Next Generation Sequencing Technologies: Data Analysis and Applications

Hands-on 1 – Data QC and Trimming

Dr. Riddhiman Dhar, Department of Biotechnology
Indian Institute of Technology Kharagpur

Good day, everyone. Welcome to the course on Negation Sequencing Technologies, Data Analysis, and Applications. In the last class, we looked at a FASTQ data file. We have explored this with the VI editor, and then we have set up the FASTQC tool and have at least run the FASTQC tool. So, what we are going to do today is actually look at the results of that FASTQC run. We will evaluate if there is any issue with the data. If there is any problem with the data, ok.

Once we do that, we will then do some pre-processing if required. And so, for example, if we require trimming, we will do that. So, these are the topics that we will actually see today, right? One is data quality evaluation based on the results that we have already generated, then we will do data trimming if it is required, and finally, after we have done trimming, we need to redo the quality control.

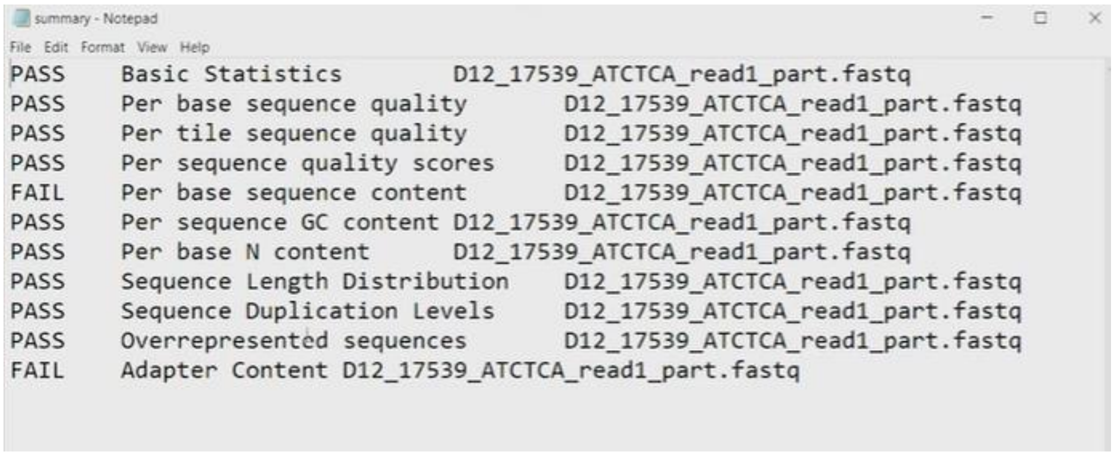
To ensure that the trimming is working fine, we have actually removed the problem that we had in the beginning. So, this is something that we will discuss. These are the steps that we will be following. So, if you can visualize it in your mind, we have done the data quality control. So, once you do that, you have to evaluate the results. Whatever results you have generated, you look at the results and then make decisions if you need to do pre-processing, for example, trimming. So, we have talked about two different types of pre-processing: one is duplicate removal, and the other is trimming.

So, if this is required, for example, trimming, we will have to do the trimming, but once we do the trimming, we have to go back to the evaluation stage again. We have to run QC, we have to do the evaluation, and if something is still not okay, we have to do pre-processing again. So, this is an iterative step, right? We do the quality control, we have the evaluation, we do the pre-processing, and we again do the quality evaluation. So, we will actually do this cycle today with the data set that we have already seen in the last class, ok?

So, let us go there. Let us move into the terminal, right where we have actually generated the quality control.

So, we are inside the terminal, right, if you remember, in the last class, we discussed how we generate these quality control results using first QC, we run the command, and inside these results first QC, we have moved all the results for these two files right read 1 read 2 ok. So, we have extracted these results, and what we are going to do is actually look into the results, make our evaluation, and then we will continue with pre-processing if required. So, let us go inside the folder now. Okay, inside this folder, we will see the results. So, this is the results first QC, and we have extracted this right. We will zoom in right away so that it is easier, and here are the results right for read 1 ok. You will find a lot of files, but there are two files: one file that will give you the summary, and that is the most important file, and then you will have some images in the right folder.

So, we will look into these two things. These are the two main things in this analysis, ok? So, if you look at the summary, you will notice some statistics. So, here are these basic statistics: what you have generated by this first QC, then you have this T12, the name of



```
PASS    Basic Statistics        D12_17539_ATCTCA_read1_part.fastq
PASS    Per base sequence quality       D12_17539_ATCTCA_read1_part.fastq
PASS    Per tile sequence quality       D12_17539_ATCTCA_read1_part.fastq
PASS    Per sequence quality scores     D12_17539_ATCTCA_read1_part.fastq
FAIL    Per base sequence content       D12_17539_ATCTCA_read1_part.fastq
PASS    Per sequence GC content D12_17539_ATCTCA_read1_part.fastq
PASS    Per base N content      D12_17539_ATCTCA_read1_part.fastq
PASS    Sequence Length Distribution    D12_17539_ATCTCA_read1_part.fastq
PASS    Sequence Duplication Levels     D12_17539_ATCTCA_read1_part.fastq
PASS    Overrepresented sequences       D12_17539_ATCTCA_read1_part.fastq
FAIL    Adapter Content D12_17539_ATCTCA_read1_part.fastq
```

