PB Project Report

Group number: 2

Aditya Singh 2018378
Amritpal Singh 2018379
Bhavay Aggarwal 2018384
Hardik Saini 2018391
Saad Ahmad 2018409
Sejal Singh 2018413

Introduction

As the new technologies are emerging every day, it is hard to keep track of what tool in bioinformatics are getting updated, what input they take, what the commands that one need to execute, what are the flags that need to be passed along, there are tens of extension for same files, which is the one should choose during alignment, wouldn't it be nice if there were some tools which do all those things on a single click, keeping that in mind we introduce **Automated SRA download and alignment pipeline.**
In which you just have to put GSE ID (with some prior configurations on your server), and it will download the SRA data and perform alignment following all the standard protocols, moreover, we introduce android application also using which you can do these things on your server by putting ssh details and it will execute the job on your ssh even when you are somewhere in a remote location away from the server.

Assumptions

1. **We consider a single organism in a experiment only**
2. **Fasterq-dump doesn't download files using FTP, instead, it downloads intermediate cache files which for some reads may end up taking huge amounts of space on HPC, and sometimes this results in an error (ran out of space) and the downloading not getting completed.**
3. **Exon specific expression and alternative splicing is not implemented for count matrix.**
4. **We assume that all fastq files have phred 33 scores**
5. **We do not take into account the strand specific experiments**
6. **Downloading the latest patch of a specific build of a genome**

Methodologies

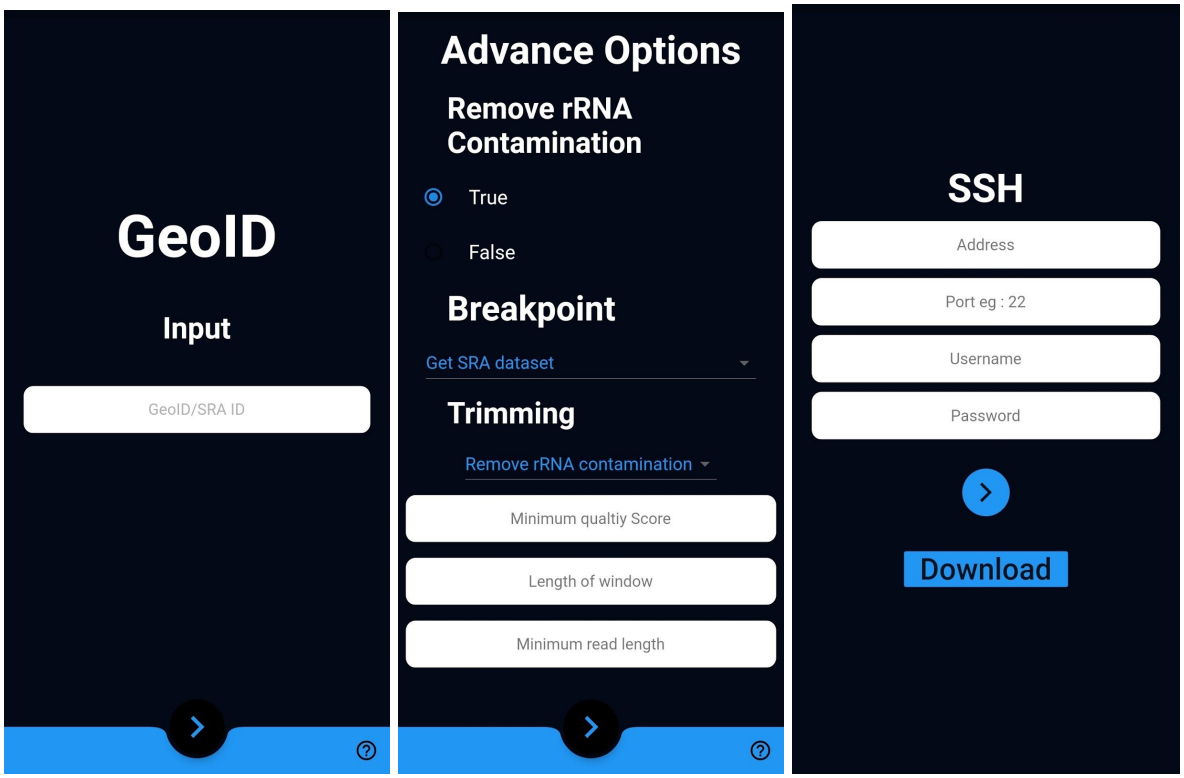
Pre-requirements
Users must set up an OpenSSH server on their server machine, in order to take benefit of the android application.

