

**“LYNX-RNA: A NEXTFLOW-BASED  
MODULAR RNA-SEQ AND MACHINE  
LEARNING PIPELINE FOR BIOMARKER  
DISCOVERY WITH LLM-SUMMARIZED  
REPORT GENERATION IN IMMUNE  
THROMBOCYTOPENIA”**

**Thesis Submitted  
in Partial Fulfillment of the Requirements for the  
Degree of**

**MASTERS OF TECHNOLOGY  
in  
BIOINFORMATICS**

**by  
DEVANSHI SHARMA  
23/BIO/05**

**Under the Supervision of  
Dr. Asmita Das  
Associate Professor**



**Department of Biotechnology**

**DELHI TECHNOLOGICAL UNIVERSITY  
(Formerly Delhi College of Engineering)  
Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India**

**May.2025**



# DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India

## CANDIDATE'S DECLARATION

I **Devanshi Sharma** hereby certify that the work which is being presented in the thesis entitled "**LYNX-RNA: A Nextflow-Based Modular RNA-Seq and Machine Learning Pipeline for Biomarker Discovery and LLM- summarized Report Generation in Immune Thrombocytopenia**" in partial fulfillment of the requirements for the award of the Degree of **Master of Technology**, submitted in the Department of Biotechnology, Delhi Technological University is an authentic record of my own work carried out during the period from January 2025 to May 2025 under the supervision of **Dr. Amita Das**.

The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

**Candidate's Signature**

This is to certify that the student has incorporated all the corrections suggested by the examiner in the thesis and the statement made by the candidate is correct to the best of our knowledge.

**Signature of Supervisor**



# DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India

## CERTIFICATE BY THE SUPERVISOR

Certified that **Devanshi Sharma (23/BIO/05)** has carried out their search work presented in this thesis entitled **“LYNX-RNA: A Nextflow-Based Modular RNA-Seq and Machine Learning Pipeline for Biomarker Discovery and LLM-Powered summarized Report Generation in Immune Thrombocytopenia”** for the award of **Master of Technology** from Department of Biotechnology, Delhi Technological University, Delhi, under my supervision. The thesis embodies results of original work, and studies are carried out by the student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Dr. Asmita Das**

Supervisor

Department of Biotechnology

Delhi Technological University

**Prof. Yasha Hasija**

Head of Department

Department of Biotechnology

Delhi Technological University

Date: 29.05.2025

# LYNX-RNA: A NEXTFLOW-BASED MODULAR RNA-SEQ AND MACHINE LEARNING PIPELINE FOR BIOMARKER DISCOVERY AND LLM-SUMMARIZED REPORT GENERATION IN IMMUNE THROMBOCYTOPENIC PERPURA

DEVANSHI SHARMA

## ABSTRACT

The increasing complexity of RNA-seq data requires analysis pipelines that are robust, scalable, and interpretable. LYNX-RNA (Language-augmented Yield for Nextflow-based RNA eXpression analysis) is a modular, Nextflow-based workflow that delivers end-to-end RNA-seq analysis—from raw FASTQ files to biological insights—with automation and reproducibility. LYNX-RNA integrates standard tools for quality control, alignment, quantification, and differential gene expression (DGE), along with advanced modules for WGCNA, PPI network modeling, and GO/KEGG enrichment.

A key feature is its built-in machine learning module (Random Forest and XGBoost) for predictive biomarker discovery, and an LLM-powered reporting system that generates natural language summaries of results. To identify DEGs, treated ITP patient data (GSE112278) was compared against external healthy controls (GSE251778) using Welch’s t-test and FDR correction. These DEGs were used to train a classifier that achieved a ROC AUC of 0.937, demonstrating high predictive accuracy. Notably, top predicted DEGs such as *EPB42*, *TNSI*, and *HAGH* overlapped with WGCNA-derived hub genes, reinforcing biological relevance. The pipeline supports deployment in low-resource environments ( $\leq 24$  GB RAM), is compatible with Conda, Docker, and HPC systems, and includes a Python-based CLI for user accessibility. We applied LYNX-RNA to a longitudinal ITP dataset spanning control, pre-treatment, and post-treatment stages, uncovering dynamic gene signatures and potential immune-metabolic biomarkers. LYNX-RNA provides a flexible, automation-ready solution for transcriptome analysis, well-suited for biomarker discovery and translational immunology.

