Next Generation Sequencing Technologies: Data Analysis and Applications

Hands-on analysis Variant Calling

Dr. Riddhiman Dhar, Department of Biotechnology

Indian Institute of Technology Kharagpur

Good day everyone. Welcome to the course Generation Sequencing Technologies Data Analysis and Applications. In the last few classes, we have learned about how to do variant calling. We have discussed different variant types, and in the last class, we focused mostly on SNVs, SNPs, and small indels. So, we have talked about the pipeline and how we use different filtering approaches, etcetera. So, in this class, the focus will be on doing the hands-on analysis, ok?

So, we will utilize a tool to actually do this analysis ourselves, and we will do some variant calls, ok? So, this is the topic that we will be covering in this class. So, let us go ahead. So, there are different variant-calling tools.

So, I will list some of them here. So, the first one is the Genome Analysis Toolkit. This is a very popular toolkit, and there are multiple options there. So, for example, you can have this GATK haplotype caller or GATK mutect2 too, and they are built for different purposes, although they have some overlaps and some special purposes. You also have these SAM tools and BCF tools that are also widely used for variant calling, and then you have something called Freebase, Stelka, and the Farescan tool, ok?

So, out of all these tools, what we will use are the SAM tools and BCF tools. It is easy to use; it is free to install and easy to use through the command line, ok? And since we have the output in the SAM file, we can use that for our analysis, ok? So, what we will do is look into these SAM tools and BCF tool commands. Of course, there are many options, but not all of them are required in this class. Also, the time will not permit us to look at the command options that are available.

just to remind you of the steps, right? So, we will follow these steps here right before we actually do the variant calling. So, we have the mapping data from the last hands-on mapping. We will do the local realignment, we will do duplicate marking, and we will not do base-quality recalibration because of the time constraint and because it is a very computationally intensive process. So, we will not be able to do this in the lab in this short period of time, but you can of course, try. I will mention the tools that we can use for doing this analysis.

Once we have this analysis ready, we will use BCF tools to call the variants, ok? So, let us get started because it will take a lot of time because we will install these tools SAM tools and BCF tools, and we will work with them, ok? So, let us go ahead and see how to install them, and then we will come back to the steps that we need to follow for doing this variant calling, ok? So, let us move to the terminal. Before I go, I just wanted to show you these places where you can find these tools and download them.

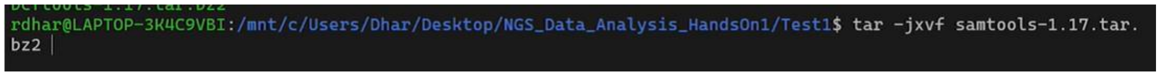
So, you have this genome analysis toolkit. So, this is the GATK, and you have this haplotype color, etcetera, in here. Then, if you go to SAM Tools or BCF Tools, you will find them here. So, you can simply search SAM tools or follow the link that I have given, and you will end up on this page, ok? And you can see these commands here for SAM tools, etcetera.

And similarly for BCF tools, you will see manuals, which means you will have these command pages, right? So, here is the list of commands you can see here, and we will use some of these commands for our analysis, ok? But before we go anywhere, we first need to install this, right? So, here, if you just go to this link: htslib.org, you will find these SAM tools, BCF tools, and htslib, ok?

So, we just need to install these SAM tools and BCF tools, and we can download them very easily here from GitHub. Right here, downloads are available. So, we will get all these tools at once, and you can install them, ok? So, I will just go let go through the instruction process. I have downloaded them already, and we will go through the instruction process, and then we can get to the variant calling part, ok? So, let us go to the terminal here. We are in this place where we have these SAM files, ok?

We generated the SAM files in the last lab, and you can see those SAM files here. So, let us see where the SAM file is, or we can simply search, right? It should be output SAM, right? So, this file is there, ok? So, we will work in this directory so that we can process the SAM file to identify the variants, ok?

So, I have downloaded these SAM tools here and see, right? So, this is SAM Tools Star Dot BZ2, ok, and we have these tools again in the same format, and we have to extract them first. Again, I have extracted. So, the extraction command you remember is: star minus zx, zxvf, right, and let us say samtools, right. So, once you run these commands for both the samtools and bcf tools, they will be extracted.

