Lecture 5

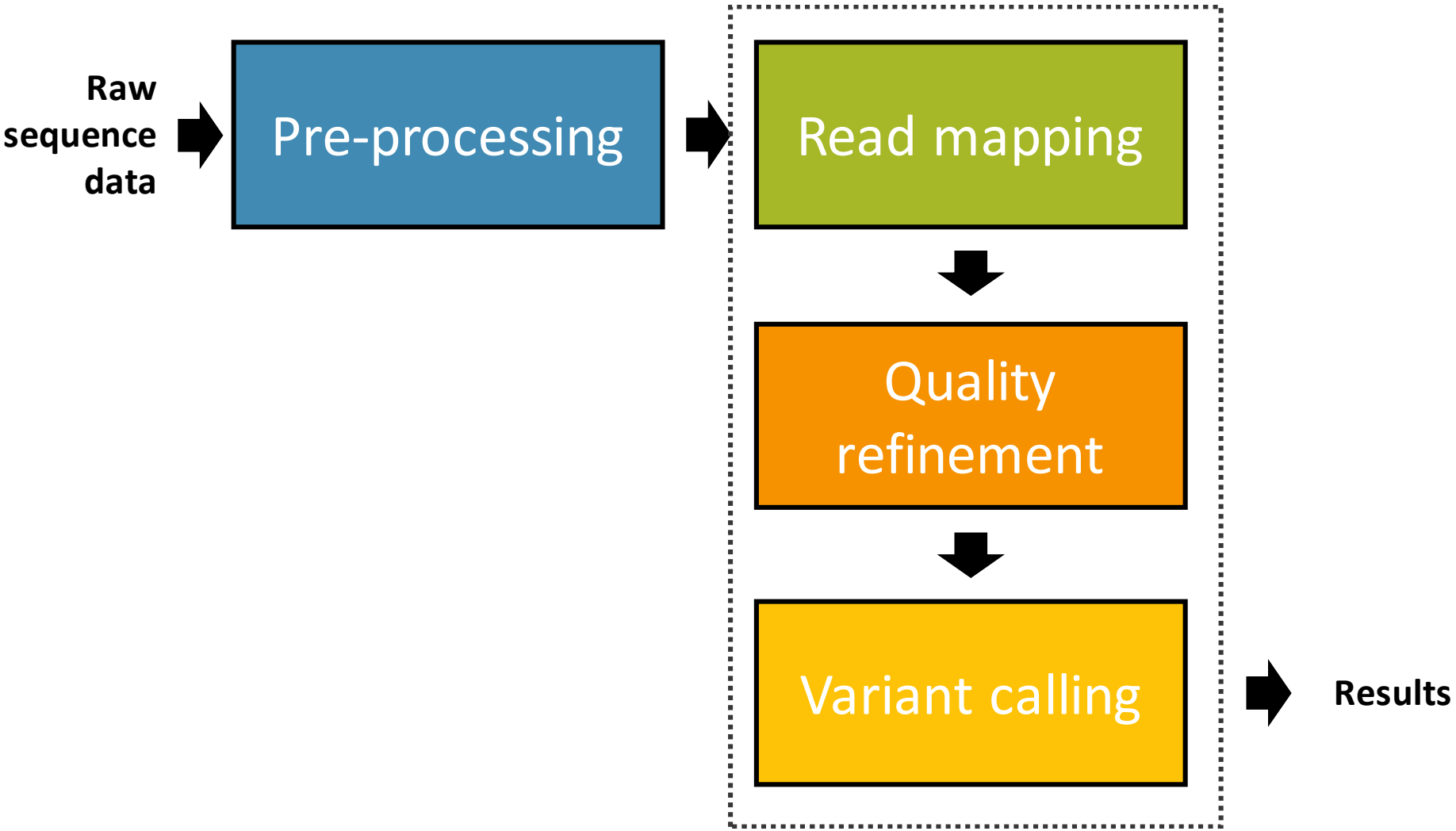
Introduction to bioinformatics (MVE510)

Autumn 2020

Additional reading: Mapping reads on a genomic sequence: an algorithmic overview and a practical comparative analysis. Schbath S, Martin V, Zytnicki M, Fayolle J, Loux V, and Gibrat JF. *Journal of Computational Biology*, 19(6) 2012.

What we did we do in the previous lecture?

