

Next Generation Sequencing Technologies: Data Analysis and Applications

Mapping reads with Bowtie2

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. We have been discussing the mapping process, and in the last class we introduced the Bowtie2 tool. So, this is a tool that utilizes the Burrows Wheeler transform with FM index and LF mapping to map reads. So, what we have discussed so far is: how do we actually utilize this tool, how do you set this tool up, and how do you actually build the index? So, in this class, what we will be doing is actually doing the read mapping using the index that we built up in the last class, ok? So, just to remind you, right, we have set up Bowtie 2. In the last class, we downloaded and extracted the tool, and it was a very simple process. It does not require a very complex installation. You download, extract, and check for permission, right? If it is required, you can change the permissions using the chmod command, right? So, once this is done, we can do the genome indexing, right, using this function called bowtie2-build, and we actually utilize that function in the last class, right, to build the index for the *Saccharomyces cerevisiae* genome. So, we downloaded the reference genome from NCBI RefSeq for *Saccharomyces cerevisiae* because we are dealing with data from this organism, ok. And during this build function, what is happening is that Burrows wheeler transform is happening, followed by the creation of this FM index, double indexing, and checkpoints, right?

Bowtie2

- Setting up Bowtie2
- Building genome index with bowtie2-build function
- BWT, FM Index, Double Indexing, Checkpoints
- Options while indexing

So, you actually have seen, right, this double indexing part because we have this forward index. Of course, there is no name for the forward index. These are like 1, 2, 3, 4, and then you also have the reverse indexes, right? We have seen those. And also, we talked about the options when you are running Bowtie2 build. You have certain options that you can utilize if required, right? If you want to speed up the process, if you are building this index for a really big genome, right, the human genome, you probably want to speed up. You want to give this minus minus threads option, and if your system permits, if you are working on a server, you can use this minus minus threads option, ah, to speed up the process. o, as we have discussed up to this point, what we are going to do today is, ah, actually do the read mapping with Bowtie 2, ok. So, there is this function, Bowtie2, that will do the read mapping. So, in that way, it is a very simple process, but of course, there are a few other things that we need to know about when we actually do the mapping. Those, ah, those, ah, parameters could be important during the mapping process, ok? So, let us first look at the command to see how we will actually run this to map reads against the reference genome. So, this is how we will specify the command, ok? So, the command is Bowtie 2, right? There is no build

or anything; it is just Bowtie 2. Then you have some sort of option, and we will come to these options, ah, later. Unlike Bowtie2 build options, these options are very useful and, in some cases, critical. So, we may have to specify certain options to get better-quality results. Then you have something called minus x bt2-idx, ok? So, what is this minus x bt2idx? It actually specifies the index file, ok? You remember we gave the index file name when we were building the genome index, right? So, we mentioned yeast underscore wg, right? Here you would have to mention this minus x yeast underscore wg that base file name, ok? We do not have to mention yeast underscore, wg, dot, dot, bt2, etcetera. We just simply give this yeast underscore wg and it will automatically take, right, all the index files, ok, in the processing, ok. So, if the system knows, right, this program and this command will know, right, ok, these are the BT2 files. It will search for them, and it will take up those files, ah, for processing.

Then you have, ah, some commands within these brackets, ok? So, this is actually specifying the input files. Okay, the first is the fastq files, right? So, you and I are working with fastq files here. So, this will be fastq files, ok? And you have three options.

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
-q	Reads in <m1>, <m2>, <r> are .fastq files



So, these three options are separated by these, ah, separated signs, right? So, you can have this minus1 m1 minus2 m2 option or minus u, ah, option or minus minus interleaved option, ok. So, these, ah, characters that are between this less than and greater than sign mean that here you would have to specify a certain value, right? This is where you should give your values, right? So, for example, here is also that bt2-idx. You see this, ah, greater than or less than sign.

This means here you will give your file name, ok? So, this is a way to actually specify where you will have to specify file names, etcetera, ok? So, what does this mean? Actually, minus 1 minus 2 will come in a moment. And then you have something called minus S, ah, and then sam means this is the, ah, some sort of file that you have to specify again, ok. So, as I have mentioned, minus x bt2 idx is the base name for the index file for the reference genome.

