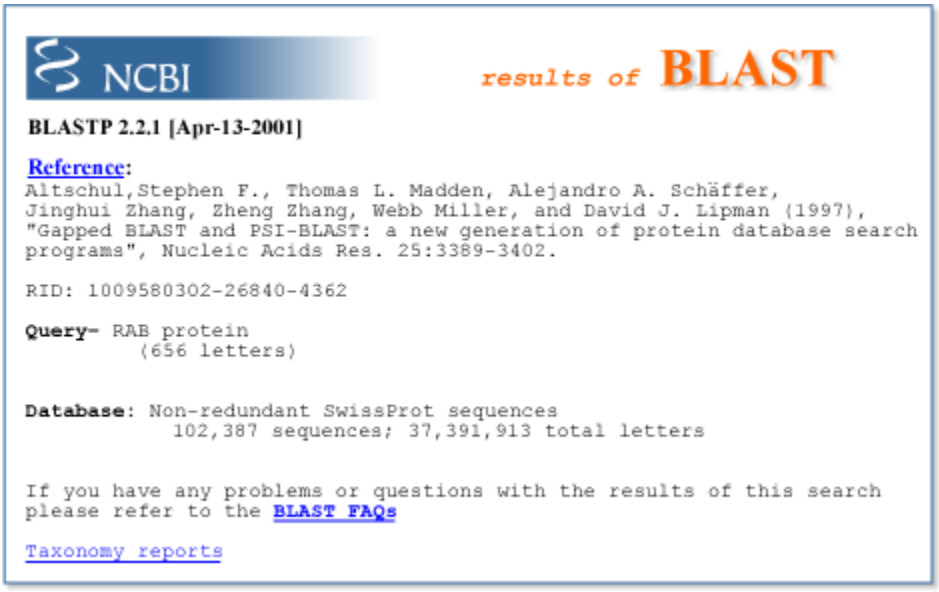


Figure 2. The BLAST report header. The *top line* gives information about the type of program (in this case, *BLASTP*), the version (*2.2.1*), and a version release date. The research paper that describes BLAST is then cited, followed by the request ID (issued by QBLAST), the query sequence definition line, and a summary of the database searched. The **Taxonomy reports** link displays this BLAST result on the basis of information in the Taxonomy database (Chapter 4).