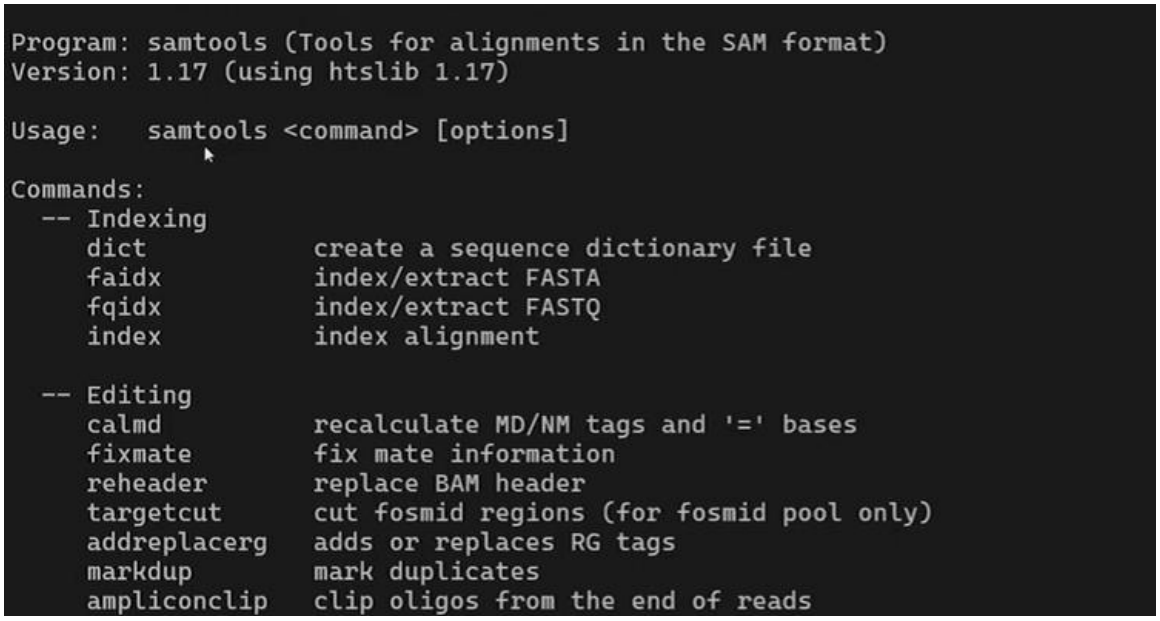


```
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ tar -jxvf samtools-1.17.tar.
bz2
```

Now, once they are extracted, you should see the folders, right? So, again, we can see, right? So, you can see this folder. So, one is this bcf tools folder here, right, and you also have the samtools folder. So, inside this folder, you have these programs: samtools and bcf tools, which you can use for running all sorts of commands.

So, I will just run this, right? So, it will give you all the options for how to run these commands and these things, right? So, you can see the usage, right? I have samtools commands and options, ok? So, there are different commands that you can use here.

They are under different operations, right? So, you can see indexing, or you can have editing, file operations, statistics, viewing, etcetera, right? So, we will use some of these commands, OK, and also some options. So, the options are not given here because you will find them in the manual on the webpage. Similarly, we can also run the bcf tools, and again, it will show you what commands you can run here.

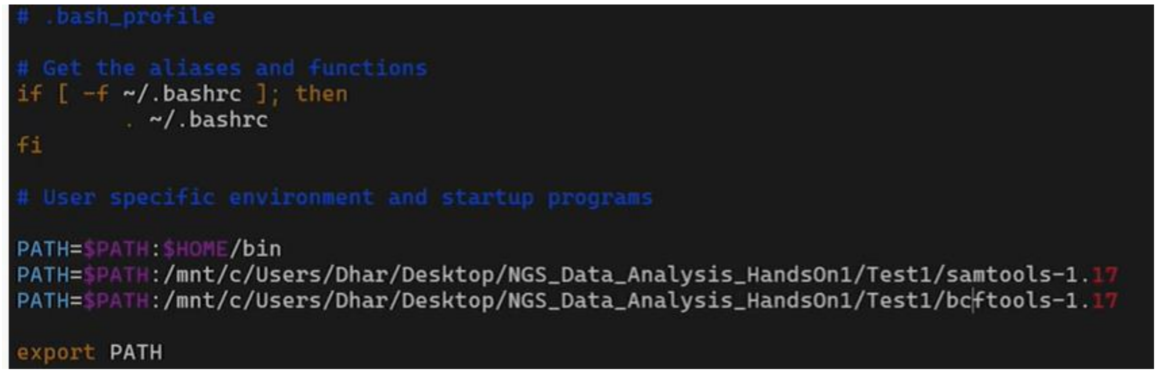


```
Program: samtools (Tools for alignments in the SAM format)
Version: 1.17 (using htslib 1.17)

Usage:   samtools <command> [options]

Commands:
  -- Indexing
     dict           create a sequence dictionary file
     faidx          index/extract FASTA
     fqidx          index/extract FASTQ
     index          index alignment

  -- Editing
     calmd          recalculate MD/NM tags and '=' bases
     fixmate        fix mate information
     reheader       replace BAM header
     targetcut      cut fosmid regions (for fosmid pool only)
     addreplacerg   adds or replaces RG tags
     markdup        mark duplicates
     ampliconclip   clip oligos from the end of reads
```

Then again, for different purposes, you have these different commands, and this is the usage command argument, right? So, this is the kind of usage that we will have to go with, right? So, again, there are a lot more options for each of these commands that are given on the webpage. So, if you go to the manual page, right, on the website, we will see some of these options, ok? So, I will show you some of these options for our analysis, but of course not everything, all right.

So, let us have this now. One of the things you will probably notice is that just calling them through this dot slash is actually quite cumbersome, right? Every time you have to direct them to the folder, right, to that folder where they have been extracted, So, if you go out of this directory and go somewhere else in the system, you have to point to that directory where samtools has been extracted and call the command samtools. Is there any way you can call Samtools from anywhere in the system? So, as it turns out, there is, right? If you are familiar with Linux, you will know that there is a way to do this by adding it to the path.

