GenEpi-BioTrain – Virtual Training 7, Session 1

Sequence alignment in theory and practice

Preparation for phylogenetic analysis

Thor Bech Johannesen
Sequencing and Bioinformatics
Department of Bacteria, Parasites and Fungi
Statens Serum Institut, Copenhagen, Denmark

Sequence alignment and phylogenetic analysis

Both sessions will be mostly focused on practical use, i.e. we will be running a lot of different programs and learn how to use them.

But it is also important that you understand the key theoretical concepts - you need to know about potential pitfalls that may lead to poor or incorrect analysis

The real fun of these sessions will be tomorrow. Then you will get to generate and visualize beautiful trees, were you can add neat markers for various types of metadata, which can help you if you want to explore evolutionary relationships or do outbreak investigations.

But you cannot infer a phylogeny from genetic data withouth first doing sequence alignment of your data!
And that will be the focus of this first session.

The format today

- A brief presentation on sequence alignment theory
- How a sequence alignment algorithm works in practice
  - Coffee break
- Multiple sequence alignment on the command line: MAFFT and MUSCLE
- Local sequence search and alignment: BLAST
  - Coffee break
- Generating core genome SNP alignments: Snippy
- Wrap up

Intended learning outcomes

At the end of todays session you should be able to

- Describe when and why we need to perform sequence alignment
- Describe the different uses of global sequence alignment, local sequence alignment, and multiple sequence alignment (MSA)
- Describe what substitution matrices and gap penalties are, and how they are used when evaluating and generating sequence alignments
- Generate a pairwise sequence alignment using the Needleman-Wunsch algorithm
- Generate a multiple sequence alignment (MSA) using two common command line alignment tools, Muscle and Mafft
- Use Basic Local Alignment Search Tool (Blast) on command line
- Use Snippy to generate a core genome alignment from reads and assemblies

Data used in this Virtual Training

The data used for the exercises in these training sessions can be downloaded from EVA or github
It includes

- A fasta file containing the v3-v4 region of the 16s rRNA gene from 14 isolates from various species
- Paired-end Illumina reads from 22 *Listeria monocytogenes* isolates from an outbreak investigation
- Genome assemblies generated from those read data
- Various metadata files for the *Listeria monocytogenes* 22 isolates

Getting the data and doing the exercises

A variety of tools and use cases will be demonstrated during these two sessions

If you wish to follow along and run the various tools yourself as we introduce them, the easiest way is likely via the markdown found here

https://github.com/ssi-dk/GenEpi-BioTrain_Virtual_Training_7/blob/main/practicals_s1_alignment.md

An html-version of this markdown file is also available on the EVA platform under Session1 -> exercise -> practicals_s1_alignment.html

For most exercises the example data used can be downloaded very swiftly, and we will do so at the beginning of each exercise.

The one exception to this is the raw read files, which are used for the very last exercises today.
Because downloading these take a while we will start right away. How to do this is detailed here:
https://github.com/ssi-dk/GenEpi-BioTrain_Virtual_Training_7/blob/main/README.md#download-raw-read-files-optional-and-only-used-in-one-optional-step-in-the-practicals

Why do we do sequence alignment?

- To investigate evolutionary relationships
  - Alignments allow us to quantify genetic drift

- To find common motifs between sequences
  - Identify the same gene in different isolates or species
  - Identify similar structures that appear in different genes

- To analyze specific variation in a gene sequence
  - Identify mutations which may alter protein structure
  - Identify mutations in active sites
  - Identify nonsense or frame shift mutations that cause loss of function

Types of sequence alignment?

- Pairwise sequence alignment (global)
  Align the full length of two similar sequences

- Multiple sequence alignment, MSA (global)
  Align the full length of multiple sequences

- Local sequence alignment (f.x BLAST)
  Identify similar regions in sequences, and align those regions

Sequence alignment, how?

Have a look at the letters and composition of these names
Would you say any of them are similar?

