



BIOMEDICAL DATA ANALYSIS

THD M.Sc Life Science Informatics - I



AMAN SATYENDRA YADAV

MATRICULATION NUMBER

22305265

aman.yadav@stud.th-deg.de

Biomedical Data Analysis

Why Biomedical data Analysis is important ?

Biomedical data analysis plays a pivotal role in enhancing our comprehension of human health, diseases, genetic mapping, and medical treatments. The progression of biomedical technology has expanded our capacity to tackle intricate biological challenges beyond the realm of disease analysis. This sophisticated field finds applications in various domains, including drug discovery, personalized medicine, the exploration of biological systems, gene sequencing, and the augmentation of scientific knowledge.

Reproducible Research

Reproducible research embodies the commitment to transparency and ease of replication in the research process and its outcomes for the benefit of others. The primary objective is to ensure that the results of a study can be independently verified and validated by fellow researchers. This can be achieved either through the utilization of the same dataset and methodologies or by applying similar approaches to different datasets.

Essential components integral to the concept of reproducible research encompass:

Data Accessibility:

The act of sharing the original dataset employed in the study facilitates others in conducting their analyses and corroborating the reported results.

Code Availability:

The provision of the code employed for data analysis, statistical modeling, and other computations empowers fellow researchers to replicate the precise procedures and analyses undertaken in the study.

Thorough Documentation:

Clear and comprehensive documentation of the research methods, parameters, and settings is imperative for understanding and replicating the study.

Version Control:

The utilization of version control systems such as Git proves invaluable in tracking alterations in code, allowing others to access specific versions of both the code and data.

Open Access Principles:

Disseminating research findings through open-access journals or repositories serves to make them freely available to the public. This practice promotes transparency and enhances accessibility to the research outcomes.

System Setup in Linux:

How to connect with the remote machine ?

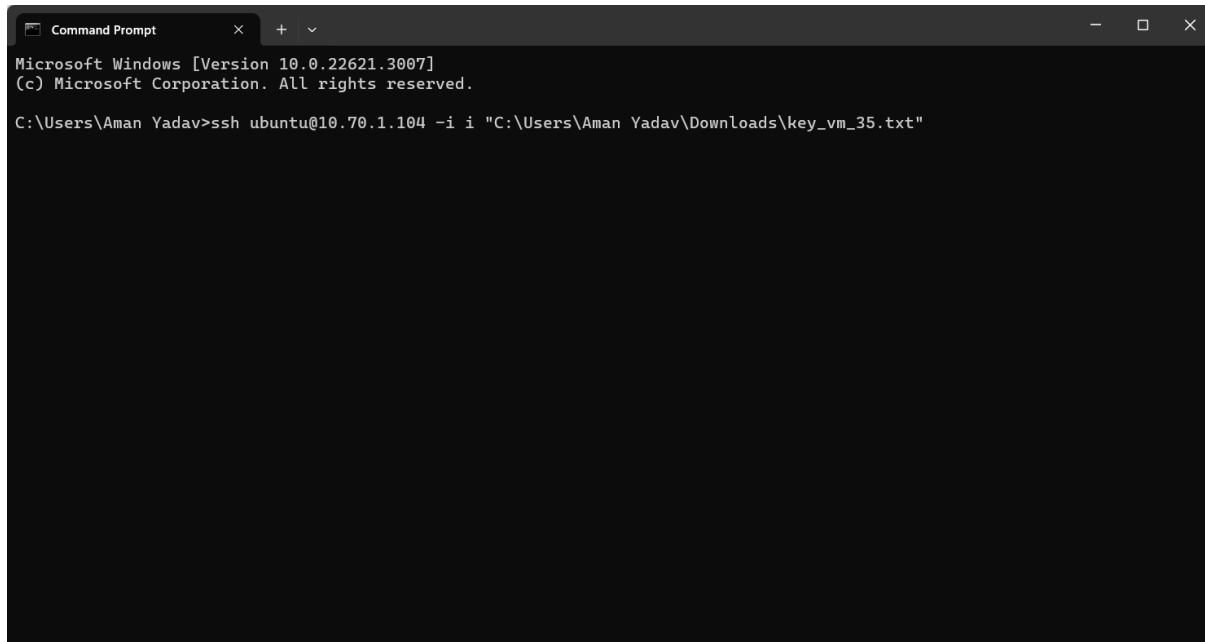
- 1) Connect to the VPN of Deggendorf institute of technology via Cisco connect.
- 2) Open the command prompt in the user computer. (Type CMD on the search bar)
- 3) Once the command prompt is open type
ssh ubuntu@10.70.1.104 -i "C:\Users\Aman Yadav\Downloads\key_vm_35.txt"