So, in our case, the base name was yeast underscore wg, ok. So, that will be specified here. Then you have something called minus 1 minus 2. So, minus 1 m1 minus 2 m2—actually, ah, this minus 1 minus 2—they are used together, and they are used only when we have paired end data, ok? So, if you are working with paired end data, you would have to specify minus 1 as the first file name, the first fastq file, and then followed by minus 2 as the second fastq file. So, you will have to let us say minus 1 yeast read 1 dot fastq, followed by minus 2 yeast read 2 dot fastq, ok? So, this is how we will specify paired N data, ok? So, this is the way the program will know, ok? We are dealing with paired n data, right, and try to align them, ah, in that, ah, way, right for paired end data specifically. If you are working with unpaired data, you will simply mention minus u, right, ah, and the file name, ok.

So, minus u yeast dot first q, right? So, that will be, ah, the format if you are working with unpaired data; that would be the input final, ok? Now, there is an interesting format, which is the minus minus interleaved, right? So, this does not happen in Illumina, right? So, where you would have a file that will contain read 1 read 2 interleaved means 1 after another.

So, you have let us say in the file that you have read 1 sequence followed by 2 sequences. Then you have another read, ah, which is read 1, then the read 2 sequence for that corresponding read, ok. So, this is the interleave format, but this is something that we do not deal with because we do not see it in Illumina, right? So, we do not have to use this, ah, option. Then you have the minus s or sam; this is actually the output file, ok? This output file will contain the mapping results or the alignment results, ok? And this file comes in sam format, dot sam, ok? So, you can say simply output dot sam or east output dot sam or something like that, right? So, that name should be given. So, ah, the program will store all the results in that file, ok? So, you might ask, like, what is this

sam format, etcetera. We will discuss this in subsequent classes, right? Why is the data stored in SAM format? What kind of information do we get? Ah, that would be for later classes, ok? So, once this is clear, So, you can, ah, ah, now also have this option minus q means reads in m 1 m 2 or r r dot fastq file.

So, this is kind of a default; this is kind of understood; you do not actually have to mention this minus q, ok? So, what are the options here? This is actually important now. So, these options will come, right, and, ah, this can be classified into three different classes. So, you can have input options, you can have run options, and you can have output options, right? So, for input options, you are specifying to the program, right, how we should deal with the input, ok? So, with the input files that you have mentioned, how should the program deal with those inputs? So, that will be the input options. The run options would be how the program should do the mapping, whether there are specific parameters you should keep in mind, right, and what specific kinds of choices you should keep in mind, right, that will come under run options. And then, finally, you have output options, right? So, these are, ah, options that will specify, right, what should be written on the final SAM file, ok? So, you can specify, ok, do not write this, do not write that, etcetera, or do not map this, etcetera, right? So, those kinds of options would be the output options, ok? So, let us look at the input options first, ok? So, here are the options, ok? So, the options would be specified by this minus 5, ok? So, you run BOWTIE 2, then say minus 5, ok? So, you have seen this throughout the Linux command line: we have a command, and if you want to specify certain options for that command, we use this minus something, right? So, that is, ah, the common, ah, notation.

Bowtie2 – Input Options



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


Command	What does it mean?
-5/--trim5 <int>	Trim <int> bases from 5' (left) end of each read before alignment (default: 0).
-3/--trim3 <int>	Trim <int> bases from 3' (right) end of each read before alignment (default: 0).

Here also is, ah, what we see is either minus 5 or you can also say minus minus trim 5 and you have to specify an integer number, right? So, what this minus 5 or minus minus trim 5 means is

that you trim a certain number of bases. So, this certain number will be specified by you, right? You can say trim, ah, 5 bases, right? So, this integer could be 5. Trim 5 bases from the 5 prime ends of each read before you map, ok? Why, why would you do this? Because sometimes in your data you might see, ok, the first few bases are of very poor quality, right, maybe 5, ah, 10 or 4, etcetera. If you see that kind of, ah, ah, through during your quality control, right, ah, you want to mention this, right, you want to do this, right, you want to say, ok, trim this, ah, 5 prime end, 5 bases in the 5 prime end and do not use them for alignment, ok.

The default value is 0, right. The program does not take any other, ah, it does not do any trimming, ah, it will just set this to 0, but if you want to do this, you have to mention, ok, minus 5 or minus minus trim 5, followed by the number of bases that you want to trim, ok. And similarly, you have another option, which is minus 3 or minus minus trim 3, ok, followed by this integer number, and it will, ah, do the trimming from the 3 prime end or the right side of the ring, ok, before it does the alignment, ok. So, again, the default is 0. Then you also have another option called, ah, minus minus trim to 3, 3 prime, or 5 prime, followed by an integer number. So, what it does is that it will trim reads exceeding certain number of bases, ok.