KARSTEN

THOR

ASTRID

KIRSTEN

TORSTEN

KASPER

KIRSTINE

Global sequence alignment

Aim: to find the best alignment over the entire length of two sequences
How do we quantify the "goodness" of an alignment?

```
KIRSTEN
|  |||||
KARSTEN
```

6 match
1 mismatch

Score = 6-1 = 5

```
KIRSTEN
   |||||
TORSTEN
```

5 match
2 mismatch

Score = 5-2 = 3

By a pre-defined Scoring scheme:

F. ex.:

Match = +1
Mismatch = -1

Global sequence alignment

If all we had to do was count matches and mismatches, this would be trivial
But we also have to account for insertions and deletions

```
KARSTEN
||
KASPER-
```

2 match
4 mismatch
1 gap

Score = 2-4-1 = -3

```
KARSTEN
||  |  |
KA-SPER
```

4 match
2 mismatch
1 gap

Score = 4-2-1 = 1

Scoring scheme:

Match = +1
Mismatch = -1
Gap = -1

Global sequence alignment of DNA

We can further optimize our scoring system by using a substitution matrix instead of simply looking at match/mismatch

Substitution matrix with even weights
(Match = +1, mismatch = -1, just as before)

	C	T	A	G
C	1	-1	-1	-1
T	-1	1	-1	-1
A	-1	-1	1	-1
G	-1	-1	-1	1

Weighted substitution matrix were purine <-> pyrimidine mutations are penalized more than purine <-> purine and pyrimidine <-> pyrimidine

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

What's the score?

Score this alignment using the weighted matrix (and a gap penalty of -2)

```
AT-CTTG
ATGCTGG
```

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

2+2-2+2+2-1+2 = +7

But how do we determine were to put the gap?

The Needleman-Wunsch algorithm

- The Needleman-Wunsch algorithm is an example of dynamic programming
- Dynamic programming is an algorithmic paradigm in which complex problems are broken down into smaller, simpler to solve subproblems that can be solved one by one
- Despite being reduced to simpler problems, the solution given by the algorithm should be optimized
- In sequence alignent, this means the solution should give the highest possible alignment score, given your choice of substitution matrix and gap penalty

This is how it works

The Needleman-Wunsch algorithm

ATGCTCG
ATCTTG

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

Gap penalty = -2

		A	T	G	C	T	C	G
	0							
A								
T								
C								
T								
T								
G								

Start in the top left corner
For each field ($M_{i,j}$), calculate three scores (C)

Deletion score:
$C_{del}(M_{i,j}) = C(M_{i,j-1}) + GapPenalty$

Insertion score:
$C_{ins}(M_{i,j}) = C(M_{i-1,j}) + GapPenalty$

Match score:
$C_{match}(M_{i,j}) = C(M_{i-1,j-1}) + SubstitutionPenalty$

Then fill out the field with whichever score is the best:

$C = max(C_{del}, C_{ins}, C_{match})$

The Needleman-Wunsch algorithm

ATGCTCG
ATCTTG

		A	T	G	C	T	C	G
	0	-2	-4	-6	-8	-10	-12	-14
A	-2	?						
T	-4							
C	-6							
T	-8							
T	-10							
G	-12							

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

Gap penalty = -2

C_{del}=
$C(M_{i,j-1})$ + GapPenalty =
-2-2 = -4

Left score

C_{ins}=
$C(M_{i-1,j})$ + GapPenalty =
-2-2 = -4

Top score

C_{match}=
$C(M_{i-1,j-1})$ + SubstitutionPenalty =
0 + 2 = 2

Diagonal score

The Needleman-Wunsch algorithm

ATGCTCG
ATCTTG

		A	T	G	C	T	C	G
	0	-2	-4	-6	-8	-10	-12	-14
A	-2	2						
T	-4							
C	-6							
T	-8							
T	-10							
G	-12							

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

Gap penalty = -2

The Needleman-Wunsch algorithm

ATGCTCG
ATCTTG

		A	T	G	C	T	C	G
	0	-2	-4	-6	-8	-10	-12	-14
A	-2	2	?					
T	-4							
C	-6							
T	-8							
T	-10							
G	-12							

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

Gap penalty = -2

C_{del}=
$C(M_{i,j-1})$ + GapPenalty =
2 - 2 = 0

C_{ins}=
$C(M_{i,j-1})$ + GapPenalty =
-4 - 2 = -6

C_{match}=
$C(M_{i-1,j-1})$ + SubstitutionPenalty =
-2 - 1 = -3

Exercise: Needleman-Wunsch algorithm

ATGCTCG
ATCTTG

		A	T	G	C	T	C	G
	0	-2	-4	-6	-8	-10	-12	-14
A	-2	2 ←	0					
T	-4							
C	-6							
T	-8							
T	-10							
G	-12							

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

Gap penalty = -2

Exercise: Needleman-Wunsch algorithm

ATGCTCG
ATCTTG

		A	T	G	C	T	C	G
	0	-2	-4	-6	-8	-10	-12	-14
A	-2	2	0	-2				
T	-4	0	4					
C	-6	-2						
T	-8							
T	-10							
G	-12							

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

Gap penalty = -2

Let's see if you're all still awake

Exercise: Needleman-Wunsch algorithm

ATGCTCG
ATCTTG

		A	T	G	C	T	C	G
	0	-2	-4	-6	-8	-10	-12	-14
A	-2	2	0	-2				
T	-4	0	4					
C	-6	-2	?					
T	-8							
T	-10							
G	-12							

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

Gap penalty = -2

C_{del}=
$C(M_{i,j-1})$ + GapPenalty =
-2 - 2 = -4

C_{ins}=
$C(M_{i,j-1})$ + GapPenalty =
4 - 2 = 2

C_{match}=
$C(M_{i-1,j-1})$ + SubstitutionPenalty =
0 + 1 = 1

Exercise: Needleman-Wunsch algorithm

ATGCTCG
AT-CTTG

		A	T	G	C	T	C	G
	0	-2	-4	-6	-8	-10	-12	-14
A	-2	2	0	-2	-4	-6	-8	-10
T	-4	0	4	2	0	-2	-4	-6
C	-6	-2	2	3	4	2	0	-2
T	-8	-4	0	1	4	6	4	2
T	-10	-6	-2	-1	2	6	7	5
G	-12	-8	-4	0	0	4	5	9

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

Gap penalty = -2

Needleman-Wunsh: The traceback

```
A T G C T C G
| |   | |   |
A T - C T T G
```

		A	T	G	C	T	C	G
	done	left	left	left	left	left	left	left
A	up	diag	left	left	left	left	left	left
T	up	up	diag	left	left	diag	left	left
C	up	up	up	diag	diag	left	diag	left
T	up	up	diag	diag	diag	diag	left	left
T	up	up	diag	diag	diag	diag	diag	left
G	up	up	up	diag	up	up	diag	diag

15 Minute Break!

We will resume at xx:xx

If you have any questions you'd like answered before we proceed, write them in the chat and we will answer as best we can after the break.
Then we'll proceed to the practical parts of today's session.

Protein sequence alignment?

We will be focusing on DNA sequence alignment today, but protein alignments work just the same, only with a more complex substitution matrix because there are 20 standard amino acids and only 4 types of nucleotides in DNA