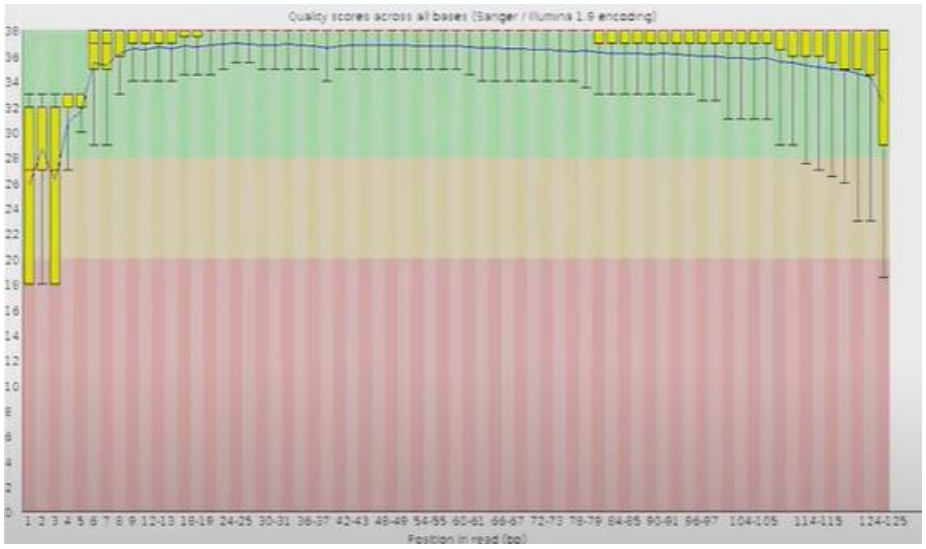
the file that you are working with, and then on the left side you will see this pass/fail and warning, ok. So, this is something that will tell us whether the quality is okay based on these statistics. So, we have talked about these statistics in the theory class. What kind of statistics do you get and what do they mean?

So, you have per-base sequence quality, part tile sequence quality, per-sequence quality scores, etcetera. All of these things we have discussed in detail and what they actually refer to. Now, what is important here is that for most of the statistics, we see a pass. So, they are of acceptable quality, ok, but for two statistics here, we see fail, which means there is probably some sort of issue that passes QC and is detected, ok, and this is where we need to actually look into the images, ok, because those images will tell us what could be the problem, ok. So, for this fail, the first fail is for per-base sequence content. Again, without looking at the image, we cannot tell what the problem is or why it is giving us this kind of fail, and then for the adapter content, you also have a fail.

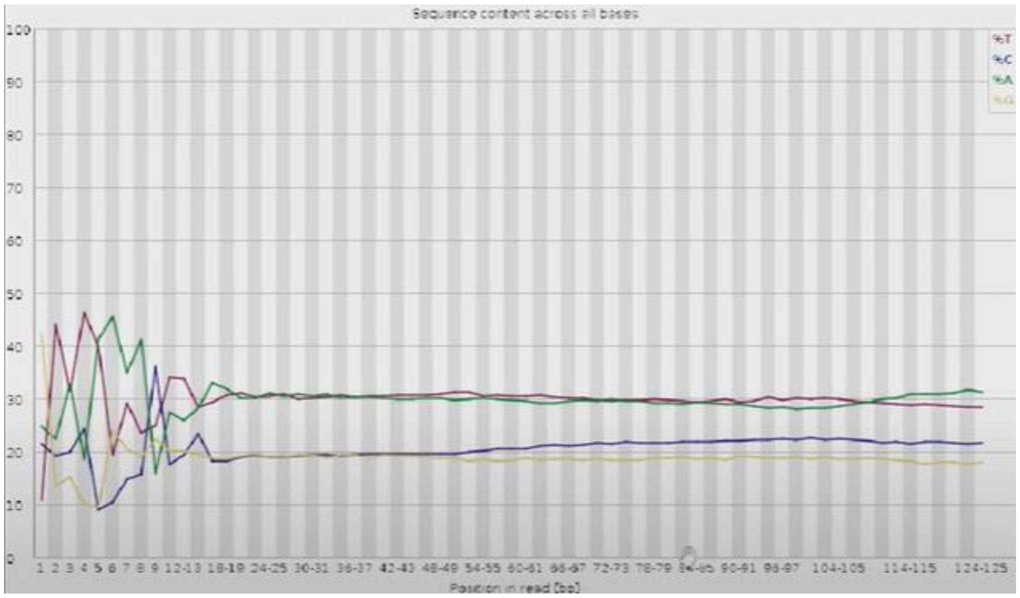
Now, this one probably raises a flag right away if you get a fail for adapter content, which means there is adapter contamination in the data. So, we need to probably do some sort of review, ok? So, let us now go into the images folder, keeping in mind that there are two statistics for which we have this fail remark, and these two statistics we have we will look into very carefully, but we can also look at the other ones. For example, we can start with this per-base quality. quality right we have discussed what this means, and we have seen these kinds of figures before. Now that we know what the x-axis is, we know what the y-axis is, and most of the quality score is in the green zone.

So, this is acceptable. We see some low scores in the beginning, but that is again expected of phenomena. We know why that is the case; we have discussed that, but most of that quality score is in the green zone. So, that means the study data is pretty good, ok? We can look into other statistics and this is one statistic where we have this fail remark. This is the sequence content across all bases. So, we have this position along the x axis, and we have a percentage ATGC along the y-axis. Now, the failure, I think, comes from this part here, right?

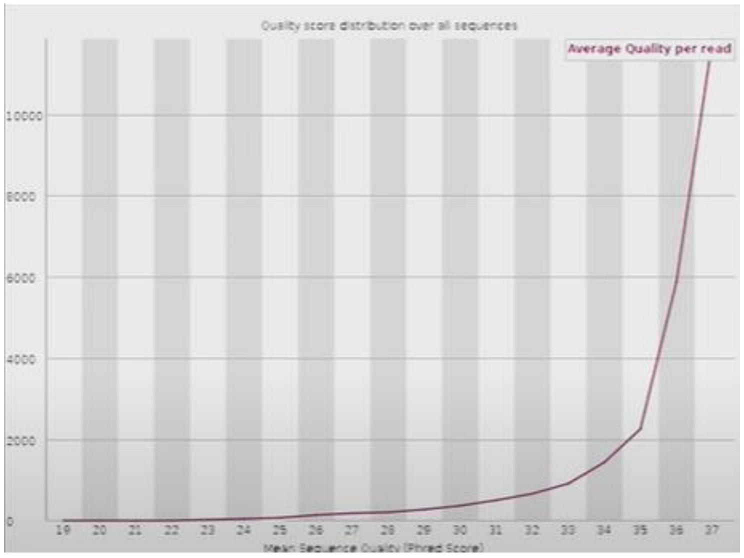
So, there is no other reason for giving this fail remark because there is this difference in ATGC content right? So, this is an AT-biassed genome. So, we will have 60 percent AT



and 40 percent GC. So, that is ok, right? Because it will vary from one organism to another, but here in the beginning, you have a lot of fluctuation. So, this can actually make this fail.