So, if you want to add this to the path, you need to have administrative access. So, it means pseudo-access, okay? And you have to modify this file, which we will call a bash profile, ok? So, this is called a bash profile, and it will ask for this password because for administrative access you need to give the password. And what we would have to do is add this folder to the path here under this path variable, ok?



```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin
PATH=$PATH:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1/samtools-1.17
PATH=$PATH:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1/bcftools-1.17

export PATH
```

So, what I will do is actually add both of the samtools and bcf tools under this path in this file, and we can then call samtools and bcf tools anywhere, ok? So, let us go inside this Samtools folder and look at the path, right? So, what I will do is just copy this path, ok? And I will add this to the bash profile. The notation is exactly the same, right?

You can see this here, and I will simply add this path, ok. And similarly, I can do this for BCF tools as well. I am using Vi. So, this means it is very easy, right? We can simply copy and paste, and BCF tools are also installed under the same path.

I just need to change the name here, bcf tools 1.17, ok? So, the versions are the same, alright? And everything remains the same. We save it, and we come out into the terminal.

So, again, if you are not sure how to modify these files, please do not do that, because this might end up causing problems for your system. So, once you have done this, one of the last commands that we want to run is source. So, it will refresh—it kind of refreshes, right? It reads the path, and once it does that, right, we can now come out of this folder, and we can call Samtools or BCF Tools from anywhere, ok? You see, I do not need to point to that directory where Samtools has been extracted, right?

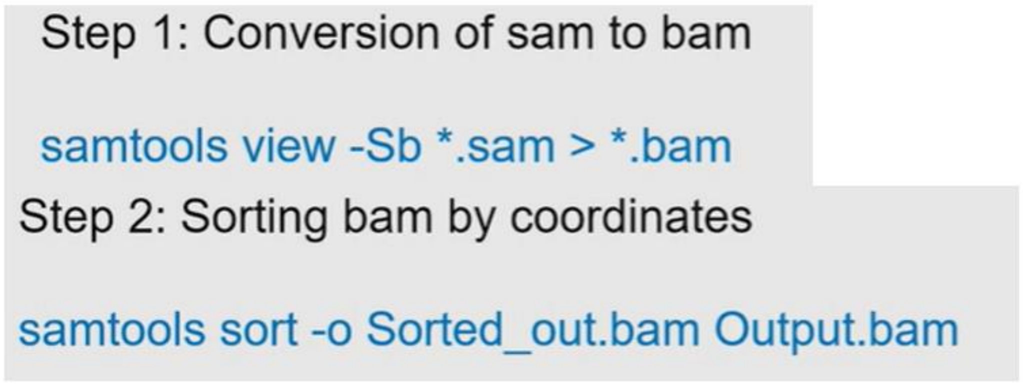
So, we can simply call Samtools from anywhere, right? Even if you are in the, let us say, desktop or ng data analysis folder, we can simply say samtools, and the system will automatically point to that directory and call samtools. And similarly, we can also call BCF tools, alright? So, this kind of makes our lives a bit easier, right? So, we have now got these tools installed, and we can call them from anywhere in the system, ok?

So, we will use these samtools and bcf tools now to identify the variants in our data, the sam file that we generated in the last hands-on, ok? So, let us now go to the presentation, and we will see the steps, multiple steps, and commands that we will have to use, right? So, again, we can search for this command through the manual pages, etcetera, but it will, of course, not be possible for us to complete the full variant calling pipeline in this class if we do that. So, what I have done is I have also compiled all those commands that we will use for doing this variant call, and we will simply run them, and I will show you right that at the end we get the variants, ok? So, there are multiple steps if we have to do this with Samtools and BCF Tools; they relate to these earlier steps here, but they are specific for these programs in this kind of way, like what these programs will need.

So, the first step is that we need to convert this sam file to a bam file, again in binary format because this is a machine-readable format. So, Samtools and BCF Tools will use this format. We need to sort the BAM file. So, sort sorting means it will by default sort according to coordinates or mapping positions in the file, and the third step is the realignment, which, as we have mentioned, Samtools and BCF tools cannot do realignment.

So, we will rely on another tool. The fourth step is to mark duplicates, right? So, we need to identify duplicates in the data. The fifth step is the base-quality recalibration. Again, as I said, we will not do this step in this class because of the time constraints; it is a very computationally expensive process, and finally, we will do the variant calls. So, this is just a few ways to give you some ideas on how you can proceed, and of course, you can try your different things, ok?

So, one of the things you should notice is that when we install Samtools, etcetera, you might end up with some issues with some libraries, ok? So, you would have to install them independently, ok? For installing Samtools and BCF tools, you have to again rely on the install commands that are there. So, let me go and show you these install files that actually mention the step-by-step installation process, ok? So, I will show that, and then we will start with the conversion of sam to bam, ok?



