Politecnico di Bari

Politecnico di Bari

Dipartimento di Ingegneria Elettrica e dell'Informazione

Corso di Laurea Magistrale in Ingegneria dei Sistemi Medicali

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE

Bioinformatica Avanzata

BLAST and Levenshtein Distance with MATLAB Bioinformatics Toolbox and C

Dr. Nicola **Altini**, Ph.D. Student

Prof. Eng. Vitoantonio **Bevilacqua**, Ph.D.

INDUSTRIAL INFORMATICS LAB

LABORATORIO DI INFORMATICA INDUSTRIALE

Anno Accademico 2019/2020

apulian bioengineering company

FASTA

- FASTA is a DNA and protein sequence alignment software package first described by David J. Lipman and William R. Pearson in 1985. Its legacy is the FASTA format which is now ubiquitous in bioinformatics.
- The **FASTA format** is a text-based format for representing either nucleotide sequences or amino acid (protein) sequences, in which nucleotides or amino acids are represented using single-letter codes. The format also allows for sequence names and comments to precede the sequences.

FASTA Format Example

- A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line (`defline`) is distinguished from the sequence data by a greater-than (">") symbol at the beginning. It is recommended that all lines of text be shorter than 80 characters in length.
- An example sequence in FASTA format is:

```
>P01013 GENE X PROTEIN (OVALBUMIN-RELATED)
QIKDLLVSSSTDLDTTLVLVNAIYFKGMWKTAFNAEDTREMPFHVTKQESKPVQMMCMNNSFNVATLPAE
KMKILELPFASGDLSMLVLLPDEVSDLERIEKTINFEKLTEWTNPNTMEKRRVKVYLPQMKIEEKYNLTS
VLMALGMTDLFIPSANLTGISSAESLKISQAVHGAFMELSEDGIEMAGSTGVIEDIKHSPESEQFRADHP
FLFLIKHNPTNTIVYFGRYWSP
```

FASTA Format Example

- Blank lines are not allowed in the middle of FASTA input.
- Sequences are expected to be represented in the standard IUB/IUPAC amino acid and nucleic acid codes, with these exceptions:
  - lower-case letters are accepted and are mapped into upper-case;
  - a single hyphen or dash can be used to represent a gap of indeterminate length;
  - in amino acid sequences, U and * are acceptable letters (see below).
- Before submitting a request, any numerical digits in the query sequence should either be removed or replaced by appropriate letter codes (e.g., N for unknown nucleic acid residue or X for unknown amino acid residue).

FASTA Format Example

- The **nucleic acid codes** supported are:

A	adenosine	C	cytidine	G	guanine
T	thymidine	N	A/G/C/T (any)	U	uridine
K	G/T (keto)	S	G/C (strong)	Y	T/C (pyrimidine)
M	A/C (amino)	W	A/T (weak)	R	G/A (purine)
B	G/T/C	D	G/A/T	H	A/C/T
V	G/C/A	-	gap of indeterminate length		

FASTA Format Example

- For those programs that use amino acid query sequences (BLASTP and TBLASTN), the accepted **amino acid codes** are:

A	alanine	P	proline
B	aspartate/asparagine	Q	glutamine
C	cystine	R	arginine
D	aspartate	S	serine
E	glutamate	T	threonine
F	phenylalanine	U	selenocysteine
G	glycine	V	valine
H	histidine	W	tryptophan
I	isoleucine	Y	tyrosine
K	lysine	Z	glutamate/glutamine
L	leucine	X	any
M	methionine	*	translation stop
N	asparagine	-	gap of indeterminate length

BLAST

- In bioinformatics, **BLAST** (**B**asic **L**ocal **A**lignment **S**earch **T**ool) is an algorithm for comparing primary biological sequence information, such as the amino-acid sequences of proteins or the nucleotides of DNA and/or RNA sequences.
- A BLAST search enables a researcher to compare a subject protein or nucleotide sequence (called a query) with a library or database of sequences, and identify library sequences that resemble the query sequence above a certain threshold.
- Different types of BLASTs are available according to the query sequences.
- For example, following the discovery of a previously unknown gene in the mouse, a scientist will typically perform a BLAST search of the human genome to see if humans carry a similar gene; BLAST will identify sequences in the human genome that resemble the mouse gene based on similarity of sequence.

Sequence Alignment

- Multiple, pairwise, and profile sequence alignments using dynamic programming algorithms;
- **BLAST searches and alignments**
  - `blastlocal` – Perform search on local BLAST database to create BLAST report
  - `blastncbi` – Create remote NCBI BLAST report request ID or link to NCBI BLAST report;
- standard and custom scoring matrices

blastformat

`blastformat`

Create local BLAST database

Syntax

```
blastformat('Inputdb', InputdbValue)
blastformat(..., 'FormatPath', FormatPathValue, ...)
blastformat(..., 'Title', TitleValue, ...)
blastformat(..., 'Log', LogValue, ...)
blastformat(..., 'Protein', ProteinValue, ...)
blastformat(..., 'FormatArgs', FormatArgsValue, ...)
```