So, this is okay again, considering the fact that you will get some sequencing issues right in the beginning when you start the alumina run. So, we again discussed why that was the case. So, what I will then say is: do not blindly believe all the remarks, right pass, fail, etcetera. For example, here we have seen per-base sequence content that looks fine to us. We can look into the other ones right per sequence GC distribution. For example, we can see per sequence quality; this is an important parameter, and what we see is that most of the reads are of very good quality.



So, they are above 30, and there are very few that are below 30, but they are all above 25. This is the average quality score of the read, and this looks very good. We can look into the sequence length distribution again, the sequence length distribution is 125, and that is expected for alumina. The right sequence length distribution will get reads of the same length. We can look into the per tile quality right again; it appears as fully blue, which is actually a good thing if you remember this is a heat-coloured ap. So, if you have a red or yellowish colour, there are some issues with the quality score in certain regions or certain tiles.

This is something that is very good. Now, the final one that we look at is the adapter content, right? We remember there was this warning right this fail remark against the adapter content, and you probably now understand why that is the case. So, if you look at this carefully, I will zoom in on the right x-axis if you have the position in the read, and the y-axis is the percentage of reads right and what percentage of reads contain adapters. So, here you see, starting from, for example, around 35 or 36 positions, many reads start to contain adapter sequences, and the colour identifies which type of adapter is right.

So, this colour yellow means we have next-era transpose sequences, right? So, during the preparation of this library this next-era transpose adapter was used. So, this appears in the read data, right? So, we have discussed why adapter content happens. What is the issue?

This happens when the fragments generated from the genomes are small, and the sequencing process will go into the adapter sequence. So, once you have this adapter sequence right, this means we need to remove adapters from the data.

So, we need to do something called trimming. We have again discussed what trimming, etcetera, is okay. So, this is for reading number one. Let us look at the two data points. What are the results? We will again have a very similar summary file, and again, you see two points where you see this fail, ok?

So, we have part of the sequence content, and then you have the adapter content, ok, and we can again, without blindly believing this statistic, actually go and look into the images. So, per base sequence content, we again see very similar things. Right in the beginning, we have these fluctuations, and that is expected in the case of Illumina, but for the rest of them, it is quite normal, and this is an AT-biassed genome. So, it looks OK. So, we are not too worried about this warning here, ok, but then we also need to check the adapter content, and here again we see a very similar pattern. We see again that we have many reads that contain adapter sequences, and in a similar observation, we have this next data transpose sequence, right?

So, this kind of helps us in evaluating the data. So, data quality score-wise, it looks great. We have checked and there are no warnings or issues, but there are a few problems. The first is the adapter content. So, this is something that we need to address before we actually go for any downstream analysis, and this is where the pre-processing step will come in. So, let us now go into the pre-processing step ok? As we have mentioned, once you have done the quality control, you get the results, you evaluate the results, and you go into the pre-processing if it is required. In our case, it is required because we have adapter contamination. We will do trimming, and then we trim it back and check it back with the FASTQC.

So, once we have done trimming, we check it through the FASTQC again. So, let us do that right. So, let us first come out of this folder, right, and let us do the trimming, ok? So, for trimming, we have discussed many tools. If you remember, we have discussed at least three tools. I have given you the link, but today we are going to use one tool, which is the bbduk. So, bbduk is a tool that utilizes some of this information about the adapters to remove it from the read data.

So, we can actually set this up, bbduk, right? We can simply say that bbmap is actually a package that contains this function. bbduk ok. So, we can go into bbmap right, and we can download and extract this bbmap. If you can see on the left side, I will zoom in a bit. On the left side you can see you have this bbduk right. So, bb bbmap has a lot of functions; we will not go into all those. So, I will just simply look at this BB Dock because this is the function that is useful for us, ok? So, as you can see, there is an explanation with a lot of details on how to use this tool, etcetera. what are the options? What are the common lines? What is the algorithm that this tool uses?

So, you have duk, which means decontamination using K-mers, right? So, if you have adapter contamination and want to do trimming, you can use these two. You can also do quality trimming or filtering, and you can do adapter trimming. You have all these options here, and you can see this adapter trimming or contaminant filtering. So, we will do this adapter trimming using this tool. Now, what is the algorithm? How does it do this? We will not go into that right now. There are a lot of details. If you are interested, you can always read right. You have all the instructions on how you should run this tool with your read data. For example, this is what we are going to use: bbduk sh iron 1 equals to read 1 dot fq and then iron 2 reads dot fastq out 1 of the output file names.

So, you give input file names if you are working with paired-end data; in our case, it is paired-end data. So, we will utilize this command, and then we will have the output file names. So, you have to give these two output file names. Now, there are a lot of parameters right in this algorithm. You can also see the memory usage, and you can actually control how much memory this tool will use. Because if you are working on a system that does not have too much memory, you want to use this flag, ok?

And then you have a lot of these parameters that you can see OK again without going into the algorithm. These parameters are part of the tool, and sometimes you need to assign certain values for getting the best results. And this is something that you have to kind of do through trial and error, right? You have to optimise these parameter values through trial and error. You kind of set a certain value and see whether the trimming has occurred properly or not. And then again, go back and modify. If it has not happened, you go back and modify the parameters and try again. So, there are some parameters that work pretty well, and you can probably find out once you have tried this out. So, what we will do is actually run this right through our data because we have adapter contamination, which means you will see these commands OK.