Step 1: Conversion of sam to bam

```
samtools view -Sb *.sam > *.bam
```

Step 2: Sorting bam by coordinates

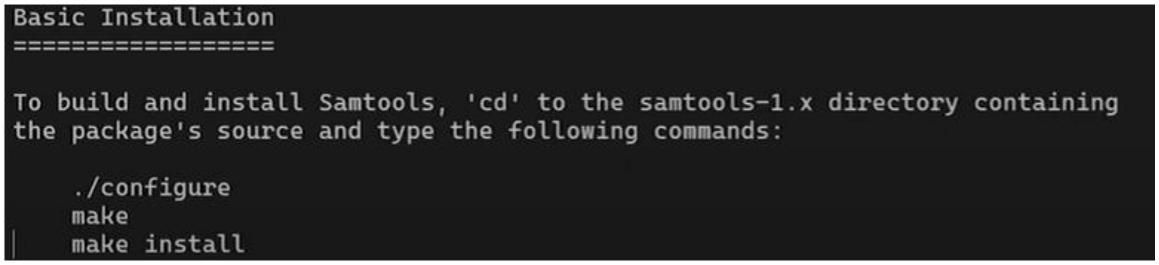
```
samtools sort -o Sorted_out.bam Output.bam
```

So, the conversion of Sam to Bam actually uses this kind of command. So, it uses this command: samtools view minus sp. Again, these options If you go to the manual, you can see what these options are doing, and then this is the notation: it gives the sam file name greater than sign and the output bam file name, ok. What the output bam file should be, ok? This is the first step; this is the conversion of sam to bam.

Then you have sorting of bam by coordinates, ok. So, again, this is the Samtools sort command, ok? And then you have the option minus o. So, here, the notation is slightly different. So, minus o means this is the output file name. So, sorted out, dot bam is the output file name, and the output dot bam is the input file, ok?

So, let us now go into the terminal; otherwise, you might get confused with all these commands, ok? So, we will start with this first one, which is the conversion of sam to bam. So, as I promised, right. So, inside these folders here at Samtools, you will have something called an install file, and this will describe everything that you require to install these programs, ok? So, these are the system requirements for the libraries that are required; you might not have all these libraries installed.

So, you may have to install them. You will find that the Samtools will give errors if you try to install them if some of them are not present in your system. So, some of these you can very easily install using pseudo-apt-install. We discussed this on the command line, right? So, if you for example, if you want to install this let us say zlib or libbz2, right you can very simply you can simply do this using this pseudo apt install libbz2, right.



```
Basic Installation
==================

To build and install Samtools, 'cd' to the samtools-1.x directory containing
the package's source and type the following commands:

    ./configure
    make
    make install
```

So, that kind of option is there. So, the errors will tell you what you have to do, what kind of requirement there is, and what kind of packages you need to install, ok? So, here are three commands that you need to execute. So, the first one is dot-slash-configure. So, inside the samtools folder, you have to go through this dot-slash configuration, right? This will configure the program, then you have to make and then make install.

So, those who are familiar with the Linux installation process will know that this is kind of a very common step for many programs, ok? So, this is what you would have to do again. I have already installed it because otherwise it will take a bit of time to install these programs. So, once you do that, you can then add it to the path. Okay, that's what I just showed you. And similarly for bcftools, you will also find this install folder inside this folder and it will again list out what you need, how you should process it, and what kind of system requirements are there.

And finally, it will also show you these commands, right? So, you have to move into the bcftools directory after extraction. So, cd then run this dot slash configure and make, ok? So, these two commands are enough; you do not have to do them again to install, ok? Once you have done this, you can add it to the path directory if you have administrative access and you can call those commands from anywhere in the system.

So, I hope that clears up how to install these tools. There could be some errors you probably notice. I mean, this is something that might happen and again, there are workarounds for those problems, right? So, if it is simply some library missing, etcetera, you can install those yourself, ok? So, now we can move into the analysis part itself, right after we have done the installation, ok? So, the analysis part is the first step, which is the conversion of the SAM file to the BAM file.

So, the notation is this, right? So, samtools minus sp output dot sam greater than sign and the output file names should be in BAM format, right? So, you can see what these options are on the site. So, if you go to let us say documentation, right and how tools, right this will under this you will find all these tools that are here manual pages and you can go to the samtools manual page and it will show you what these commands do, ok?

So, right. So, for example, yeah. So, the first command that we are using is the view command, right? So, the view command is here, ok, and if you click on this view command, you see these options: ok minus p minus h, etcetera, right? So, with these kinds of options, you can go through them and see what they actually do, ok? Of course, we will not have time to discuss all these options, etcetera, but you can actually find them out in detail, ok?

So, the command is sorry, samtools view, right, view minus sp. So, this will generate the BAM file, ok? So, we have generated the BAM file. You can check, right, whether we have the BAM file, and yes, the BAM file is there. We cannot see the BAM file; this is a binary format, right? We cannot see or read the BAM file using Vi, but we can use Samtools to actually read or at least see some of the data in the BAM file.

So, the command is again samtools view rput. bam, but we do not want to see the full, right? We just want to see the starting part, let us say, right? So, we just type this. So, again, I will repeat the command samtools view rput.

bams slash head, right. So, this head gives you the first few lines of this BAM file, right? And again, this is very similar to the SAM file; of course, we cannot read it using Vi. So, this is the first step done, right?

We have generated the BAM file. Now we have to sort this out, ok? So, if you go to the presentation, sort this BAM file by coordinates. So, there is a function called samtools sort.