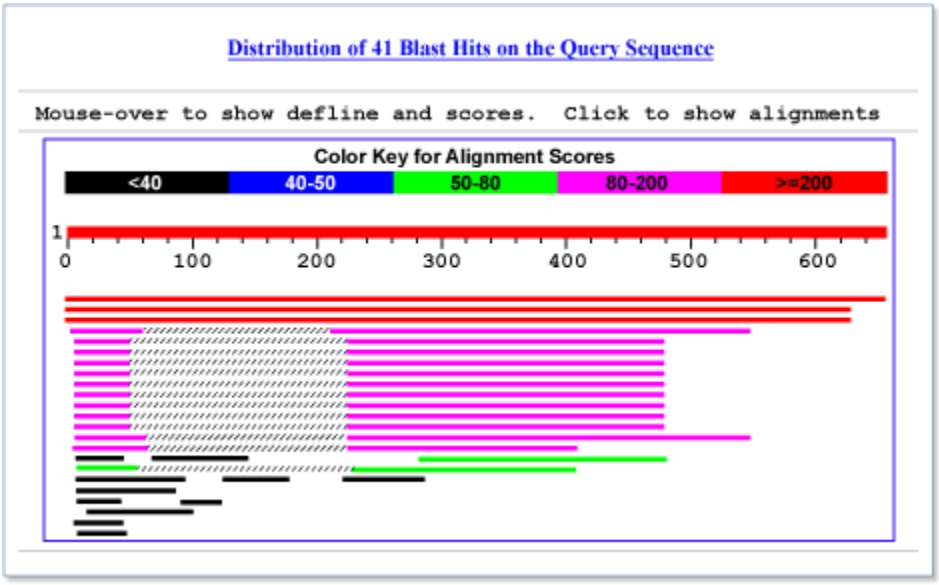


Figure 3. Graphical overview of BLAST results. The query sequence is represented by the *numbered red bar* at the *top* of the figure. Database hits are shown aligned to the query, *below* the red bar. Of the aligned sequences, the most similar are shown closest to the query. In this case, there are three high-scoring database matches that align to most of the query sequence. The next twelve bars represent lower-scoring matches that align to two regions of the query, from about residues 3–60 and residues 220–500. The *cross-hatched parts* of the these bars indicate that the two regions of similarity are on the same protein, but that this intervening region does not match. The remaining bars show lower-scoring alignments. Mousing over the bars displays the definition line for that sequence to be shown in the window above the graphic.

Sequences producing significant alignments:		Score (bits)	E Value
(a)	(b)	(c)	(d)
gi|116365|sp|P26374|RAE2_HUMAN	Rab proteins geranylgeranylt...	1216	0.0
gi|21431807|sp|P24386|RAE1_HUMAN	Rab proteins geranylgerany...	879	0.0
gi|585775|sp|P37727|RAE1_RAT	Rab proteins geranylgeranyltra...	846	0.0
gi|13626886|sp|Q61598|GDIC_MOUSE	RAB GDP dissociation inhib...	127	5e-29
gi|729566|sp|P39958|GDI1_YEAST	SECRETORY PATHWAY GDP DISSOC...	127	5e-29
gi|13626813|sp|O97556|GDIB_CANFA	Rab GDP dissociation inhib...	126	1e-28
gi|13638229|sp|P50397|GDIB_MOUSE	RAB GDP dissociation inhib...	125	3e-28
gi|1707888|sp|P50398|GDIA_RAT	RAB GDP dissociation inhibito...	124	7e-28
gi|121108|sp|P21856|GDIA_BOVIN	Rab GDP dissociation inhibit...	124	7e-28
gi|21903424|sp|P50396|GDIA_MOUSE	Rab GDP dissociation inhib...	124	7e-28
gi|13626812|sp|O97555|GDIA_CANFA	RAB GDP dissociation inhib...	124	8e-28
gi|1707886|sp|P31150|GDIA_HUMAN	Rab GDP dissociation inhibi...	123	9e-28
gi|13638228|sp|P50395|GDIB_HUMAN	Rab GDP dissociation inhib...	122	2e-27
gi|1707891|sp|P50399|GDIB_RAT	RAB GDP DISSOCIATION INHIBITO...	121	5e-27
gi|1723467|sp|Q10305|YD4C_SCHPO	Putative secretory pathway ...	120	8e-27
gi|585776|sp|P32864|RAEP_YEAST	RAB proteins geranylgeranylt...	97	7e-20
gi|10720243|sp|O93831|RAEP_CANAL	RAB proteins geranylgerany...	74	9e-13
gi|2498411|sp|Q49398|GLF_MYCGE	UDP-galactopyranose mutase	35	0.63
gi|11135401|sp|Q9XBQ9|STHA_AZOVI	Soluble pyridine nucleotid...	34	1.0
gi|11135075|sp|O05139|STHA_PSEFL	Soluble pyridine nucleotid...	33	1.3
gi|11135195|sp|P57112|STHA_PSEAE	Soluble pyridine nucleotid...	33	1.8
gi|22257022|sp|Q8TZJ8|RLA0_PYRFU	Acidic ribosomal protein P...	33	2.1
gi|3915516|sp|P94488|YNAJ_BACSU	Hypothetical symporter ynaJ	32	3.4
gi|231788|sp|P30599|CHS2_USTMA	CHITIN SYNTHASE 2 (CHITIN-UD...	32	3.7
gi|2498412|sp|P75499|GLF_MYCPN	UDP-galactopyranose mutase	32	4.2
gi|547891|sp|P36225|MAP4_BOVIN	Microtubule-associated prote...	32	4.2
gi|586602|sp|P37747|GLF_ECOLI	UDP-galactopyranose mutase	32	4.6