Application
The application is created on Flutter.
Flutter Packages used in the Application:
SSH
Permission Handler
Path Provider
Why we used these packages

- **SSH**
  - It is used as a bridge between the application and the host machine.
  - We used this package while initialising the code to the machine from our application and downloading the output file using SFTP when the process is completed.
- **Permission Handler**
  - For the file to get downloaded in our mobile we require Storage permission and this package deals with it.
- **Path Provider**
  - It provides us with the path to where our output file is downloaded.

Working of the Application

- The first page is where we fill in the GeoID/SRA ID as an input.
- The second page consists of various options.
- The third page is where the magic happens. It consists of all the inputs provided on both pages. On this page, we provide our SSH details.
  - Execution: When we click on the arrow button which provides our SSH details to the SSH package which executes our command which is being generated according to the inputs we entered/selected on page one and two. These inputs act as a parameter in the command. When the process completes, we get a notification on our device regarding that.
  - Download: The command we provided creates an output file of .html format. To get that file to our device we provide our SSH details to the SSH package and click on the Download button which then downloads the output file to our mobile phone. We have to provide Storage permission to this application for it to save the output .html file into our mobile phones to a path provided by Path Provider package.



Requirements on the Server-Side (Not working right now because of execution in sh shell, paths are hardcoded in python script)

The server will contain python script in the home directory of the user which is used for ssh login

Create environment variable PB_PROJECT=<to_your_python_virtual-env>

- Example: PB_PROJECT=$HOME/saad-env/bin/activate
- TRIMHOME=$HOME/Trimmomatic-0.39 {or whatever your version of Trimmomatic installed}

Download executable binaries of the following

- Edirect
- Hisat2,
- fastQC
- Trimmomatic
- Sortmerna
- sratoolkit

Append the path to these binaries in $PATH environment variable

- Example: export PATH=$PATH:$HOME/FastQC

Additionally, these all are the packages that must be installed on python virtual environment, which was set up above

- Biopython
- Pysradb
- Beautifulsoup
- Pandas

Why we used Hisat2 and fasterq-dump

- **Hisat2**
    - It is a splice aware aligner, what this means is that it can somehow figure out where an intron is and does not unnecessarily disregard a read when it is corresponding to two exons which are separated by an intron in the genome.
    - Thus this would also help with longer reads
    - It is faster and more memory efficient than its contemporaries (primarily STAR which reports slightly better results on low quality reads)
    - Hisat2 handles both WGS and transcriptomic data as well

Fasterq-dump and prefetch

- Although we would have loved to write a custom script to download data using ftp with curl/wget, we did not because that would have required extensive texting and also there was time constraints
- Fasterq-dump is considerably stabler and faster that its legacy version fastq-dump. The defaults on fasterq-dump command are much better set. However it still sufferers from the caching problem (downloading intermediate files and reading them which often caused errors on our systems.
- We use prefetch to download data in .sra format which is a faster and more memory efficient way to download sra data.