ssh – This command is used to establish connection with the main virtual machine.

Ubuntu - This is the username used to log in to the remote server

10.70.1.104 – It is the IP address of the remote server which we want to connect with.

i "C:\Users\Aman Yadav\Downloads\key_vm_35.txt" – Path of the private Key file which is required to connect with the virtual machine. Any one with the private key file can connect to your virtual machine



```
Command Prompt
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Aman Yadav>ssh ubuntu@10.70.1.104 -i "C:\Users\Aman Yadav\Downloads\key_vm_35.txt"
```

Creating a new environment .

Conda and Bioconda-

Conda: (Definitions of the topic is taken from internet and chat gpt)

Conda is an open-source package management and environment management system that runs on Windows, macOS, and Linux. It is developed by Anaconda, Inc. Conda simplifies the process of installing, managing, and sharing software packages in different computing environments. It is often used in the data science and scientific computing communities to create isolated environments with specific versions of packages and dependencies.

Bioconda:

Bioconda is a distribution of bioinformatics software for Conda. It is a channel (repository) within the Conda ecosystem that is dedicated to providing bioinformatics tools and packages. Bioconda aims to make it easier for researchers and bioinformaticians to install and manage a wide range of bioinformatics software, ensuring that dependencies are handled efficiently.

We will be using miniconda instead:

Miniconda is designed to provide a lightweight and efficient way to install and manage software packages and dependencies. Unlike the full Anaconda distribution, which comes with a large number of pre-installed packages, Miniconda allows users to install only the packages they need. This makes it a more streamlined option for users who prefer a more customized and minimal installation.

Step 1 : Installation :

The instructions to download miniconda is given on the official page.

<https://docs.conda.io/projects/miniconda/en/latest/index.html#quick-command-line-install>

Command lines can also be used to download miniconda in your virtual machine.

For windows :

These three commands quickly and quietly install the latest 64-bit version of the installer and then clean up after themselves. To install a different version or architecture of Miniconda for Windows, change the name of the .exe installer in the curl command.

```
curl https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86_64.exe -o miniconda.exe
```

```
start /wait "" miniconda.exe /S
del miniconda.exe
```

for Linux :

```
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
```

The first step of creating a new environment has been done by downloading miniconda.

Step 2 : Creating a new environment :

Creating a new environment in miniconda can be done very easily and quickly using a single command line .

“ conda create bio”

To check whether the new environment has been created or not then type. *This command will activate the environment “bio”*

“ conda activate bio”

```
bash
```

```
(base) $ # This is the base (root) environment
(bio) $  # This is the "bio" environment
```

It can also be deactivated by entering the command :

“conda deactivate bio”

To install package in the new environment enter the command:

“conda install package_name”

Jupyter Lab: (Only Definitions of this topic is taken from internet and chat gpt)

Jupyter Lab is an interactive development environment for working with Jupyter notebooks, code, and data. It is an evolution of the classic Jupyter Notebook interface and provides a more flexible and feature-rich environment for interactive computing. Jupyter Lab allows you to work with documents, code, and data in a more integrated and interactive way.

Jupyter Lab supports extensions, which are additional plugins that enhance its functionality. These extensions can provide features such as enhanced visualization, debugging tools, and more.