blastformat

`blastformat`

Arguments

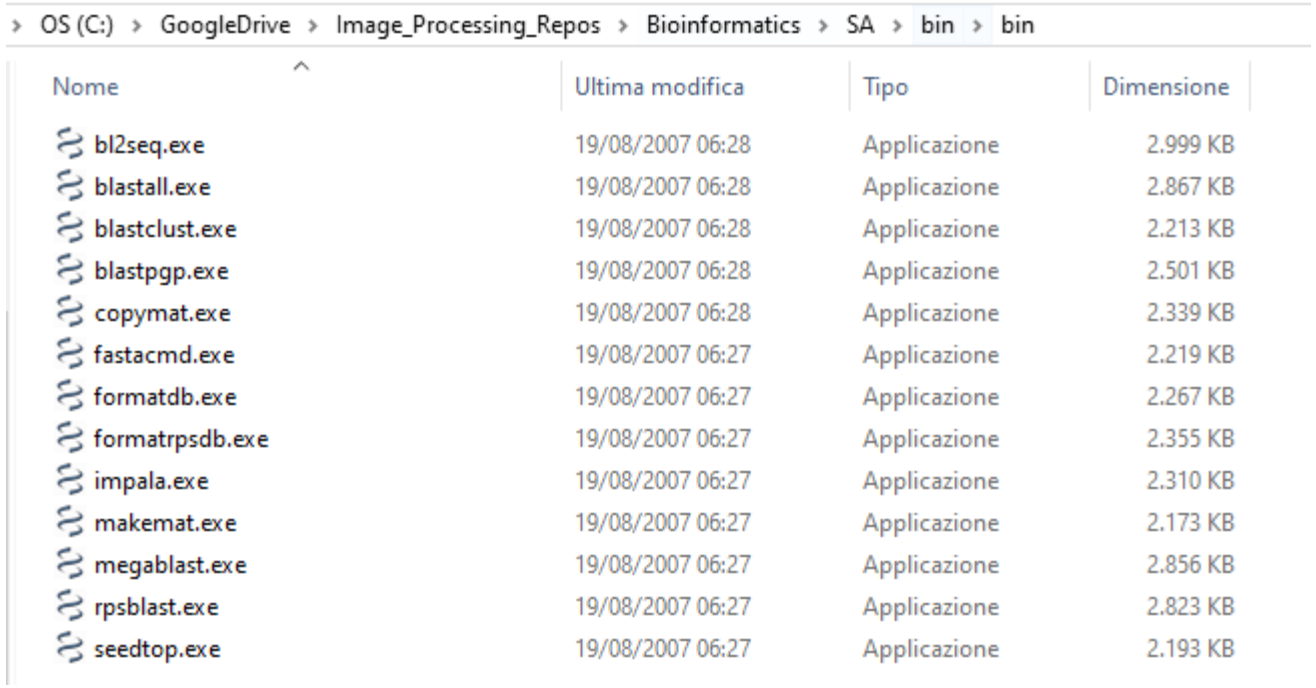
- `InputdbValue` Character vector or string specifying a file name or path and file name of a FASTA file containing a set of sequences to be formatted as a blastable database. If you specify only a file name, that file must be on the MATLAB® search path or in the current folder. (This corresponds to the formatdb option -i.)
- `FormatPathValue` Character vector or string specifying the full path to the formatdb executable file, including the name and extension of the executable file. Default is the system path.
- `TitleValue` Character vector or string specifying the title for the local database. Default is the input FASTA file name. (This corresponds to the formatdb option -t.)
- `LogValue` Character vector or string specifying the file name or path and file name for the log file associated with the local database. Default is formatdb.log. (This corresponds to the formatdb option -l.)
- `ProteinValue` Specifies whether the sequences formatted as a local BLAST database are protein or not. Choices are true (default) or false. (This corresponds to the formatdb option -p.)
- `FormatArgsValue` NCBI formatdb command, that is, a character vector or string containing one or more instances of -x and the option associated with it, used to specify input arguments.

blastformat

- To use the blastformat function, you **must have a local copy of the NCBI `formatdb` executable** file available from your system. You can download the `formatdb` executable file by accessing BLAST executables.
- Run the downloaded executable and configure it for your system. For convenience, consider placing the NCBI `formatdb` executable file on your **system path**.
- URL: ftp://ftp.ncbi.nlm.nih.gov/blast/executables/legacy.NOTSUPPORTED/2.2.17/

