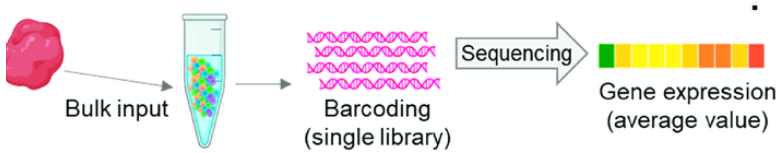
Bulk RNA-seq analysis using R (Rsubread) and python (pyrpipe)



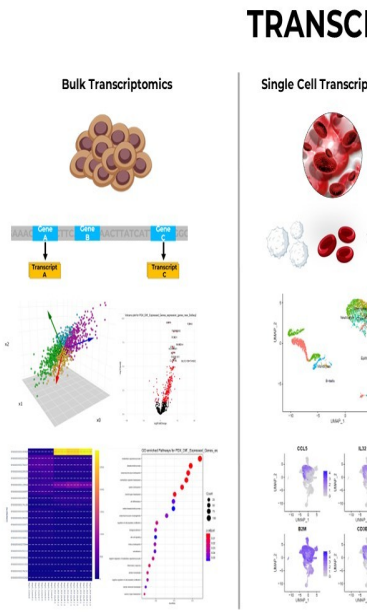
Dr Priyanka Jain
Assistant Professor
Amity University Uttar Pradesh

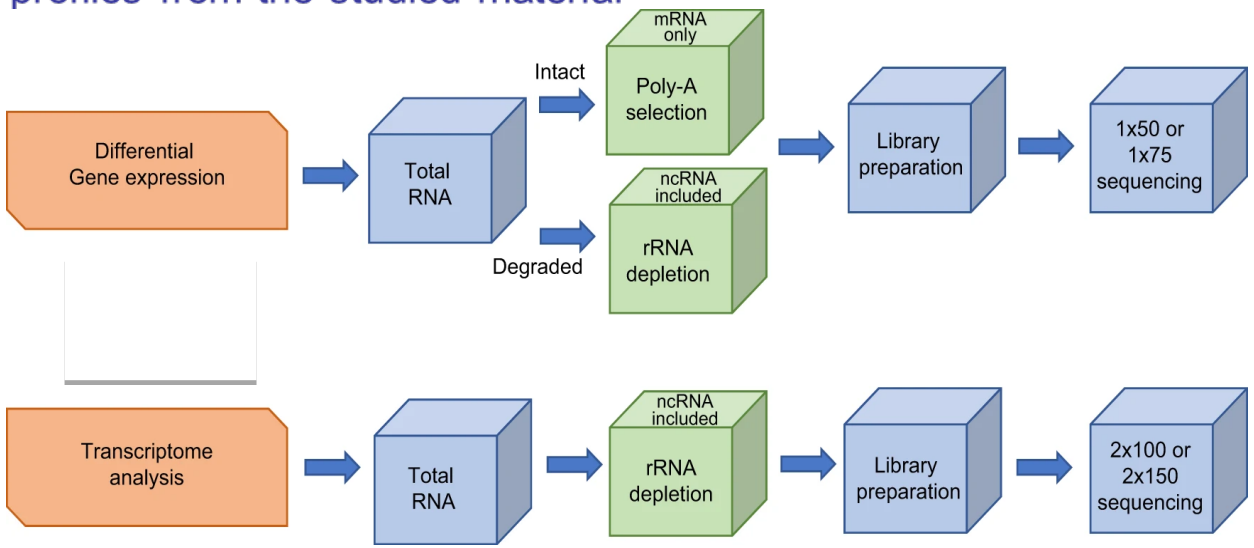
Hands-on session
Priyanka Thareja
Aryaman Sajwan
Amity University Uttar Pradesh

BULK RNA SEQUENCING

RNA sequencing (RNA-Seq) is a high-throughput sequencing methods that provides insight into the transcriptome of a cell. It provides far higher coverage and greater resolution of the dynamic nature of the transcriptome.

