Next Generation Sequencing Technologies: Data Analysis and Applications

Bowtie2 output

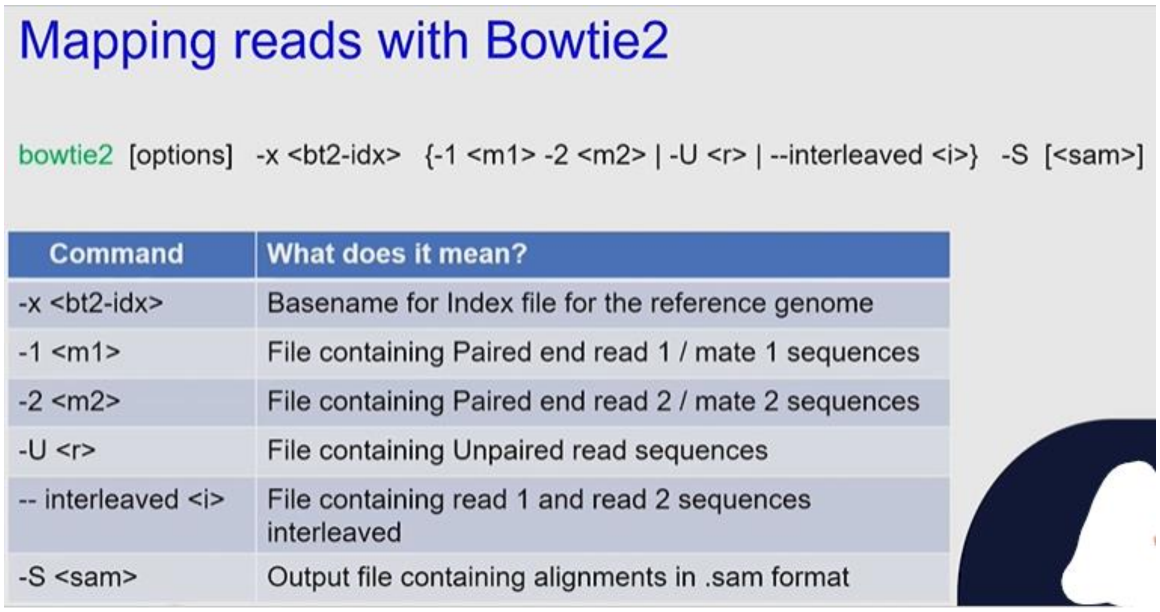
Dr. Riddhiman Dhar, Department of Biotechnology

Indian Institute of Technology Kharagpur

Good day, everyone. Welcome to the course on Next Generation Sequencing Technologies, Data Analysis, and Applications. In the last few classes, we have discussed the mapping algorithms, and then we have followed that up with some hands-on. So, we have introduced a tool called Bowtie2, which actually does maps and we installed this tool in our system. I have shown you how we can build a genome index, right? So, there are two steps to using this tool. The first step is building the genome index, which we did use a reference genome, and then we can do the read mapping once we have built that index, and we have also seen how we can actually do this mapping.

So, in today's class, we will be discussing how we interpret this data. So, once we have generated this output file from Bowtie2, how do we actually interpret this data? So, this is the concept that we will be covering in this class. So, we will interpret the Bowtie2 output because this is a very complex file, and there are a lot of details in that file, ok?

So, these are the keywords that we will come across. One is called SAM format, and the other is header lines. So, just to summarize what we have discussed so far and what we have done in the last few classes, we have set up Bowtie2. We have used this Bowtie2 build function for building this genome index, and there are certain options that you can give in the command line when you are using this Bowtie2 build. Then, we use the Bowtie2 function itself.



Mapping reads with Bowtie2

```
bowtie2  [options]   -x <bt2-idx>   {-1 <m1> -2 <m2> | -U <r> | --interleaved <i>}   -S  [<sam>]
```

Command	What does it mean?
-x <bt2-idx>	Basename for Index file for the reference genome
-1 <m1>	File containing Paired end read 1 / mate 1 sequences
-2 <m2>	File containing Paired end read 2 / mate 2 sequences
-U <r>	File containing Unpaired read sequences
-- interleaved <i>	File containing read 1 and read 2 sequences interleaved
-S <sam>	Output file containing alignments in .sam format

So, this is the function that we use for mapping, ok? And with Bowtie2, there are a lot of options. We have mentioned that some of these options could be useful, and we have also discussed concordant and discordant mapping. So, just to remind you again, right, this is how we run this command, ok? So, there are these inputs that we have to give, right? So, we have this minus x, right?

So, that actually is the base index, right? So, this is the index that we built in the earlier step. Then we have minus 1 and minus 2. If you are dealing with paired end data, we are giving input from these first few files. If you are dealing with unpaired data, we use this minus u option.

We do not use this interleaved option; we do not have that kind of data. And minus s is the output file format. So, this is the same format, ok? So, this is something we will discuss towards the end of this class, ok? So, again, there are three types of options that you can give.

One is input options, right? So, you can specify what you want to do with the input data. You can ask Bowtie2 to do something with the input data, or you can specify how the data is provided, what the quality score format is, etcetera. You also have run options. When you are running the code, what should you run, what kind of alignment should it do, and then finally output options?

What kind of output you want—whether you want certain outputs or not—you can specify using output options. So, what are we going to do? So, I introduced them in the last class. So, what I will just summarize is some of these because what we will do today is also start with using them. So, I will show you how you actually use them and what kind of effect you will see on the data analysis, ok? So, these are the input options; some of the input options that we have mentioned are minus 5 and minus 3, and then you can also specify this quality score format, right?