Now I will explain this command and what you have in here, ok? So, this is the function, this is the command that will remove the adapters from your data. We have this in data in 1 in 2 because we have paired in data. We will use out 1 in 2 because we will be dealing with paired in data. So, we will get 2 output files, then you have something called ref equals to adapters dot fa ok. So, this actually refers to the adapters file that is present within this folder. So, bbmap has its own adapter sequences stored.

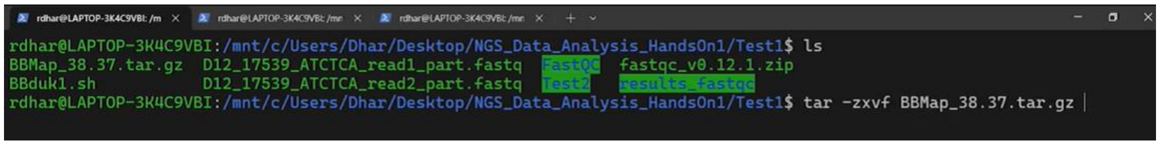
So, this contains a list of all adapters that are utilised by different platforms. So, these adapters can then be removed by this tool, ok? So, if that is so, what will this tool do? It will match the sequences of the reads against these adapters, and whenever it finds some match, it will remove those adapter sequences. And then you have certain parameters. Okay, again, we are not going into these parameter values or how this algorithm works, but these are the parameters that they have mentioned we can use, and you can also change them in certain sequences. So, there is a very nice explanation following this about how we actually remove these adapters.

And if you are interested in quality streaming, then you have to follow certain other parameters, etcetera, and we will not go into that because our quality score is pretty good. So, let us actually run this right through our command line, ok? And so what I have done is actually optimise these parameters and I have stored these parameter values in this file.

So, the first step is setting up this bbmap. Now, you can see you can download this bbmap from here, right? You have this bbmap download here, ok?

You can download this star file here, and you need to set it up, ok? So, what I have done is I have downloaded this already in our system. You can see this here. And we need to extract this, and we need to utilise this tool, ok? So, to extract this is a tar dot gz file, right? You remember, hopefully, this compression that we have talked about and that information will be very useful. This is a tar compression archive with gz compression, ok?

So, we need to use this command: tar minus xvf or zxvf right, and bbmap ok. So, this will decompress the whole thing, ok? So, once we do that right, it is extracting all the functions and all the tools. So, it is done now, and we can check what has happened. You can see it has created a folder called bbmap, inside which you have all the files, all the functions, and all the tools.

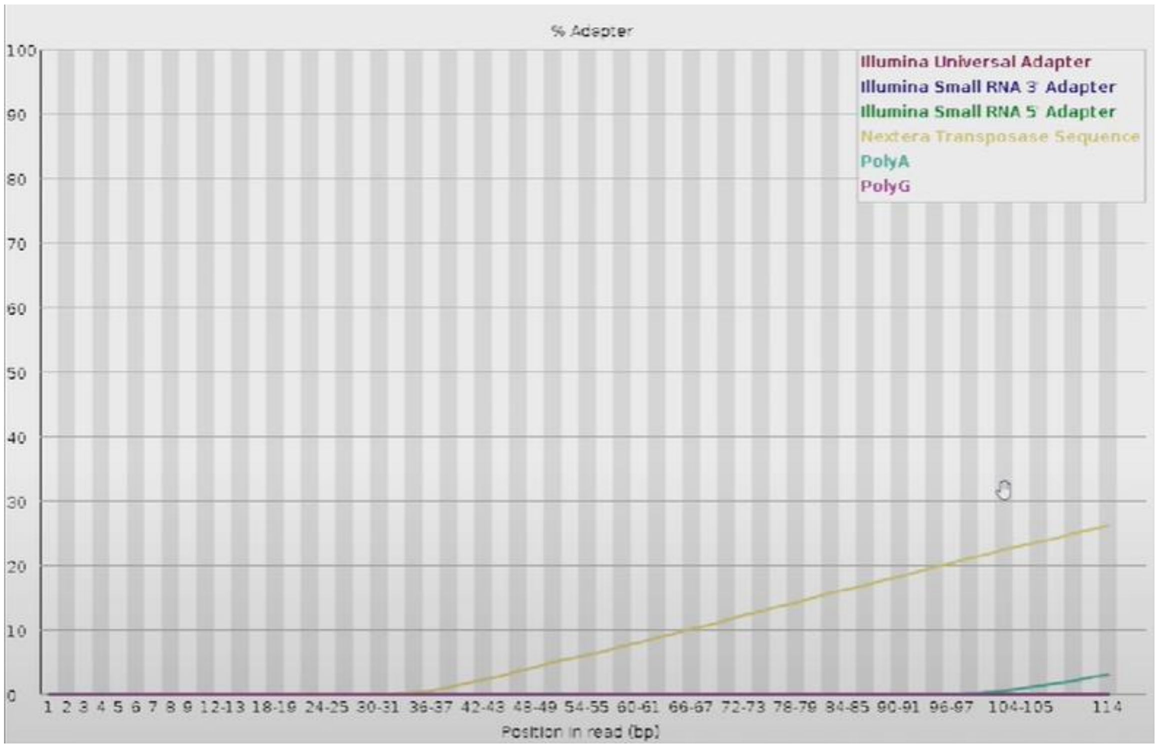


```
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ ls
BBMap_38.37.tar.gz  D12_17539_ATCTCA_read1_part.fastq  FastQC  fastqc_v0.12.1.zip
BBduk1.sh           D12_17539_ATCTCA_read2_part.fastq  Test2   results_fastqc
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ tar -zxvf BBMap_38.37.tar.gz
```

So, let us look at those functions, etcetera. You have a lot of functions again. And as you have seen, bbmap can do a lot of things, but we are interested in only one function, which is the bb dot. So, let us check if this function has the right bb dot. We can also check right now to see if this has execution permission. Again, if we download sometimes, then there is no execution permission, but in this case, we have the right.