Working of Pipeline

- Input to our pipeline is GSE ID (GEO Experiment ID, example: GSE146443) this is given on android application & the breakpoint i.e. where to stop the whole process
- Next, the application will ask the user to put ssh details, in order to execute a job on the server
- On Succesful execution, two-parameter were passed to the python script, namely GSE_ID & Breakpoint
- **Getting hands-on Fastq Files from GSE ID (End Point: fastq files)**
- Using esearch (edirect-utility) we extract SRP_ID from GSE_ID.
- Then using SRP ID in pysradb (python) we get metadata for SRP ID
- Using metadata pysradb return a data frame of that metadata this data frame contains SRR ID
- These SRR ID's can be used to download SRA Samples using fasterq-dump resulting in **fastq-files.** If the user only wants .sra files then prefetch is used.
- **Generating Quality Report (End Point: HTML files)**
- These above downloaded fastq files are input to fastqc which generates **HTML report file** which users can see on their android application, the app supports download/fetch quality report from the server.
- **Pre-Processing fastq files (Optional, End Point: Trimmed fastq files)**

- The user gets an option to preprocess the fastq-files if desired these above downloaded fastq files can be trimmed by using trimmomatic or sortmerna which results in **fastq files** with trimmed off bad quality the option to use trimmomatic or sortmerna is also provided in frontend
- The users also get the new quality report of the new sample after pre-processing
- **Getting the Genome build name which was used in Experiment (End Point: txt genome build version)**
- Using GSE ID with esearch we can get GSM ID (Scrape/Extracting)
- Then using this GSM ID with esearch we scrape the NCBI sample page & scrape the **genome build**
- **Downloading Reference Genome/Index (End Point: Hisat2 index files)**
- **There are three cases**
- **1.** Pre-built index already exists for that genome build (we scraped above)
- In case we got the genome build hit on scraping the NCBI website, download that **hisat2 pre-built index** that is hosted on the website using wget along with annotations
- **2.** Genome build is mentioned in GSM ID Page of the experiment on the NCBI site
- In case we got the genome build hit on scraping the NCBI website, but pre-built hisat2 index isn't hosted on the website then, using Genome build (scraped above) along with esearch (assembly database) we get Refseq FTP Path to that build
- The using wget along with FTP-path to download that reference genome + annotations
- Then use hisat-build to create a .**ht2 index file**.
- **3.** Genome Build isn't mentioned on the NCBI Webpage (Scraping doesn't result in hit)
- In that case, We download the latest reference genome for that,
- first, we get organism name using pysradb (metadata searched above)
- Then we search that organism name in esearch assembly database (CLI) and scrape the output and get the latest RefSeq genome build FTP-path
- The using that FTP-path along with wget to download the latest reference genome build + annotations
- Then use this downloaded genome to create **hisat2 index** file using hisat2-build
- **Generating Count Matrix (End Point: Count Matrix)**
- The above-sorted bam files are given as input to feature-counts which results in **count matrix**

Alignment of reads with genome/index (End Point: SAM Files)

Using hisat2 index files + fastq files from above alignment is done which results in sam files.

Processing SAM Files(End Point: Sorted BAM Files)