Intall jupyter Lab

conda install jupyterlab

Activate the "bio" environment in case it is not already active. First, we configure the Jupyter server.

\$ jupyter server --generate-config \$ jupyter server password

The terminal will ask you for a password. Type in a password and remember it. Next, the program asks to confirm the password. Start a server by using the following command:

\$ jupyter lab --no-browser --ip "*"

```
Last login: Wed Jan 31 22:52:07 2024 from 172.25.4.106
(base) ubuntu@lsi:~$ jupyter lab --no-browser --ip "*"
$: command not found
(base) ubuntu@lsi:~$ conda activate Bio
(Bio) ubuntu@lsi:~$ jupyter lab --no-browser --ip "*"
[I 2024-01-31 23:13:26.964 ServerApp] Package jupyterlab took 0.0000s to import
[I 2024-01-31 23:13:26.985 ServerApp] Package jupyter_lsp took 0.0201s to import
[W 2024-01-31 23:13:26.985 ServerApp] A '_jupyter_server_extension_points' function was not found in jupyter_lsp.
d, a '_jupyter_server_extension_paths' function was found and will be used for now. This function name will be dep
d in future releases of Jupyter Server.
[I 2024-01-31 23:13:26.993 ServerApp] Package jupyter_server_terminals took 0.0074s to import
[I 2024-01-31 23:13:26.993 ServerApp] Package notebook_shim took 0.0000s to import
[W 2024-01-31 23:13:26.993 ServerApp] A '_jupyter_server_extension_points' function was not found in notebook_shim
ead, a '_jupyter_server_extension_paths' function was found and will be used for now. This function name will be d
ted in future releases of Jupyter Server.
[I 2024-01-31 23:13:26.994 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2024-01-31 23:13:26.996 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2024-01-31 23:13:27.000 ServerApp] jupyterlab | extension was successfully linked.
```

Connect to jupyter lab

**"ssh -i "C:\Users\Aman Yadav\Downloads\key_vm_35 (1).txt" -L 8888:localhost:8888
[ubuntu@10.70.1.104](#)"**

Sequence Data (Only Definitions of this topic is taken from course ppt)

To get the course data, use the command wget, it allows to download the files on the terminal. First step, is to download the files from the links given in the course by using the wget command in front of the link.

**wget https://nextcloud.th-deg.de/s/BZdNP4BQqRLae7L/download/genome.fa wget
https://nextcloud.th-deg.de/s/DFEkkW8aArwFPWN/download/annotation.gtf wget
https://nextcloud.th-deg.de/s/zrjyWijeAjekRr7/download/illumina_adapter.fa wget
https://nextcloud.th-deg.de/s/jsbPzAmNfYRBgpk/download/seqdata.tar** Unpacking sequence
data:

To unpack the files with .tar extension use this command tar -xvf seqdata.tar

```
(base) ubuntu@lsi:~$ tar -xvf seqdata.tar
G1_rep1_read1.fastq.gz
G1_rep1_read2.fastq.gz
G1_rep2_read1.fastq.gz
G1_rep2_read2.fastq.gz
G1_rep3_read1.fastq.gz
G1_rep3_read2.fastq.gz
G2_rep1_read1.fastq.gz
G2_rep1_read2.fastq.gz
G2_rep2_read1.fastq.gz
G2_rep2_read2.fastq.gz
G2_rep3_read1.fastq.gz
G2_rep3_read2.fastq.gz
(base) ubuntu@lsi:~$
```

FastQC (Only Definitions of this topic is taken from internet and chat gpt)

FastQC aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis.