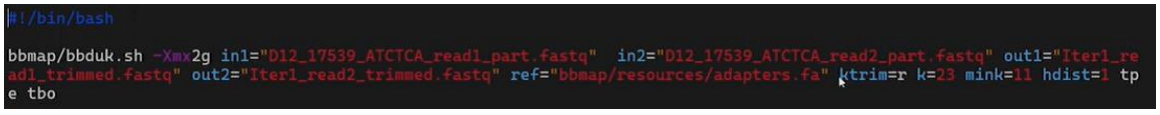
If you do not have this execution permission you can change the permission using chmod ok. So, here is the function that we will use. So, it is a download. It is very easy to set up, right? You just simply download the tar.dot.cz file you extracted, and you have the function. You do not need to do anything in this programme, ok? So, as I was saying, we have tested different parameter values for this data and we have optimised which parameters actually work the best for these files. So, let us see, and I have stored this in this file. Let us look at this right.

So, here is this I have stored in something called bb dot 1 dot sh just to distinguish from the bb dot sh right, and it actually stores the full command that would be using ok right. So, if you are working on the command line, you can run this as a script itself, but if you want to just run in the terminal, you can copy this command and run in the terminal. So,



this is what we will do, ok? So, we will copy this command and we will run it in the terminal. Before we do that, I will explain what this command actually means.

In the first part here, bb map bb dot dot sh, we are calling that programme bb dot dot sh inside this bb map folder. Then we have this minus xmx 2g, ok? So, it means it will limit the usage of RAM. So, it should not use more than 2 GB of beta, right? So, if you are working on a system where you do not have too much memory, you want to limit this memory usage; otherwise, it might crash the system.



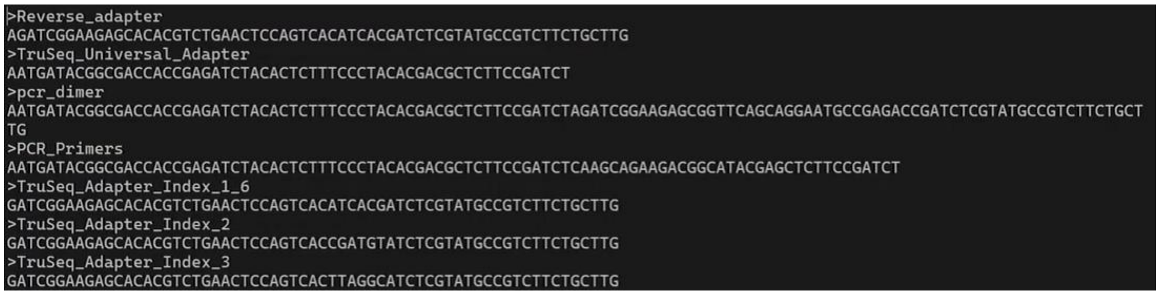
```
#!/bin/bash

bbmap/bbduk.sh -Xmx2g in1="D12_17539_ATCTCA_read1_part.fastq"  in2="D12_17539_ATCTCA_read2_part.fastq" out1="Iter1_re
ad1_trimmed.fastq" out2="Iter1_read2_trimmed.fastq" ref="bbmap/resources/adapters.fa" ktrim=r k=23 mink=11 hdist=1 tp
e tbo
```

So, this is how it is done: minus xmx2g, then you have the input file names ok? In 1 d 12 1753 9 etc., read 1 part dot first q. This is the first first q file, then you have the second first q file right. So, this is required because we are dealing with paired-end data. So, we need to give both files at once. Then you have the output file names that we can specify, and we can specify anything that we like.

So, I have mentioned iteration 1 read 1 trimmed dot first q just to distinguish from the original file names, right? This is the read-only file, but this is also trimmed, and I have also mentioned iteration 1 because we might need multiple iterations to get to the optimal parameter or optimal data quality. So, that is why I mentioned iteration 1. You have these two files right out of the box; this is also because you have paired end data. If you are dealing with single-end data, you do not need to mention this in 2 or out 2, ok?

Then you have to give the reference sequences. So, what are these reference sequences? These are the adapted sequences that bbduk will use to remove those adapters from the read data. So, this contains again all the adapted sequences. So, bbmap resource adapters dot fa I will show you in a moment what this file is, where it is, and what it looks like. Then you have certain parameters that this bbduk has mentioned. So, before we run, let us run this right, and I also want to see this right.

So, let us quit, right? You remember now the vi-editor ok colon q? We do not want to make any changes; we want to just quit. We want to check this file; it is in this directory vi resources adapters dot fa right. So, this is where the adapters dot fa file is; this is in the first apartment, right? Dot fa means fasta format. And here you are, right? You have these adapter sequences: reverse adapter, trueseq, universal adapter. So, these are adapter sequences utilized by different platforms, including Illumina as well as other platforms.

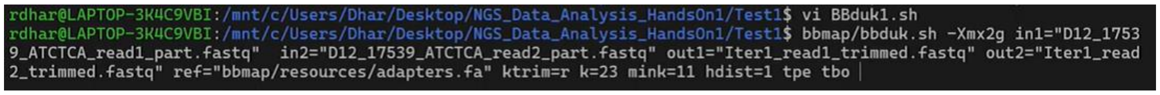


```
>Reverse_adapter
AGATCGGAAGAGCACACGTCTGAACTCCAGTCACATCACGATCTCGTATGCCGTCTTCTGCTTG
>TruSeq_Universal_Adapter
AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT
>pcr_dimer
AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCTAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCGTCTTCTGCT
TG
>PCR_Primers
AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCTCAAGCAGAAGACGGCATACGAGCTCTTCCGATCT
>TruSeq_Adapter_Index_1_6
GATCGGAAGAGCACACGTCTGAACTCCAGTCACATCACGATCTCGTATGCCGTCTTCTGCTTG
>TruSeq_Adapter_Index_2
GATCGGAAGAGCACACGTCTGAACTCCAGTCACCGATGTATCTCGTATGCCGTCTTCTGCTTG
>TruSeq_Adapter_Index_3
GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCATCTCGTATGCCGTCTTCTGCTTG
```