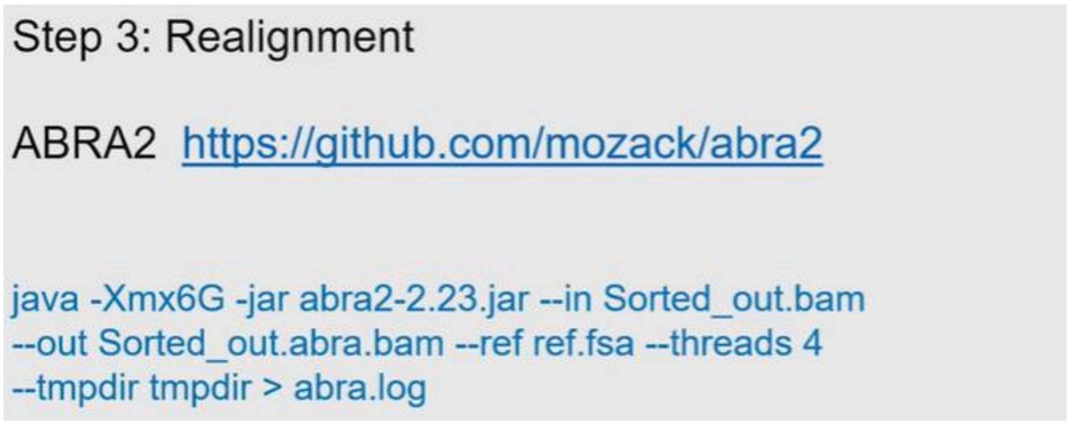
We need to give the output name, right, and then the input file name. So, the input file name is output.

bam, and the new output file name is sorted out. bam, ok. So, remember, right after minus, the output file name will come. And again, you can see here, right, that there is this sort option, ok? And how we call this is also given here, right? So, how do you call this, and then again, if you click on it, you will find all the options that are available?

So, let us now go there. Samtools sort minus o sorted out dot bam output dot bam, ok. So, we have given the output file name after the minus o, right, and this output dot bam is the input file from the last step, right. So, we converted this and it will run, and then we will get this sorted out, right, and again we can see this by Samtools view and head, right, ok. So, what we will notice now compared to the other ones, right, in the earlier file is that this is now sorted according to the chromosome here, right? You can see this in most of these reads; they start with this chromosome number, and position-wise they are also sorted, right?

You have this 274, 275, 282, 300 range; they are sorted, right? So, this is sorted by coordinates, ok? That is what this command will do by default. We will use another type of sort a bit later, and you will see you can also sort by reading names, ok? So, that will be useful for the next steps, ok? So, once we have sorted, right, we can go back to the presentation, right, and we have to do something called indexing, ok?

So, why do you need this indexing? You see, in a moment, for the realignment program, it will require this index file of the bam files, ok? So, we will use this sorted-out dot bam for our realignment purposes, ok? So, realignment will require this index file for the map. In addition, if you want to visualize these bam files, you also need the index file.



Step 3: Realignment

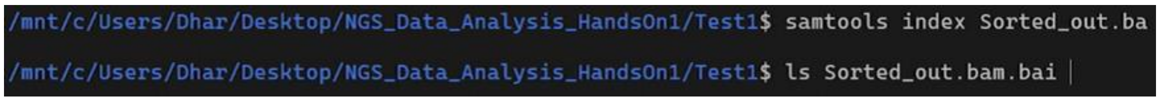
ABRA2 https://github.com/mozack/abra2

```
java -Xmx6G -jar abra2-2.23.jar --in Sorted_out.bam
--out Sorted_out.abra.bam --ref ref.fsa --threads 4
--tmpdir tmpdir > abra.log
```

So, coming to the next step, which is the realignment, ok. So, realignment cannot be done with SAMTOOLS or BCFTOOLS. It does not have any such function, but we can use another standalone tool called ABRA2. So, here is the link where you will find this package, which you can download and install, and you can use it, ok? So, it is very simple. You just download one single file, the compiled version, and you simply run run, ok.

It requires some, and this is the command that we will use. Again, I will explain what this command actually means, and we will run this command. So, again, I have compiled this because it will be much faster to complete this process, ok? You can find out what each of these commands means, and I can show you there is a page where these commands are also given and how you should run them. So, let us now go to that, right? So, let us build the index and then do the realignment, and we will follow that with the next step, ok?

So, let us go to the terminal, ok? So, for building indexes, we simply use, so, sorry, Samtools index sorted out dot bam, ok. So, the index file you will see will be created, and you will see that it ends with dot b a i. So, this is the index file of this BAM file, ok? So, whenever you run this index command, it will create this dot bam dot bai, ok?



```
/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ samtools index Sorted_out.ba
/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ ls Sorted_out.bam.bai
```

you can see

So, this file has the same data; this is the index file, ok? So, this index file has been created now. We can then go for the realignment, ok, and we will do the realignment using the ABRA2 package and I will show you the package where it is. So, this is the github link, right, where you find this package, ok? And from here, you can download this, right?

So, on the right side, you can see this release, right? I can download version 2.23, which I will use for this analysis, and down here, if you scroll down, you can see the commands that you have to use. And this is the use; this is the command, and it gives you the input file, the output file, the reference sequence, etcetera, and some of the temporary directories that So, this is again explained in detail; you can read through it and understand how you should run this command, ok? So, I will just simply run that command. Before that, we need to actually see whether we have this temporary directory already created. If not, we need to create it, and we also need to see this file. This is the program that will execute whether this file is installed, ok?