To download FastQc on the virtual machine

“conda install fastqc” (we are installing it in the bio environment)

The version of the FastQc and installation can be confirmed by the following code

“Fastqc --version”

```
*** System restart required ***
Last login: Wed Jan 31 22:37:18 2024 from 172.25.4.107
(base) ubuntu@lsi:~$ fastqc --version
FastQC v0.11.9
(base) ubuntu@lsi:~$
```

To check the quality of any raw sequence the following command can be used

```
(Bio) ubuntu@lsi:~$ fastqc G2_rep3_read1.fastq.gz
application/gzip
Started analysis of G2_rep3_read1.fastq.gz
Approx 5% complete for G2_rep3_read1.fastq.gz
Approx 10% complete for G2_rep3_read1.fastq.gz
Approx 15% complete for G2_rep3_read1.fastq.gz
Approx 20% complete for G2_rep3_read1.fastq.gz
Approx 25% complete for G2_rep3_read1.fastq.gz
Approx 30% complete for G2_rep3_read1.fastq.gz
Approx 35% complete for G2_rep3_read1.fastq.gz
Approx 40% complete for G2_rep3_read1.fastq.gz
Approx 45% complete for G2_rep3_read1.fastq.gz
Approx 50% complete for G2_rep3_read1.fastq.gz
Approx 55% complete for G2_rep3_read1.fastq.gz
Approx 60% complete for G2_rep3_read1.fastq.gz
Approx 65% complete for G2_rep3_read1.fastq.gz
Approx 70% complete for G2_rep3_read1.fastq.gz
Approx 75% complete for G2_rep3_read1.fastq.gz
Approx 80% complete for G2_rep3_read1.fastq.gz
Approx 85% complete for G2_rep3_read1.fastq.gz
Approx 90% complete for G2_rep3_read1.fastq.gz
Approx 95% complete for G2_rep3_read1.fastq.gz
Analysis complete for G2_rep3_read1.fastq.gz
(Bio) ubuntu@lsi:~$
```

“fastqc G2_rep3_read1.fastq.gz” (G2_rep3_read1.fastq.gz is my raw sequence data)

```
(Bio) ubuntu@lsi:~/fastqc_outputs$ ls
G1_rep1_read1_fastqc.html  G1_rep2_read1_fastqc.html  G2_rep2_read1_fastqc.html
G1_rep1_read2_fastqc.html  G1_rep3_read1_fastqc.html
(Bio) ubuntu@lsi:~/fastqc_outputs$
```

To access the FastQc output files we must log in to our jupyterlab.

Cutadapt (Only Definitions of this topic is taken from internet and chat gpt)

Cutadapt is a bioinformatics tool that is commonly used for removing adapter sequences, primers, and other types of unwanted or contaminating sequences from high-throughput sequencing data. It is particularly useful in the preprocessing step of analyzing next-generation sequencing (NGS) data, such as data generated from Illumina platforms.

Install cut adap in the Bio enviornment:

“conda install cutadapt”

Check if it is installed or not

“cutadapt --version”


```
(Bio) ubuntu@lsi:~$ cutadapt --version
3.5
(Bio) ubuntu@lsi:~$
```

Run the cutadapt command to trim the sequence

```
“cutadapt -u 10 -l 25 -q 30 G2_rep3_read1.fastq.gz G2_rep3_read2.fastq.gz -o
trimmed_G2_rep3_read1.fastq.gz -p trimmed_G2_rep3”
```

```
“cutadapt -u 10 -l 25 -q 30 -o trimmed_G1_rep3_read1.fastq.gz -p
trimmed_G1_rep3_read2.fastq.gz G1_rep3_read1.fastq.gz G1_rep3_read2.fastq.gz”
```

```
“cutadapt -u 10 -l 25 -q 30 G1_rep1_read1.fastq.gz G1_rep1_read2.fastq.gz -o
trimmed_G1_rep1_read1.fastq.gz -p trimmed_G1_rep1”
```

-u 10: This trims the first 10 bases from the 5' end of each read.

-l 25: This discards read pairs if, after trimming, either read is shorter than 25 bases.

-q 30: This specifies a quality cutoff of 30 for trimming bases.