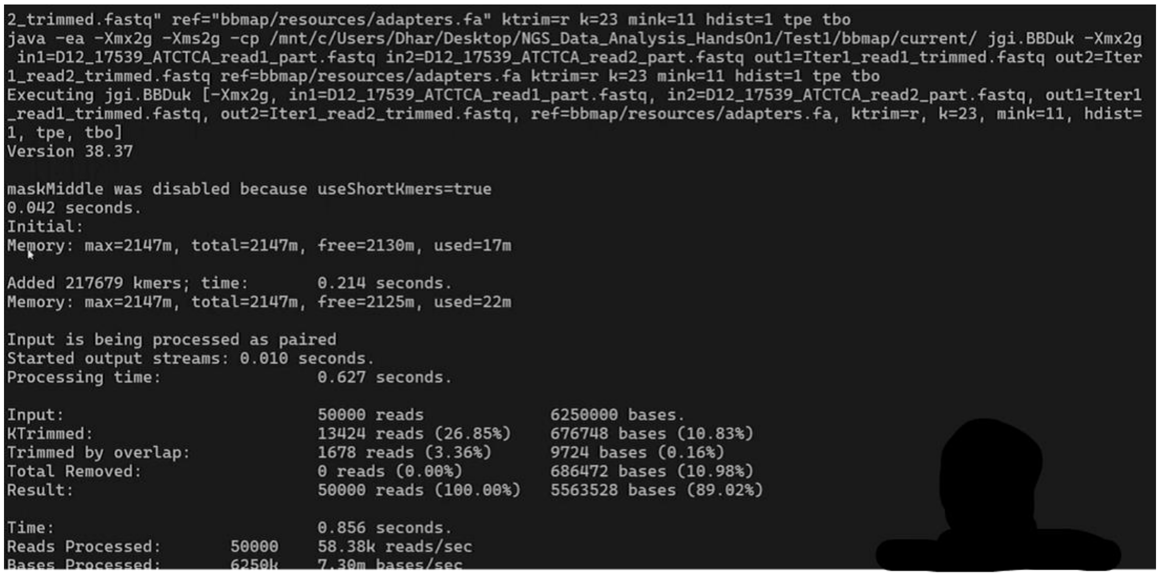
And you can see this, and perhaps you can also search if you see this next error, right? The search function in Vi comes in really handy here, right? So, you can see a slash, and the moment you type next, immediately you see this next error, right? So, you have this next adapter sequence here, which means we can now use this right file to remove our adapter sequences. Now, why am I showing this adapter file? Why do you need to know about this file? Imagine you are using some adapter sequences that you have yourself designed.

These are not commercially available adapters; these are the adapter sequences that you have designed yourself for your experiment for whatever reason. Now, these adapter dot fa files will not contain those adapter sequences, right? So, in your data, if you see some sort of adapter contamination, you want to update this file to remove that adapter contamination. So, this is something where you have to add this, ok? You can simply add this by vi right. You can simply add add that sequence, saying your adapter name right, and then give the sequence and just save it with a colon wq right.

So, here we are not saving anything; we have not written anything, but in case you do modify this file you can save it with a colon, ok? So, we have now seen this adapter file, and we can simply run this now, which will hopefully remove this adapter data. So, let us run this. So, I will simply copy this right, and then I will paste it here, OK, and we have explained and understood what each of these commands actually means. So, let us now enter. The moment I press enter, it will start this streaming, and this is done, ok, and you will get certain statistics.



```
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ vi BBduk1.sh
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ bbmap/bbduk.sh -Xmx2g in1="D12_1753
9_ATCTCA_read1_part.fastq"  in2="D12_17539_ATCTCA_read2_part.fastq" out1="Iter1_read1_trimmed.fastq" out2="Iter1_read
2_trimmed.fastq" ref="bbmap/resources/adapters.fa" ktrim=r k=23 mink=11 hdist=1 tpe tbo
```



```
2_trimmed.fastq" ref="bbmap/resources/adapters.fa" ktrim=r k=23 mink=11 hdist=1 tpe tbo
java -ea -Xmx2g -Xms2g -cp /mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1/bbmap/current/ jgi.BBDuk -Xmx2g
 in1=D12_17539_ATCTCA_read1_part.fastq in2=D12_17539_ATCTCA_read2_part.fastq out1=Iter1_read1_trimmed.fastq out2=Iter
1_read2_trimmed.fastq ref=bbmap/resources/adapters.fa ktrim=r k=23 mink=11 hdist=1 tpe tbo
Executing jgi.BBDuk [-Xmx2g, in1=D12_17539_ATCTCA_read1_part.fastq, in2=D12_17539_ATCTCA_read2_part.fastq, out1=Iter1
_read1_trimmed.fastq, out2=Iter1_read2_trimmed.fastq, ref=bbmap/resources/adapters.fa, ktrim=r, k=23, mink=11, hdist=
1, tpe, tbo]
Version 38.37

maskMiddle was disabled because useShortKmers=true
0.042 seconds.
Initial:
Memory: max=2147m, total=2147m, free=2130m, used=17m

Added 217679 kmers; time:       0.214 seconds.
Memory: max=2147m, total=2147m, free=2125m, used=22m

Input is being processed as paired
Started output streams: 0.010 seconds.
Processing time:                0.627 seconds.

Input:                          50000 reads             6250000 bases.
KTrimmed:                       13424 reads (26.85%)    676748 bases (10.83%)
Trimmed by overlap:             1678 reads (3.36%)      9724 bases (0.16%)
Total Removed:                  0 reads (0.00%)         686472 bases (10.98%)
Result:                         50000 reads (100.00%)   5563528 bases (89.02%)

Time:                           0.856 seconds.
Reads Processed:       50000    58.38k reads/sec
Bases Processed:       6250k    7.30m bases/sec
```

[redacted]

So, it had, so it will give you some statistics regarding how much memory it utilised, etcetera, in the system, and then it will give you some statistics about the reads that it has processed. So, here it gives you some input about the input. Right, it has about 50,000 reads in total. Right, it processes 25,000 plus 25,000 into files, and it processes about 600,6.