In summary, LYNX-RNA bridges key gaps in usability, scalability, and interpretability in transcriptomic workflows. It serves as a versatile, automation-ready platform for both academic research and translational applications in systems immunology, precision medicine, and biomarker discovery

**Keywords:** RNA-seq pipeline, LYNX-RNA, Nextflow, Biomarker discovery, Immune Thrombocytopenia (ITP), Machine learning, Random Forest, XGBoost, WGCNA, Immune infiltration, Gene expression analysis, Differential expression, Functional enrichment, Large Language Model (LLM), Systems biology, Transcriptomics, Workflow automation, Natural language reporting, Co-expression network analysis

## ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Asmita Das, for her unwavering support, expert guidance, and continuous encouragement throughout this research journey. Her insights were instrumental in shaping the foundation and direction of this thesis.

I am also thankful to my PhD candidates Simran Singh and Yuvraj Sharma for their collaborative spirit, stimulating discussions, and timely help at every step of pipeline development and validation.

I extend my heartfelt thanks to the Department of Biotechnology, Delhi Technological University, for fostering an excellent academic and research environment, and to all faculty and administrative staff whose support made this journey smoother.

Lastly, I express my gratitude to my family, friends and loved ones whose patience, motivation, and emotional support have been my anchor throughout this endeavour.

Above all, I thank God for granting me the perseverance, clarity, and grace to complete this work.

## TABLE OF CONTENTS

Title	Page no.
Candidate's Declaration.	ii
Certificate.	iii
Abstract.	iv
Acknowledgement.	v
Contents.	vi
List of Figures.	x
List of Tables.	xii
List of Abbreviations.	xiii
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
1.1 Background.....	1
1.2 Motivation.....	3
1.3 Objective.....	4
1.4 Scope.....	7
<b>CHAPTER 2: LITERATURE REVIEW.....</b>	<b>8</b>
2.1 RNA-seq Analysis and Challenges.....	8
2.2 Existing Pipelines.....	11
2.3 Nextflow as a Workflow Manager.....	17
2.3.1 Workflow Construction.....	18
2.3.2 Environment Management.....	26
2.3.3 Output Structure.....	27
2.4 Statistical Foundations From RNA-seq.....	28
2.4.1 Quality Assessment and Processing.....	28
2.4.2 Alignment and Mapping Quality.....	28
2.4.3 Transcript Quantification and Normalization.....	28
2.4.4 Statistical Modelling for Differential Expression.....	29
2.4.5 Functional Enrichment and Gene Ontology.....	30
2.4.6 Network and Module based Analysis.....	30

2.4.7 Immune Profiling and Sample level Scoring.....	30
2.4.8 Machine learning.....	31
<b>CHAPTER 3: METHODOLOGY – PIPELINE DESIGN.....</b>	<b>32</b>
3.1 Development Environment.....	32
3.1.1 Workflow orchestration with workflow.....	32
3.1.2 Programming environment and tool chain.....	32
3.2 Architecture of LYNX-RNA.....	34
3.3 Quality Control & Preprocessing.....	36
3.3.1 Check Quality with FastQC.....	36
3.3.2 Trim reads with Trimmomatic.....	37
3.4 Transcript Quantification.....	39
3.4.1 STAR Aligner.....	39
3.4.2 Generating Genome Indexes.....	39
3.4.3 Output of Genome Index Generation.....	41
3.5 Differential Expression Analysis.....	41
3.5.1 Core workflow.....	42
3.5.2 Data Transformation and Visualization.....	43
3.5.3 Additional Features.....	43
3.5.4 Outputs.....	43
3.6 Network Analysis 34.....	45
3.6.1 Conceptual Framework.....	45
3.6.2 Core workflow.....	46
3.7 Functional Enrichment.....	47
3.7.1 Gene ontology and Pathway Databases.....	47
3.7.2 Statistical Methodology.....	48
3.8 Machine Learning Integration.....	48
3.8.1 Goals of machine learning.....	48
3.8.2 Data Flow and processing.....	49