The -o option is used to specify the output file for the first read in the pair, and -p is used for the second read in the pair. Adjust the parameters based on your specific data and trimming requirements.

```
(Bio) ubuntu@lsi:~$ cutadapt -u 10 -l 25 -q 30 -o trimmed_G1_rep3_read1.fastq.gz -p trimmed_G1_rep3_read2.fastq.gz G1_rep3_read1.fastq.gz G1_rep3_read2.fastq.gz
This is cutadapt 3.5 with Python 3.10.12
Command line parameters: -u 10 -l 25 -q 30 -o trimmed_G1_rep3_read1.fastq.gz -p trimmed_G1_rep3_read2.fastq.gz G1_rep3_read1.fastq.gz G1_rep3_read2.fastq.gz
Processing reads on 1 core in paired-end mode ...
Done          00:00:00      129,786 reads @   6.6 µs/read;   9.05 M reads/minute
Finished in 0.87 s (7 µs/read; 8.93 M reads/minute).

=== Summary ===

Total read pairs processed:      129,786
Pairs written (passing filters): 129,786 (100.0%)

Total basepairs processed:      25,957,200 bp
  Read 1:    12,978,600 bp
  Read 2:    12,978,600 bp
Quality-trimmed:                863,788 bp (3.3%)
  Read 1:      325,977 bp
  Read 2:      537,811 bp
Total written (filtered):        6,432,842 bp (24.8%)
  Read 1:      3,229,025 bp
  Read 2:      3,203,817 bp
```

Cutadapt Result

Ubuntu.

```
$ sudo apt-get update
```

```
$ sudo apt-get install g++
```

```
$ sudo apt-get install make
```

Basic STAR workflow consists of 2 steps: (Taken from official STAR Manual)

1. Generating genome indexes files.

2. Generating genome indexes. In this step user supplied the reference genome sequences (FASTA files) and annotations (GTF file), from which STAR generate genome indexes that are utilized in the 2nd (mapping) step. The genome indexes are saved to disk and need only be generated once for each genome/annotation combination

```
--runThreadN 1
```

```
--runMode genomeGenerate
```

```
--genomeDir /path/to/genomeDir
```

```
--genomeFastaFiles /path/to/genome/fast1 /path/to/genome/fast2 ...
```

```
--sjdbGTFfile /path/to/annotations.gtf --sjdbOverhang ReadLength-1
```

Here is the Example :

```
STAR --genomeDir /home/ubuntu/Star --genomeFastaFiles /home/ubuntu/genome.fa --  
readFilesIn /home/ubuntu/G2_rep3_read1.fastq.gz /home/ubuntu/G2_rep3_read2.fastq.gz --  
sjdbGTFfile /home/ubuntu/annotations.gtf --sjdbOverhang 99
```

```
Bio) ubuntu@lsi:~$ STAR --genomeDir /home/ubuntu/Star --genomeFastaFiles /home/ubuntu/genome.fa --readFilesIn /home/ubu  
tu/G2_rep3_read1.fastq.gz /home/ubuntu/G2_rep3_read2.fastq.gz --sjdbGTFfile /home/ubuntu/annotations.gtf --sjdbOverhang  
99  
      /home/ubuntu/miniconda3/envs/Bio/bin/STAR-avx2 --genomeDir /home/ubuntu/Star --genomeFastaFiles /home/ubuntu/gen  
me.fa --readFilesIn /home/ubuntu/G2_rep3_read1.fastq.gz /home/ubuntu/G2_rep3_read2.fastq.gz --sjdbGTFfile /home/ubuntu/  
nnotations.gtf --sjdbOverhang 99  
      STAR version: 2.7.11a   compiled: 2023-09-15T02:58:53+0000 :/opt/conda/conda-bld/star_1694746407721/work/source  
eb 01 20:06:27 ..... started STAR run  
eb 01 20:06:27 ..... loading genome
```

2. Running mapping jobs (Taken from official STAR Manual)

The basic options to run a mapping job are as follows:

```
--runThreadN NumberOfThreads
```

```
--genomeDir /path/to/genomeDir
```

```
--readFilesIn /path/to/read1 [/path/to/read2 ]
```