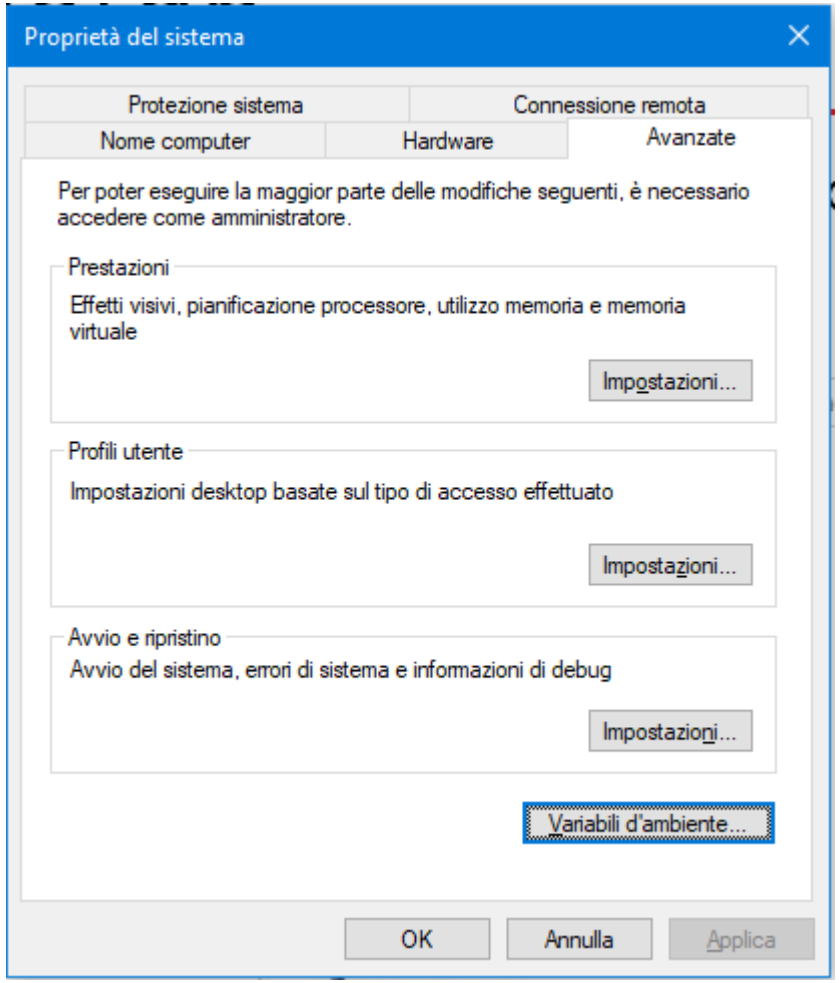
formatdb

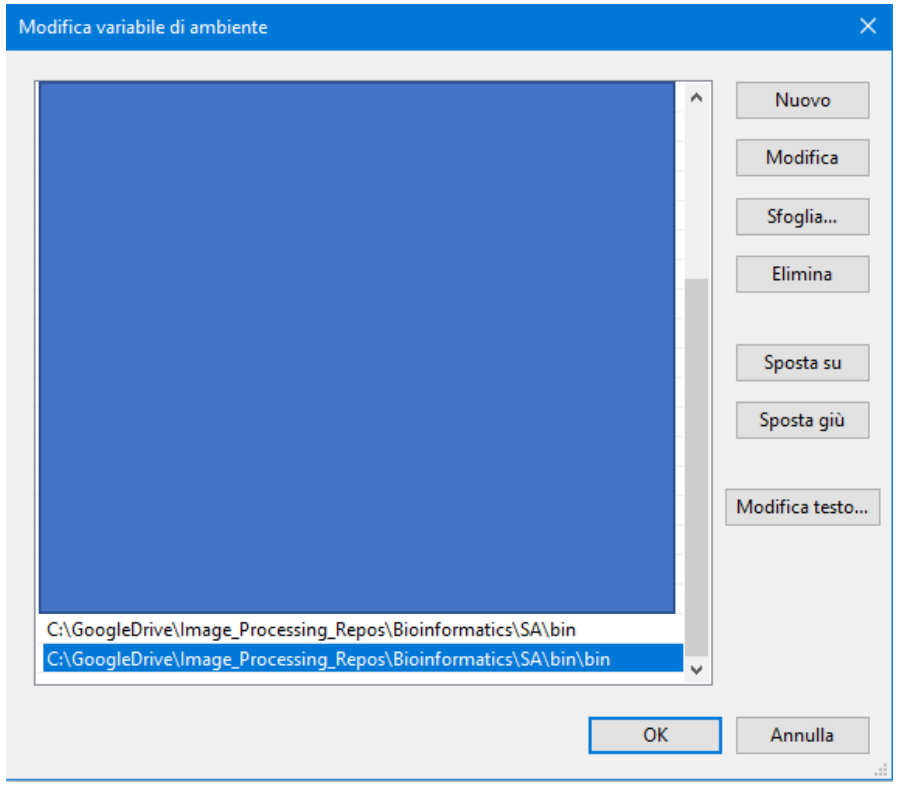
- Remember to add `formatdb` to your system path!



formatdb

- Remember to add `formatdb` to your system path!





blastlocal

blastlocal

Perform search on local BLAST database to create BLAST report

Syntax

```
blastlocal('InputQuery', InputQueryValue)
Data = blastlocal('InputQuery', InputQueryValue)
... blastlocal(..., 'Program', ProgramValue, ...)
... blastlocal(..., 'Database', DatabaseValue, ...)
... blastlocal(..., 'BlastPath', BlastPathValue, ...)
... blastlocal(..., 'Expect', ExpectValue, ...)
... blastlocal(..., 'Format', FormatValue, ...)
... blastlocal(..., 'ToFile', ToFileValue, ...)
... blastlocal(..., 'Filter', FilterValue, ...)
... blastlocal(..., 'GapOpen', GapOpenValue, ...)
... blastlocal(..., 'GapExtend', GapExtendValue, ...)
... blastlocal(..., 'BLASTArgs', BLASTArgsValue, ...)
```

blastlocal

Arguments

- `InputQueryValue`
  Character vector or string specifying the file name or path and file name of a FASTA file containing query nucleotide or aminoacid sequence(s). (This corresponds to the `blastall` option `-i`.)

- `ProgramValue`
  Character vector or string specifying a BLAST program. Choices are:
  - `'blastp'` (default) — Search protein query versus protein database.
  - `'blastn'` — Search nucleotide query versus nucleotide database.
  - `'blastx'` — Search translated nucleotide query versus protein database.
  - `'tblastn'` — Search protein query versus translated nucleotide database.
  - `'tblastx'` — Search translated nucleotide query versus translated nucleotide database.

  (The ProgramValue argument corresponds to the `blastall` option `-p`.)

- `DatabaseValue`
  Character vector or string specifying a file name or path and file name of a local BLAST database (formatted using the NCBI `formatdb` function) to search. Default is a local version of the nr database in the MATLAB® current folder. (This corresponds to the `blastall` option `-d`.)

blastlocal

Arguments

- `FilterValue`

  Controls the application of a filter (DUST filter for the blastn program or SEG filter for other programs) to the query sequence(s). Choices are true (default) or false. (This corresponds to the `blastall` option `-F`.)

- `ExpectValue`

  Value specifying the statistical significance threshold for matches against database sequences. Choices are any real number. Default is 10. (This corresponds to the `blastall` option `-e`.)

getgenbank

getgenbank

Retrieve sequence information from GenBank database

Syntax

```
Data = getgenbank(AccessionNumber)
getgenbank(AccessionNumber)
Data = getgenbank(..., 'PartialSeq', PartialSeqValue, ...)
Data = getgenbank(..., 'ToFile', ToFileValue, ...)
Data = getgenbank(..., 'FileFormat', FileFormatValue, ...)
Data = getgenbank(..., 'SequenceOnly', SequenceOnlyValue, ...)
```

BLAST Example

- Download the FASTA nucleotide file and a FASTA amino acid file for E. coli, and save them in a folder beloning to MATLAB path.

```
fullURL = 'http://www.scfbio-iitd.res.in/chemgenome/genomedataset/NC_004431.fna'
filename_fna = 'Bioinformatics/SA/NC_004431.fna'
urlwrite(fullURL,filename_fna)
fullURL = 'https://www.brinkman.mbb.sfu.ca/~mlangill/public_tmp/NC_004431.faa'
filename_faa = 'Bioinformatics/SA/NC_004431.faa'
urlwrite(fullURL,filename_faa)
```