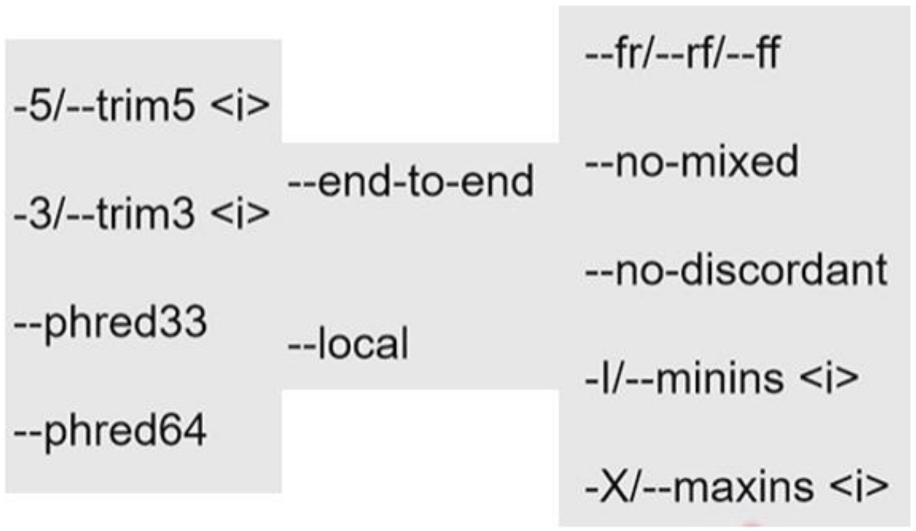
So, if you are dealing with old Illumina data, you will probably specify PHRED 64; otherwise, by default, it will take PHRED 33. In the run options, you have two options, right? So, end-to-end and local, and inside this end-to-end and local, you can also specify the speed and the accuracy, right? So, there is the speed versus accuracy trade-off. You can say you want a very sensitive or very accurate result.

So, you can say very sensitive, right, or you want very fast; you can also specify very fast. Coming to the output options, you have this: minus-minus FR minus-minus RF minus-minus FF. We discussed this in the last class. Minus-minus FR is useful for Illumina data. This is valid for Illumina data when you are dealing with paired-end data, right? So, these options mostly apply to paired-end data, not single-end data.

You also have this minus-minus no mixed option, right? So, this actually gives you or leaves out a certain type of alignment, which you may not require. You might say we do not require that kind of alignment. So, specifically, what minus-minus no mixed will do is leave out this single-end alignment, right? So, what happens in paired-end alignment? So, the program will try to map both, both reading 1 and reading 2, right? If it does not find any valid concordant or discordant alignment, what will it try to do? It will take these individual reads and try to align them separately, ok?

So, if you do not want that because this may not be useful, we can always specify minus-

minus no mixed. In some cases, we may not really require minus-minus no discordant. For example, we are doing some sort of genotyping, right? We are looking at genetic variation at certain loci or in a few regions of the genome, right? So, we might want to say we do not really need this discordant mapping.



```
-5/--trim5 <i>
-3/--trim3 <i>
--phred33
--phred64

--end-to-end
--local

--fr/--rf/--ff
--no-mixed
--no-discordant
-I/--minins <i>
-X/--maxins <i>
```

So, we can specify minus-minus no discordant, right? So, we discuss what is concordant mapping and discordant mapping in the last class. And then you can also mention the minimum fragment length and maximum fragment length using these options, minus i and minus x. So, we have not used them right in the run that we actually did in the last class, ok? So, what we are going to do is actually try some of these variations because they are useful, and you will see how the results change.

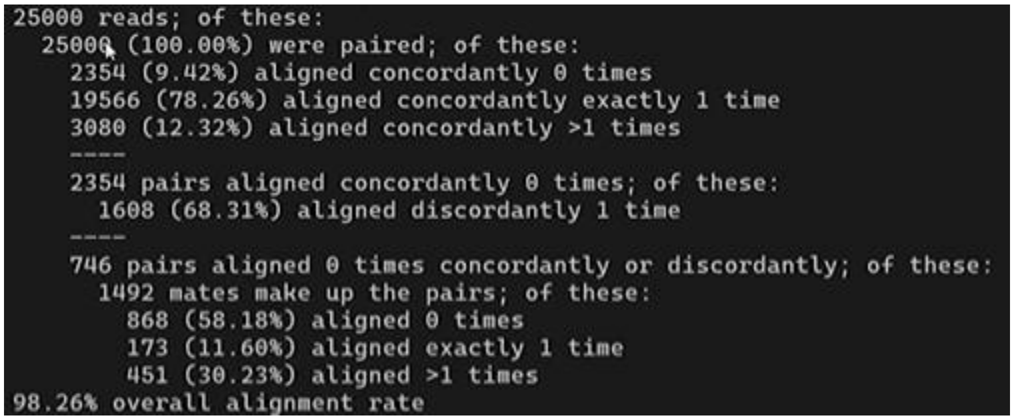
So, let us move on to the terminal, right, and see how these options will change the output. So, coming to the terminal, right, you remember this from the last class. So, this is where we actually left. We actually ran this bow tie to write using this command. We are calling this program bow tie to here. We are giving minus x, right?

This is the index file name, yeast_wg, right? We created this when we built the reference index using the bowtie2 build function. Then we are giving these input file names, right? These are minus 1, and minus 2 means this is paired-end data. So, that is why we are giving two input file names.

So, one is the iteration 1 read 1 trimmed dot fastq, and the minus 2 is the read 2 trimmed fastq, right? So, I mentioned why we need to provide trimmed data because this is pre-processed data, right? So, we have done some preprocessing; for example, we have done adapted trimming. So, this is the most appropriate file for the mapping, right? And then, followed by that, we have this minus s output dot sam.

This is the output file where the alignment results will be stored, ok? So, let us try a few options, and then we will see what the alignment file actually looks like. As I mentioned, this output file is quite complex. There is a lot of data in there, and we need to understand how this data is organized and what it means. So, if you want to process the SAM files for further analysis, you need to understand this format and the information that is stored inside.

So, let us try different options. So, once you run this, what you get is some sort of summary statistics, right? So, here we have. So, what is telling us is that there are 25,000 reads, and out of these 25,000, all of them were paired data, right? So, and then there are 2350 that cannot be matched concurrently.