3.8.3 Algorithm Used.....	50
3.8.4 Model Training and validation.....	50
3.8.5 Evaluation Metrics.....	51
3.8.6 Feature Importance and Biomarker Identification.....	51
3.8.7 Interpretability enhancements.....	52
<b>CHAPTER 4: DATASET AND EXPERIMENTAL SETUP.....</b>	<b>53</b>
4.1 Dataset Description.....	53
4.1.1 Eltrombopag-Treated ITP Dataset.....	53
4.1.2 Control Dataset.....	54
4.2 Data Source.....	54
4.3 Preprocessing & TPM Calculation.....	55
4.3.1 TPM vs other normalization metrics.....	56
4.3.2 Output and integration in LYNX-RNA.....	57
4.4 ML Dataset Preparation.....	57
4.4.1 Input Data overview.....	57
4.4.2 Feature and target extaction.....	58
4.4.3 Train and test split with stratification.....	59
4.4.4 Output Processing Step.....	60
4.5 Tools and Platforms Used.....	61
<b>CHAPTER 5: RESULTS AND DISCUSSION.....</b>	<b>63</b>
5.1 Quality Assurance and sample consistency.....	63
5.3 DEG Analysis.....	67
5.2 WGCNA and Hub Gene Identification.....	70
5.4 Functional and Pathway Enrichment.....	73
5.5 Integrated Interpretation with LLM Summarization.....	76
5.6 ML Performance.....	77
5.6.1 Dataset Composition.....	77
5.6.2 Model Performance.....	77
5.6.3 Confusion Matrix.....	78



5.6.4 Bulk DEGs Prediction.....	79
5.6.5 Top DEGs Visualisation.....	79
<b>CHAPTER 6: DISCUSSION.....</b>	<b>80</b>
6.1 Pipeline Performance.....	80
6.2 ML interpretability.....	80
6.3 Benchmarking.....	81
6.4 Predicted DEGs and pipeline validation.....	81
<b>CHAPTER 7: CONCLUSION AND FUTURE WORK.....</b>	<b>82</b>
7.1 Summary of Findings.....	82
7.2 Advantages of LYNX-RNA.....	82
7.3 Limitations.....	83
7.4 Future Enhancement.....	83
<b>REFERENCES.....</b>	<b>89</b>
<b>List of Publications and Proofs.....</b>	<b>90</b>
<b>Plagiarism Verification Certificate.....</b>	<b>91</b>
<b>Plagiarism Report.....</b>	<b>92</b>

## LIST OF FIGURES

<b>FIGURE NO.</b>	<b>TITLE OF FIGURES</b>	<b>PAGE NO.</b>
Fig 3.1.	Workflow of LYNX-RNA rna-seq pipeline	34
Fig 3.2.	Per base sequence quality	37
Fig 3.3.	Adapter Content	37
Fig 3.4.	Per base quality after trimming	38
Fig 3.5	Adapter content after trimming	38
Fig 3.6	DESeq2 result output layout	44
Fig 3.7	MA-plot generated by the plot MA function in DESeq2.	44
Fig 3.5	Volcano plot by DEGs	45
Fig 4.1	layout of trained data	58
Fig 4.2.	Training Snapshot	61
Fig 5.1.	Bar Chart	63
Fig 5.2	PCA plot	64
Fig 5.3	Heat Map	65
Fig 5.4	Box Plot	66
Fig 5.5	Heat Map	67
Fig 5.6	Line Graph	68
Fig 5.7	Box Plot	68
Fig 5.8	Volcano Plot	69
Fig 5.9	MA plot	69
Fig 5.10	Gene Dendrogram Showing Module Assignment by Dynamic Tree Cut.	70
Fig 5.11	Gene Co-expression Network of Hub Genes from Key Module	70
Fig 5.12	Scale Independence Plot for Soft Thresholding Power Selection	71

Fig 5.13	Mean Connectivity Plot Across Soft Thresholding Powers	72
Fig 5.14	Top 25 Enriched Gene Ontology (GO) Biological Processes	73
Fig 5.15	Semantic Similarity Heatmap of Enriched GO Biological Processes	74
Fig 5.16	Comparative Bubble Plot of Enriched GO Biological Processes	75
Fig 5.17	GSEA of genes	76
Fig 5.18	LLM Report	77
Fig 5.19	Model Performance Metrics	78
Fig 5.20	Confusion matrix showing the distribution of actual vs. predicted DEG classes.	80
Fig 5.21	Top DEGs	81

**LIST OF TABLES**

<b>TABLE NO.</b>	<b>TITLE OF TABLES</b>	<b>PAGE NO.</b>
Table 4.1.	Clinical Characteristics of ITP Patients and Healthy Donors (control)	55
Table 4.2.	Comparison of TPM with others	56
Table 4.3.	Pipeline stage and respective tools	62
Table 5.1.	Confusion Matrix	79