- SNPs, indels and structural variants can be identified by whole genome sequencing
- Three main steps: read mapping, quality refinement and SNP calling
- SNP calling is done through statistical models describing the errors in the data
- Analysis of whole genome sequencing results in a massive data reduction: from billions of observations to a handful of mutations


Raw sequence data
Pre-processing
Read mapping
Quality refinement
Variant calling
Results

Today's agenda

- Pair-wise alignment of DNA sequences
  - The Needleman-Wunsch algorithm
  - The Smith-Waterman algorithm
- Computational complexity
- BLAST: The Basic Local Alignment Search Tool

Our main problem this lecture

The sequence reads – many and short

- Hundreds of million reads from sequencing, each only a few hundred nucleotides long

The reference – few and large

- The human genome consisting of 23 chromosomes (total 3.2×10^9 nucleotides)
- A bacterial genome (one chromosome, 4.6×10^6 nucleotides for *E. coli*)

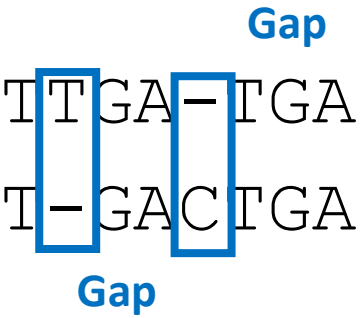
Our main problem this lecture

How do we find where our sequence read match ('align') to the reference sequences?

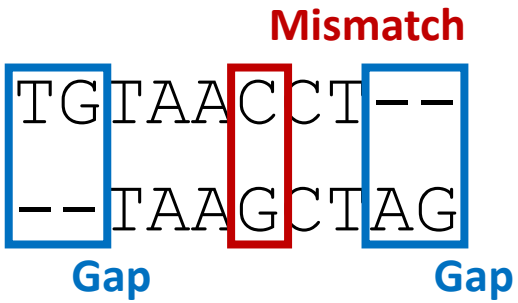
Example of alignments

Alignment 1

Alignment 2



Alignment 3



Two main forms of alignment

Global alignment

Two sequences are aligned over their full length.

Local alignment

Two sequences are aligned based on their best matching subsequences.

Local alignments are used to match short sequence reads against long reference sequences.

Example

Let $S(a,b) = \begin{cases} 5, & a = b \\ -4, & a \neq b \end{cases}$ and $d = -7$.

Score the following alignments.

Alignment 1

```
AGTCTAGT
AGTATAGT
```

S=5+5+5-4+5+5+5+5=31

Alignment 2

```
TTGA-TGA
T-GACTGA
```

S=5-7+5+5-7+5+5+5=16

Alignment 3

```
TGTAACCT--
--TAAGCTAG
```

S=-7-7+5+5+5-4+5+5-7-7=-9

Example

			1	2	3	4	5	6
			T	C	G	G	A	T
		0	-5	-10	-15	-20	-25	-30
1	T	-5						
2	C	-10						
3	C	-15						
4	A	-20						
5	T	-25						

Example

			1	2	3	4	5	6
			T	C	G	G	A	T
		0	-5	-10	-15	-20	-25	-30
1	T	-5	5	0	-5	-10	15	-20
2	C	-10	0	10	5	0	-5	-10
3	C	-15	-5	5	6	1	-4	-9
4	A	-20	-10	0	1	2	6	1
5	T	-25	-15	-5	-4	-3	1	11