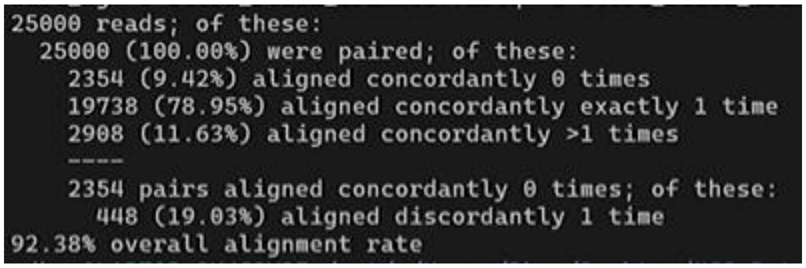


```
25000 reads; of these:
  25000 (100.00%) were paired; of these:
    2354 (9.42%) aligned concordantly 0 times
    19566 (78.26%) aligned concordantly exactly 1 time
    3080 (12.32%) aligned concordantly >1 times
    ----
    2354 pairs aligned concordantly 0 times; of these:
      1608 (68.31%) aligned discordantly 1 time
    ----
    746 pairs aligned 0 times concordantly or discordantly; of these:
      1492 mates make up the pairs; of these:
        868 (58.18%) aligned 0 times
        173 (11.60%) aligned exactly 1 time
        451 (30.23%) aligned >1 times
98.26% overall alignment rate
```

So, it was not mapped. There are about 19,500 reads that align concurrently exactly one time, and then you have 3000 reads that align concurrently more than one time, which means they are mapping to repeat sequences, right? There are multiple mappings. Then out of those 2354 that did not align concurrently, So, the program tried doing this discordant mapping and it found discordant mapping for 1608 reads, and the rest of them were aligned independently, right? So, that is what this minus-minus no mixed option is; if you give it, it will stop this alignment, ok?

So, let us try those first two commands: right minus-minus no mixed, and then minus-minus no discordant, ok? So, see what actually happens, right? So, we can run these commands, right? So, I will run the same command, but now with these options, ok? So, here, I am going to give these options.

You can see here, right my cursor is minus-minus no mixed, ok. So, here is this minus-minus no mixed, and the rest of the options remain the same, ok? So, the options come after the program, right? So, as you have seen now, in Linux, we run a program or a command in the terminal, right? Every time we do that, if you want to give some options, we have to give this after this command, ok?



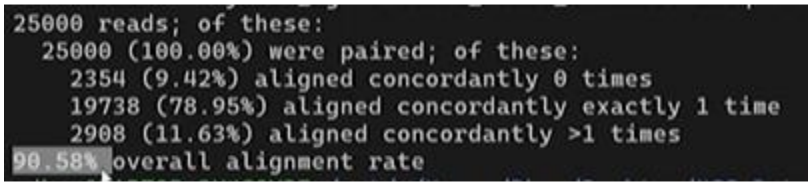
```
25000 reads; of these:
  25000 (100.00%) were paired; of these:
    2354 (9.42%) aligned concordantly 0 times
    19738 (78.95%) aligned concordantly exactly 1 time
    2908 (11.63%) aligned concordantly >1 times
    ----
    2354 pairs aligned concordantly 0 times; of these:
      448 (19.03%) aligned discordantly 1 time
92.38% overall alignment rate
```

So, this is what we are doing. So, we can just give this minus-minus no mixed and run, right press enter and we run and you will see the output statistics, right and you will see that statistics are a bit different, ok because we have switched off this mixed mapping means it will not try to align these reads individually, ok. And immediately, you probably notice that this overall alignment now drops from 98 percent to 92 percent, roughly. So, this is an effect because we have stopped some parts of the alignment, ok? So, this is actually closer to the actual mapping percentage, right? This single-end data may not always be useful, and individual mapping may not always be useful, ok?

And the rest of the statistics remain kind of the same; right, you have seen something like 25000 reads, right, 2350; they did not align concurrently. 1907–38, they aligned concurrently exactly one time, and then 2900 aligned concurrently more than one time. So, what is happening is that you will probably see a slight increase here in these numbers. What this means is that the program is trying to somehow align these concordant numbers.

It is reporting some concordant alignment which it was probably ignoring last time. So, this will change some of the results a little bit—ok, not a lot.

We can also try this minus-minus no discordant, right? We can simply add this to this command, right? We can have both minus-minus no mix as well as minus-minus no discordant, ok? So, let's try that, ok? And I have just written minus-minus no discordant, and then I will just press enter and it will run again and give you some statistics, right?



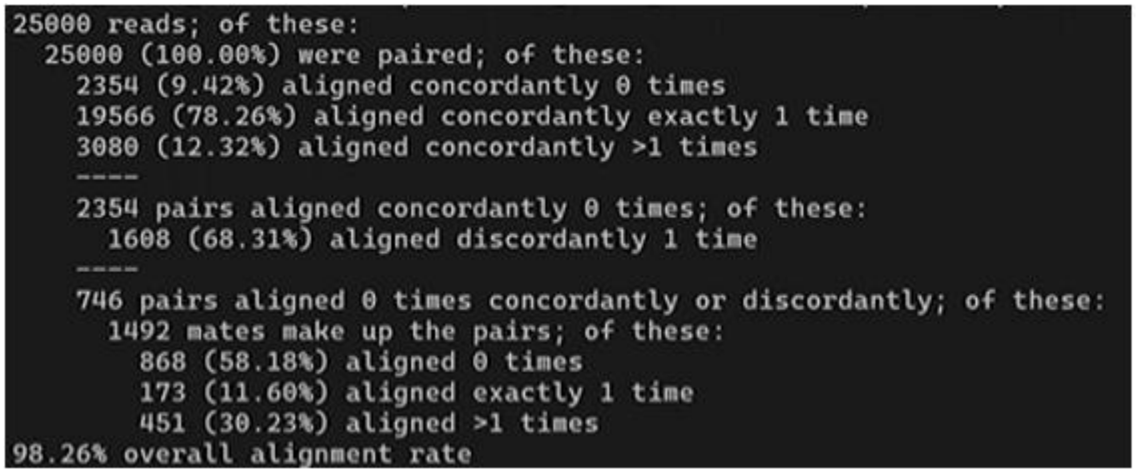
```
25000 reads; of these:
  25000 (100.00%) were paired; of these:
    2354 (9.42%) aligned concordantly 0 times
    19738 (78.95%) aligned concordantly exactly 1 time
    2908 (11.63%) aligned concordantly >1 times
90.58% overall alignment rate
```

And what do you expect? You will probably expect the alignment rate to go down further, right, because we are not allowing discordant mapping either. So, this is something that you will notice here immediately: this 90.58 percent overall alignment rate, right? So, which is lower than 92 and much lower than 98 percent, right?