Blosum62 substitution matrix

	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
A	0	1	0	4																	A
G	-3	0	-2	0	6																G
P	-3	-1	-1	-1	-2	7															P
D	-3	0	-1	-2	-1	-1	6														D
E	-4	0	-1	-1	-2	-1	2	5													E
Q	-3	0	-1	-1	-2	-1	0	2	5												Q
N	-3	1	0	-2	0	-2	1	0	0	6											N
H	-3	-1	-2	-2	-2	-2	-1	0	0	1	8										H
R	-3	-1	-1	-1	-2	-2	-2	0	1	0	0	5									R
K	-3	0	-1	-1	-2	-1	-1	1	1	0	-1	2	5								K
M	-1	-1	-1	-1	-3	-2	-3	-2	0	-2	-2	-1	-1	5							M
I	-1	-2	-1	-1	-4	-3	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-1	-4	-3	-4	-3	-2	-3	-3	-2	-2	2	2	4					L
V	-1	-2	0	0	-3	-2	-3	-2	-2	-3	-3	-3	-2	1	3	1	4				V
W	-2	-3	-2	-3	-2	-4	-4	-3	-2	-4	-2	-3	-3	-1	-3	-2	-3	11			W
Y	-2	-2	-2	-2	-3	-3	-3	-2	-1	-2	2	-2	-2	-1	-1	-1	-1	2	7		Y
F	-2	-2	-2	-2	-3	-4	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	1	3	6	F
	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	

Multiple sequence alignments

Multiple sequence alignment is considerably more complex than pairwise alignment

But we want you to take home some practical skills, so we will rather focus on some exercises

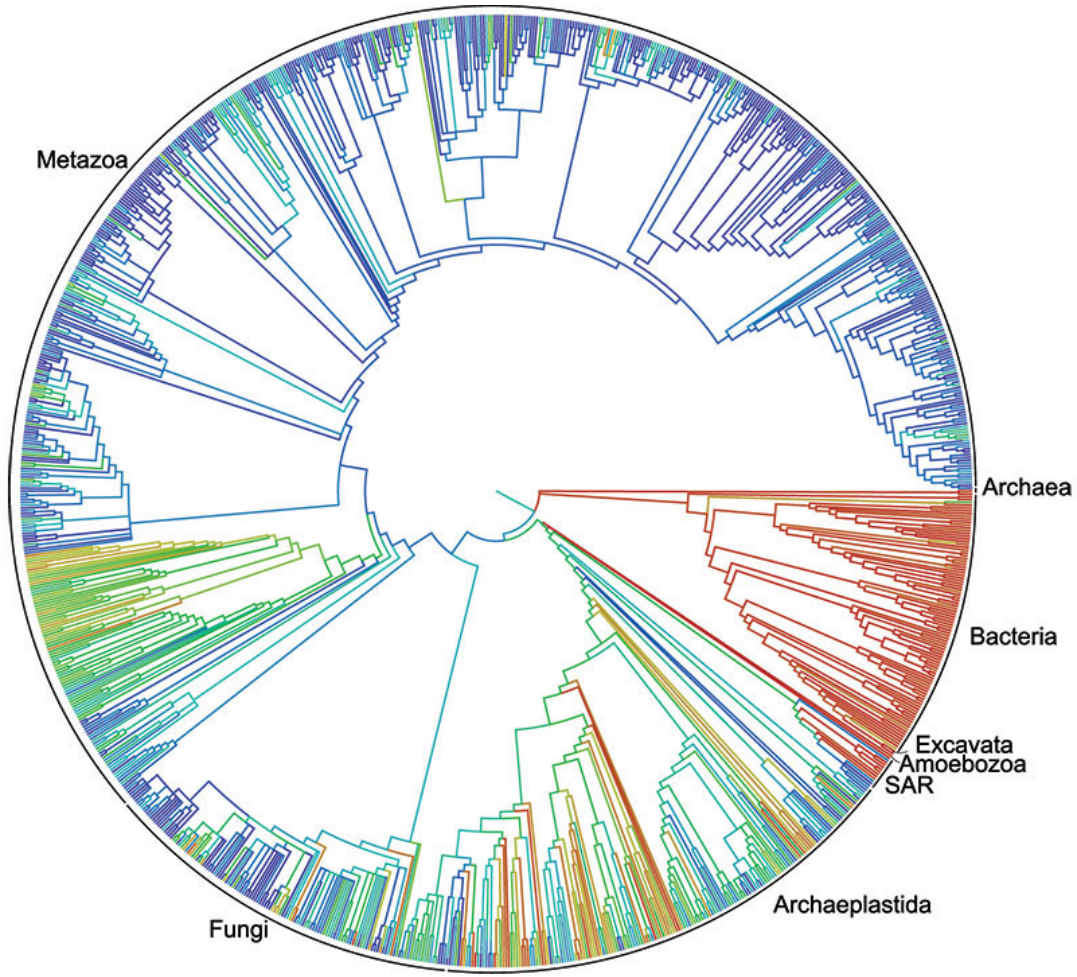
Note however, that Multiple Sequence Alignments are (for the most part) not guaranteed to provide you the best possible alignment.

```
KIRSTEN     K-IRSTEN-
KARSTEN     K-ARSTEN-
TORSTEN     T-ORSTEN-
KASPER      K-A-SPER-
THOR        THOR-----
KIRSTINE    K-IRSTINE
ASTRID      --A-STRID
```