Bowtie2 – Input Options



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

Command	What does it mean?
<code>--trim-to [3: 5:]<int></code>	Trim reads exceeding <int> bases. Bases will be trimmed from either the 3' (right) or 5' (left) end of the read. If not specified, bowtie 2 will default to trimming from the 3' (right) end of the read. <code>--trim-to</code> and <code>-3/-5</code> are mutually exclusive.

And you can also do something, right, you can, so once you, ah, mention this number of bases, right, so bases can be trimmed from either 3 prime end or 5 prime end, but they will not be done from both, right. So, you have to mention which side you want to do, right, whether 3 prime end or 5 prime end. Ah, by default, if you do not specify what I do, I will trim from the 3 prime ends on the right side, ok. So, again, ah, this is a different option, right? So, if that means that it exceeds a certain number of bases you want to trim, right? Again, probably you know that beyond, let us say, 40 bases or 50 bases, there is no point in reading the data because the quality is very poor. If

you are working with such a data set, you want to, let us say, trim beyond 50 bases, right? So, so, so when you are running this part I 2, you can specify, ok, do not align, ah, beyond these 40 bases for mapping, right? So, this is, ah, something that you can also mention, ok? And another point you need to know is that trim 2 and minus 3 minus 5 that we have just discussed are mutually exclusive. We cannot, ah, mention both, right? So, you cannot have minus 3 as well as minus minus trim 2, right? So, you cannot combine multiple options among these two, right? Otherwise, you can actually combine multiple options, and we will see, right, how we actually do that, ok? Then there are two other options: input options, which are also very useful, right? So, the earlier options were related to trimming reads, right, for mapping, ah, while while mapping the reads, Bowtie 2 will just trim the ends and then do the mapping.

Bowtie2 – Input Options



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

Command	What does it mean?
--phred33	Base qualities are in "Phred+33" encoding
--phred64	Base qualities are in "Phred+64" encoding

Here these two, ah, options, ah, ah, they relate to the quality of the bases, right? So, when Bowtie 2 is aligning, right, it will read the bases along with the quality score, and the quality score can be in FRET 33 format or in FRET 64 format, right? So, we have talked about the Phred Plus 33 encoding and the Phred Plus 64 encoding, right? So, for the older illuminator, they utilized Phred Plus 64 encoding. So, if you are working with an old illuminator, you want to mention this in the option, right? So, ah, if you are working with new data, it will be Phred Plus 33, and this is the default option now, ok? So, there are other options now, which are the run options. We have talked about what the run options are, right? So, what do we mean by that? Is that how you actually do the mapping or the alignment? So, this will come under the run option. So, what you have seen now is the input options; they talk about the input, right, the quality, whether their quality is scored or the sequence trimming; they all talk about input. Here in the run options, it talks about the mapping process, ok? If there is a specific way that it should do the mapping, or there should be, there are certain, ah, parameters that need to be specified for the mapping process those will be done here in these options, ok? So, there are two useful commands that we can use.

So, one is called end-to-end mode, ok? So, this actually means the full read will be aligned against the reference sequence, ok? So, ah, you have perhaps learned about global alignment, right? So, where the two sequence if you are aligning against each other, right, you look for full alignment, right, from one end to the other end, ok. So, you do not leave out any part of the sequences, ok? So, it is similar here that the idea is very similar; you align from one end of the read to the other end, ok? So, this is the end-to-end mode, which is very similar to the global alignment. Inside this end-to-end mode, you can also say very fast, right, minus minus very fast, minus minus fast, minus minus sensitive, or minus very sensitive, ok. What does it mean, ah, actually? So, very fast means you want very, very fast mapping, because you are dealing with a huge amount of data. So, you want to do this very quickly, but in the process, you are willing to sacrifice some accuracy. So, one of the things you should remember is that when you do this read mapping with BWT-based map methods such as Bowtie II, the backtracking process, or mapping reads with mismatches, that may not always give you the best results because of certain, ah, ah, features of the algorithms, or certain features of the implementation.