Computational complexity

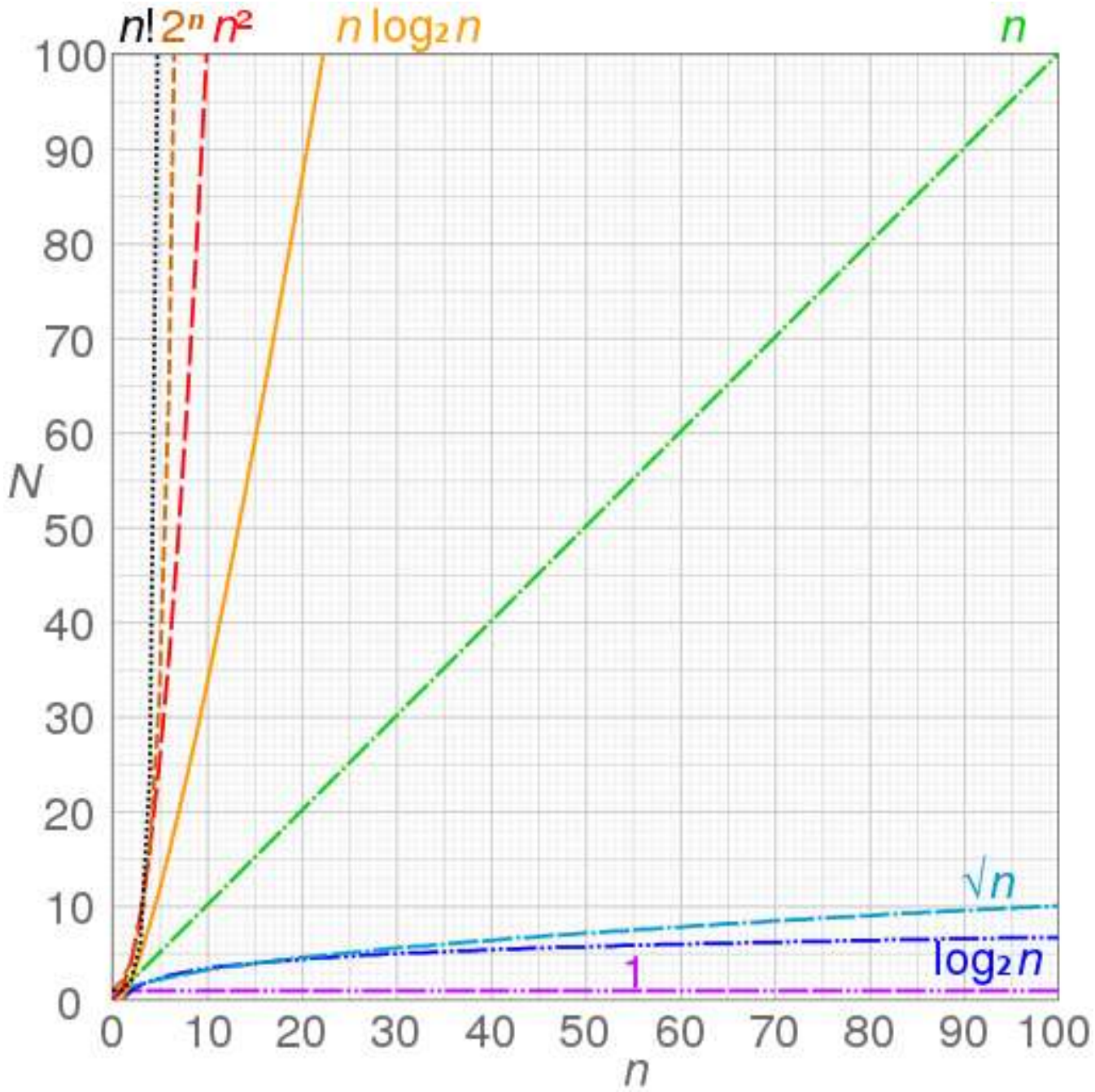
- Measures how many computations that are needed to execute an algorithm
- Fewer computations means faster algorithm

Computational complexity is measured using *O* notation (*O* stands for *ordo* which means order in latin).

Computational complexity

If an algorithm is *O(n)* this means that the number of computations grows linearly with respect to *n* (which can, for example, be the number of input data).

Other examples: $O(n^2)$, *O(log n), O(nm), etc.*


n!
2ⁿ
n²
n log₂ n
n
√n
log₂ n
1
N
n
0
10
20
30
40
50
60
70
80
90
100

Computational complexity

In our application, Smith-Waterman will have a complexity of

$$O(L_R L_G)$$

for each read. Here,

L_R= the length of the reads,

L_G= the length of the genome (reference).

This can become very slow if L_G is large (which it is in our case).

Futhermore….

Alignment of a 100 bases long sequence read against the first human chromosome requires a matrix of dimension 100×247,199,720. This which will take 100 gigabytes of RAM if stored in memory.

Smith-Waterman are too slow and resource heavy!
Can we make the alignment faster?

The BLAST algorithm (Basic Local Alignment Search Tool)

BLAST can be used to match any sequence (‘query’) against a reference (‘database’). The algorithm work as follows

1. Create a table of words (subsequences) of size W and their location in the reference (index, hash).
2. Find the position of the words present in the sequence. These positions are called ‘seeds’.
3. Extend the alignment around the seeds.

BLAST is around 50 times faster than Smith-Waterman.

The BLAST algorithm (Basic Local Alignment Search Tool)

Reference ('database')