Then we convert sam files to bam files using Samtools

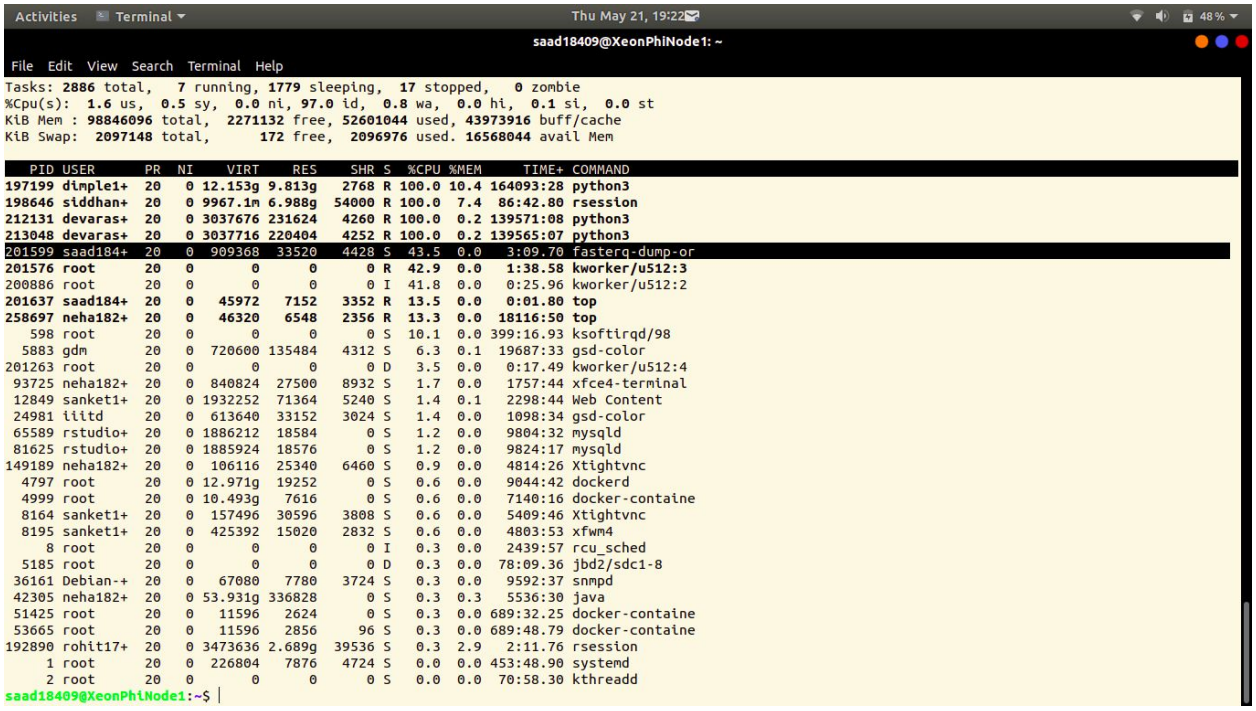
Then we sort the bam file using Samtools again which results in **sorted BAM Files**

Challenges (in decreasing order of difficulty)

1. Figuring out connectivity between the Flutter app and HPC and executing a nohup command through the app. Cannot read environment variables
2. Getting the correct reference genome and annotation.
3. Retrying commands after they failed to complete (most of the times this is because of memory issue like with fasterq-dump or hisat2-build)
4. Debugging in general

Results

This line of code is executed from the app and starts a nohup command on the HPC.



Using the top command we see the command has successfully been executed on the HPC and is now running the fasterq-dump command to download fastq files

```
saad18409@XeonPhiNode1:~$ cat report.txt
--2020-05-21 19:18:52--  https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM741695
Resolving www.ncbi.nlm.nih.gov (www.ncbi.nlm.nih.gov)... 130.14.29.110, 2607:f220:41e:4290::110
Connecting to www.ncbi.nlm.nih.gov (www.ncbi.nlm.nih.gov)|130.14.29.110|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: '/home/saad18409/experimentSample.html'

     0K ......... ......... ...                              94.3K=0.3s

2020-05-21 19:18:54 (94.3 KB/s) - '/home/saad18409/experimentSample.html' saved [24285]

join   :|-------------------------------------------------- 100%
|-------------------------------------------------- 100%
spots read      : 25,541,142
reads read      : 25,541,142
reads written   : 25,541,142
join   :|-------------------------------------------------- 100%
|------------------------------------------- 89.19%saad18409@XeonPhiNode1:~$
```

We save the output in a file named report.txt which shows the fastq files being downloaded

```
 18150K ......... ...                                     100% 36.6M=3.7s

2020-05-21 19:30:05 (4.83 MB/s) - 'GCF_000001405.25_GRCh37.p13_annotations.gtf.gz' saved [18599335]

FINISHED --2020-05-21 19:30:05--
Total wall clock time: 7.9s
Downloaded: 1 files, 18M in 3.7s (4.78 MB/s)
WARNING: combining -O with -r or -p will mean that all downloaded content
will be placed in the single file you specified.

--2020-05-21 19:30:05--  ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/GCF_000001405.25_GRCh37.p13/GCF_000001405.25_GRCh37.p13_genomic.fna.gz
           => 'ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/GCF_000001405.25_GRCh37.p13/.listing'
Resolving ftp.ncbi.nlm.nih.gov (ftp.ncbi.nlm.nih.gov)... 130.14.250.10, 2607:f220:41e:250::10
Connecting to ftp.ncbi.nlm.nih.gov (ftp.ncbi.nlm.nih.gov)|130.14.250.10|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD (1) /genomes/all/GCF/000/001/405/GCF_000001405.25_GRCh37.p13 ... done.
==> PASV ... done.    ==> LIST ... done.

     0K ...                                                   297K=0.01s

2020-05-21 19:30:08 (297 KB/s) - 'ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/GCF_000001405.25_GRCh37.p13/.listing' saved [3263]

Removed 'ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/GCF_000001405.25_GRCh37.p13/.listing'.
--2020-05-21 19:30:08--  ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/GCF_000001405.25_GRCh37.p13/GCF_000001405.25_GRCh37.p13_genomic.fna.gz
           => 'GCF_000001405.25_GRCh37.p13_genome.fna.gz'
==> CWD not required.
==> PASV ... done.    ==> RETR GCF_000001405.25_GRCh37.p13_genomic.fna.gz ... done.
Length: 943912753 (900M)
```