Bowtie2 – Run Options

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

Command	What does it mean?
--local mode	Not entire read is required to align (soft clipping/trimming)
--very-fast-local	
--fast-local	
--sensitive-local	(default in --local mode)
--very-sensitive-local	


If you are mapping reads, ah, with mismatches, it might not give you the best results, right? So, unless you probably run this multiple times, you will not get the best results, right? You have to exhaustively search all the alignments; only then might you end up with the best results, right? But this exhaustive searching or exhaustive mapping is very time-consuming. So, you often ignore that, right? So, there might be some inaccuracy, right? So, the bowtie 2 may not always find the

best match in case you have reads with mismatches, ok? So, here, what you are saying is that, ah, we can sacrifice certain accuracy, right? We do not really need absolutely, ah, the exact, ah, or the absolutely best match. We can sacrifice certain accuracy, but we want speed, ok? So, you are sacrificing accuracy, ah, for speed. You can then have these different levels, right, where you have this balance between, ah, speed and accuracy, right? So, if you come, come down to fast, minus minus fast, it means you want slightly more accuracy, right, and, ah, you actually can, ah, sacrifice certain speed, ok. And then you have sensitive and very sensitive. So, very sensitive is the slowest one. So, it will actually do the exhaustive searching and try to find the best match for all reads, and this will be the slowest, but it will give you the most accurate results, right? So, if you are working with a small data set with very accurate results, then you probably run with this very sensitive option. The default option is management-sensitive, right? So, if you are running this in end-to-end mode, this is the default option, and this actually works quite well. So, as you will see, once you have, ah, kind of run a lot of these, ah, or utilized a lot of read data for alignment, you will see this option is actually good, good enough for us. You can also run in local mode, which is actually in contrast to the end-to-end mode, right? So, local mode means you are doing something called local alignment, right? Again, this is very similar to local element, which means when you are aligning two sequences, local element only looks for matches in certain parts. You will not try to do the end-to-end mapping, right? So, it might, ah, be a kind of trim or something called soft clipping here.

It might leave out a certain basis when it is aligned, right? So, it will not align the full read, and in some cases, this is very useful, right? Why? Because, as you have seen in Illumina, especially, your quality score suffers at the ends, right? So, on the first few and last few basis, the quality score is low. So, if you are doing this end-to-end mode, sometimes what will happen is that the program will force, ah, this alignment against the reference sequence, which might not be the best alignment, right because of the sequencing errors, etcetera, in the ends of this, ok. So, if you are running in local mode, what the aligner will do is kind of soft-clip those ends, and it will find the best match, right? It will not try to forcibly match everything against the reference sequence, ok? And so, in that case, it will probably find better matches in the reference sequence, ok? So, in some cases, this minus-minus local mode is very useful and we use this option. Again, as in end-to-end mode, we have different options. You can say very fast local, ah, fast local, sensitive local, and

very sensitive local, right? Again, there is the speed and accuracy trade-off. So, very fast would be, as the name suggests, right? It will be quick, but it will sacrifice some accuracy. Whereas a very sensitive local is at the other end of the spectrum, right? So, it is very slow compared to a very fast option. So, it is the slowest option of all, but it will give you the most accurate results. Again, the default is this minus-minus-sensitive local, which actually works very well for most of our applications, ok.


Bowtie2 – Output Options



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

Command	What does it mean?
<code>--fr/--rf/--ff</code>	The upstream/downstream mate orientations for a valid paired-end alignment against the forward reference strand. E.g., if <code>--fr</code> is specified and there is a candidate paired-end alignment where mate 1 appears upstream of the reverse complement of mate 2 Default: <code>--fr</code>

Also, if mate 2 appears upstream of the reverse complement of mate 1, that is also valid



So, there are other options that we also use; these are called output options, ok. So, these are also useful in some cases. For example, if you are aligning paired end data, you want to specify what you will consider a valid paired end alignment. So, these are the options; what you can mention is something called minus minus fr, minus minus rt, or minus minus ff. So, these are the three options that you have. So, what do they mean is that they actually specify the upstream downstream orientation of the read 1 and read 2 for a paired end data, ok? So, for example, if you specify minus minus fr, ok? So, if you see, let us say, a paired end alignment, ok, and the mate 1 appears upstream of the reverse complement of the mate 2, right? So, you have mate 2 here, right, and then you see this mate 1 AH in the upstream region and the alignment of mate 2 is in the reverse complement, ok? So, you see this reverse complement. So, this will be a valid alignment, right? So, the default is minus minus minus. As we see, we expect the mate 1 to align right in the forward direction, and you see the reverse complement of the mate 2. The other option can happen; other options can

happen as well, right? So, mate 2 can appear upstream, right, but in the forward direction, and we see the reverse complement of mate 1 downstream, right. That is also a valid option for paired end alignment when you have minus and minus options specified. You have the other options, right? You can see that you can think about the opposite ah 1, right minus minus rf, which is the reverse complement of mate 1 ah, which is upstream of the forward sequence of mate 2. You can also specify minus minus if you expect to see both in the forward direction in some regions, right? So, for Illumina, it is mostly minus minus minus, which makes sense, ok? So, we will not touch this option, but it is good to know that you have these variations.