ACGAGTGAGTGCCGAGTACGTAGCGTAGGAGTGAGTTGGAGTGAGACGTGAGT

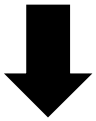
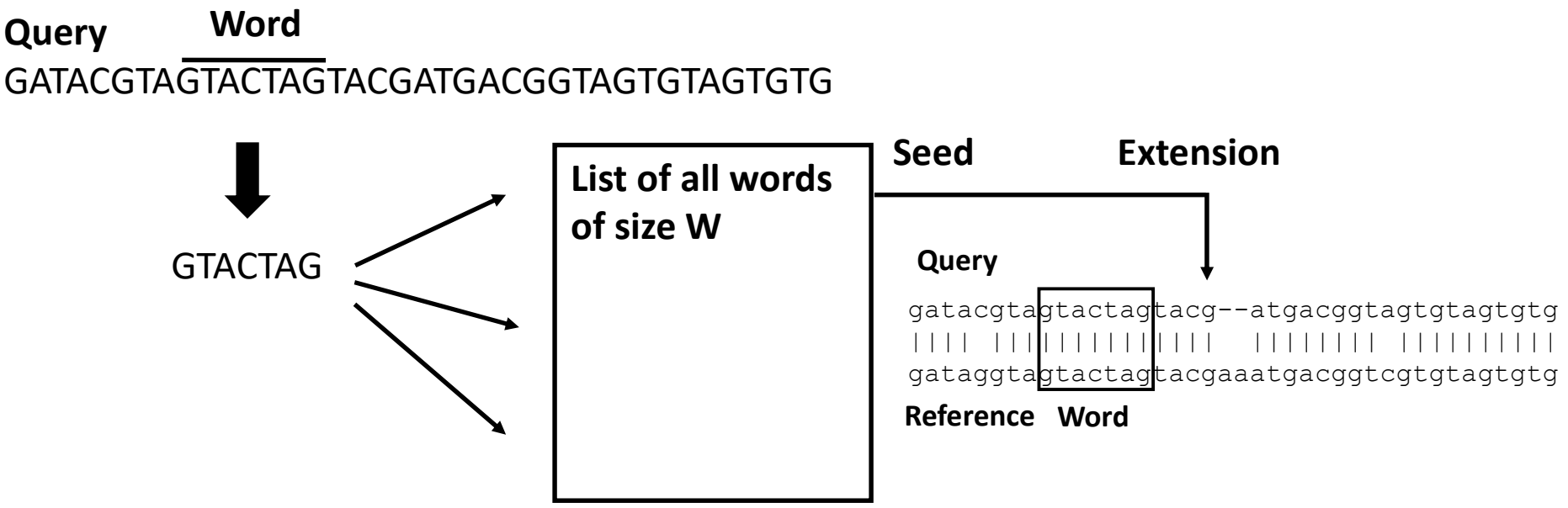


Table (hash) of words of size 7

Maximum $4^7=2^{14}=16384$ rows

Word	Position
ACGAGTG	1
CGAGTGA	2
GAGTGAG	3, 38
AGTGAGT	4, 30

The BLAST algorithm (Basic Local Alignment Search Tool)