Phylogenetics

Investigating the evolutionary relationship between organisms

We have to align our sequences before we can infer a phylogenetic relationship



https://www.science.org/content/article/first-comprehensive-tree-life-shows-how-related-you-are-millions-species

Exercise: Inferring the evolutionary relationship between bacteria from the 16s rRNA gene

Run this command to download the fasta file with example data
wget https://github.com/ssi-dk/GenEpi-BioTrain_Virtual_Training_7/raw/main/data/16s_data/16s_sequences.fasta

The file contains part of the 16s rRNA gene from 14 isolates, the v3-v4 region which is widely used for microbiome analysis

We can use these sequences to infer the evolutionary relationship between these species

But first we have to align them

Exercise: MAFFT, Multiple Alignment using Fast Fourier transform

MAFFT is one of the first published methods for rapid multiple sequence alignment, and is still widely used
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC135756/

First, lets have a look at what the helper file can tell us

```
mafft -h
```

Now make an alignment for the 16s_sequences.fasta

```
mafft 16s_sequences.fasta > 16s_sequences_mafft_alignment.fasta
```

Exercise: MUSCLE, MUltiple Sequence Comparison by Log-Expectation

MUSCLE was published a few years after MAFFT, and is very fast when aligning large numbers of sequences (1000+)
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC390337/

Again, lets have a look at what the helper file can tell us

```
muscle -h
```

Now make an alignment for the 16s_sequences.fasta

```
muscle -align 16s_sequences.fasta -output
16s_sequences_muscle_alignment.fasta
```

Exercise: MAAFT and MUSCLE

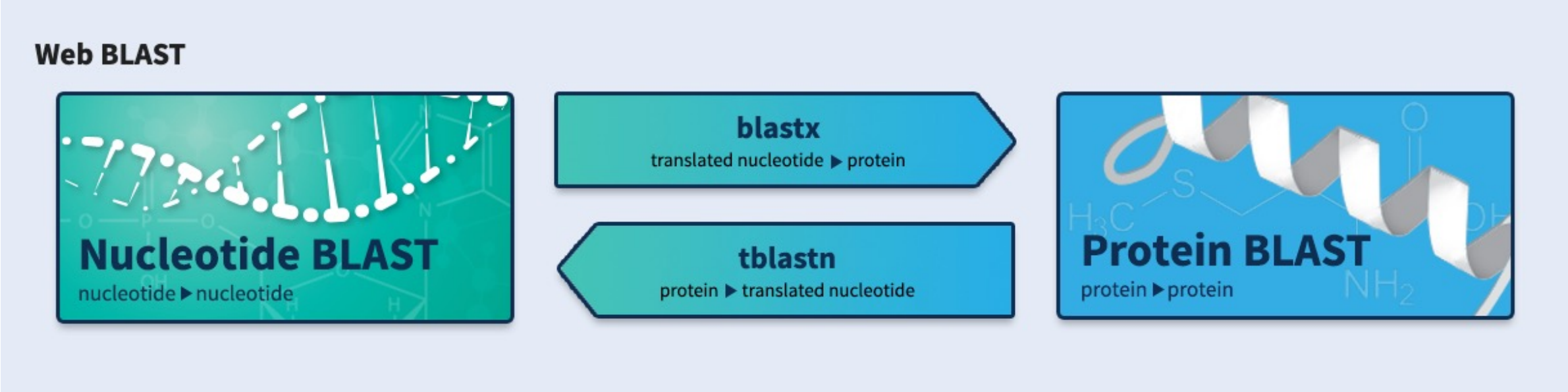
Have a look at the two alignments you generated