Figure 4. One-line descriptions in the BLAST report. Each line is composed of four fields: (*a*) the gi number, database designation, Accession number, and locus name for the matched sequence, separated by vertical bars (Appendix 1); (*b*) a brief textual description of the sequence, the definition. This usually includes information on the organism from which the sequence was derived, the type of sequence (e.g., mRNA or DNA), and some information about function or phenotype. The definition line is often truncated in the one-line descriptions to keep the display compact; (*c*) the alignment score in bits. Higher scoring hits are found at the top of the list; and (*d*) the E-value, which provides an estimate of statistical significance. For the first hit in the list, the gi number is *116365*, the database designation is *sp* (for SWISS-PROT), the Accession number is *P26374*, the locus name is *RAE2_HUMAN*, the definition line is *Rab proteins*, the score is *1216*, and the E-value is *0.0*. Note that the first 17 hits have very low E-values (much less than 1) and are either RAB proteins or GDP dissociation inhibitors. The other database matches have much higher E-values, 0.5 and above, which means that these sequences may have been matched by chance alone.

Information about the Alignment Is Contained within a SeqAlign

SeqAlign is the ASN.1 object that contains the alignment information about the BLAST search. The SeqAlign does not contain the actual sequence that was found in the match but does contain the start, stop, and gap information, as well as scores, E-values, sequence identifiers, and (DNA) strand information.