So, this is the alignment that we get. So, mostly the concordant alignment, ok? And in many cases and in many applications, maybe we are just interested in this concordant mapping, and we really do not want or need that discordant mapping or mixed mapping, right? So, if that is not required, we can actually stop those, and that will make the program a bit faster, right? So, because it will not try all those things, all right? So, if this is clear, right now we can actually try other options.

So, these are mostly output options; right, what do you want in the output file, ok? What we now can do? We can actually also change the run parameters, ok? So, one of the run parameters that we often use is the local option, right? So, as I have mentioned, it makes a lot of sense to use the local alignment instead of the global or end-to-end alignment because in reads, especially in Illumina, we have seen the quality score being low in the beginning and dropping toward the end. So, if you are doing local alignment, the program can ignore some of these bases that are of bad quality, and they may not be matching very well with the reference sequence, ok?

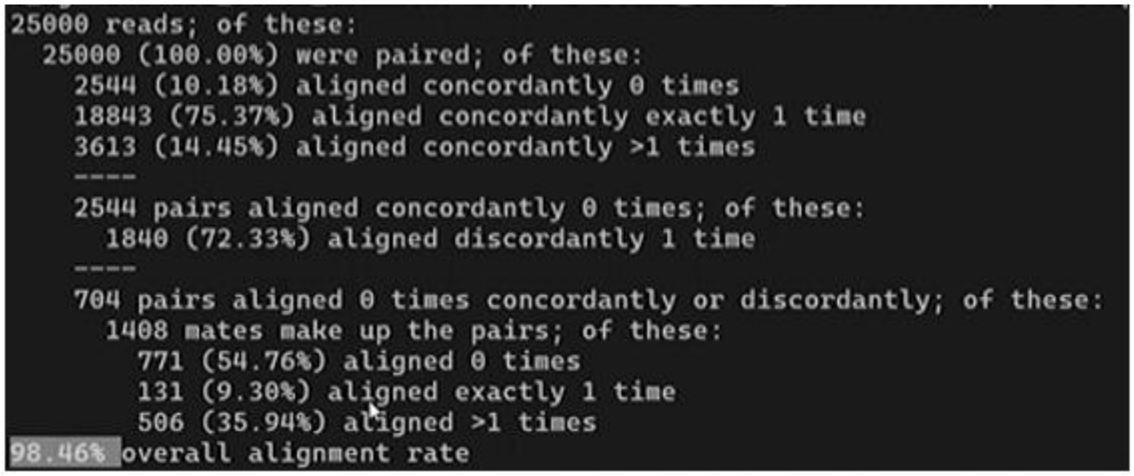
So, we will try that now, right? So, we will come back to that original alignment, right? So, let me clear the screen, ok, and we will come back to that original command, right, and I will run it again and here. So, the default is end-to-end, which is global. So, it will try to match the whole read against the reference sequence, even if the match is a bit poor, right?



```
25000 reads; of these:
  25000 (100.00%) were paired; of these:
    2354 (9.42%) aligned concordantly 0 times
    19566 (78.26%) aligned concordantly exactly 1 time
    3080 (12.32%) aligned concordantly >1 times
    ----
    2354 pairs aligned concordantly 0 times; of these:
      1608 (68.31%) aligned discordantly 1 time
    ----
    746 pairs aligned 0 times concordantly or discordantly; of these:
      1492 mates make up the pairs; of these:
        868 (58.18%) aligned 0 times
        173 (11.60%) aligned exactly 1 time
        451 (30.23%) aligned >1 times
98.26% overall alignment rate
```

So, all right. So, these are the results, right? So, you have this 98.26 percent overall alignment rate, ok? Now, what will we do? Again, give this option minus-minus local, ok? As you can see, I have just given this minus-minus local, right? The local option means I am asking the program to run in local alignment only, ok?

And I will run this. What you will notice is that the overall alignment rate will go up. It should go up because it will be able to now align more reads properly to the reference because it can now remove or maybe ignore certain bases at the end of the read. So, let us see, right? It will give the results in a moment or so, yeah. So, you will probably notice this increase in the overall alignment rate, right?



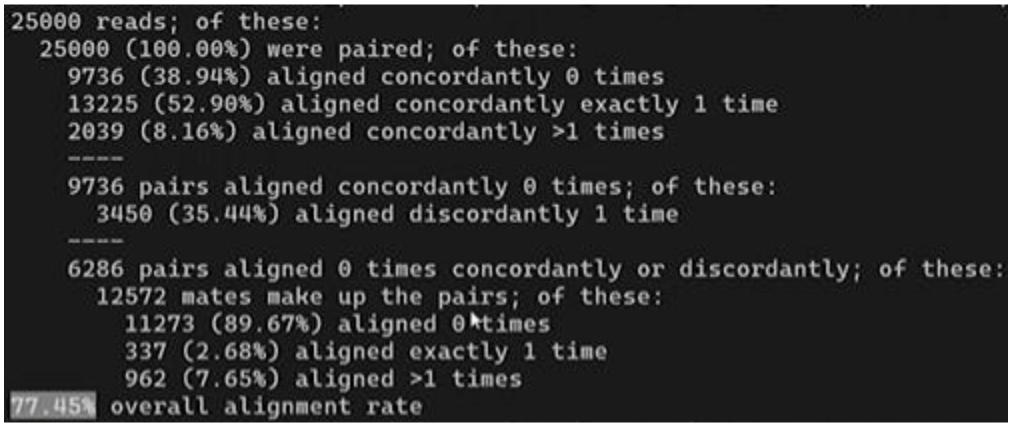
```
25000 reads; of these:
  25000 (100.00%) were paired; of these:
    2544 (10.18%) aligned concordantly 0 times
    18843 (75.37%) aligned concordantly exactly 1 time
    3613 (14.45%) aligned concordantly >1 times
    ----
    2544 pairs aligned concordantly 0 times; of these:
      1840 (72.33%) aligned discordantly 1 time
    ----
    704 pairs aligned 0 times concordantly or discordantly; of these:
      1408 mates make up the pairs; of these:
        771 (54.76%) aligned 0 times
        131 (9.30%) aligned exactly 1 time
        506 (35.94%) aligned >1 times
98.46% overall alignment rate
```