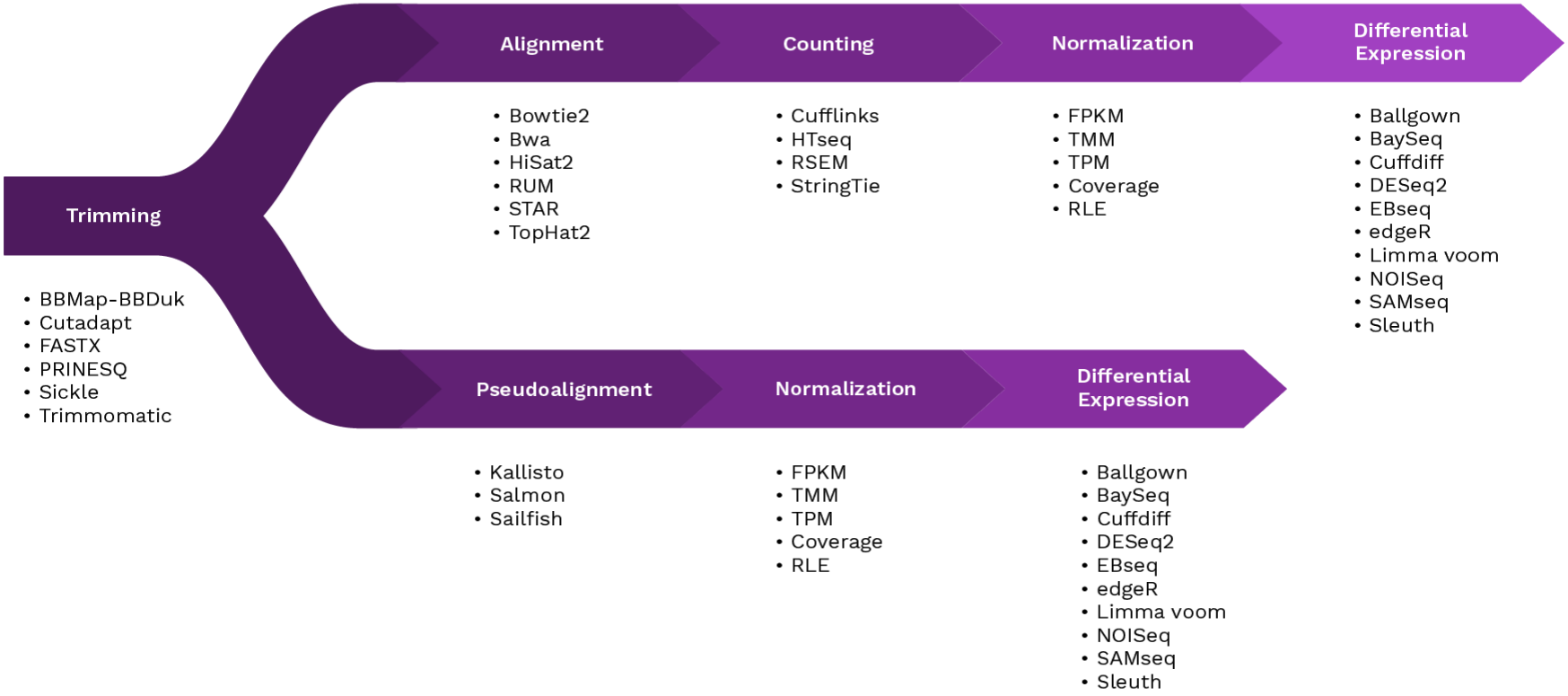
Bulk RNAseq uses a tissue or cell population as a starting material, and results in a mixture of different gene expression profiles from the studied material





From bulk, single-cell to spatial RNA sequencing. *Int J Oral Sci* **13**, 36 (2021)