Example:

```
STAR --runThreadN 1 --genomeDir /home/ubuntu/Star --readFilesIn <(zcat
/home/ubuntu/fastqc_outputs/ G2_rep3_read1.fastq.gz)
<(zcat/home/ubuntu/fastqc_outputs/G2_rep3_read2.fastq.gz) --outFileNamePrefix G2_rep3
```

```
(Bio) ubuntu@lsi:~$ STAR --runThreadN 1 --genomeDir /home/ubuntu/Star --readFilesIn <(zcat /home/ubuntu/fastqc_outputs/G2_rep3_read
1.fastq.gz) <(zcat /home/ubuntu/fastqc_outputs/G2_rep3_read2.fastq.gz)
/home/ubuntu/miniconda3/envs/Bio/bin/STAR-avx2 --runThreadN 1 --genomeDir /home/ubuntu/Star --readFilesIn /dev/fd/63 /dev/f
d/62
STAR version: 2.7.11a compiled: 2023-09-15T02:58:53+0000 :/opt/conda/conda-bld/star_1694746407721/work/source
Feb 01 20:29:58 ..... started STAR run
Feb 01 20:29:58 ..... loading genome
Feb 01 20:29:58 ..... started mapping
Feb 01 20:30:05 ..... finished mapping
Feb 01 20:30:05 ..... finished successfully
(Bio) ubuntu@lsi:~$
```

```
(Bio) ubuntu@lsi:~$ ls
_1AAligned.out.sam      G1_rep1_read2_fastqc.zip  G2_rep3Aligned.out.sam
_1ALog.final.out        G1_rep1_read2_fastq.gz    G2_rep3Log.final.out
_1ALog.out              G1_rep1SJ.out.tab         G2_rep3Log.out
_1ALog.progress.out     G1_rep2Aligned.out.sam    G2_rep3Log.progress.out
_1ASJ.out.tab           G1_rep2Log.final.out      G2_rep3_read1_fastqc.html
_2AAligned.out.sam      G1_rep2Log.out            G2_rep3_read1_fastqc.zip
_2ALog.final.out        G1_rep2Log.progress.out   G2_rep3_read1_fastq.gz
_2ALog.out              G1_rep2_read1_fastqc.zip  G2_rep3_read2_fastq.gz
_2ALog.progress.out     G1_rep2_read1_fastq.gz    G2_rep3SJ.out.tab
_2ASJ.out.tab           G1_rep2_read2_fastq.gz    Genomedir
Aligned.out.sam         G1_rep2SJ.out.tab         genome.fa
annotation.gtf          G1_rep3_read1_fastqc.zip  home
fastqc_outputs          G1_rep3_read1_fastq.gz    illumina
G1_rep1Aligned.out.sam  G1_rep3_read2_fastq.gz    Log.final.out
G1_rep1Log.final.out    G2_rep1_read1_fastq.gz    Log.out
G1_rep1Log.out          G2_rep1_read2_fastq.gz    Log.progress.out
G1_rep1Log.progress.out G2_rep2_read1_fastqc.zip  mapped
G1_rep1_read1_fastqc.zip G2_rep2_read1_fastq.gz    miniconda3
G1_rep1_read1_fastq.gz  G2_rep2_read2_fastq.gz    readsAligned.out.sam
```

```

                                UNIQUE READS:
      Uniquely mapped reads number |      181149
        Uniquely mapped reads %   |      97.68%
          Average mapped length   |      199.82
      Number of splices: Total    |      38297
Number of splices: Annotated (sjdb) |      37950
        Number of splices: GT/AG |      38171
        Number of splices: GC/AG |         96
        Number of splices: AT/AC |         10
      Number of splices: Non-canonical |         20
      Mismatch rate per base, %   |         0.14%
        Deletion rate per base    |         0.01%
        Deletion average length    |         1.16
        Insertion rate per base    |         0.01%
        Insertion average length    |         1.09
                                MULTI-MAPPING READS:
      Number of reads mapped to multiple loci |      3661
        % of reads mapped to multiple loci |      1.97%
      Number of reads mapped to too many loci |         6
        % of reads mapped to too many loci |      0.00%
                                UNMAPPED READS:
      Number of reads unmapped: too many mismatches |         0
        % of reads unmapped: too many mismatches |      0.00%
      Number of reads unmapped: too short |        625
        % of reads unmapped: too short |      0.34%
      Number of reads unmapped: other |         1
        % of reads unmapped: other |      0.00%
                                CHIMERIC READS:
      Number of chimeric reads |         0
        % of chimeric reads |      0.00%
[END]