So, here are the results. You can see a slight increase, ok, 98.46 percent, ok. So, you might say this is not a lot, ok? And the reason is that we have already trimmed the data quite well, right? So, we have done this pre-processing step, and we have already trimmed the data, and we have seen that the quality score is pretty good, right, in the beginning, and in the end, it is not really bad for this data.

If you had this low-quality score at the end or at the beginning, right, we might see a better improvement in this alignment rate, ok? And this will be clear if we now try to run this using the original data where the data is not trimmed, ok? Because if the data is not trimmed, the program will have a hard time finding proper alignment for some of the reads, right, where the adapter is still there, ok? So, what do we do? We will demonstrate, right, how this will change. You will see a much more significant improvement if we are doing this alignment or mapping with the original data, right, without any preprocessing or trimming.

So, let us do that. So, to do that, what will I do? I am running this in normal mode, which is the end-to-end mode, the default mode. So, the only thing we need to change are the input files, ok? So, the name of the input files So, sorry, it is D12, right? These are the original files; right, read 1, and then we have read 2, ok.

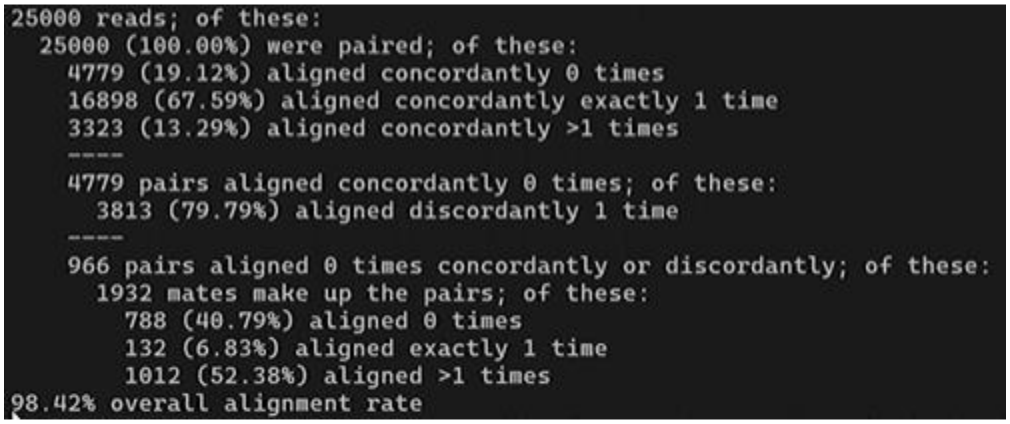
So, I have given these original file names now, and we are running this port on these original files. So, let us run and see what the output is, ok? And of course, what we will expect compared to the earlier scenario is that the alignment rate will be much lower because you have these reads that are not trimmed. So, the program will not be able to find proper alignment, ok? And immediately you can see that the alignment rate drops to 77 percent, right?



```
25000 reads; of these:
  25000 (100.00%) were paired; of these:
    9736 (38.94%) aligned concordantly 0 times
    13225 (52.90%) aligned concordantly exactly 1 time
    2039 (8.16%) aligned concordantly >1 times
    ----
    9736 pairs aligned concordantly 0 times; of these:
      3450 (35.44%) aligned discordantly 1 time
    ----
    6286 pairs aligned 0 times concordantly or discordantly; of these:
      12572 mates make up the pairs; of these:
        11273 (89.67%) aligned 0 times
        337 (2.68%) aligned exactly 1 time
        962 (7.65%) aligned >1 times
77.45% overall alignment rate
```

So, from 98 percent, we have dropped to 77 percent because you have these adapters at the end and the program is trying to use them for alignment, but it is not finding a good match in the reference sequence, right? So, that is why this problem is, ok, all right. So, what else will we do now? Let us try this in local mode and see if there is any difference or if there is an impact on this alignment rate. So, use this option minus local, right? And let us run this and see the output; there should be a significant improvement, ok?

And that kind of will tell you, right, if your pre-processing step is not very effective and there are some issues, or maybe you have not noticed certain issues and probably did not take any action, then you will probably see a good improvement in your alignment rate, ok? So, let us see, right, it is taking a bit of time because it is running in the local mode and you have all the options open, right, the mixed alignment as well as the discordant, and immediately you see there is a huge jump in the alignment rate, ok? You can see here that the alignment rate now is 98.4 percent compared to the earlier case, right?



```
25000 reads; of these:
  25000 (100.00%) were paired; of these:
    4779 (19.12%) aligned concordantly 0 times
    16898 (67.59%) aligned concordantly exactly 1 time
    3323 (13.29%) aligned concordantly >1 times
    ----
    4779 pairs aligned concordantly 0 times; of these:
      3813 (79.79%) aligned discordantly 1 time
    ----
    966 pairs aligned 0 times concordantly or discordantly; of these:
      1932 mates make up the pairs; of these:
        788 (40.79%) aligned 0 times
        132 (6.83%) aligned exactly 1 time
        1012 (52.38%) aligned >1 times
98.42% overall alignment rate
```

So, it is very comparable to that, right? So, compared to 77 percent, you have reached 98 percent; this is comparable to the run using the pre-processed data, ok? So, maybe the statistics are slightly different, etcetera. This will happen because the program will not run exactly in the same manner because you have these adapters, etcetera, but still, we now see a pretty good alignment rate, ok? So, this kind of tells you, right, probably it makes sense to run the program in the minus-minus local mode because in case there is some sort of quality issues remaining with the data and you have not done any pre-processing for those minus-minus local option, it can still take care of those issues, and it can actually map reads, and you know those quality issues or maybe the contamination issues, ok, alright.