less 16s_sequences_mafft_alignment.fasta

less 16s_sequences_muscle_alignment.fasta

Tomorrow we will use these alignments to infer phylogenies and investigate the evolutionary relationship between these species
Before then we have a few more topics of sequence alignment we need to go over

Basic Local Alignment Search Tool (BLAST)



https://blast.ncbi.nlm.nih.gov/Blast.cgi

Command line BLAST

There are two mandatory inputs for a BLAST search

- -query: query sequence(s) (fasta format)
- -subject OR -db: subject sequence(s) (fasta format) or blast-formatted database

Blast will perform local alignments between similar regions in your query and subject/database

There are many different options for output format.

We will look at two today:

- Default: Easy to read with the naked eye, neatly visualizes hits and alignments between query and subject sequences
- -outfmt 6: Tab-separated format where you choose what information to include in each column. Much nicer to use for anything programmatic

BLAST

Blast uses the Smith-Waterman algorithm, which is a variation of the Needleman-Wunsch algorithm, to perform local sequence alignment

Like Needleman-Wunsch , the Smith-Waterman algorithm is a dynamic programming algorithm, and therefore guaranteed to provide you the best possible alignment between your two sequences.

But Blast is not just an alignment tool!

The “search” part of Blast is a search for exact kmer matches between your query and subject sequences. This is necessary because aligning all parts of large query and subject sequences is way too demanding computationally, and exact kmer matching is very fast.

BLAST

Therefore there are three important parameters in blast that dictate what your alignment output looks like.

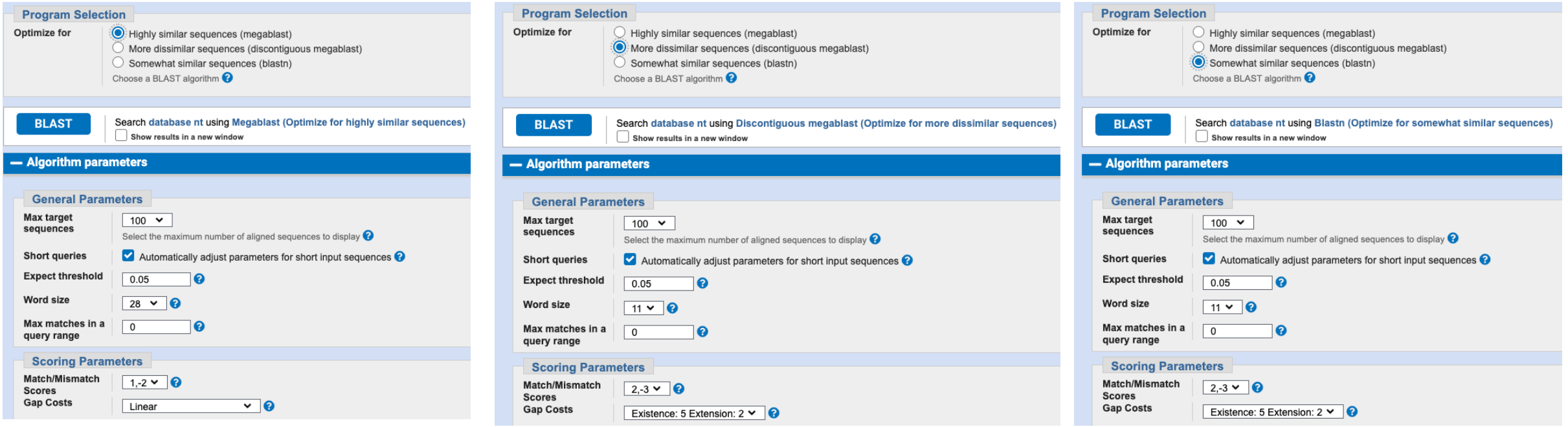
- The substitution matrix (match and mismatch score)
- The gap penalty
- The kmer size (or word size) used for initial search (seeding)