Then you have some of the options that you can use, right? So, one of the things that will happen when you are aligning paired end data in BowTie 2 is that it will first look for something called concordant mapping. So, we will discuss in a moment what concordant mapping is, and if it cannot find any concordant mapping, it will go for something called discordant mapping. Again, I will explain in a moment. If it cannot find any concordant mapping or discordant mapping, what will you try to do? It will try to find mapping for individual reads, ok?

Bowtie2 – Output Options



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

Command	What does it mean?
--no-mixed	By default, when bowtie2 cannot find a concordant or discordant alignment for a pair, it then tries to find alignments for the individual mates. This option disables that behavior.

So, instead of finding an AH mapping for both read 1 and read 2 in paired end data, it will try to find individual mappings of read 1 and read 2. And if you mention this minus minus no mixed, it will see it will do what it will do; it will not try to find these individual mappings, right? So, in some cases, it might not be useful to have this individual mapping because you want to actually get the full paired end mapping because you have repeat sequences or structural variance that you want to resolve.

Bowtie2 – Output Options



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

Command	What does it mean?
--no-discordant	By default, bowtie2 looks for discordant alignments if it cannot find any concordant alignments. A discordant alignment is an alignment where both mates align uniquely, but that does not satisfy the paired-end constraints (<u>--fr/--rf/--ff</u>).

You can also mention minus minus no discordant, right? So, if you do not want this discordant mapping, you can simply say I do not want discordant mapping. So, just discard, and just stop at concordant mapping. If there is no concordant mapping just discard those reads and do not do the discordant mapping. In some applications, if you are not interested in looking at this structural variance, or some sort of recombination, you probably would not do this discordant mapping. Ah, so, again, you can mention this, and this will speed up the process even further, right? So, you will not try to do the concordant mapping, then the discordant mapping, and then the individual mapping.

So, it will just stop at concordant mapping. If it finds the concordant mapping fine, it will report. If it does not find it, it will just discard the read. It will say, "N match found or no location found, ok.

Bowtie2 – Output Options



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

Command	What does it mean?
-l/--minins <int>	The minimum fragment length for valid paired-end alignments, e.g. if -l 60 is specified and a paired-end alignment consists of two 20-bp alignments in the appropriate orientation with a 20-bp gap between them, that alignment is considered valid (default: 0)

So, you will also have other options. These are called minus-i options. So, this actually again will specify the fragment length in the case of paired end data, right? So, this specifies the minimum

fragment length that it requires to call a valid alignment, ok? So, imagine a situation where you have ah if you have specified minus i as 60, right? So, you have minus i followed by the number 60, right? So, this is the integer saying that fragment length should be a minimum of 60 base pairs, ok. ah So, if you have let us say two reads, right, paired end, right. So, you have 20 base pair alignments for both in both directions, and you have 20 base pair gaps between them, right? So, this will be considered a valid alignment. The default value is 0, right? So, we do not usually impose any minimum AH fragment length. So, ah, we usually would not change this either. The other one is minus x, ok? So, sometimes this one could be useful, ok? So, this is the maximum fragment length for valid paired end alignment, right? So, if you say minus x is 100, right, and if you have ah, So, what you are mentioning is that we will consider only 100 base pair fragments, ok? So, we expect the fragments to be smaller than 100 base pairs and not bigger than those, right? So, if you let us say you have 220 base pair fragments on both directions and you have a 60 base pair gap, that will be a valid alignment, right? So, by default, the tool actually takes it as 500, which means it will consider fragments of length 500, ok, ok.