BLAST Example

- NC_004431.fna

```
>gi|26245917|ref|NC_004431.1| Escherichia coli CFT073, complete genome
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC
TTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGGTCACTAAATACTTTAACCAA
TATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTACACAACATCCATGAAACGCATTAGCACCACC
ATTACCACCACCATCACCATTACCACAGGTAACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAG
CCCGCACCTGACAGTGCGGGCTTTTTTTTGTGCACAGAAAACCCCCAGCTAGGCTGGGGGTTCCGGAAAG
CTTTCAGCTTTGAGCCAGTTATTAAAACCCCTTTTGATTTGTTAAAACACCTTGCGGTCTGGCAACTGCA
AGTGTCAAACAAGAAATCAAAAGGGGGTCCCAATGGGGAACGAAAAGAGCTTAGCGCACACCCGATGGAA
CTGTAAATATCACATAGTATTTGCGCCAAAATACCGAAGACAGGTGTTCTACAGAGAGAAGCGTAGAGCA
ATAGGCTGTATTTTGAGAAAGCTGTGTGAGTGGAAAAGTGTACGGATTCTGGAAGCTGAATGCTGTGCAG
ATCATATCCATATGCTTGTGGAGATCCCGCCCAAAATGAGCGTATCAGGCTTTATGGGATATCTGAAAGG
GAAAAGCAGTCTGATGCCTTACGAGCAGTTTGGTGATTTGAAATTCAAATACAGGAACAGGGAGTTCTGG
TGCAGAGGGTATTACGTCGATACGGTGGGTAAGAACACGGCGAAGATACAGGATTACATAAAGCACCAGC
TTGAAGAGGATAAAATGGGAGAGCAGTTATCGATTCCCTATCCGGGCAGCCCGTTTACGGGCCGTAAGTA
ACGAAGTTGGATGCAAATGTCAGATCGTGTGCGCCTGTTAGGGCGCGGCTGGTAAGAGAGCCTTATAGGC
GCATTTGAAAAACCTCCGGCTATGCCGGAGGATATTTATTTCGACCAAAGGTAACGAGGTAACAACCATG
CGAGTGTTGAAGTTCGGCGGTACATCAGTGGCAAATGCAGAACGTTTTCTGCGGGTTGCCGATATTCTGG
AAAGCAATGCCAGGCAGGGGCAGGTGGCCACCGTCCTCTCTGCCCCCGCCAAAATCACCAACCATCTGGT
AGCGATGATTGAAAAAACCATTAGCGGCCAGGATGCTTTACCCAATATCAGCGATGCCGAACGTATTTTT
GCCGAACTTCTGACGGGACTCGCCGCCGCCCAGCCGGGATTTCCGCTGGCACAATTGAAAACTTTCGTCG
ACCAGGAATTTGCCCAAATAAAACATGTCCTGCATGGCATCAGTTTGTTGGGGCAGTGCCCGGATAGCAT
CAACGCTGCGCTGATTTGCCGTGGCGAGAAAATGTCGATCGCCATTATGGCCGGCGTGTTAGAAGCGCGT
```

BLAST Example

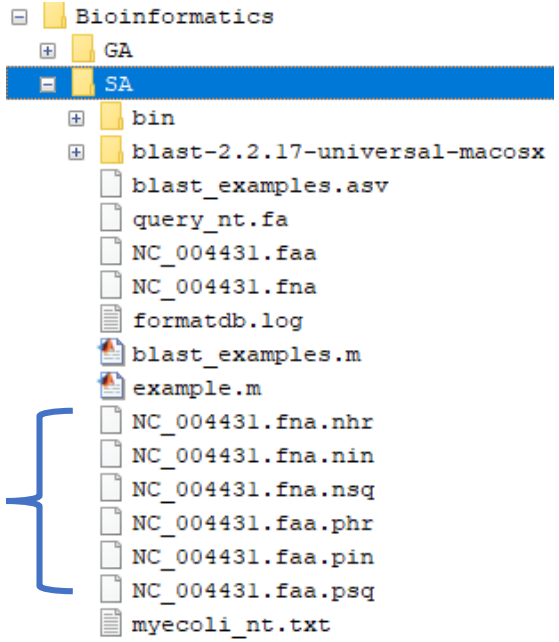
- NC_004431.faa

```
>gi|26245918|ref|NP_751957.1| thr operon leader peptide [Escherichia coli CFT073]
MKRISTTITTTITITTGNGAG
>gi|26245919|ref|NP_751958.1| hypothetical protein c0002 [Escherichia coli CFT073]
MFYREKRRAIGCILRKLCEWKSVRILEAECCADHIHMLVEIPPKMSVSGFMGYLKGKSSLMPYEQFGDLK
FKYRNREFWCRGYYVDTVGKNTAKIQDYIKHQLEEDKMGEQLSIPYPGSPFTGRK
>gi|26245920|ref|NP_751959.1| bifunctional aspartokinase I/homeserine dehydrogenase I [Escherichia coli CFT073]
MKNLRLCRRIFISTKGNEVTTMRVLKFGGTSVANAERFLRVADILESNARQGQVATVLSAPAKITNHLVA
MIEKTISGQDALPNISDAERIFAELLTGLAAAQPGFPLAQLKTFVDQEFAQIKHVLHGISLLGQCPDSIN
AALICRGEKMSIAIMAGVLEARGHNVTVIDPVEKLLAVGHYLESTVDIAESTRRIAASRIPADHMVLMAG
FTAGNEKGELVVLGRNGSDYSAAVLAACLRADCCEIWTDVDGVYTCDPRQVPDARLLKSMSYQEAMELSY
FGAKVLHPRTITPIAQFQIPCLIKNTGNPQAPGTLIGASRDEDELPVKGISNLNNMAMFSVSGPGMKGMV
GMAARVFAAMSRARISVVLITQSSSEYSISFCVPQSDCVRAERAMQEEFYLELKEGLLEPLAVTERLAII
SVVGDGMRTLRGISAKFFAALARANINIVAIAQGSSERSISVVVNNDDATTGVRVTHQMLFNTDQVIEVF
VIGVGGVGGALLEQLKRQQSWLKNKHIDLRVCGVANSKALLTNVHGLNLENWQEELAQAKEPFNLGRLIR
LVKEYHLLNPVIVDCTSSQAVADQYADFLREGFHVVTPNKKANTSSMDYYHQLRYAAEKSRRKFLYDTNV
GAGLPVIENLQNLLNAGDELMKFSGILSGSLSYIFGKLDEGMSFSEATTLAREMGYTEPDPRDDLSGMDV
ARKLLILARETGRELELADIEIEPVLPAEFNAEGDVAAFMANLSQLDNLFAARVAKARDEGKVLRYVGNI
DEDGVCRVKIAEVDSNDPLFKVKNGENALAFYSHYYQPLPLVLRGYGAGNDVTAAGVFADLLRTLSWKLG
V
```