Blast will only perform local alignment in regions where it identifies an exact match of at least k nucleotides between your query and subject sequences.

When you use default "blastn" on command line, this is equivalent of what is called "megablast" in the online version of blast. This uses a "word size" of 28.

Even if your query and subject sequence share a region with 96% similarity, Megablast will fail to do an alignment if the mismatches between the two are spread out so that there is no exact match of 28 nucleotides!

Basic Local Alignment Search Tool (BLAST)



You can pick which program to use with the -task option, i.e.
blastn -task blastn -query [query.fasta] -subject [subject.fasta]

You can also modify the exact match (-reward), mismatch (-penalty) and gap scores (-gapopen, -gapextend)

BLAST command line input and output

Megablast with bare minimum input options

```
blastn -query [query.fasta] -subject [subject.fasta] -out [blast_output.txt]
```

Megablast with a more programmatic friendly output format

```
blastn -query [query.fasta] -subject [subject.fasta] -out [blast_output.tsv] -outfmt 6
```

Megablast, programmatic friendly output with select info

```
blastn -query [query.fasta] -subject [subject.fasta] -out [blast_output.tsv] -outfmt "6
qseqid sseqid pident length qlen evalue bitscore sseq"
```

BLAST exercises

Identifying the v3-v4 region of the 16s gene in genome assemblies

- Running megablast (blastn)
- Modifying output format
- Setting up a blast database
- Running megablast against a database

End of part 2 quiz!

That is it for part 2

We will do a very short speed quiz

15 Minute Break!

We will resume at xx:xx

If you have any questions you'd like answered before we proceed, write them in the chat and we will answer as best we can after the break.
Then we'll proceed to the last part of today's session:
core genome allignments

Core genome SNP analysis

Earlier we created the alignments necessary to infer the evolutionary relationship between isolates based on the 16s rRNA gene.

Using only a region of ~500 nucleotides naturally provides very low discriminatory power. Often different species will be identical in the specific region.

With whole-genome sequencing we have the option to not just look at a single gene (or a few genes as is the case for Multilocus Sequence Typing), but instead compare the full genomes

Core genome SNP analysis tools

- Snippy

https://github.com/tseemann/snippy

- NASP (Northern Arizona SNP pipeline)