As mentioned above, the actual database sequences are fetched from the BLAST databases when needed. This means that an identifier must uniquely identify a sequence in the database. Furthermore, the query sequence cannot have the same identifier as any sequence in the database unless the query sequence itself is in the database. If one is using stand-alone BLAST with a custom database, it is possible to specify that every sequence is uniquely identified by using the **–O** option with formatdb (the program that converts FASTA files to BLAST database format). This also indexes the entries by identifier. Similarly, the **–J** option in the (stand-alone) programs blastall, blastpgp, megablast, or rpsblast certifies that the query does not use an identifier already in the database for a different sequence. If the **–O** and **–J** options are not used, BLAST assigns unique identifiers (for that run) to all sequences and shields the user from this knowledge.

```
>gi|116365|sp|P26374|RAE2_HUMAN   Rab proteins geranylgeranyltransferase component A 2 (Rab escort
          protein 2) (REP-2) (Choroideraemia-like protein)
         Length = 656

 Score =  846 bits (2186), Expect = 0.0
 Identities = 432/632 (68%), Positives = 489/632 (77%), Gaps = 13/632 (2%)

Query: 1   MADNLPTEFDVVIIGTGLPESILAAACSRSGQRVLHIDSRSYYGGNWASFSFSGLLSWLK 60
           MADNLP++FDV++IGTGLPESI+AAACSRSGQRVLH+DSRSYYGGNWASFSFSGLLSWLK
Sbjct: 1   MADNLPSDFDVIVIGTGLPESIIAAACSRSGQRVLHVDSRSYYGGNWASFSFSGLLSWLK 60

Query: 61  EYQQNNDIGEESTVVWQDLIHETEEAITLRKKDETIQHTEAFPYASQDMEDNVEEIGALQ 120
           EYQ+NND+  E++ +WQ+ I E EEAI L  KD+TIQH E F YASQD+  +VEE GALQ
Sbjct: 61  EYQENNDVVTENS-MWQEQILENEEAIPLSSKDKTIQHVEVFCYASQDLHKDVEEAGALQ 119

Query: 121 KNPSLGVS----NTFTEVLDSALPEESQLSYFNSDEMPAKHTQKSDTEISLEVTDVEESV 176
           KN +   S          S LP   +     S E+PA+ +Q    E S EV D E +
Sbjct: 120 KNHASVTSAQSAEAAEAAETSCLPTAVEPLSMGSCEIPAEQSQCPGPESSPEVNDAEATG 179

Query: 177 EKEKYCGDKTCMHTVXXXXXXXXXXXXTVEDKADEPIRNRITYSQIVKEGRRFNIDLVSK 236
           +KE      +                 V+D  + P +NRITYSQI+KEGRRFNIDLVS+
Sbjct: 180 KKENSDAKSS--------TEEPSENVPKVQDNTETPKKNRITYSQIIKEGRRFNIDLVSQ 231

Query: 237 LLYSQGLLIDLLIKSDVSRYVEFKNVTRILAFREGKVEQVPCSRADVFNSKELTMVEKRM 296
           LLYS+GLLIDLLIKS+VSRY EFKN+TRILAFREG VEQVPCSRADVFNSK+LTMVEKRM
Sbjct: 232 LLYSRGLLIDLLIKSNVSRYAEFKNITRILAFREGTVEQVPCSRADVFNSKQLTMVEKRM 291

Query: 297 LMKFLTFCLEYEQHPDEYQAFRQCSFSEYLKTKKLTPNLQHFVLHSIAMTSESSCTTIDG 356
           LMKFLTFC+EYE+HPDEY+A+    +FSEYLKT+KLTPNLQ+FVLHSIAMTSE++  T+DG
Sbjct: 292 LMKFLTFCVEYEEHPDEYRAYEGTTFSEYLKTQKLTPNLQYFVLHSIAMTSETTSCTVDG 351

Query: 357 LNATKNFLQCLGRFGNTPFLFPLYGQGEIPQGFCRMCAVFGGIYCLRHKVQCFVVDKESG 416
           L ATK FLQCLGR+GNTPFLFPLYGQGE+PQ FCRMCAVFGGIYCLRH VQC VVDKES
Sbjct: 352 LKATKKFLQCLGRYGNTPFLFPLYGQGELPQCFCRMCAVFGGIYCLRHSVQCLVVDKESR 411

Query: 417 RCKAIIDHFGQRINAKYFIVEDSYLSEETCSNVQYKQISRAVLITDQSILKTDLDQQTSI 476
           +CKA+ID FGQRI +K+FI+EDSYLSE TCS VQY+QISRAVLITD S+LKTD DQQ SI
Sbjct: 412 KCKAVIDQFGQRIISKHFIIEDSYLSENTCSRVQYRQISRAVLITDGSVLKTDADQQVSI 471

Query: 477 LIVPPAEPGACAVRVTELCSSTMTCMKDTYLVHLTCSSSKTAREDLESVVKKLFTPYTET 536
           L VP  EPG+  VRV ELCSSTMTCMK TYLVHLTC SSKTAREDLE VV+KLFTPYTE
Sbjct: 472 LAVPAEEPGSFGVRVIELCSSTMTCMKGTYLVHLTCMSSKTAREDLERVVQKLFTPYTEI 531

Query: 537 EINEEELTKPRLLWALYFNMRDSSGISRSSYNGLPSNVYVCSGPDCGLGNEHAVKQAETL 596
           E   E++ KPRLLWALYFNMRDSS ISR  YN LPSNVYVCSGPD GLGN++AVKQAETL
Sbjct: 532 EAENEQVEKPRLLWALYFNMRDSSDISRDCYNDLPSNVYVCSGPDSGLGNDNAVKQAETL 591

Query: 597 FQXXXXXXXXXXXXXXXXXXXXDGDDKQPEAP 628
           FQ                   DGD  Q E P
Sbjct: 592 FQQICPNEDFCPAPPNPEDIVLDGDSSQQEVP 623
```