Annotations and Reference genome downloaded using wget

```
                                                        saad18409@XeonPhiNode1: ~
File  Edit  View  Search  Terminal  Help
Tasks: 2878 total,   6 running, 1775 sleeping,  17 stopped,   0 zombie
%Cpu(s):  1.9 us,  0.2 sy,  0.0 ni, 97.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 98846096 total,  3494288 free, 56852432 used, 38499372 buff/cache
KiB Swap:  2097148 total,        0 free,  2097148 used. 12189632 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
197199 dimple1+  20   0 12.153g 9.813g   2768 R 100.0 10.4 164109:13 python3
198646 siddhan+  20   0 10.330g 7.584g  53272 R 100.3  8.0 102:27.91 rsession
201780 saad184+  20   0 3851824 3.661g   4064 R 100.0  3.9   5:18.20 hisat2-build-s
212131 devaras+  20   0 3037676 245220   4260 R 100.0  0.2 139586:52 python3
213048 devaras+  20   0 3037716 254064   4252 R 100.0  0.3 139580:51 python3
201846 saad184+  20   0   45976   7232   3416 R  13.5  0.0   0:01.80 top
258697 neha182+  20   0   46320   6548   2356 S  13.5  0.0  18118:57 top
  5883 gdm       20   0  720600 135488   4312 S   4.6  0.1  19688:22 gsd-color
 12849 sanket1+  20   0 1932252  71364   5240 S   2.0  0.1   2298:59 Web Content
 26822 iiitd     20   0 3678820 1.532g  10336 S   1.4  1.6   2289:37 Web Content
 81625 rstudio+  20   0 1885924  18568      0 S   1.2  0.0   9824:28 mysqld
```

Now as seen on in the top command, building index has started (no pre-built indexes or ensembl annotations for the genome build)

The hisat2 index files being created as ht2idxes.*.ht2 (This failed to build due to memory issues which happen frequently, we had to download pre-built indexes for it and then run the later part of code)


```
16411841 reads; of these:
  16411841 (100.00%) were unpaired; of these:
    1437776 (8.76%) aligned 0 times
    9626479 (58.66%) aligned exactly 1 time
    5347586 (32.58%) aligned >1 times
91.24% overall alignment rate
15589765 reads; of these:
  15589765 (100.00%) were unpaired; of these:
    3619672 (23.22%) aligned 0 times
    7864037 (50.44%) aligned exactly 1 time
    4106056 (26.34%) aligned >1 times
76.78% overall alignment rate
19211406 reads; of these:
  19211406 (100.00%) were unpaired; of these:
    5797673 (30.18%) aligned 0 times
    7230523 (37.64%) aligned exactly 1 time
    6183210 (32.19%) aligned >1 times
69.82% overall alignment rate
```

Alignment summary

Conclusions & Future Work

- Extend our work to differential expression
- Extend our work to Gene ontologies
- Automate the workflow for the installation of required libraries/binaries on server
- Download pre-built index for refseq genome build IDs
- Add options for exome sequencing, strandedness, etc.
- Search by name/author of a research paper (possibly integrate search by image)
- Add option for more aligners based on user's requirements