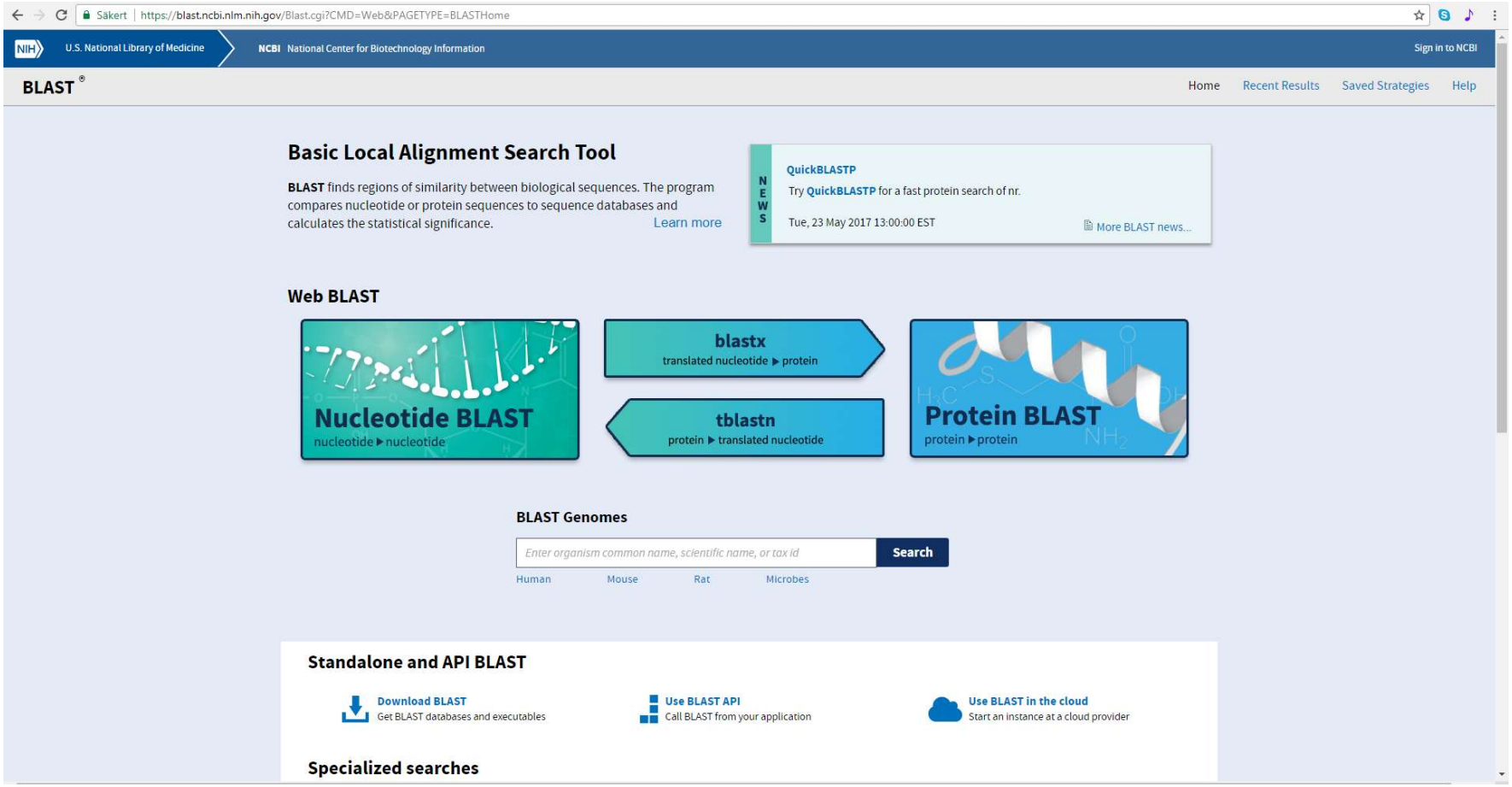


```
BLASTN 2.2.26 [Sep-21-2011]

Query= read1
         (50 letters)

Database: CCDS_nucleotide.20160908.fna
           32,554 sequences; 55,631,471 total letters

Searching.....................................................done

                                                                 Score    E
Sequences producing significant alignments:                      (bits) Value

CCDS73971.1|Hs108|chr17                                            84   1e-16
CCDS73970.1|Hs108|chr17                                            84   1e-16
CCDS73969.1|Hs108|chr17                                            84   1e-16

>CCDS73971.1|Hs108|chr17
          Length = 909

 Score = 83.8 bits (42), Expect = 1e-16
 Identities = 48/50 (96%)
 Strand = Plus / Plus


Query: 1   aggctgctcccccggtggcctctgcaccagcagctcctacaccggcggcc 50
           |||||||||||| |||||| ||||||||||||||||||||||||||||
Sbjct: 86  aggctgctccccccgtggcccctgcaccagcagctcctacaccggcggcc 135
```

The BLAST algorithm (Basic Local Alignment Search Tool)