```

Feature Count (Only Definitions of this topic is taken from internet and chat gpt)

FeatureCounts is a program used for counting reads associated with genomic features (such as genes, exons, and other genomic elements) in RNA-seq data. It is part of the Subread package and is commonly used in bioinformatics analysis to quantify gene expression levels from RNA-seq experiments.

FeatureCounts is primarily used for quantifying the number of reads that align to specific genomic features, providing a count matrix that represents the number of reads associated with each feature.

Install Feature count in your Bio Environment.

“conda install -c bioconda subread”

Workflow

Input:

The typical input for FeatureCounts includes aligned reads in BAM format and a genomic annotation file (such as a GTF or GFF file) that defines the genomic features of interest (e.g., genes, exons).

Usage:

It is commonly used as part of the analysis pipeline after aligning RNA-seq reads to a reference genome using aligners like STAR, HISAT2, or others.

Output:

The main output of FeatureCounts is a count matrix, where rows correspond to genomic features (e.g., genes) and columns correspond to samples. Each entry in the matrix represents the number of reads that align to a particular feature in a specific sample.

Features:

FeatureCounts can be used to count reads at various levels of genomic features, such as gene, exon, or transcript. The level of counting is specified by the user based on the research question and the available annotation file.

Example:

```
featureCounts -a annotations.gtf -o counts.txt -p -B -T 8 G2_rep3Aligned.out.sam  
G1_rep1Aligned.out.sam _1AAligned.out.sam _2AAligned.out.sam Aligned.out.sam
```



Input Files (5 SAM Files):

G2 rep3Aligned.out.sam:

SAM file containing aligned reads for the sample G2_rep3.

G1 rep1Aligned.out.sam:

SAM file containing aligned reads for the sample G1_rep1.

1AAligned.out.sam:

SAM file containing aligned reads for an unnamed sample (denoted as 1A).

2AAligned.out.sam:

SAM file containing aligned reads for an unnamed sample (denoted as _2A).

Aligned.out.sam:

Output Files:

counts.txt: The output file where FeatureCounts will store the counts of reads mapped to each feature.

counts.txt.summary: A summary file providing additional information about the counting process.

FeatureCounts Parameters:

Paired-end: Indicates that the input data is from paired-end sequencing.

Count read pairs: Specifies that read pairs should not be counted individually but as a single unit.

Annotation: The genomic annotation file in GTF format (annotations.gtf) defining the features (e.g., genes) used for counting.

Dir for temp files: The directory for storing temporary files generated during the counting process (./ indicates the current directory).

Threads: The number of threads or CPU cores to be used during counting (8 threads in this case).

Level: Specifies the counting level. In this case, it is set to "meta-feature level," which typically means counting at the level of genes.

Multimapping reads: Specifies that multimapping reads should not be counted.

Multi-overlapping reads: Specifies that multi-overlapping reads should not be counted.