BLAST Example

- Create local blastable databases from the `NC_004431.fna` and `NC_004431.faa` FASTA files by using the `blastformat` function.

```
blastformat('inputdb', filename_fna, 'protein', 'false');
blastformat('inputdb', filename_faa);
```



BLAST Example

- Use the `getgenbank` function to retrieve sequence information for the E. coli threonine operon from the GenBank® database.

```
S = getgenbank('M28570');
```

S

1x1 struct with 24 fields

Field	Value
LocusName	'ECOTHROPA'
LocusSequenceLength	'351'
LocusNumberofStrands	''
LocusTopology	'linear'
LocusMoleculeType	'DNA'
LocusGenBankDivision	'BCT'
LocusModificationDate	'11-JUN-1993'
Definition	'Escherichia coli threonine operon (thrPLA) genes, complete cds.'
Accession	'M28570'
Version	'M28570.1'
GI	''
Project	[]
DBLink	[]
Keywords	'threonine operon.'
Segment	[]
Source	'Escherichia coli'
SourceOrganism	3x64 char
Reference	1x1 cell
Comment	[]
Features	26x57 char
CDS	1x2 struct
Sequence	'ttgcaacgggcaatatgtctctgtgtggattaaaaaaagagtgtctgatagcagcttctgaactggttacctgccgtgagtaaattaaaattttattgacttaggtcactaaat...
SearchURL	'https://www.ncbi.nlm.nih.gov/entrez/viewer.fcgi?db=nuccore&id=M28570'
RetrieveURL	'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nuccore&id=290476&rettype=gb&retmode=text'

BLAST Example

- Create a query file by using the fastawrite function to create a FASTA file named `query_nt.fa` from this sequence information, using only the accession number as the header.

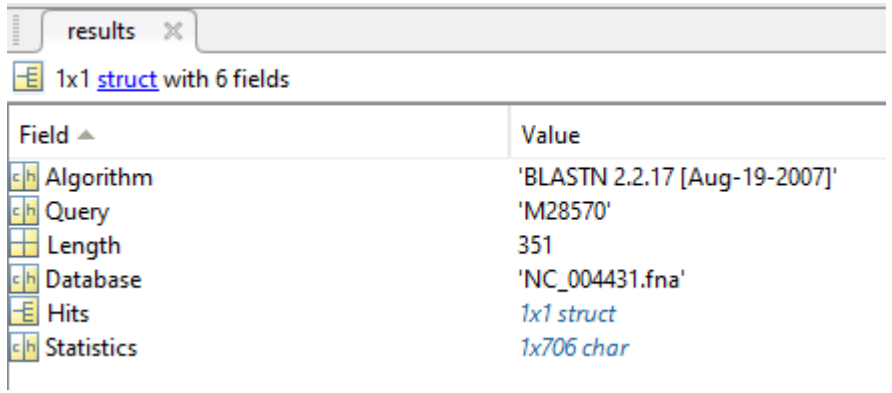
```
S.Header = S.Accession;
path_query = 'Bioinformatics/SA/query_nt.fa';
fastawrite(path_query, S);
```

```
>M28570
ttgcaacgggcaatatgtctctgtgtggattaaaaaaagagtgtctgatagcagcttctgaactggttac
ctgccgtgagtaaattaaaattttattgacttaggtcactaaatactttaaccaatataggcatagcgca
cagacagataaaaattacagagtacacaacatccatgaaacgcattagcaccaccattaccaccaccatc
accattaccacaggtaacggtgcgggctgacgcgtacaggaaacacagaaaaaagcccgcacctgacagt
gcgggctttttttttcgaccaaaggtaacgaggtaacaaccatgcgagtgttgaagttcggcggtacatc
a
```

BLAST Example 1

- Example 1: **Search nucleotide query versus nucleotide database**
- Use MATLAB syntax to submit the query sequence in the `query_nt.fa` FASTA file for a BLAST search of the local nucleotide database `NC_004431.fna`. Specify the BLAST program `blastn`. Return the BLAST search results in `results`, a MATLAB structure.

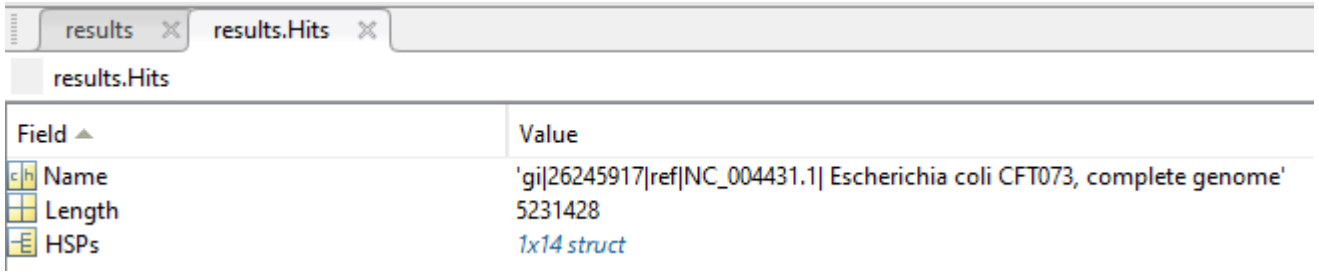
```
results = blastlocal('inputquery', path_query,...
                     'database', filename_fna,...
                     'program',  'blastn');
```



- `Algorithm` NCBI algorithm used to do a BLAST search.
- `Query` Identifier of the query sequence submitted to a BLAST search.
- `Length` Length of the query sequence.
- `Database` All databases searched.

BLAST Example 1

- Example 1: **Search nucleotide query versus nucleotide data**



`Hits.Name`	Name of a database sequence (subject sequence) that matched the query sequence.
`Hits.Score`	Alignment score between the query sequence and the subject sequence.
`Hits.Expect`	Expectation value for the alignment between the query sequence and the subject sequence.
`Hits.Length`	Length of a subject sequence.