Figure 5. A pairwise sequence alignment from a BLAST report. The alignment is preceded by the sequence identifier, the full definition line, and the length of the matched sequence, in amino acids. Next comes the bit score (the raw score is in *parentheses*) and then the E-value. The following line contains information on the number of identical residues in this alignment (*Identities*), the number of conservative substitutions (*Positives*), and if applicable, the number of gaps in the alignment. Finally, the actual alignment is shown, with the query on *top*, and the database match is labeled as *Sbjct*, below. The numbers at *left* and *right* refer to the position in the amino acid sequence. One or more dashes (–) within a sequence indicate insertions or deletions. Amino acid residues in the query sequence that have been masked because of low complexity are replaced by Xs (see, for example, the *fourth* and *last* blocks). The line between the two sequences indicates the similarities between the sequences. If the query and the subject have the same amino acid at a given location, the residue itself is shown. Conservative substitutions, as judged by the substitution matrix, are indicated with +.

```
# BLASTN 2.2.1 [Aug-1-2001]
# Database: ecoli
# Query:gi|4730899|dbj|AP000130.1|Homo sapiens genomic DNA of 21q22.1, GART and AML, f43D11-119B8 region, segment 5/10.
# Fields: Query id, Subject id, % identity, alignment length, mismatches, gap openings, q. start, q. end, s. start, s. end, e-value, bit
gi|4730899|dbj|AP000130.1|AP000130     gi|2367099|gb|AE000133.1|AE000133      100.00   1198    0       0       52913   54110   10943
gi|4730899|dbj|AP000130.1|AP000130     gi|1789919|gb|AE000427.1|AE000427      100.00   1198    0       0       52913   54107   2347
gi|4730899|dbj|AP000130.1|AP000130     gi|1789607|gb|AE000401.1|AE000401      100.00   1198    0       0       52913   54107   6037
gi|4730899|dbj|AP000130.1|AP000130     gi|1788338|gb|AE000294.1|AE000294      100.00   1198    0       0       52913   54107   5700
gi|4730899|dbj|AP000130.1|AP000130     gi|1787588|gb|AE000231.1|AE000231      100.00   1198    0       0       52913   54107   4146
gi|4730899|dbj|AP000130.1|AP000130     gi|1786875|gb|AE000170.1|AE000170      100.00   1198    0       0       52913   54107   2321
gi|4730899|dbj|AP000130.1|AP000130     gi|1786751|gb|AE000160.1|AE000160      100.00   1198    0       0       52913   54107   9133
gi|4730899|dbj|AP000130.1|AP000130     gi|1788508|gb|AE000308.1|AE000308      99.92    1198    1       0       52913   54107   11740
gi|4730899|dbj|AP000130.1|AP000130     gi|2367181|gb|AE000381.1|AE000381      99.83    1198    2       0       52913   54107   2030
gi|4730899|dbj|AP000130.1|AP000130     gi|1788290|gb|AE000291.1|AE000291      99.58    1198    5       0       52910   54107   5290
gi|4730899|dbj|AP000130.1|AP000130     gi|1787633|gb|AE000234.1|AE000234      93.21    1105    72      3       52912   54014   1146
```

Figure 6. BLAST output in hit table format. This shows the results of a search of an *E. coli* database using a human sequence as a query. The lines starting with a # sign should be considered comments and ignored. The *last comment line* lists the fields in the table.

Any BLAST database or FASTA file from the NCBI Web site that contains gi numbers already satisfies the uniqueness criterion. Unique identifiers are normally a problem only when custom databases are produced and care is not taken in assigning identifiers. The identifier for a FASTA entry is the first token (meaning the letters