So, I think we can now go back and look at the output file, ok? So, the output file is the SAM file output dot SAM that has been generated, ok, and we will start with that file and we will start looking at what you have and what kind of information you have inside those files, ok. So, I will first go back to the presentation. I will just mention some of the terms because I do not want to show the file immediately because you might get a bit confused with all the information in there. So, let us go back, and I will introduce this SAM and BAM format, and then we can come back and see what you have in the actual data, ok? So, the output file that you have seen is generated in SAM format, ok, and you also sometimes get this in BAM format, ok.

So, the SAM stands for sequence alignment map, and this is a human-readable format. So, we can read this, we can open this file, and we can read this, which we will do a bit later, ok? But corresponding to the SAM file, you also have a BAM file, right? You can generate this BAM file from a SAM file, and this is a binary alignment map, which means the machine can read it. So, this is a machine-readable format that we can supply to the machine, but we are not able to read this file.

So, if you want to read this file, if you want to write a code to process the alignment results and to identify, for example, genetic variants, etcetera, you will go with the sequence alignment map, ok? So, the SAM file and we will see what the fields are there, ok? So, let us discuss the SAM format and what it contains, because this is the file that we will read. And in the SAM file, you have two sections. The first section is the header section, which you will see at the top of the file, and this header section is optional. So, in some cases, in some files, you may not see this header section at all, and you can also remove this header section yourself, ok?

By default, Bowtie 2 will generate this header section in the SAM file, but you can switch off this header section. We will see when you actually come to the alignment section. We want to see the result from the alignment section, so we can switch off the header section. And the second part is the alignment section, which actually contains the actual alignment or mapping results. So, the header section kind of describes the reference sequence, what program was used, etcetera. All this information is there in the reads, etcetera.

The alignment section contains the actual results for each read. So, it will have the information where the read mapped, which location, which chromosome for example, and also the quality of the mapping, whether it found some mismatches, some mutations, some insertions, some deletions, etcetera. All that information is there. In addition, it will also show whether you have concordant mapping or discarded mapping if you are working with paired end data. So, all sorts of information are there, and we will go over those in the next class, ok? So, let us look at the header section to see what it means, what kind of information it has, and how you interpret this data.

So, the header section starts with the read sign, ok? So, it is like a FASTQ format, right? You have this header section that will start with the read, as we will see in the actual file and we will see that you have this at the read sign, ok? And then you have the SAM format version that is specified using the HD vn tag, ok?

vn means version. So, at hd. So, hd means header, right? It is part of the header, and then you have vn, which means version, ok? So, in this SAM format, there is an upgrade, right from some version to another, and then you need to know which version you are working with because there might be certain changes here and there, ok with the information that is present, ok. So, that is why this version of information will be there. Now, you can have multiple header lines, not just one, and each header line will also start with the read sign.

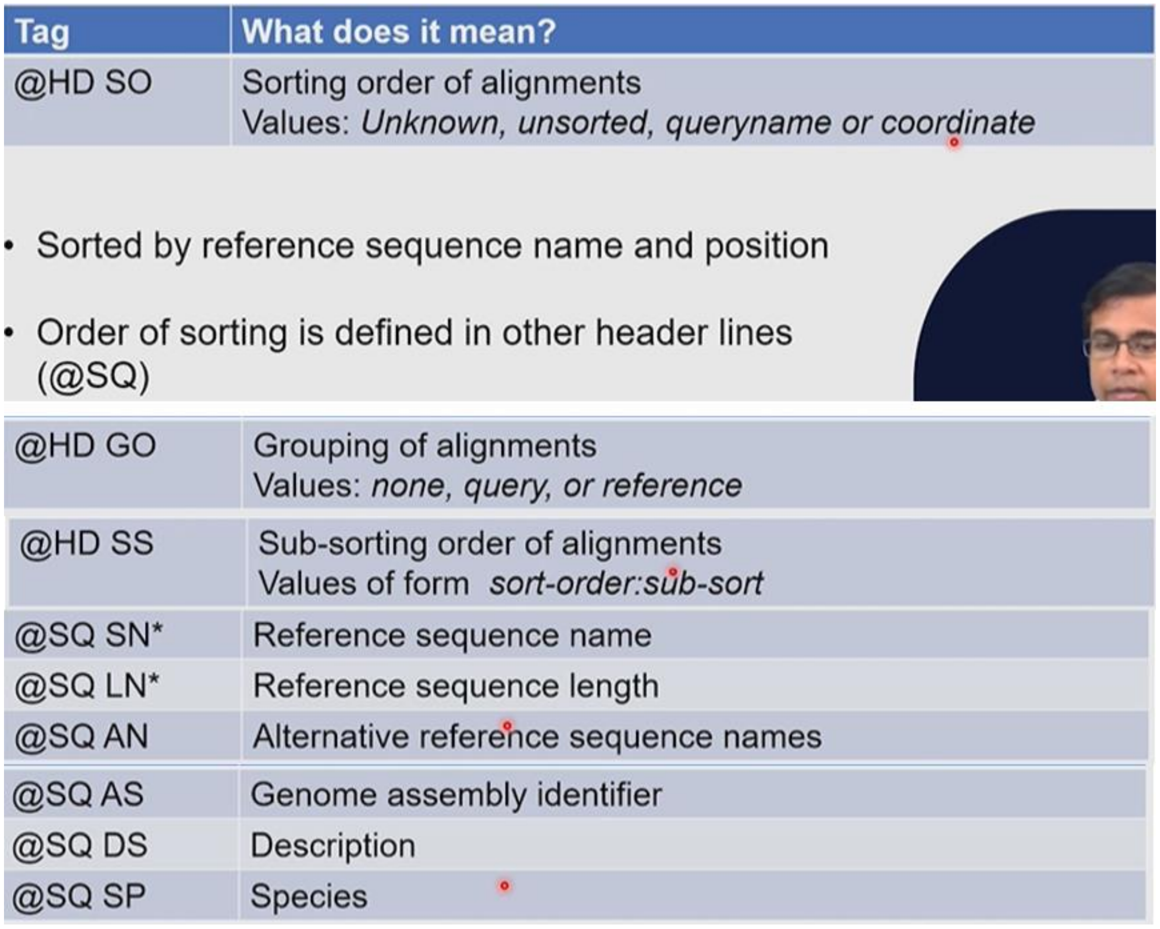
So, it will convey some information, but everything should start with this at the beginning, ok? Because then, if you are writing a program, you can identify this as the header line, and maybe some information you are looking for you can also parse or gather from the header, ok?