BLAST Example 1

- Example 1: **Search nucleotide query versus nucleotide data**

results | results.Hits | results.Hits.HSPs

results.Hits.HSPs

Fields	Score	Expect	Identities	Strand	Alignment	QueryIndices	SubjectIndices
1	329	1.0000e-90	*1x1 struct*	'Plus/Plus'	*tgcaacgggcaatatgtctctgtgtgg...*	[2,188]	[17,203]
2	129	2.0000e-30	*1x1 struct*	'Plus/Plus'	*aggtaacggtgcgggctgacgcgtac...*	[222,286]	[237,301]
3	111	5.0000e-25	*1x1 struct*	'Plus/Plus'	*cgaccaaaggtaacgaggtaacaac...*	[296,351]	[1022,1077]
4	30.2000	1.4000	*1x1 struct*	'Plus/Minus'	*cacagaaaaaagcccgcac|||| ||||||||...*	[254,272]	[312,294]
5	28.2000	5.5000	*1x1 struct*	'Plus/Minus'	*gaaaaaagcccgca||||||||||||gaaa...*	[258,271]	[700336,700323]
6	28.2000	5.5000	*1x1 struct*	'Plus/Plus'	*atccatgaaacgca||||||||||||atccat...*	[171,184]	[1628663,1628676]
7	28.2000	5.5000	*1x1 struct*	'Plus/Plus'	*ccatgaaacgcatt||||||||||||ccatga...*	[173,186]	[1985409,1985422]
8	28.2000	5.5000	*1x1 struct*	'Plus/Plus'	*agtgttgaagttcg||||||||||||agtgtt...*	[327,340]	[2374649,2374662]
9	28.2000	5.5000	*1x1 struct*	'Plus/Plus'	*aacgggcaatatgt||||||||||||aacgg...*	[5,18]	[3309835,3309848]
10	28.2000	5.5000	*1x1 struct*	'Plus/Minus'	*tctgaactggttac||||||||||||tctgaac...*	[57,70]	[3868117,3868104]
11	28.2000	5.5000	*1x1 struct*	'Plus/Minus'	*cagcttctgaactg||||||||||||cagcttct...*	[52,65]	[4081677,4081664]
12	28.2000	5.5000	*1x1 struct*	'Plus/Plus'	*taaaaattacagag||||||||||||taaaa...*	[149,162]	[4249562,4249575]
13	28.2000	5.5000	*1x1 struct*	'Plus/Minus'	*acagaaaaaagccc||||||||||||acag...*	[255,268]	[4587817,4587804]
14	28.2000	5.5000	*1x1 struct*	'Plus/Minus'	*accaaaggtaacga||||||||||||acca...*	[298,311]	[5101728,5101715]

`Hits.HSPs.Score`	Pairwise alignment score for a high-scoring sequence pair between the query sequence and a subject sequence.
`Hits.HSPs.Expect`	Expectation value for a high-scoring sequence pair between the query sequence and a subject sequence.
`Hits.HSPs.Identities`	Identities (match, possible, and percent) for a high-scoring sequence pair between the query sequence and a subject sequence.
`Hits.HSPs.Strand`	Sense (Plus = 5' to 3' and Minus = 3' to 5') of the DNA strands for a high-scoring sequence pair between the query sequence and a subject sequence.
`Hits.HSPs.Alignment`	Three-row matrix showing the alignment for a high-scoring sequence pair between the query sequence and a subject sequence.
`Hits.HSPs.QueryIndices`	Indices of the query sequence residue positions for a high-scoring sequence pair between the query sequence and a subject sequence.
`Hits.HSPs.SubjectIndices`	Indices of the subject sequence residue positions for a high-scoring sequence pair between the query sequence and a subject sequence.

BLAST Example 1

- Example 1: **Search nucleotide query versus nucleotide data**

```
>> align = results.Hits.HSPs(1).Alignment;
>> align(:,1:20)
ans =
  3×20 char array
    'tgcaacgggcaatatgtctc'
    '||||||||||||||||||||'
    'tgcaacgggcaatatgtctc'
>> align(:,20:40)
ans =
  3×21 char array
    'ctgtgtggattnnnnnnngag'
    '||||||||||        |||'
    'ctgtgtggattaaaaaaagag'
```

BLAST Example 1

- Example 1: **Search nucleotide query versus nucleotide data**
- A key element in evaluating the quality of a pairwise sequence alignment is the "substitution matrix", which assigns a score for aligning any possible pair of residues.
- A possible and simple way to implement a custom substitution matrix is the following:

1. Define a substitution matrix
2. Define a function which maps residues to matrix indices
3. Calculate the cost/reward associated to the alignment

BLAST Example 1

- Example 1: **Search nucleotide query versus nucleotide data**

1. Define a substitution matrix

	A	T	C	G
A	+1	-5	-5	-1
T	-5	+1	-1	-5
C	-5	-1	+1	-5
G	-1	-5	-5	+1

```
pen_matrix = ...
      [+1  -5  -5  -1;
       -5  +1  -1  -5;
       -5  -1  +1  -5;
       -1  -5  -5  +1];
```

BLAST Example 1

- Example 1: **Search nucleotide query versus nucleotide data**

2. Define a function which maps nucleotides to matrix indices

```
function idx = idx_from_atcg(char1)
    if char1 == 'a'
        idx = 1;
    elseif char1 == 't'
        idx = 2;
    elseif char1 == 'c'
        idx = 3;
    elseif char1 == 'g'
        idx = 4;
    else
        idx = -1;
    end
end
```

BLAST Example 1

- Example 1: **Search nucleotide query versus nucleotide data**

3. Calculate the cost/reward associated to the alignment

```
function pen = score_from_matrix(char1, char2, pen_matrix)
    idx1 = idx_from_atcg(char1);
    idx2 = idx_from_atcg(char2);
    if idx1 > 0 && idx2 > 0
        pen = pen_matrix(idx1, idx2);
    else
        pen = 0;
    end
end
```