WORKFLOW OF RNA SEQUENCING DATA ANALYSIS



PREREQUISITES for Rsubread

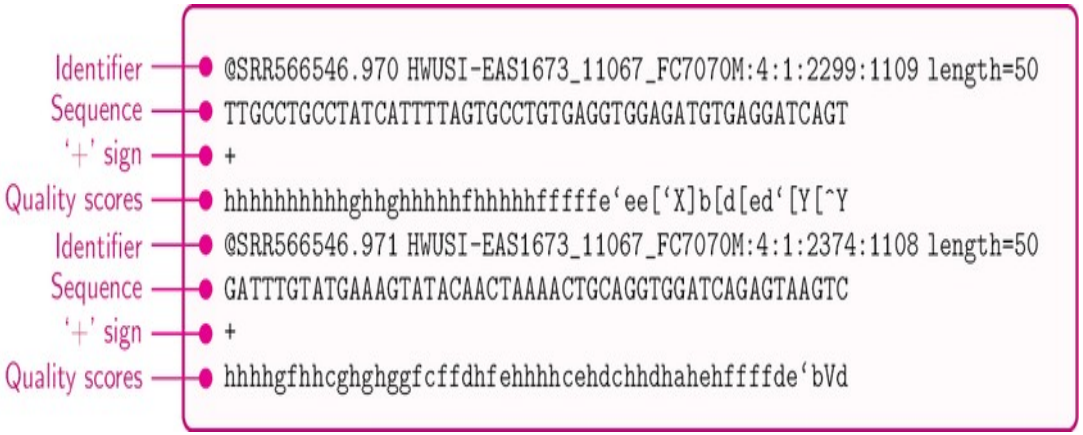
- Windows OS (at least 8GB RAM) with working Power shell
- Install R(version "4.4")
- After installing R run the following command to install Rsubread:
- if (!require("BiocManager", quietly = TRUE))
- install.packages("BiocManager")
- BiocManager::install("Rsubread")
- https://figshare.com/s/f5d63d8c265a05618137 - download the file and unzip it

LOADING R PACKAGES

```
library(Rsubread)
library(limma)
library(edgeR)
```

- **Rsubread** provides functions for read alignment and feature counting. It is particularly useful for handling large RNA-seq datasets efficiently.
- **Limma** stands for Linear Models for Microarray Data. It is widely used for the analysis of gene expression data.
- **edgeR** stands for Empirical Analysis of Digital Gene Expression Data in R. It is designed for differential expression analysis of RNA-seq count data. It uses statistical methods to model the read counts and test for differential expression.

LISTING FASTQ FILES FOR ANALYSIS



FASTQ files are a common format used to store raw sequence data from high-throughput sequencing experiments. Each FASTQ file contains sequences of nucleotides along with their corresponding quality scores.

FastQ files contains raw sequencing reads. Each file represents reads from a specific sample.

This step is important because it allows us to identify all the FASTQ files in the directory, ensuring that we process all available samples.

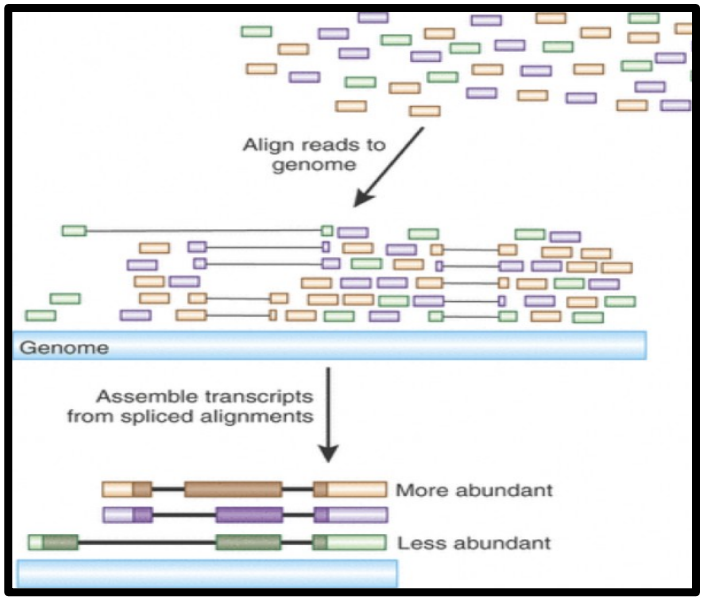
```
reads1 <- list.files(path=".", pattern="*.fastq.gz$")
```

BUILDING INDEX FOR REFERENCE GENOME

- Indexing a reference genome is the process of generating a set of data structures that facilitate rapid access to specific locations within the genomic sequence.
- This process involves mapping the sequence of nucleotides in the genome into a format that can be efficiently searched, enabling bioinformatics tools to quickly locate and align sequencing reads to the reference genome. Indexing is crucial for improving the performance and speed of downstream analyses, such as alignment and variant detection.

```
buildindex(basename="chr1_mm10",reference="chr1.fa")
```

ALIGNMENT



The next step is to align the RNA-seq reads to the reference genome using the “align” function from the Rsubread package.