https://tgennorth.github.io/NASP/readme.html

- CSI Phylogeny

https://bioinformaticshome.com/tools/descriptions/CSI_Phylogeny.html#gsc.tab=0

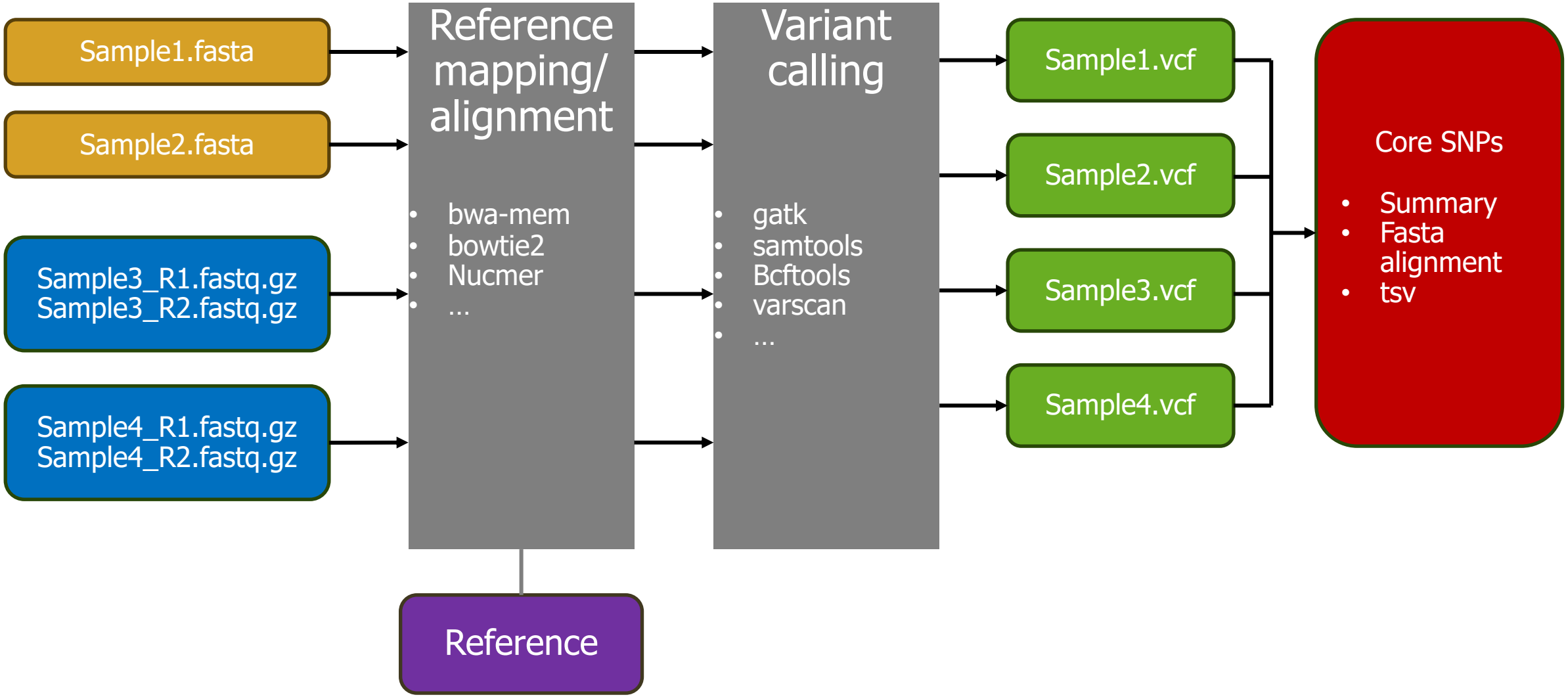
Core genome SNP analysis

All these tools work by aligning reads and/or assembled genomes to a specified reference genome, in order to identify variant sites (Single Nucleotide Polymorphisms, SNPs)

They can then compare the results across multiple isolates, and generate a output with all the variable sites in a convenient matrix or fasta-format. The fasta output can be used directly to generate a core genome phylogeny.

Some tools contain a bit more functionality than others, and can also assist you a lot in quality control.

Core genome SNP calling workflow



Exercise: Snippy

The last exercise for today is to run Snippy to create a core genome alignment based on the reads or genome assemblies from the 22 Listeria isolates.

In tomorrows session on phylogenies we will use this alignment to do phylogenetic reconstruction and use this to aid in an oubtreak investigation.

This exercise takes a while to finish. Do not worry if you don’t have an output for tomorrows exercises! We will of course provide you with a finished core genome alignment to use.

Acknowledgements

The creation of this training material was commissioned by ECDC to Statens Serum Institut (SSI) and developed by Thor Bech Johannesen, for use in training at the ECDC Virtual Academy (EVA) platform