**Setting up a new workstation**:

Firstly, open the command prompt or linux shell and run:

ssh [Kerberos-id@baadal.iitd.ac.in](mailto:Kerberos-id@baadal.iitd.ac.in)

Enter the kerberos password

After logging in, run:

ssh ip address

Enter the password and type yes.

**Alternate:** To save your time from doing login again and again just download an application called mobaXterm, which is free of cost. Extract the installer zip and install the application.

Steps to setup a new workstation:

1. Click on the session option present at the top left corner.
2. Now click the ssh session type.
3. In the remote host, enter the ip address of the server like 10.222.72.25. Check the specify username and enter the username like genome1.
4. Click on the network settings option displayed on the same page.
5. Now click on the SSH gateway (jump host) which pops up a new window.
6. In that window fill baadal.iitd.ac.in in the gateway host, Kerberos id in the username and check the use ssh key option and finally click ok.
7. Enter the password for the server and for the Kerberos id and click on save password.
8. Now to open the server, click on to the ip address listed on the User sessions dropdown menu and you are good to go.
9. To restart the session type R.

**Setting up a new conda environment:**

**Installing conda:** Firstly, download the .sh file into your computer.

Now via terminal go to your download directory and run the .sh file using the command:

bash Anaconda3-2020.11-linux-x86\_64.sh , this will start the installation.

**Creating a new conda environment:**

conda create -n environment\_name

**To activate the environment:**

conda activate environment\_name

**To deactivate the environment:**

conda deactivate

**To enlist all the conda environment:**

conda env list

**Note:** In the current workstation, conda is already installed and working perfectly. So, please do not update the current version on your own.

**Downloading a new dataset:** For this purpose, we will use the SRA toolkit.

To download SRA toolkit in our conda environment:

conda install -c bioconda sra-tools

To download a new dataset from SRA just type the command:

prefetch accession\_number (download the file in SRA format)

Converting the SRA file into fastq format:

fastq-dump accession\_number.sra (converts the SRA file into fastq format)

Note: This method is only for single end data and if we use this method for pair end data, we will use another script.

**For Pair End Data:**

**input=$1 (**$1 is describing the 1st positional argument)

**for SRR in `cat $input`** (For loop will search for the SRR term)

**do**

**/home/genome1/mitochondria\_assembly/tools/sratoolkit.2.9.6-1-centos\_linux64/bin/fastq-dump.2.9.6 --gzip --skip-technical --origfmt --readids --read-filter pass --dumpbase --split-3 --clip -A $SRR (** /home/genome1/mitochondria\_assembly/tools/sratoolkit.2.9.6-1-centos\_linux64/bin/fastq-dump.2.9.6 is the pathway of fastq-dump tool.

gzip option is zipping the file and we get the file in .gz file format.

The --skip-technical option tells the tool to not bother downloading the barcodes and primers.

-–origfmt option is used to keep the data file in the original format.

--readids option disables the clipping of read ids.

--read-filter pass filter out reads that are all N’s or otherwise completely useless and pass other reads.

--dumpbase option is useful for Illumina reads as this option converts color codes into the A, T, G, C.

--split3: This option splits the pair end data into three separate files, one containing the forward reads, second containing the reverse reads and third containing orphan reads( these are the reads which are non-paired.

--clip: It is a default option.

-A $SRR: This signifies that the compressed files are going to store in the SRR directory address.

**echo $SRR (**Prints the content of SRR directory on the screen.)

**echo 'retrival\_successful\_FASTQ DUmped'; (**Prints retrival\_successful\_FASTQ Dumped on the screen)

**done (**end of for loop)

**Note:** SRA toolkit can be used for downloading both SRR and ERR files. The problem using this script is that as we know at IIT Delhi we work behind a proxy wall and because of this often connection gets timed out, so it is suggested to run this script using nohup command.

**Code:**

**input=$1**

**for SRR in `cat $input`**

**do**

**/home/genome1/mitochondria\_assembly/tools/sratoolkit.2.9.6-1-centos\_linux64/bin/fastq-dump.2.9.6 --gzip --skip-technical --origfmt --readids --read-filter pa$**

**echo $SRR**

**echo 'retrival\_successful\_FASTQ DUmped';**

**done**

**Running an alignment:**

**#!/bin/bash** (tells the OS to invoke the specified shell to execute the commands that follows in the script.)

**raw="/home/genome1/mitochondria\_assembly/raw"** (setting the path to a new variable raw)

**hg38="/home/genome1/mitochondria\_assembly/raw/index/GRCh38.primary\_assembly.genome.fa"** (setting the reference genome fasta sequence to the hg38 variable)

**#mkdir $raw/alignments** (making a directory alignment inside the raw directory)

**ls -1 $raw/\*.fastq.gz | grep 'pass\_1' >> $raw/alignments/fastqlist** (This command is consisting of three steps. In first step the ls command is listing all the files inside the raw directory having .fastq.gz format which is then passes to the second command grep. The grep command only selects files having pass\_1 on their name. All the files selected using grep command is then stored inside the fastqlist directory.)

**while read p** (while loop reads each fastqlist row contents one by one via p variable)

**do**

**nam=$(echo $p | xargs -n 1 basename | cut -f 1-2 -d"\_");** (echo command will print the p variable, xargs takes the printed value as an input and -n 1 option is specifying that the number of arguments that are read from the standard input is limited to 1. Cut -f 1-2 combine and cut the 1 and 2 portion of the file before \_. This result is then stored in the nam variable.)

**bwa mem -t 8 -M $hg38 $p $raw/${nam}\_2.fastq.gz | samtools sort -@8 -o ${nam}.sorted.bam - 2> $raw/alignments/${nam}.sorted.stderr**

**(**bwa is an aligner tool and mem option is ensuring maximal exact matches. -t option define the number of threads which is 8 in this case. Some reads partially aligns with two different chromosomes and -M option (stands for mask) is used to mark such reads as secondary. The .fastq.gz file is aligned to the reference genome and the BWA outputs alignment in the new standard SAM format. The output SAM file is then taken as the input into the samtools which then sort the sam file coordinate wise. The @ option is representing the number of sorting and compression threads which is set to 8 in this case. The output of this command is stored in .sorted.bam file format and if the samtools is giving an error it is stored in the alignment directory in .stderr file format. File descriptor 2 represents the standard error. The output of bwa tool is dumped into the USER/stdin and – is representing that the standard input is given to the samtools.)

**done < "$raw/alignments/fastqlist"** (done command is used to close the do while loop and “$raw/alignments/fastqlist” is the target folder from which the while loop is taking contents).

**Code:**

**#!/bin/bash**

**raw="/home/genome1/mitochondria\_assembly/raw"**

**hg38="/home/genome1/mitochondria\_assembly/raw/index/GRCh38.primary\_assembly.genome.fa"**

**#mkdir $raw/alignments**

**ls -1 $raw/\*.fastq.gz | grep 'pass\_1' >> $raw/alignments/fastqlist**

**while read p**

**do**

**nam=$(echo $p | xargs -n 1 basename | cut -f 1-2 -d"\_");**

**bwa mem -t 8 -M $hg38 $p $raw/${nam}\_2.fastq.gz | samtools sort -@8 -o ${nam}.sorted.bam - 2> $raw/alignments/${nam}.sorted.stderr**

**done < "$raw/alignments/fastqlist"**