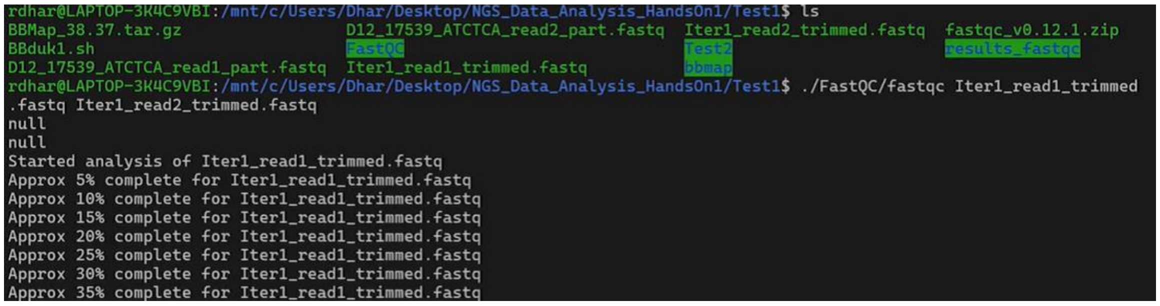
25 million bases right? So, in this read data, we have 6.25 million bases. It has streamed certain readings. So, it has streamed about 13,000 reads and about 676,000 bases, roughly, which is about 10 percent of the data and 11 percent of the data. It has also trimmed other reads; these are two different ways it has streamed.

So, 1600 reads again, 9700 bases. It has not completely removed any So, total removed: 0 reads. So, it has all the bases right, and the result is that it has all the reads right. 50,000 reads in the trimmed data contain all of these, and the bases will be around 556.

56 million is okay. So, from 6.25 million, we have now come down to 5.56 million, right? So, it means some sort of trimming has happened, ok, and then there are some speeds, how much time it has taken, etcetera, etcetera, ok. So, once this is done, the next step is to actually run it through the quality control pipeline. We want to check quality again, ok? This is important because we want to make sure that these adapter sequences have been

removed.

So, let us do that, and here we will get this trimmed data right. So, we have named this iteration 1 read 1 trimmed dot first q and iteration 1 read 2 trimmed first q right. Now, again, the running of quality control is very easy. We will have to run through the first QC, and we follow the same command as before. Here, the input file names will be different, right? So, the input file names will be iteration 1 read 1 trimmed dot first q right and iteration 1 read 2 trimmed dot first q right.

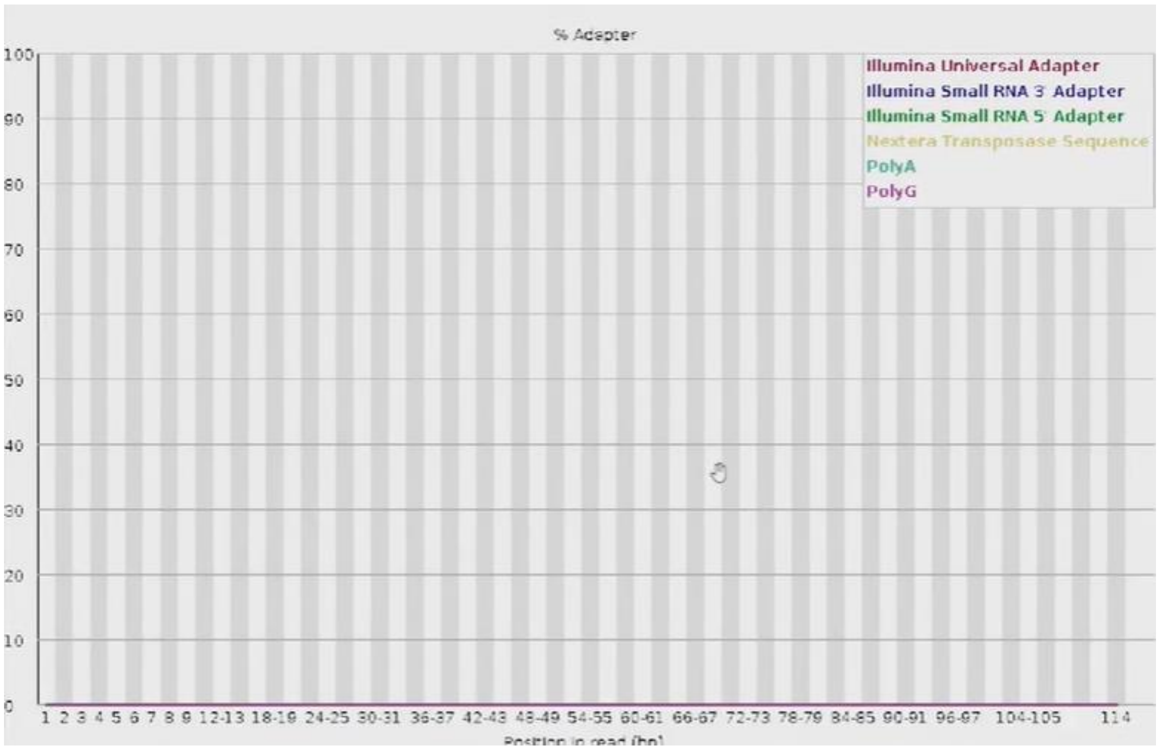


```
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ ls
BBMap_38.37.tar.gz                D12_17539_ATCTCA_read2_part.fastq  Iter1_read2_trimmed.fastq  fastqc_v0.12.1.zip
BBduk1.sh                         FastQC                             Test2                      results_fastqc
D12_17539_ATCTCA_read1_part.fastq Iter1_read1_trimmed.fastq          bbmap
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ ./FastQC/fastqc Iter1_read1_trimmed
.fastq Iter1_read2_trimmed.fastq
null
null
Started analysis of Iter1_read1_trimmed.fastq
Approx 5% complete for Iter1_read1_trimmed.fastq
Approx 10% complete for Iter1_read1_trimmed.fastq
Approx 15% complete for Iter1_read1_trimmed.fastq
Approx 20% complete for Iter1_read1_trimmed.fastq
Approx 25% complete for Iter1_read1_trimmed.fastq
Approx 30% complete for Iter1_read1_trimmed.fastq
Approx 35% complete for Iter1_read1_trimmed.fastq
```