So, here we go. Yeah, so we have this version 2.23, right? So, which will we use for our analysis? So, for the sake of time, I will simply copy this command here, and I will explain in a moment what this command actually means. So, we are calling this program ABRA 2; this is version 2.

23, right. This xmx6g means we are limiting memory usage to 6 GB. Again, you can reduce it; it will, of course, take more time depending on your system, right? If your system has only 4 GB, you might want to say, Do not use more than 2 GB. So similarly, in this system, I will use 6 GB. I will say, OK, we can use 6 up to 6 GB. Then you have the minus-minus in this is the input file name, right, on which you want to run the realignment.

Then this is the output file name, right? We have given an output file name sorted out as dot abra dot pm. Then you have to give the reference genome sequence, ok? So, I will give you the reference genome that I have copied here.

This is the S28HC reference sequence that we used for mapping, ok? Then minus minus thread. So, this is for speeding up the process depending on the system that you are working with and minus-minus tmp date. So, this is a temporary directory that is required for storing some processes. Right, the program for this command, ok. So, I am not sure if we have created this temp directory yet, right?

So, what will I do? I will check again, right? I will cancel this command. I will not run this. What will I do? Let us check whether we have created No. So, it is not there, right?

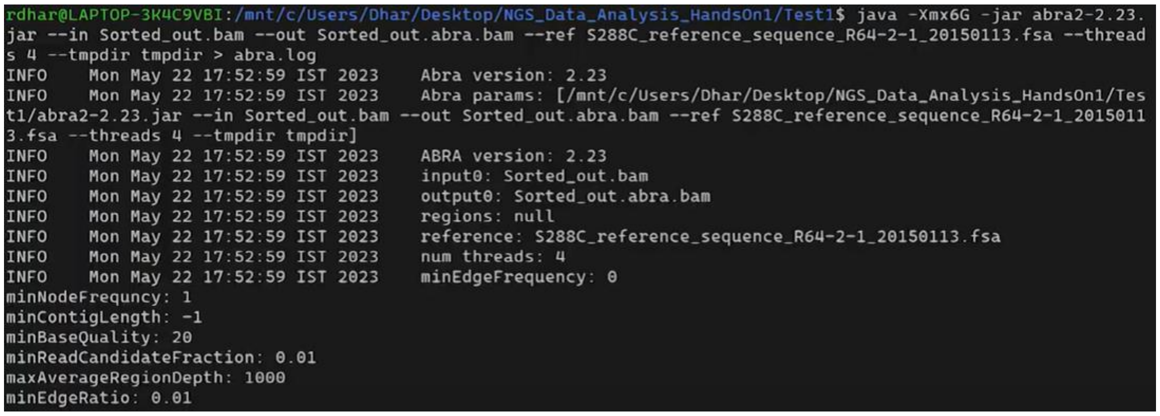
So, I will create this directory, and then now I will run this command, ok? And ok. So, maybe it is okay. So, maybe during the copying, it did not work out very well.

So, I will run it again. Let us see, right? It is 4. So, I am seeing only part of it, right? So, let us put the full command again: minus, ok. So, here we are, right? We are running this command minus xmxxg, limiting the memory usage, input file name, output file name, reference file name, threads, temporary directory, right, and the log file that will be created.

Sorry. So, the command is not correct again. So, it is not; it is actually something to do with the copying part when you copy the command. So, it should be Java, right? So, let us, I think, start from here again, right?

We think there is an issue. So, we need to provide the reference file name. So, I have just copied the command. We need to specify the reference file name that is stored here, the estimated reference.

Now, it is running. You can see, all right. So, here, it will be done in a moment. And once this is done, we will get the alignment done, and we can go for a duplicate mark within the samples, ok? So, when it is running, let us talk about the duplicate marking part, right? It is running in the background. So, in the duplicate marking, we have to first sort reads by name, right? So, when you have to mark duplicates, what will this program do? It will look for pairs and reads, right, and their locations, and then compare them against other pairs and reads, right.



```
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Test1$ java -Xmx6G -jar abra2-2.23.
jar --in Sorted_out.bam --out Sorted_out.abra.bam --ref S288C_reference_sequence_R64-2-1_20150113.fsa --thread
s 4 --tmpdir tmpdir > abra.log
INFO    Mon May 22 17:52:59 IST 2023    Abra version: 2.23
INFO    Mon May 22 17:52:59 IST 2023    Abra params: [/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn1/Tes
t1/abra2-2.23.jar --in Sorted_out.bam --out Sorted_out.abra.bam --ref S288C_reference_sequence_R64-2-1_2015011
3.fsa --threads 4 --tmpdir tmpdir]
INFO    Mon May 22 17:52:59 IST 2023    ABRA version: 2.23
INFO    Mon May 22 17:52:59 IST 2023    input0: Sorted_out.bam
INFO    Mon May 22 17:52:59 IST 2023    output0: Sorted_out.abra.bam
INFO    Mon May 22 17:52:59 IST 2023    regions: null
INFO    Mon May 22 17:52:59 IST 2023    reference: S288C_reference_sequence_R64-2-1_20150113.fsa
INFO    Mon May 22 17:52:59 IST 2023    num threads: 4
INFO    Mon May 22 17:52:59 IST 2023    minEdgeFrequency: 0
minNodeFrequncy: 1
minContigLength: -1
minBaseQuality: 20
minReadCandidateFraction: 0.01
maxAverageRegionDepth: 1000
minEdgeRatio: 0.01
```

So, that is why you need to first do this sorting by name. Then, we have to use something called the fixmate command, right? So, this will fill in mate coordinates and insert size, right? So, it will compare during duplicate marking, right?