Now, at the read sign, this is followed by a two-letter code to denote what kind of record you are looking at. So, there are specific notations. So, we say at the read-hd, so, right? So, at the read hd means header, right, and so signifies you are using sorting order of alignment. So, you are looking at how the alignments are sorted in the file or whether they are unsorted, ok?

So, these are the values you will get: unknown or unsorted. So, unsorted is the default option; you will see this in most of the files, and then you have sorting by query name, right by the read name, or sorting by the alignment sequence position, right. So, you can have this sorting by reference sequence name and position and this order of sorting will be defined in another header line. Okay, that will start with the read-s cube. So, we will come to what this is at the read-s cube, ok? So, you have at the read hd So, then you also have to understand how the alignments are grouped. So, this is again specific; you can be specific that you can have grouping by none; right, there is no grouping of alignments; their alignments are grouped by query name, right where the name of the read is.

So, the query is the name of the read, and then you can also group by the reference, ok? So, if, let us say, some reads are aligning to chromosome 1, you probably want to group them based on the position or the reference genome where they actually align. So, chromosome 1 reads mapping to chromosome 1, and they will be grouped together. You can have reads mapping to other chromosomes grouped together as well, ok? So, this is not necessarily sorted, but you can have this grouping by query name or grouping by reference to the mapping location, ok? So, you can also have the so; beyond the SO tag, which is the sorting order, you can also have something called a sub-sorting order, right?

So, this is how it will be given: right sort, order, colon, subsort. Now, what is this sub-sorting? So, if you have sorted these alignments based on something like, for example, you have sorted based on reference position right where they or the coordinate system are, right where they actually map, you can do a sub-sorting after the first sorting using the query name, for example, right. So, you can have two levels of sorting, right? So, you can first sort by coordinate or the position where it is aligned, followed by the query name, ok? And this sort order is the same; the sort order value will be the same as whatever the value is in the HDSO tag, ok?



Tag	What does it mean?
@HD SO	Sorting order of alignments Values: *Unknown, unsorted, queryname or coordinate*

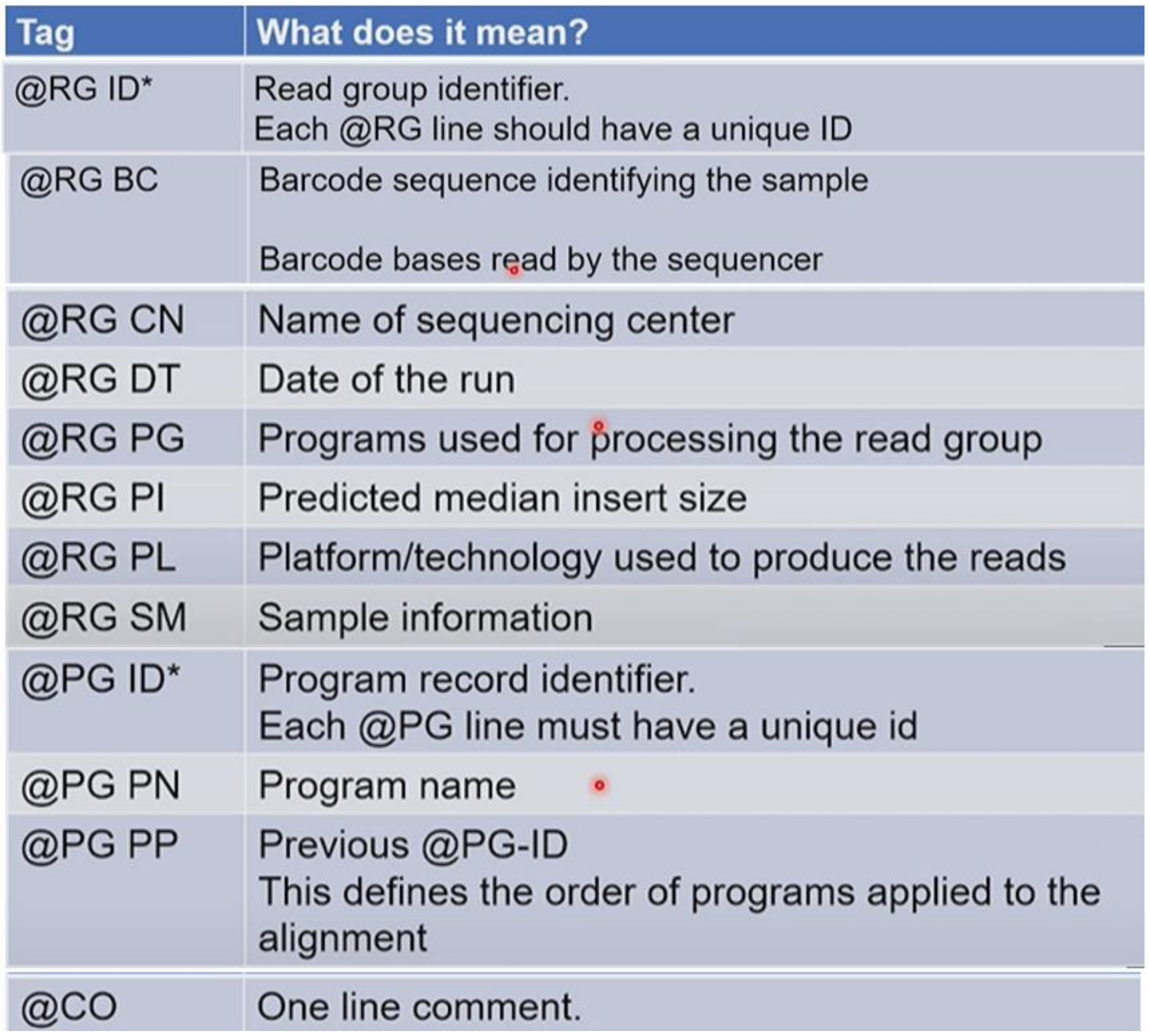
- Sorted by reference sequence name and position
- Order of sorting is defined in other header lines (@SQ)