BLAST Example 1

- Example 1: **Search nucleotide query versus nucleotide data**

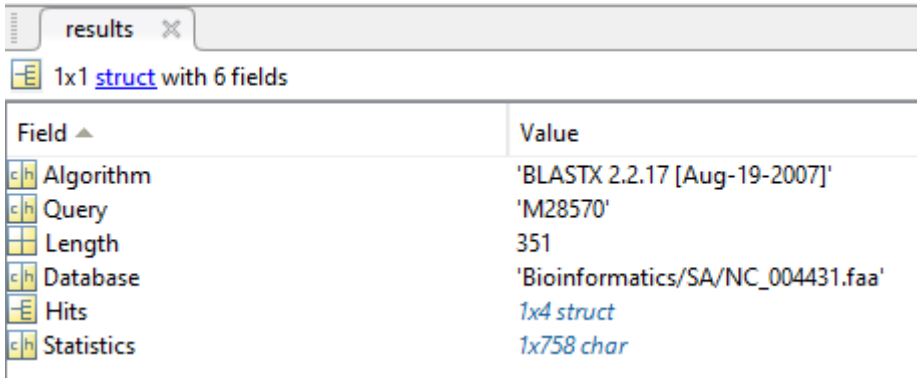
```
>> align = results.Hits.HSPs(4).Alignment;
>> align
align =
  3×19 char array
    'cacagaaaaaagcccgcac'
    '|||| ||||||||||||||'
    'cacaaaaaaaagcccgcac'

char1 = g char2 = a
Penalty is -1
Final score is 35
```

BLAST Example 2

- Example 2: **Performing a Nucleotide Translated Search**
- Use MATLAB syntax to submit the query sequence in the `query_nt.fa` FASTA file for a BLAST search of the local amino acid database `NC_004431.faa`. Specify the BLAST program `blastx`. Return the BLAST search results in `results`, a MATLAB structure.

```
results = blastlocal('inputquery', path_query,...
                     'database', filename_faa,...
                     'program',  'blastx');
```



BLAST Example 2

results | results.Hits | results.Hits(1).HSPs | results.Hits(1).HSPs.Alignment | results.Hits(2).HSPs | results.Hits(2).HSPs.Alignment

results.Hits

Fields	Name	Length	HSPs
1	'gi|26245920|ref|NP_751959.1| bifunctional aspartokinase I/homeserine dehydrogenase I [Escherichia coli CFT073]'	841	*1x1 struct*
2	'gi|26248808|ref|NP_754848.1| hypothetical protein c2966 [Escherichia coli CFT073]'	191	*1x1 struct*
3	'gi|26247297|ref|NP_753337.1| hypothetical protein c1428 [Escherichia coli CFT073]'	43	*1x1 struct*
4	'gi|26250140|ref|NP_756180.1| hemin importer ATP-binding subunit [Escherichia coli CFT073]'	266	*1x1 struct*

results | results.Hits | results.Hits(1).HSPs | results.Hits(1).HSPs.Alignment

results.Hits(1).HSPs.Alignment

```
val =
    'LFFSTKGNEVTTMRVLKFGGTS'
    '+F STKGNEVTTMRVLKFGGTS'
    'IFISTKGNEVTTMRVLKFGGTS'
```

results | results.Hits | results.Hits(1).HSPs | results.Hits(1).HSPs.Alignment | results.Hits(2).HSPs | results.Hits(2).HSPs.Alignment

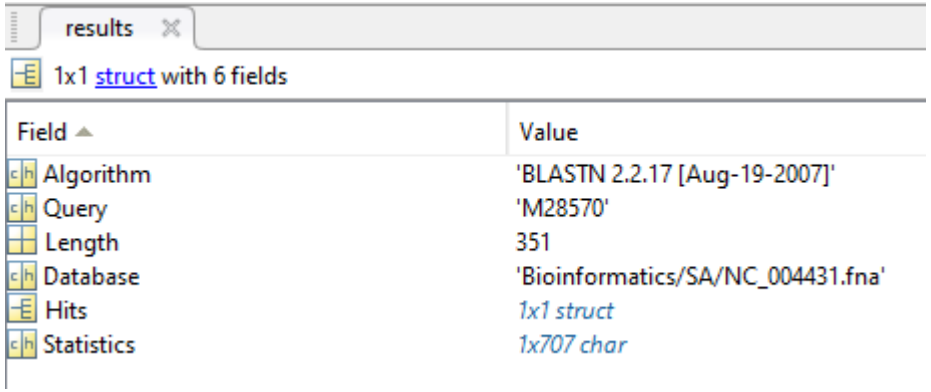
results.Hits(2).HSPs.Alignment

```
val =
    'ICLCAMPILVKVFSDLSQ*NFNLLTAGNQFRS'
    '+ LCAMP+++   S +S  N++     N F S'
    'LMLCAMPLMLTGCSTMSSVNWSAANPWNWFGS'
```

BLAST Example 3

- Example 3: **Performing a Nucleotide Search Using `blastall` Syntax**
- Use `blastall` syntax to submit the query sequence in the `query_nt.fa` FASTA file for a BLAST search of the local nucleotide database `NC_004431.fna`. Specify the BLAST program `blastn` and an expectation value of 0.0001. Return the BLAST search results in `results`, a MATLAB structure.

```
query = sprintf('-i %s -d %s -p blastn -e 0.0001', path_query, filename_fna);
results = blastlocal(query);
```



BLAST Example 4

- Example 4: **Performing a Nucleotide Search and Creating a Formatted Report**
- Submit the query sequence in the `query_nt.fa` FASTA file for a BLAST search of the local nucleotide database `NC_004431.fna`. Specify the BLAST program `blastn` and a tabular alignment format. Save the contents of the BLAST report to a file named `myecoli_nt.txt`.

```
blastlocal('inputquery', path_query,...
           'database', filename_fna, ...
           'tofile', 'Bioinformatics/SA/myecoli_nt.txt', ...
           'blastargs', '-p blastn -m 8');
```