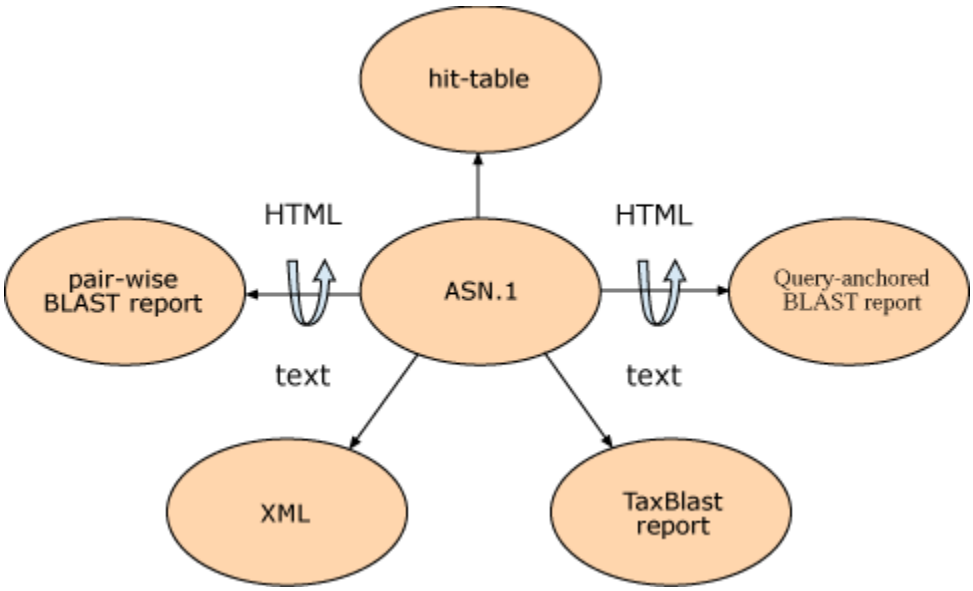


Figure 7. The different output formats that can be produced from ASN.1. Note that some nodes can be viewed as both HTML and text. XML is also structured output but can be produced from ASN.1 because it has equivalent information.

up to the first space) after the > sign on the definition line. The simplest case is to simply have a unique token (e.g., 1, 2, and so on), but it is possible to construct more complicated identifiers that might, for example, describe the data source. For the FASTA identifiers to be reliably parsed, it is necessary for them to follow a specific syntax (see Appendix 1).

More information on the SeqAlign produced by BLAST can be found here or be downloaded as a PowerPoint presentation, as well as from the NCBI Toolkit Software Developer's handbook.

XML

XML and ASN.1 are both structured languages and can express the same information; therefore, it is possible to produce a SeqAlign in XML. Some users do not find the format of the information in the SeqAlign to be convenient because it does not contain actual sequence information, and when the sequence is fetched from the BLAST database, it is packed two or four bases per byte. Typically, these users are familiar with the BLAST report and want something similar but in a format that can be parsed reliably. The XML produced by BLAST meets this need, containing the query and database sequences, sequence definition lines, the start and stop points of the alignments (one offset), as well as scores, E-values, and percent identity. There is a public DTD for this XML output.

BLAST Code

The BLAST code is part of the NCBI Toolkit, which has many low-level functions to make it platform independent; the Toolkit is supported under Linux and many varieties of UNIX, NT, and MacOS. To use the Toolkit, developers should write a function “Main”, which is called by the Toolkit “main”. The BLAST code is contained mostly in the tools directory (see Appendix 2 for an example).

The BLAST code has a modular design. For example, the Application Programming Interface (API) for retrieval from the BLAST databases is independent of the compute engine. The compute engine is independent from the formatter; therefore, it is possible (as mentioned above) to compute results once but view them in many different modes.

Readdb API

The readdb API can be used to easily extract information from the BLAST databases. Among the data available are the date the database was produced, the title, the number of letters, number of sequences, and the longest sequence. Also available are the sequence and description of any entry. The latest version of the BLAST databases also contains a taxid (an integer specifying some node of the NCBI taxonomy tree; see Chapter 4). Users are strongly encouraged to use the readdb API rather than reading the files associated with the database, because the the files are subject to change. The API, on the other hand, will support the newest version, and an attempt will be made to support older versions. See Appendix 2 for an example of a simple program (db2fasta.c) that demonstrates the use of the readdb API.

Performing a BLAST Search with C Function Calls

Only a few function calls are needed to perform a BLAST search. Appendix 3 shows an excerpt from a Demonstration Program doblast.c.

Formatting a SeqAlign

MySeqAlignPrint (called in the example in Appendix 3) is a simple function to print a view of a SeqAlign (see Appendix 4).

Appendix 1. FASTA identifiers

The syntax of the FASTA definition lines used in the NCBI BLAST databases depends upon the database from which each sequence was obtained (see Chapter 1 on GenBank). Table 1 shows how the sequence source databases are identified.