Bowtie2 – Output Options

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

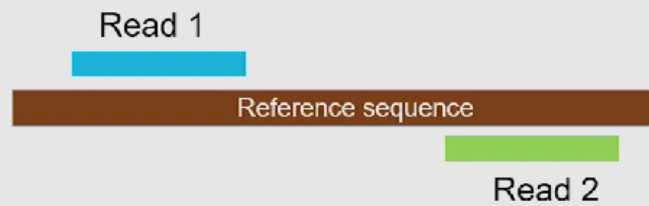
Command	What does it mean?
-X/--maxins <int>	The maximum fragment length for valid paired-end alignments. E.g. if -X 100 is specified and a paired-end alignment consists of two 20-bp alignments in the proper orientation with a 60-bp gap between them, that alignment is considered valid. Default: 500.

So, I will just mention what is concordant mapping and what is discordant mapping, and then we can go and see how we actually map the reads. So, as you have seen, concordant and this kind of mapping, which actually refers to only paired end sequencing data, are these terms, and what is concordant mapping? So, you expect this mapping, right? If you have read 1 and 2, we will expect this kind of location ah with certain ah within a certain distance, right? Let us say a fragment length of 500 or 1000. Within that, you will expect this read 1 read 2 mapping, right? But what will happen, right? So, read 1 and read 2 also satisfy this minus-minus-fr option, right? So, read 1 is forward direction, read 2 in reverse complement, or read 1 is reverse ah complement and read 2 in

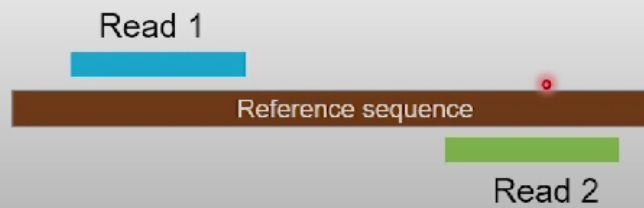
forward direction, or something like that, ok? So, what we observe is that, ok, read 1 is upstream of read 2, and this is actually what is expected, ok? So, if we see if the observed is similar to what is expected, we call this a concordant mapping.

Concordant mapping

Expected mapping

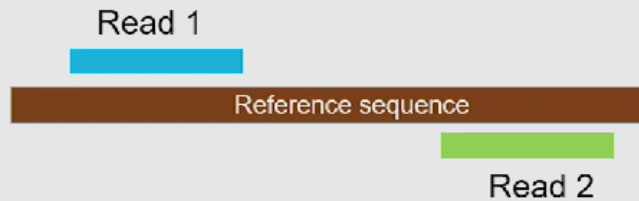


Observed mapping



Discordant mapping

Expected mapping



Observed mapping

- Far apart than expected
- Mapping to two different chromosomes

Observed
 \neq
Expected

But sometimes what will happen is that this is expected, but what you observe is that these two reads they map far apart from each other than expected, right? So, you say they map ah, 10,000 bases apart in the genome, which is very unlikely to happen, right? So, why would you see those 2 reads map to 10000 bases apart because the fragment length when you generate the library we do not have fragment lengths ah of 10000 bases, right or words that they map to 2 different chromosomes altogether, right and again this is not likely to happen, ok. So, this actually refers to discordant mapping that is observed is not ah equal to expected, right? Now, this is something very interesting in certain situations, right? For example, if there is a structural variant or there is some sort of chromosomal fusion, you will likely see this kind of discordant mapping. So, if you let us say in your data you kind of find similar discordant mapping for many reads, right, you probably can carefully look into that and say, ok, maybe there is some sort of chromosomal fusion or gene fusion, where one read is mapping to one chromosome and the other read is mapping to another chromosome, and if you have multiple observational observations like that, you can probably go ahead and kind of test that kind of situation, ok.

So, I think that kind of gives you an overview of the options, right? What we will do now is just