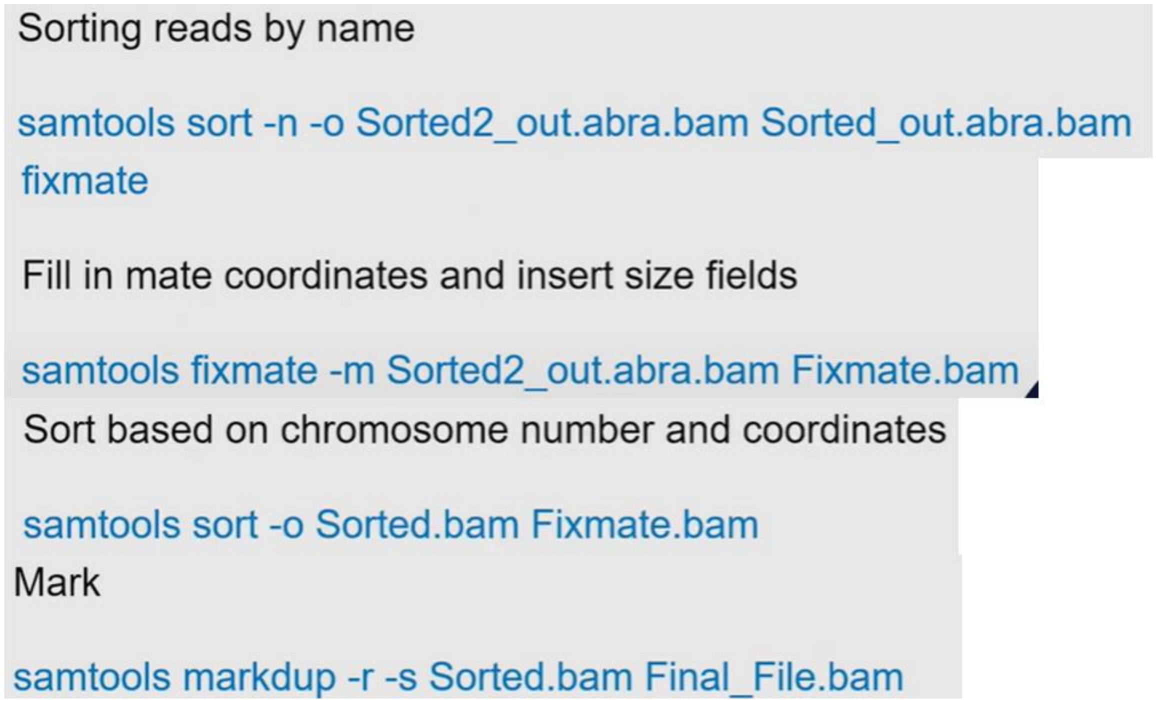
It will compare. The program will compare these mate coordinates and insert sizes right before it marks them. And after that is done, right, so we have to again sort based on chromosome number and coordinates and these will be used, right, so for the actual marking step, ok. So, there are actually four steps right here. So, first, we have to, after this realignment, sort reads by name, right, using this command.

And you see, we have used this sample sort. We simply need to use this minus n option. Once we have done that, we have to use this fixmate command. So, it will identify the mates, fill in the mate coordinates, and then calculate the insert sizes, etc. And once this is done, we have to again sort based on chromosome number and coordinates because these duplicates will be identified by this position, by their positions in the reference sequence, right? So, as we have discussed, they should start at the same position and end at the same position.

And finally, the actual marking—right, this is the mark to command that we can execute, ok? So, let us see if the realignment is done, and then we can do this step by step, ok? So, this realignment is done, as you can see. And let us find out if this file has been created. Yes, this has been created, ok?

So, we cannot right now see what kind of changes have been made because we have to visualize those changes. Now what we will do is run this actually, right? So, we will sort by the read names, ok, with the option minus n, right? That means the read names or by the name. And then, if you go to the options for SAM tools, you will find this, ok?

And it should be done in a moment, ok? So, the new file is now sorted to out dot abra dot bm, ok. So, with this bam file, we can now go through the next step, which is the fix, right? So, again, this is the command, right? So, here is the SAM Tools fixmate; the option is minus m.


Sorting reads by name

```
samtools sort -n -o Sorted2_out.abra.bam Sorted_out.abra.bam
```

fixmate

Fill in mate coordinates and insert size fields

```
samtools fixmate -m Sorted2_out.abra.bam Fixmate.bam
```

Sort based on chromosome number and coordinates

```
samtools sort -o Sorted.bam Fixmate.bam
```

Mark

```
samtools markdup -r -s Sorted.bam Final_File.bam
```

So, it will actually fill in the mate coordinates. So, that is why the minus m option is required. Again, you will find this in the SAM tools manual. Of course, if you have to go through all this, it will take much longer if you have to find out what these options are doing. So, I am simply copying these options that we have that I found out about early, ok?