The input format is indicated as “FASTQ” to show the input files are in FASTQ format.

The output format is specifies as “BAM” to indicate that the output should be in BAM format.

The RNA seq reads are aligned with indexed reference genome.

align(index="chr1_mm10", readfile1=reads1, input_format="FASTQ", output_format="BAM")

BAM FILE - OUTPUT OF ALIGNMENT

A **BAM** file (*.bam) is the compressed binary version of a SAM file that is used to represent aligned sequences up to 128 Mb.

BAM files store aligned sequence data, which includes information on where each read maps to the reference genome.

BAM files contain a header section and an alignment section:

Header—Contains information about the entire file, such as sample name, sample length, and alignment method. Alignments in the alignments section are associated with specific information in the header section.

Alignments—Contains read name, read sequence, read quality, alignment information, and custom tags. The read name includes the chromosome, start coordinate, alignment quality, and the match descriptor string.

The alignments section includes the following information for each or read pair:

RG: Read group, which indicates the number of reads for a specific sample.

BC: Barcode tag, which indicates the demultiplexed sample ID associated with the read.

SM: Single-end alignment quality.

AS: Paired-end alignment quality.

NM: Edit distance tag, which records the Levenshtein distance between the read and the reference.

XN: Amplicon name tag, which records the amplicon tile ID associated with the read.

```
bamfiles <- list.files(path=".", pattern="*.BAM$")
```

FEATURE COUNTS

- **Feature counting quantifies the number of reads that map to each gene, providing the raw data for subsequent analysis of gene expression.**

- **FeatureCounts** is a function under RSubread used in bioinformatics for counting the number of reads (from RNA sequencing) that map to genomic features such as genes, exons, or genomic regions. It is part of the **Subread** package.
- The next step is to count the number of reads that map to each gene using the “featureCounts” function.
- This step is crucial because it provides the raw data for differential expression analysis.

```
fc <- featureCounts(files=bamfiles, annot.inbuilt="mm10")
names(fc)
fc$stat
head(fc$annotation)
```

	1	2	3	4	5	6	7	8	9	10	11	12
497097	438	300	65	237	354	287	0	0	0	0	0	0
100503874	1	0	1	1	0	4	0	0	0	0	0	0
100038431	0	0	0	0	0	0	0	0	0	0	0	0
19888	1	1	0	0	0	0	10	3	10	2	0	0
20671	106	182	82	105	43	82	16	25	18	8	3	10
27395	309	234	337	300	290	270	560	464	489	328	307	342

LOADING SAMPLE INFORMATION FROM CSV FILE

Reading sample information provides the metadata needed for differential expression analysis, linking read counts to experimental conditions and ensuring accurate statistical analysis.

A sample file needs to be created with the information given in the image and save the file with .csv extension.

Sample Info is used to describe the experimental condition associated with each sample. Conditions being control and treatment.

Control : Untreated or baseline state **Treatment** : Manipulated for experiment

```
sample,condition
```

FileName	Sample Name	Cell Type	Status	SRA FileName
MCL1.DG_BC2CTUACXX_ACTTGA_L002_R1	MCL1.DG	basal	virgin	SRR1552450.fastq.gz
MCL1.DH_BC2CTUACXX_CAGATC_L002_R1	MCL1.DH	basal	virgin	SRR1552451.fastq.gz
MCL1.DI_BC2CTUACXX_ACAGTG_L002_R1	MCL1.DI	basal	pregnant	SRR1552452.fastq.gz
MCL1.DJ_BC2CTUACXX_CGATGT_L002_R1	MCL1.DJ	basal	pregnant	SRR1552453.fastq.gz
MCL1.DK_BC2CTUACXX_TTAGGC_L002_R1	MCL1.DK	basal	lactate	SRR1552454.fastq.gz
MCL1.DL_BC2CTUACXX_ATCACG_L002_R1	MCL1.DL	basal	lactate	SRR1552455.fastq.gz
MCL1.LA_BC2CTUACXX_GATCAG_L001_R1	MCL1.LA	luminal	virgin	SRR1552444.fastq.gz
MCL1.LB_BC2CTUACXX_TGACCA_L001_R1	MCL1.LB	luminal	virgin	SRR1552445.fastq.gz

```
sampleInfo <- read.table("sample_info.csv", header=TRUE, sep=",", row.names=1)
```