For example, if the identifier of a sequence in a BLAST result is gb|M73307|AGMA13GT, the gb tag indicates that sequence is from GenBank, M73307 is the GenBank Accession number, and AGMA13GT is the GenBank locus.

The bar (|) separates different fields. In some cases, a field is left empty, although the original specification called for including this field. To make these identifiers backwards-compatible for older parsers, the empty field is denoted by an additional bar (||).

A gi identifier has been assigned to each sequence in NCBI's sequence databases. If the sequence is from an NCBI database, then the gi number appears at the beginning of the identifier in a traditional report. For example, gi|16760827|ref|NP_456444.1 indicates an NCBI reference sequence with the gi number 16760827 and Accession number NP_456444.1. (In stand-alone BLAST, or when running BLAST from the command line, the **–I** option should be used to display the gi number.)

The reason for adding the gi identifier is to provide a uniform, stable naming convention. If a nucleotide or protein sequence changes (for example, if it is edited by the original submitter of the sequence), a new gi identifier is assigned, but the Accession number of the record remains unchanged. Thus, the gi identifier provides a mechanism for identifying the exact sequence that was used or retrieved in a given search. This is also useful when creating crosslinks between different Entrez databases (Chapter 15).

Table 1. Database identifiers in FASTA definition lines.

Database name	Identifier syntax
GenBank	gb|accession|locus
EMBL Data Library	emb|accession|locus

Table 1 continued from previous page.

Database name	Identifier syntax
DDBJ, DNA Database of Japan	dbj|accession|locus
NBRF PIR	pir||entry
Protein Research Foundation	prf||name
SWISS-PROT	sp|accession|entry name
Brookhaven Protein Data Bank	pdb|entry|chain
Patents	pat|country|number
GenInfo Backbone Id	bbs|number
General database identifier[a]	gnl|database|identifier
NCBI Reference Sequence	ref|accession|locus
Local Sequence identifier	lcl|identifier

a gnl allows databases not included in this list to use the same identifying syntax. This is used for sequences in the **trace databases**, e.g., gnl|ti|53185177. The combination of the second and third fields should be unique.

Appendix 2. Readdb API

A simple program (db2fasta.c) that demonstrates the use of the readdb API.

```
Int2 Main (void)
{
    BioseqPtr bsp;
    Boolean is_prot;
    ReadDBFILEPtr rdfp;
    FILE *fp;
    Int4 index;
if (! GetArgs ("db2fasta", NUMARG, myargs))
{
    return (1);
}
    if (myargs[1].intvalue)
    is_prot = TRUE;
    else
    is_prot = FALSE;
    fp = FileOpen("stdout", "w");
    rdfp = readdb_new(myargs[0].strvalue, is_prot);
    index = readdb_acc2fasta(rdfp, myargs[2].strvalue);
    bsp = readdb_get_bioseq(rdfp, index);
    BioseqRawToFasta(bsp, fp, !is_prot);
    bsp = BioseqFree(bsp);
    rdfp = readdb_destruct(rdfp);
    return 0;
}
```

Note that:

1. Readdb_new allocates an object for reading the database.
2. Readdb_acc2fasta fetches the ordinal number (zero offset) of the record given a FASTA identifier (e.g., gb|AAH06776.1|AAH0676).

3. Readdb_get_bioseq fetches the BioseqPtr (which contains the sequence, description, and identifiers) for this record.
4. BioseqRawToFasta dumps the sequence as FASTA.

Note also that Main is called, rather than "main", and a call to GetArgs is used to get the command-line arguments. db2fasta.c is contained in the tar archive ftp://ftp.ncbi.nih.gov/blast/demo/blast_demo.tar.gz.