simply look at this run; it is actually quite simple, right? So, what we will do is run this bowtie 2. So, back to this terminal where we have created the index files, I can see these index files here yeast underscore wg 1 2 3 4 and the reverse ones reverse indexes, right? So, what we will do is use this to actually map these things against the reference sequence. So, we have to actually call Bowtie 2 from that folder because we do not have this system-wide installation.

```
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ bowtie2
Command 'bowtie2' not found, but can be installed with:
sudo apt install bowtie2
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ ./bowtie2-2.5.1-mingw-x86_64/bowtie
2 --help
Bowtie 2 version 2.5.1 by Ben Langmead (langmea@cs.jhu.edu, www.cs.jhu.edu/~langmea)
Usage:
  bowtie2 [options]* -x <bt2-idx> {-1 <m1> -2 <m2> | -U <r> | --interleaved <i> | -b <bam>} [-S <sam>]

<bt2-idx>  Index filename prefix (minus trailing .X.bt2).
           NOTE: Bowtie 1 and Bowtie 2 indexes are not compatible.
<m1>      Files with #1 mates, paired with files in <m2>.
           Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
<m2>      Files with #2 mates, paired with files in <m1>.
           Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
<r>       Files with unpaired reads.
           Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
<i>       Files with interleaved paired-end FASTQ/FASTA reads
           Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
<bam>     Files are unaligned BAM sorted by read name.
<sam>     File for SAM output (default: stdout)

<m1>, <m2>, <r> can be comma-separated lists (no whitespace) and can be
specified many times.  E.g. '-U file1.fq,file2.fq -U file3.fq'.

Options (defaults in parentheses):

Input:
  -q          query input files are FASTQ .fq/.fastq (default)
  --tab5     query input files are TAB5 .tab5
  --tab6     query input files are TAB6 .tab6
```

So, we are calling this from the folder, and we will just call this with minus minus help because then it will show all the options that we have just discussed, right, and it will show the usage pattern of how you should run this command, ok. Here it is, right, and there are a lot of commands. You can, of course, go through all of them, and we have discussed some of the relevant ones. The most important one is here. Right here is bow tie 2 usage. Right here is how we should run this command. You have bow tie 2 options minus x p t to idx, and then the read file names the input file followed by the output file. Now, in the output file again, we will have to mention this as the same file, right? So, let us do that. Right in the input file, you can also see another option, which is the minus b bam. So, this is ah, some data sets might be in bam format, right? We have, as we have seen, and this can be specified as bam, ok? So, in our data, it is in ah-fastq format, right? So, what we will do is call this bowtie 2 now, ok, minus x the index file base name, right? So, which is underscored by g, right? Now, we would have to give the index file -x yeast_wg input file, which is the fastq files, and remember, we will here provide the trimmed files that we created last time, right? When we were doing the hands on, we did some trimming, right? So, these are the files that

were created after the pre-processing state, right? So, these are the files that we will use for mapping, right. So, the input files will be these trimmed files; right now, we have them in that folder. So, because we have paired in data, we are using this option: minus 1 iteration 1 read 1 trimmed dot fastq, followed by minus 2 iteration 1 read 2 trimmed dot pass q, ok. And the only thing remaining is the output file name, which should be in sample, ok. So, we have given this minus s option, and the output file name I am just mentioning the output dot sample, right? So, the alignment will be generated here, right in this output dot sample, ok? So, it will take a very, very long time to generate, but it is actually very, very fast, as you will see, and we are dealing with very small data; we are only mapping 25000 reads and 25000 paired entries. So, we have 25000 pairs, and it is done, right? So, if you are dealing with the big data sets, you will see it will take quite a bit of time and much longer than this, but it will still be in hours, not days, ok?

```
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ ./bowtie2-2.5.1-mingw-x86_64/bowtie
2 -x yeast_wg -1 Iter1_read1_trimmed.fastq -2 Iter1_read2_trimmed.fastq -S Output.sam
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
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$
```

So, here are some statistics that we can actually read, right? So, it found 25000 pairs, out of which it could not concurrently map 2354 reads, ok, but it could map about 19500 reads concurrently exactly one time, right? So, there is unique mapping; press the reference you know, and there are 3000 reads for which it found concurrent mapping, but more than one time, right? So, it means there are multiple mappings, right? They are mapping to repeat sequences in the genome. Then it is looking at this 2354 that did not align concurrently, right? So, it tried discordant mapping. So, you can actually stop this using this minus minus, ah, no discordant option, right? So, you can stop this with those options, and then finally, it also looked at this, which did not align concurrently discordantly; it is looking at independent alignment of the names, right? Again, you can ah stop this option, right? Say using this minus minus no mixed option that we discussed, ok? So, we will look at some of the variants of this command when we actually discuss the output files, and we will see, like, what are the useful options in these cases? So, here are the references, right, and

resources that we have used, and finally, to conclude. So, what I do is function. We have used this function to map reads, and we have discussed these options in a lot more detail, but as we go along, we will see that we actually understand the utility of these options a bit more. We have also talked about these global local alignment methods, and they have varying speeds and accuracy. We can implement them again in the subsequent classes, and that is it for today. Thank you. .