@HD GO	Grouping of alignments Values: *none, query, or reference*
@HD SS	Sub-sorting order of alignments Values of form *sort-order:sub-sort*
@SQ SN*	Reference sequence name
@SQ LN*	Reference sequence length
@SQ AN	Alternative reference sequence names
@SQ AS	Genome assembly identifier
@SQ DS	Description
@SQ SP	Species

Then you have something called SQ, ok? So, this will give you the alignment sorting order, depending on how the alignment is sorted. So, that is given in this @ SQ. @ SQ, this will be the reference sequence name. So, a star means this information will be there by default, and then you have a cube in the star; this is the reference sequence length.

So, reference sequence: name the genome that you are working with. So, this information will be there. You can also have @SQ a n, which is an alternative reference sequence name, right? So, some reference sequences may not have another alternative name that is also given here. You can also have other fields, such as cube AS DS SP.

So, you can have a genome assembly identifier that indicates which assembly this program is using. There is some description of this reference, and you can also have the species name, right? Then you have something called the rate RG which is the read group. This is usually an unordered number of lines, and each line will also have an identifier called the read group identifier, and each RG line should have a unique ID. So, these fields you will not see very often, but in case you do, you know how to interpret them, right? Of course, you cannot remember all the details, but you now understand how to interpret these fields if you see them in your SAM file.



Tag	What does it mean?
@RG ID*	Read group identifier. Each @RG line should have a unique ID
@RG BC	Barcode sequence identifying the sample Barcode bases read by the sequencer
@RG CN	Name of sequencing center
@RG DT	Date of the run
@RG PG	Programs used for processing the read group
@RG PI	Predicted median insert size
@RG PL	Platform/technology used to produce the reads
@RG SM	Sample information
@PG ID*	Program record identifier. Each @PG line must have a unique id
@PG PN	Program name
@PG PP	Previous @PG-ID This defines the order of programs applied to the alignment
@CO	One line comment.

Then you have @ of RG BC. So, this is actually the barcode sequence that is used by the sequencer to identify the sample, and then you also have some details about the sequencing center name and some details about the run date. Programs that were used for processing the read group, right, and you have predicted media and insert size, right size, etcetera, and platform that was used some sample information, etcetera, then you have another type of tag that you can get, which is called PG.

This is a program, right? So, which program actually generated this SAM file? So, this is important, right? So, for example, in the case of bow tie 2, you should see bow tie or bow tie2 under this value, right? So, PG ID means there will be this identifier that will tell you, OK, this is the program that was used. The program name will be there if you have some other name, and then you can also have something called PG PP, right?

So, this kind of defines the previous program that was used to generate this. So, you can kind of define a chain of these programs that have been used on this data before you got this SAM file, ok? So, that is useful information when you are processing the data. You also get this program version, which is called the PG PM.

And then, finally, you can get some comment lines. So, they are given this @ of C, right? So, these are one-line comments, and you can have multiple comment lines describing something or some sort of experimental detail or the data processing details, etcetera, ok? So, let us have a look at the data very briefly in the header part, and you will probably see some of these values that we have just described, ok? So, let us move to the terminal, right, and we will see this header part, ok? So, how do you check the header part of this output file? So, the output file is output dot SAM.

So, let us check the header file, and then you can use this colon-C no-wrap. So, you can remove this wrapping, and you can identify this header part here, right? This will start with this @ sign, ok? So, up to this point, you have the header, ok?

So, @ of HD, you are using version 1.5 of SAM, right? The first line tells you VN, right? Then you have SO; this is unsorted; their alignment is not sorted; and then you have the query. And then you have this @ of SQ, ok?

So, this is the reference that is there. So, here you can see some reference numbers. So, these are actually chromosome numbers, right? So, this information is there, as is the length of each of these reference sequences. So, if you check the reference file, the reference file genome data is actually organized in this way. So, that is why the program is reporting these references, ok?

And then finally, you have this @ of PG, which is what gives you the program, right? So, this is @ of PG; you have the bowtie 2, right? The program name and ID are bowtie2, and the program version is 2.

5.1. And then finally, it gives you the common line that was used to actually get this alignment. Get this map, ok? So, this is what the header line looks like. Beyond this you have the alignment section, which we will describe in the next class. So, we can quit now and go back here, right? So, these are the references that we have used in this class.

And finally, just to summarize, the bow tie 2 options can be useful. So, in the first part of this class we have seen, right, we can use these bow tie 2 options, and they can be useful for obtaining better alignment results. And bow tie 2 alignment results are stored in the specific formats, right? So, SAM and BAM, where SAM is the human-readable format and BAM is the machine-readable format. And we have also looked into the interpretation of the SAM format header lines.

So, there could be some useful information that you might need when you are processing the data. And in the next class, we will be describing the rest of the parts of the SAM format files, specifically the alignment section. And thank you.