Appendix 3. Excerpt from a demonstration program doblast.c

```
/* Get default options. */
options = BLASTOptionNew(blast_program, TRUE);
if (options == NULL)
    return 5;

options->expect_value = (Nlm_FloatHi) myargs [3].floatvalue;

/* Perform the actual search. */
seqalign = BioseqBlastEngine(query_bsp, blast_program, blast_database, options,

    NULL, NULL, NULL);

/* Do something with the SeqAlign... */
MySeqAlignPrint(seqalign, outfp);

/* clean up. */
seqalign = SeqAlignSetFree(seqalign);
options = BLASTOptionDelete(options);

sep = SeqEntryFree(sep);
FileClose(infp);
FileClose(outfp);
```

The main steps here are:

1. BLASTOptionNew allocates a BLASTOptionBlk with default values for the specified program (e.g., blastp); the Boolean argument specifies a gapped search.
2. The expect_value member of the BLASTOptionBlk is changed to a non-default value specified on the command-line.
3. BioseqBlastEngine performs the search of the BioseqPtr (query_bsp). The BioseqPtr could have been obtained from the BLAST databases, Entrez, or from FASTA using the function call FastaToSeqEntry.

The BLASTOptionBlk structure contains a large number of members. The most useful ones and a brief description for each are listed in Table 2.

Table 2. The most frequently used BLAST options in the BLASTOptionBlk structure.

Type[a]	Element	Description
Nlm_FloatHi	expect_value	Expect value cutoff
Int2	wordsize	Number of letters used in making words for lookup table
Int2	penalty	Mismatch penalty (only blastn and MegaBLAST)
Int2	reward	Match reward (only blastn and MegaBLAST)
CharPtr	matrix	Matrix used for comparison (not blastn or MegaBLAST)
Int4	gap_open	Cost for gap existence

Table 2 continued from previous page.

Type[a]	Element	Description
Int4	gap_extend	Cost to extend a gap one more letter (including first)
CharPtr	filter_string	Filtering options (e.g., L, mL)
Int4	hitlist_size	Number of database sequences to save hits for
Int2	number_of_cpus	Number of CPUs to use

a The types are given in terms of those in the NCBI Toolkit. Nlm_FloatHi is a double, Int2/Int4 are 2- or 4-byte integers, and CharPtr is just char*.

Appendix 4. A function to print a view of a SeqAlign: MySeqAlignPrint

```
#define BUFFER_LEN 50

/*
    Print a report on hits with start/stop. Zero-offset is used.
*/
static void MySeqAlignPrint(SeqAlignPtr seqalign, FILE *outfp)
{
    Char query_id_buf[BUFFER_LEN+1], target_id_buf[BUFFER_LEN+1];
    SeqIdPtr query_id, target_id;
    while (seqalign)
    {
        query_id = SeqAlignId(seqalign, 0);
        SeqIdWrite(query_id, query_id_buf, PRINTID_FASTA_LONG, BUFFER_LEN);

        target_id = SeqAlignId(seqalign, 1);
        SeqIdWrite(target_id, target_id_buf, PRINTID_FASTA_LONG, BUFFER_LEN);

        fprintf(outfp, "%s:%ld-%ld\t%s:%ld-%ld\n",
            query_id_buf, (long) SeqAlignStart(seqalign, 0), (long)
SeqAlignStop(seqalign, 0),
            target_id_buf, (long) SeqAlignStart(seqalign, 1), (long)
SeqAlignStop(seqalign, 1));
        seqalign = seqalign->next;
    }
return;
}
```

Note that:

1. SeqAlignId gets the sequence identifier for the zero-th identifier (zero offset). This is actually a C structure.
2. SeqIdWrite formats the information in query_id into a FASTA identifier (e.g., gi|129295) and places it into query_buf.
3. SeqAlignStart and SeqAlignStop return the start values of the zero-th and first sequences (or first and second).

All of this is done by high-level function calls, and it is not necessary to write low-level function calls to parse the ASN.1.

References

1. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic Local Alignment Search Tool. J Mol Biol. 1990;215:403–410. PubMed PMID: 2231712.
2. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 1997;25:3389–3402. PubMed PMID: 9254694.