Min overlapping bases: The minimum number of overlapping bases required for counting a read

Output

ERCC-00002	ERCC-00002	1	1061	+	1061	17753	21842	19681	18560	11688	15499
ERCC-00003	ERCC-00003	1	1023	+	1023	1386	1500	1454	1659	1030	1267
ERCC-00004	ERCC-00004	1	523	+	523	433	508	468	4363	3112	3438
ERCC-00009	ERCC-00009	1	984	+	984	297	362	327	644	432	507
ERCC-00012	ERCC-00012	1	994	+	994	0	0	0	1	0	0
ERCC-00013	ERCC-00013	1	808	+	808	0	0	0	2	2	0
ERCC-00014	ERCC-00014	1	1957	+	1957	11	9	4	10	2	7
ERCC-00016	ERCC-00016	1	844	+	844	0	0	0	0	0	0
ERCC-00017	ERCC-00017	1	1136	+	1136	0	0	0	0	0	1
ERCC-00019	ERCC-00019	1	644	+	644	3	4	2	12	10	11
ERCC-00022	ERCC-00022	1	751	+	751	140	149	156	192	88	141
ERCC-00024	ERCC-00024	1	536	+	536	0	0	0	0	0	0
ERCC-00025	ERCC-00025	1	1994	+	1994	73	74	56	118	86	93
ERCC-00028	ERCC-00028	1	1130	+	1130	2	0	0	5	7	5
ERCC-00031	ERCC-00031	1	1138	+	1138	0	1	4	4	0	0
ERCC-00033	ERCC-00033	1	2022	+	2022	0	0	0	0	0	3

Workflow manager. (Only Definitions of this topic is taken from internet and chat gpt)

A workflow manager is a software tool or system designed to facilitate, automate, and streamline the execution of complex processes or workflows in various domains such as scientific research, data analysis, bioinformatics, business processes, and more. It helps organize, coordinate, and manage the sequence of tasks involved in a workflow, often involving multiple steps, dependencies, and participants.

Snakemake (Only Definitions of this topic is taken from internet and chat gpt)

Snakemake is a workflow management system that is used to create, manage, and execute data analysis workflows. It provides a flexible and scalable way to define and organize complex workflows, especially in the context of bioinformatics and computational biology. Snakemake allows users to specify the dependencies between different tasks in the workflow and automatically manages the execution of these tasks based on their dependencies.

My code:

```
SAMPLES=["G1_rep1", "G1_rep2", "G1_rep3", "G2_rep1", "G1_rep2", "G1_rep3"]
```

```
rule all:
```

```
    input:
        expand("/home/ubuntu/smake_example/workflows/fastq_raw/{sample}_read1.fastq.gz", sample
= SAMPLES),
        expand("/home/ubuntu/smake_example/workflows/fastq_raw/{sample}_read1.fastq.gz", sample
= SAMPLES),
        expand("/home/ubuntu/smake_example/workflows/fastq_raw/{sample}_read2.fastq.gz", sample
= SAMPLES),
```

```
rule fastqcs:
```

```
    input:
        "../fastq_raw/{sample}_read1.fastq.gz"
    output:
        "../fqc_out/{sample}_read1_fastqc.html"
    shell:
        "fastqc -o {output} {input}"
```

```
rule cutadapt:
```

```
    input:
        fastq1=expand("/home/ubuntu/smake_example/workflows/fastq_raw/{sample}_read1.fastq.gz",
sample = SAMPLES),
        fastq2=expand("/home/ubuntu/smake_example/workflows/fastq_raw/{sample}_read2.fastq.gz",
sample = SAMPLES)
    output:
        o1=expand("/home/ubuntu/smake_example/workflows/trimmed/trimmed_read1.fastq.gz",
sample = SAMPLES),
        o2=expand("/home/ubuntu/smake_example/workflows/trimmed/trimmed_read2.fastq.gz",
sample = SAMPLES)
    shell:
        "cutadapt -u 10 --minimum-length 25 -q 30 -o {output.o1} -p {output.o2} {input.fastq1}
{input.fastq2}"
```

```
rule star_index:
```

```
    input:
        fasta=("/home/ubuntu/genome.fa")
    output:
        index=("/home/ubuntu/smake_example/workflows/star_index")
    shell:
```