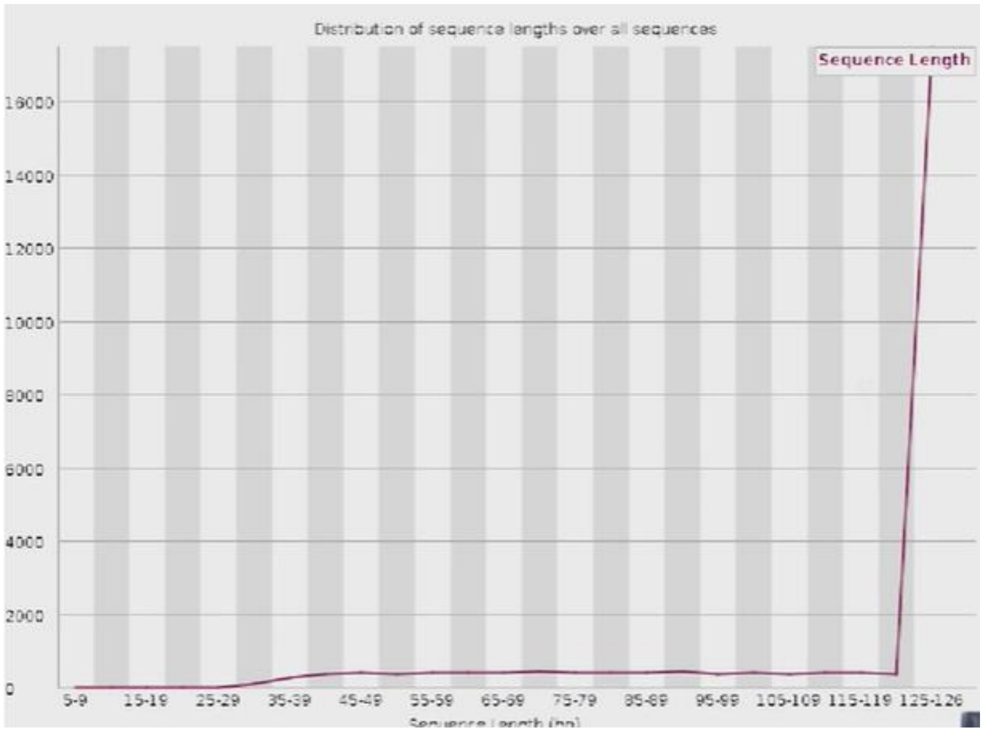
These are the trimmed files, which we want to check through the first QC. So, let us run this, and again, we get very similar output. So, this is quite quick because we are dealing with small data. We are not dealing with the full data, as you can imagine. If you are dealing with the full data, it will take more time. So, this is complete, right, and we have generated these first QC files. Again, you can see these first QC files here: read 1 trimmed first QC dot html and first QC dot c ok. We follow the same process: we create a folder right where we say mkdir trimmed data right.

So, we can move all the trimmed data there, ok, and we can move this first QC now again using this command mv, and we go there to trimmed data, ok. We move inside the trimmed data folder and we again unzip. We are following the same steps as before. We have done this for read 1, now we do it for read 2, ok, and we have generated these folders. So, we have this first QC folder for 2 of the 2 reads and 2 data sets, and then we can now go inside these folders and check the quality again, ok, and the quality that we will be focusing on mostly is the adapter content, right? That is what was the problem earlier, right? We had adapter contamination in both of them, ok? So, let us go into that folder and see what actually happened, ok?

So, let us see where the data is. So, here is the trimmed data, and we can go inside, and again, let us look at the summary. Now, the summary says everything is passed except per base sequence content, okay? We are not worried about that too much, but it now gives a warning about sequence length distribution. So, this is something that is new, and if you think about it, this is expected. If you are trimming sequences, your sequence length distribution will change, and some of the reads might be very, very small. So, let us go into the images, and this will tell us the full story, right? If we first look at adapter content now, it looks great; there is no adapter contamination here. Okay, everything is gone. Earlier, earlier used to see this as increasing adapter content, but that is gone. So, that problem is resolved, but there is some other warning right where is this sequence length distribution, and you can now realise why this warning comes right because you have certain reads that are very, very small.



You have most of the reads of length 125 right here; you can see that this is the peak for this sequence length, but then you also now have some reads that are smaller, and especially here you have reads that are very, very small. So, this is why the warning is right, ok? So, we can now look into the other one as well, which is for the read 2 OK, and again, we can look at the summary file first, and you see similar results. Again, the warning for sequence length distribution is gone, but the adapter contamination is gone.



It is passed; there is no adapter contamination, and we can confirm this through these images again. So, here is the adapter content, and it looks great. We also have the sequence length distribution and it is very similar compared to the other file. Right here, you have sequence length, where most of the reads are of length 125, but you have some reads that are very small, but that is ok. I think we can go ahead with this data, and it looks fine; it does not need any further pre-processing.

So, I think coming back to the presentation, we have actually gone through these steps: we have done this data quality control, we have evaluated the data, we saw some problems and then we did some pre-processing. After pre-processing, we have again run the quality control and we have evaluated again. So, to summarise what we have seen, data pre-processing is an iterative step. So, it requires trimming and re-evaluation of quality and whether it might require some other sort of pre-processing that is again dependent on the data that you are working with, and data trimming is required when there are concerns regarding read quality or adapter contamination. So, in our case, it was adapter contamination. So, we went to pre-process data trimming and then also verified that the trimming actually worked OK. I will stop here. Thank you.