DIFFERENTIAL GENE EXPRESSION

Differential expression is the process of determining the differences in gene expression levels between different biological conditions. It identifies which set of genes are expressed at different levels under varying experimental conditions, such as treatments, time points, or disease states.

edgeR stores data in a simple list-based data object called a DGEList.
This type of object is easy to use because it can be manipulated like any list in R.

dgeFull is a variable here in which we are saving DGEList.

**dgeFull <-DGEList(counts=fc$counts, gene=fc$annotation[,c("GeneID","Length")]
,group=sampleInfo$condition)**

Filtering out genes that have zero counts across all samples:

**dgeFull <- DGEList(dgeFull$counts[apply(dgeFull$counts, 1, sum) != 0,],
group=dgeFull$samples$group) head(dgeFull$counts)**

NORMALIZATION

- ✓ The normLibSizes function normalizes the library sizes in such a way to minimize the log-fold changes between the samples for most genes. The default method for computing these scale factors uses a trimmed mean of M-values (TMM) between each pair of samples.
- ✓ TMM stands for Trimmed Mean of M-values. It is a normalization method that adjusts for compositional differences between samples. TMM aims to make the majority of genes have similar expression levels across samples by trimming extreme values and calculating a scaling factor for each sample.
- ✓ We call the product of the original library size and the scaling factor the **effective library size**, i.e., the normalized library size. The effective library size replaces the original library size in all downstream analyses.

```
dgeFull <- calcNormFactors(dgeFull, method="TMM")
dgeFull$samples
head(dgeFull$counts)
eff.lib.size <- dgeFull$samples$lib.size*dgeFull$samples$norm.factors
normCounts <- cpm(dgeFull)
```

Estimate Common Negative Binomial Dispersion by Conditional Maximum Likelihood

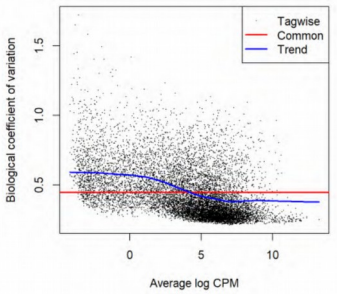
The estimateCommonDisp function in the edgeR package is used to estimate a common dispersion parameter for a set of counts following a negative binomial distribution. This helps in accurately modeling the data and improving the reliability of downstream analyses, such as identifying differentially expressed genes.

dgeFull <- estimateCommonDisp(dgeFull)

EstimateTagwiseDisp Estimate Empirical Bayes Tagwise Dispersion Values

The estimateTagwiseDisp function refines the RNA-seq data analysis by providing gene-specific dispersion estimates using the empirical Bayes method. This process enhances the accuracy of differential expression analysis by accounting for the unique variability of each gene.

dgeFull <- estimateTagwiseDisp(dgeFull)



count-per-million (CPM)

- ✓ A statistical method used will be used identify differentially expressed genes between experimental groups.
- ✓ This test compares the read counts for each gene between groups, taking into account the estimated dispersion.
- ✓ By performing the exact test, one can determine which genes show statistically significant differences in expression between conditions, providing insights into the underlying biological processes and responses.

```
dgeTest <- exactTest(dgeFull)
 dgeTest
 write.csv(dgeTest, file = "path/to/your/directory")
 hist(dgeTest$table[,"PValue"], breaks=50)
 hist(dgeTestFilt$table[,"PValue"], breaks=50)
```

DIFFRENTIALLY EXPRESSED GENE LIST

GeneID	logFC	logCPM	PValue
319263	2.431961	15.3897	0.05
72265	-2.22407	15.38969	0.01
212442	-2.22407	15.38969	0.05
19989	0.6268	16.12666	0.01
98711	-2.22407	15.38969	0.05
66755	-2.22407	15.38969	0.01
68421	-2.22407	15.38969	0.05
19243	-2.22407	15.38969	0.01
13518	4.77364	16.02713	0.03125


THANK YOU