BLAST Example 4

myecoli_nt.txt

M28570	gi|26245917|ref|NC_004431.1|	100.00	187	0	0	2	188	17	203	1e-090	329
M28570	gi|26245917|ref|NC_004431.1|	100.00	65	0	0	222	286	237	301	2e-030	129
M28570	gi|26245917|ref|NC_004431.1|	100.00	56	0	0	296	351	1022	1077	5e-025	111
M28570	gi|26245917|ref|NC_004431.1|	94.74	19	1	0	254	272	312	294	1.4	30.2
M28570	gi|26245917|ref|NC_004431.1|	100.00	14	0	0	258	271	700336	700323	5.5	28.2
M28570	gi|26245917|ref|NC_004431.1|	100.00	14	0	0	171	184	1628663	1628676	5.5	28.2
M28570	gi|26245917|ref|NC_004431.1|	100.00	14	0	0	173	186	1985409	1985422	5.5	28.2
M28570	gi|26245917|ref|NC_004431.1|	100.00	14	0	0	327	340	2374649	2374662	5.5	28.2
M28570	gi|26245917|ref|NC_004431.1|	100.00	14	0	0	5	18	3309835	3309848	5.5	28.2
M28570	gi|26245917|ref|NC_004431.1|	100.00	14	0	0	57	70	3868117	3868104	5.5	28.2
M28570	gi|26245917|ref|NC_004431.1|	100.00	14	0	0	52	65	4081677	4081664	5.5	28.2
M28570	gi|26245917|ref|NC_004431.1|	100.00	14	0	0	149	162	4249562	4249575	5.5	28.2
M28570	gi|26245917|ref|NC_004431.1|	100.00	14	0	0	255	268	4587817	4587804	5.5	28.2
M28570	gi|26245917|ref|NC_004431.1|	100.00	14	0	0	298	311	5101728	5101715	5.5	28.2

Levenshtein distance

- In information theory, linguistics and computer science, the **Levenshtein distance** is a string metric for measuring the difference between two sequences.
- Informally, the Levenshtein distance between two words is the minimum number of single-character edits (*insertions*, *deletions* or *substitutions*) required to change one word into the other.
- It is named after the Soviet mathematician Vladimir Levenshtein, who considered this distance in 1965.

Levenshtein distance

- **Iterative with full matrix**
- Computing the Levenshtein distance is based on the observation that if we reserve a matrix to hold the Levenshtein distances between all prefixes of the first string and all prefixes of the second, then we can compute the values in the matrix in a dynamic programming fashion, and thus find the distance between the two full strings as the last value computed.

Levenshtein distance

```c
int Levenshtein_distance(char *x, char *y) {
        int m = strlen(x);
        int n = strlen(y);
        printf("m = %d , n = %d\n", m, n);
        int i, j;
        int distance;
        int **d = malloc((m + 1) * sizeof(int*));
        for(i = 0; i <= m; i++)
            d[i] = malloc((n + 1) * sizeof(int));
        for(i = 0; i <= m; i++)
            d[i][0] = i;
        for(j = 1; j <= n; j++)
            d[0][j] = j;
        print_matrix(d, m, n);
        for(i = 1; i <= m; i++) {
            for(j = 1; j <= n; j++) {
                if(x[i - 1] != y[j - 1]) {
                    int k = minimum(
                            d[i][j - 1],
                            d[i - 1][j],
                            d[i - 1][j - 1]
                     );
                    d[i][j] = k + 1;
                } else {
                    d[i][j] = d[i - 1][j - 1];
                }
                 print_matrix(d, m, n);
             }
        }
        distance = d[m][n];
        for(i = 0; i <= m; i++)
            free(d[i]);
        free(d);
        return distance;
}
```

Levenshtein distance

- **Example**
- The Levenshtein distance between `"attccag"` and `"atgcaa"` is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

- `atgcaa` → `attcaa` (substitution of `"g"` for `"t"`)
- `attcaa` → `attcca` (substitution of `"a"` for `"c"`)
- `attcca` → `attccag` (insertion of `"g"` at the end).

Levenshtein distance

- str1: **attccag**
- str2: **atgcaa**

		a	**t**	**g**	**c**	**a**	**a**
	0	1	2	3	4	5	6
a	1	0	1	2	3	4	5
t	2	1	0	1	2	3	4
t	3	2	1	1	2	3	4
c	4	3	2	2	1	2	3
c	5	4	3	3	2	2	3
a	6	5	4	4	3	2	2
g	7	6	5	4	4	3	**3**

The Levenshtein Distance between **attccag** and **ctgcaa** is **3**

Levenshtein distance

- str1: **att** str2: **cta**

		c	**t**	**a**
	0	1	2	3
a	1	0	0	0
t	2	0	0	0
t	3	0	0	0

		c	**t**	**a**
	0	1	2	3
a	1	1	2	2
t	2	0	0	0
t	3	0	0	0

		c	**t**	**a**
	0	1	2	3
a	1	1	2	2
t	2	2	1	2
t	3	0	0	0

		c	**t**	**a**
	0	1	2	3
a	1	1	0	0
t	2	0	0	0
t	3	0	0	0

		c	**t**	**a**
	0	1	2	3
a	1	1	2	2
t	2	2	0	0
t	3	0	0	0

		c	**t**	**a**
	0	1	2	3
a	1	1	2	2
t	2	2	1	2
t	3	3	0	0

		c	**t**	**a**
	0	1	2	3
a	1	1	2	0
t	2	0	0	0
t	3	0	0	0

		c	**t**	**a**
	0	1	2	3
a	1	1	2	2
t	2	2	1	0
t	3	0	0	0

		c	**t**	**a**
	0	1	2	3
a	1	1	2	2
t	2	2	1	2
t	3	3	2	0

		c	**t**	**a**
	0	1	2	3
a	1	1	2	2
t	2	2	1	2
t	3	3	2	**2**

The Levenshtein Distance between **att** and **cta** is **2**

References

- MATLAB Documentation
  - *Bioinformatics Toolbox*.

    URL: https://it.mathworks.com/help/bioinfo/
  - *Sequence Alignment*.

    URL: https://it.mathworks.com/help/bioinfo/sequence-alignment.html
- FASTA Format
  - *NCBI*. URL: https://www.ncbi.nlm.nih.gov/BLAST/fasta.shtml
- Wikipedia.
  - *Levenshtein Distance*. URL: https://en.wikipedia.org/wiki/Levenshtein_distance