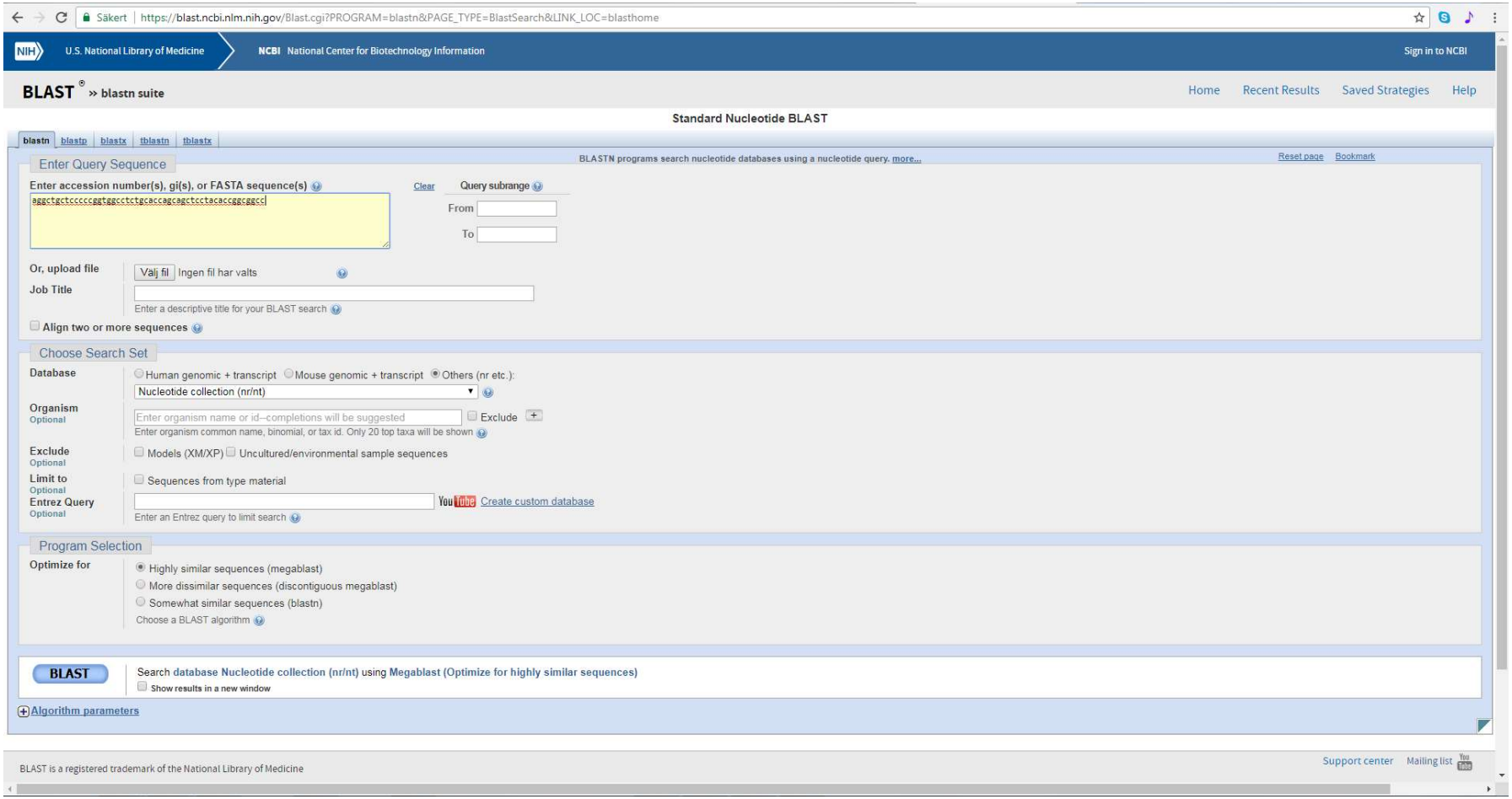
- The word size changes the sensitivity
  - Large word size means faster algorithm but lower sensitivity
  - Smaller word size means slower algorithms but higher sensitivity.
- BLAST can compare
  - Nucleotide vs nucleotide (‘blastn’)
  - Protein vs protein (‘blastp’)
  - Nucleotide vs protein (‘blastx’)
  - Protein vs nucleotide (‘tblastn’)

The BLAST algorithm (Basic Local Alignment Search Tool)