So, this is done. So, we can see that fixmate dot bam has been created, ok? And once this fix is done, we can now sort based on the chromosome number and coordinates. So, that is the default sort here, right? So, this is the SAM tool sort minus sorted dot bam, which is the output file name that you want to create now from the fixed dot bam, which is the input file name, ok?

And it will run in a moment, right? So, see if this is done, okay? So, here it is: the right-sorted dot bam is there. And we can now finally do the actual duplicate marking, ok? So, let us do this, right? So, let us do the duplicate marking here.

And this is the command-mark loop, right? And minus r means to remove those duplicates, and minus s means to create small statistics. And the sorted dot bam is the input file name, and then you have the final file dot bam, which is the output file name. And this is very, very quick, as you can see that there are only a few duplicate pairs that have been identified, right?

The duplicate total is about 38. So, that is okay. It is like, out of the many reads, we have like 50000. So, 25000 paired ends, right? So, we have to throw out about 38. So, that is not a big number for this data. But if you are going with bigger data sets, these numbers will go much higher.

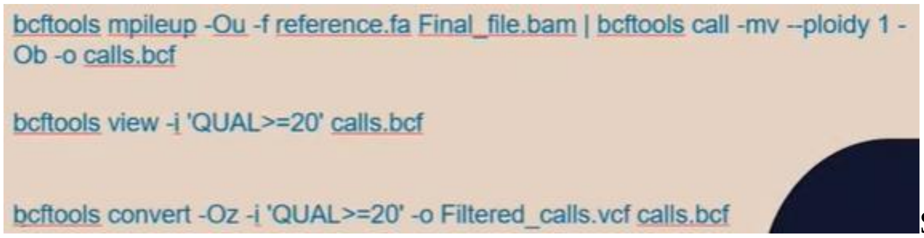
The next step is BQSR. Of course, as I mentioned, we will not discuss this. And I will just point you to this program, BaseRecalibrator, that actually does this process for you. You can install this tool and try it yourself. And once we are done, the final step is the variant calls, and this step will be done with BCF tools, ok?

So, there are two parts here. As you can see, the first part of this command is bcftools.mpileup. This is what you have to do to the final file that we generated, the final BAM file. And again, you have these options that actually mention what is happening here. And then you have the second part, which will actually do the variant calls, ok? So, in this part, you will see this minus mv and then something called minus minus ploidy 1, ok.

So, we are working with the haploid genome here, not the diploid genome. So, we have to mention this; otherwise, it will consider this a diploid genome. And it will generate this output in BCF format, and the output file name will be dot bcf. So, minus o is the output file name, so call dot bcf. So, that means you can find these options under the BCF Tools manual page, right?

What these commands are doing, how you should use them, what options you should use, and what these options mean, ok? So, I am simply taking these commands, and I will run them here, ok? So, we need to define the reference file again, right? So, the reference file name would be, let us see, this is the reference genome that we used for mapping, and it should be done very soon, ok? Now one of the things it will tell you, right? This number of reads per input file set to minus D 250 means it is taking maximum date depth as 250 or maximum coverage for each position as 250, ok?

So, that is fine. Then you have, once you have generated this calls dot bcf, you can actually filter out or visualize this bcf file saying bcftools view, and you can filter saying like I want only those variants that for which you have a quality score greater than equal to 20, ok. So, here it is, and once we type this, you see some of this list, ok? So, this is really cumbersome, right? So, what we want to do is write this in a file, okay, and this is how we do this, right? So, write this in this format called vcf dot vcf, and we can then read that file and see what these mutations are, what these variants are, and what these types of variants are, right?



```
bcftools mpileup -Ou -f reference.fa Final_file.bam | bcftools call -mv --ploidy 1 -
Ob -o calls.bcf

bcftools view -i 'QUAL>=20' calls.bcf


bcftools convert -Oz -i 'QUAL>=20' -o Filtered_calls.vcf calls.bcf
```

So, whether they are SNPs or indels, etcetera, ok. So, again, the command is vcf tools convert, and we are filtering by this quality score. The output file name is filtered dot filtered calls dot vcf, and this is not vcf; this is vcf, and then the original input file name, right from which we are creating this output file. Why are we writing this CVF file? Because this is a human-readable format. So, we can read it ourselves, right? We just simply open it with vi, right, and we can read all the data that is there, and you can again use this colon now. Rap, you can see these variants that have been identified from these data sets, ok?

So, we will talk about this vcf format in the next class, this vcf and vcf, what these are and how this data is stored. So, we will discuss that in the next class. So, once you have done this up to this step, the final step that is remaining is the visualization step. So, we want to do something called a manual check where we want to identify these variants in the data and see whether they are actually there, and we will use this tool called the integrated genome browser. So, that will be for the next class. So, we have learned how to do the variant calling ourselves, and we have seen that this is a really multi-step analysis, and of course, within a very short period of time, it is not possible to show all the steps and all the options that are available, and we can also customize these options based on data sets and requirements. Thank you.