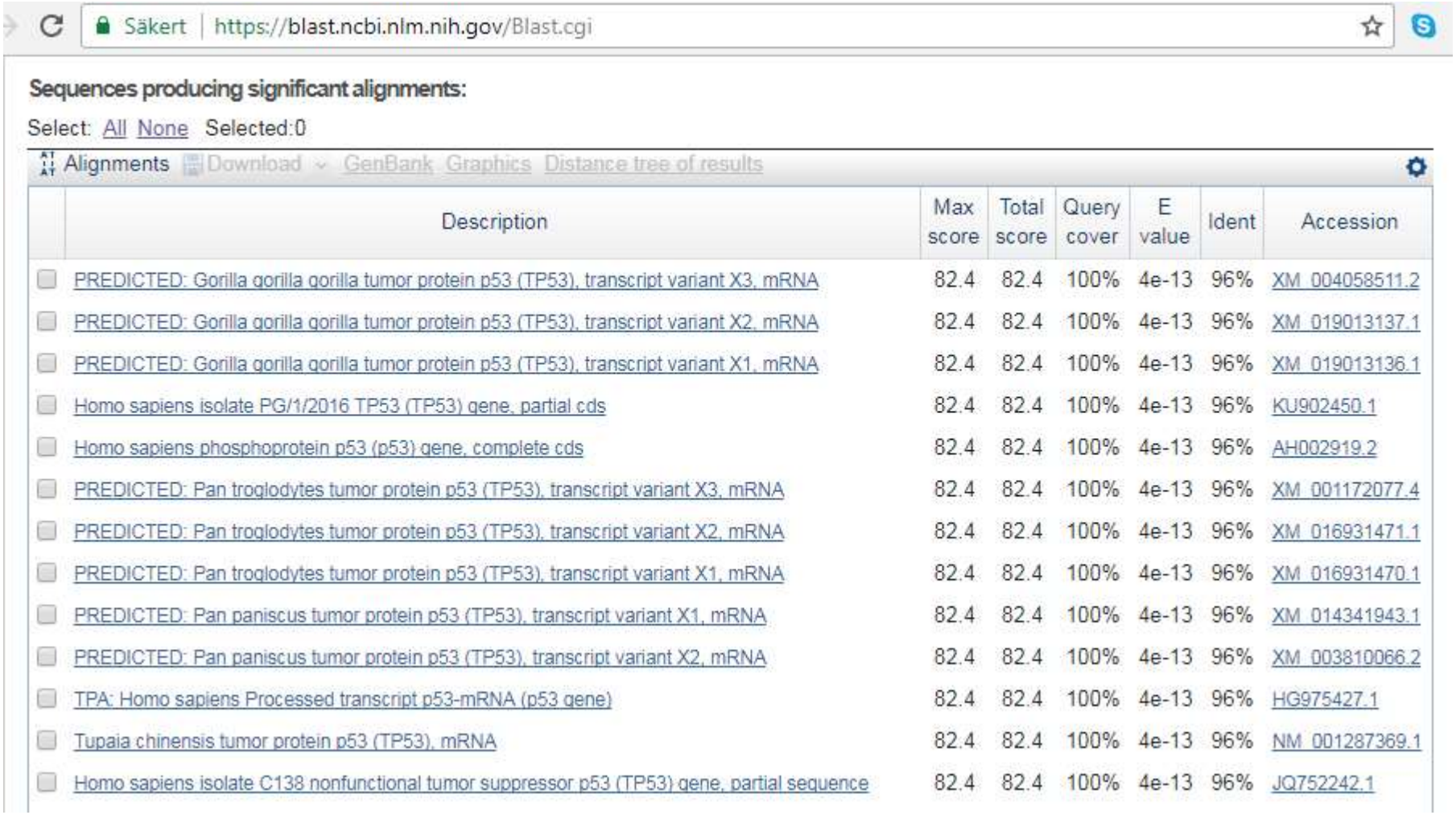
https://www.ncbi.nlm.nih.gov/genbank/

The BLAST algorithm (Basic Local Alignment Search Tool)



https://www.ncbi.nlm.nih.gov/genbank/

The BLAST algorithm (Basic Local Alignment Search Tool)



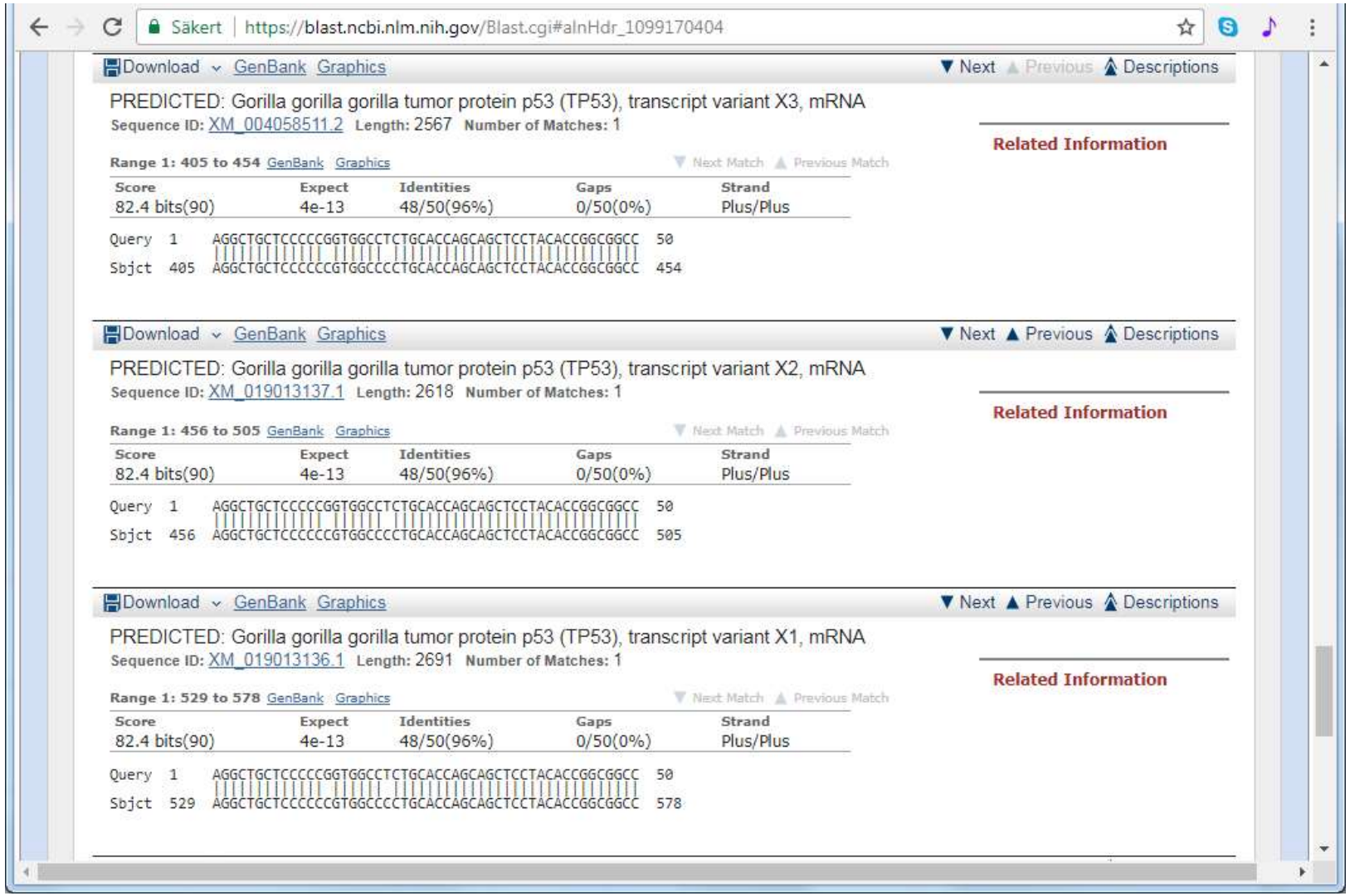
Sequences producing significant alignments:

Select: All None Selected:0

Alignments Download GenBank Graphics Distance tree of results

	Description	Max score	Total score	Query cover	E value	Ident	Accession
☐	PREDICTED: Gorilla gorilla gorilla tumor protein p53 (TP53), transcript variant X3, mRNA	82.4	82.4	100%	4e-13	96%	XM_004058511.2
☐	PREDICTED: Gorilla gorilla gorilla tumor protein p53 (TP53), transcript variant X2, mRNA	82.4	82.4	100%	4e-13	96%	XM_019013137.1
☐	PREDICTED: Gorilla gorilla gorilla tumor protein p53 (TP53), transcript variant X1, mRNA	82.4	82.4	100%	4e-13	96%	XM_019013136.1
☐	Homo sapiens isolate PG/1/2016 TP53 (TP53) gene, partial cds	82.4	82.4	100%	4e-13	96%	KU902450.1
☐	Homo sapiens phosphoprotein p53 (p53) gene, complete cds	82.4	82.4	100%	4e-13	96%	AH002919.2
☐	PREDICTED: Pan troglodytes tumor protein p53 (TP53), transcript variant X3, mRNA	82.4	82.4	100%	4e-13	96%	XM_001172077.4
☐	PREDICTED: Pan troglodytes tumor protein p53 (TP53), transcript variant X2, mRNA	82.4	82.4	100%	4e-13	96%	XM_016931471.1
☐	PREDICTED: Pan troglodytes tumor protein p53 (TP53), transcript variant X1, mRNA	82.4	82.4	100%	4e-13	96%	XM_016931470.1
☐	PREDICTED: Pan paniscus tumor protein p53 (TP53), transcript variant X1, mRNA	82.4	82.4	100%	4e-13	96%	XM_014341943.1
☐	PREDICTED: Pan paniscus tumor protein p53 (TP53), transcript variant X2, mRNA	82.4	82.4	100%	4e-13	96%	XM_003810066.2
☐	TPA: Homo sapiens Processed transcript p53-mRNA (p53 gene)	82.4	82.4	100%	4e-13	96%	HG975427.1
☐	Tupaia chinensis tumor protein p53 (TP53), mRNA	82.4	82.4	100%	4e-13	96%	NM_001287369.1
☐	Homo sapiens isolate C138 nonfunctional tumor suppressor p53 (TP53) gene, partial sequence	82.4	82.4	100%	4e-13	96%	JQ752242.1

The BLAST algorithm (Basic Local Alignment Search Tool)



The BLAST algorithm (Basic Local Alignment Search Tool)

Is BLAST fast enough?

No! Next generation DNA sequencing produce too many reads (50x faster than SW is not enough!).

Can we further improve the speed?

Yes, but we need some more concepts from computer science!

Summary of lecture 5

- The Needleman-Wunsch algorithm can be used to find global alignments. The best alignment is identified by iteratively fill a alignment matrix and backtrack from the highest value.
- The Smith-Waterman algorithm can be find local alignments. The best alignment is found similarly to the NW algorithm but where all negative values has been replaced by zeros and the alignment can start and end at any positions.

Summary of lecture 5

- BLAST uses a ‘seed-and-extend’ algorithm to efficiently calculate local alignments. This makes it 50 times fast than Smith-Waterman.
- BLAST is however still to slow to efficiently map reads from next generation sequencing to a reference.