## LIST OF ABBREVIATIONS

Abbreviation	Full Form
ANOVA	Analysis of Variance
AUC	Area Under the Curve
BAM	Binary Alignment Map
BiocManager	Bioconductor Package Manager
Bioconductor	Open-source Software Project for Genomic Data
CPM	Counts Per Million
CSV	Comma-Separated Values
Cytoscape	Cytoscape Network Visualization Tool
DEG	Differentially Expressed Gene
DESeq2	Differential Expression Sequencing 2 (R package)
EdgeR	Empirical Analysis of Digital Gene Expression Data in R
ENA	European Nucleotide Archive
Ensembl	European Bioinformatics Institute Genome Database
FASTA	FASTA Sequence Format
FASTQ	FASTQ File Format (raw sequencing reads)
FDR	False Discovery Rate
FPKM	Fragments Per Kilobase of transcript per Million mapped reads
GEO	Gene Expression Omnibus
GO	Gene Ontology
GSEA	Gene Set Enrichment Analysis
gseapy	Gene Set Enrichment Analysis in Python
GTF	Gene Transfer Format
HISAT2	Hierarchical Indexing for Spliced Alignment of Transcripts 2
HPC	High-Performance Computing
HTSeq	High-Throughput Sequencing (counting tool)
KEGG	Kyoto Encyclopedia of Genes and Genomes
kNN	k-Nearest Neighbors
LFC	Log Fold Change
LIME	Local Interpretable Model-Agnostic Explanations

limma	Linear Models for Microarray and RNA-seq Data
LLM	Large Language Model
lncRNA	Long Non-coding RNA
ML	Machine Learning
mRNA	Messenger RNA
ncRNA	Non-coding RNA
Nextflow	Workflow Management System
nf-core	Nextflow Core Community Pipelines
ORF	Open Reading Frame
PCA	Principal Component Analysis
PPI	Protein-Protein Interaction
Python	Python Programming Language
QC	Quality Control
R	R Programming Language
RF	Random Forest
RNA-seq	RNA Sequencing
ROC	Receiver Operating Characteristic
RPKM	Reads Per Kilobase of transcript per Million mapped reads
rRNA	Ribosomal RNA
SAM	Sequence Alignment/Map
Seurat	Single-cell RNA-seq Analysis Toolkit
Shap	SHapley Additive exPlanations
snRNA	Small Nuclear RNA
SRA	Sequence Read Archive
STAR	Spliced Transcripts Alignment to a Reference
STRING	Search Tool for the Retrieval of Interacting Genes/Proteins
SVM	Support Vector Machine
TMM	Trimmed Mean of M-values
TPM	Transcripts Per Million
tRNA	Transfer RNA
UCSC	University of California Santa Cruz
VCF	Variant Call Format
WGCNA	Weighted Gene Co-expression Network Analysis
XGBoost	Extreme Gradient Boosting

## CHAPTER – 1

### INTRODUCTION

#### 1.1 BACKGROUND

The development of high-throughput sequencing technologies has made it possible to uncover many organisms' transcriptional profiles at the whole-genome level. The technology of RNA-seq, or messenger RNA (transcriptome) sequencing, has permitted researchers to explore gene expression patterns with great precision in hundreds of model and non-model organisms. RNA sequencing (RNA-seq) has emerged as a strong tool for assessing genome-wide gene expression, revolutionizing various fields of biology. Even without a reference genome, a wealth of understanding related to processes such as cellular development, gene function, and responses to environmental stimuli, among others, has been uncovered. The RNA-seq methodology has often set the basis for developing molecular genetic analysis in non-model organisms and has become an essential tool. Researchers focused mainly on wet lab and field work sometimes struggle to exploit the data available from “next-generation sequencing” because they lack experience in bioinformatics, which is perceived to require in-depth computational and programming skills[3]. Transcriptome sequencing has become a commonplace technique which is employed in many scientific settings. RNA-seq analysis has various advantages over microarray[16]. Current next-generation sequencing methods yield fastq files that contain the sequencing reads captured from the sample. These reads are typically aligned to a specific reference genome. In RNA-seq, the reads after alignment are quantified on a per gene or per transcript basis to discern information regarding the level of gene expression in a population of cells. Additional analyses may include technical quality control of the sequencing libraries and clustering analysis for experimental quality control. Often, analysis is done to compare samples of two conditions against each other, and determine the statistically significant differences in the level of transcripts per gene.[2]

Further analysis can investigate the pathways associated with these differentially expressed genes, perform various read metrics to assess the variability of the data, and identify single nucleotide changes or deletions that occur throughout the coding regions or the genome. In this contribution we address the problem of creating robust, easily adaptable software for the quality control and analysis of RNA-seq data. This is a difficult problem because the field is moving very rapidly with new and improved algorithms for key tasks being published frequently. Also, novel applications of RNA-seq are constantly being enabled by new analytic approaches. For example, innovations in analysis now permit tools to be developed that aid in the discovery of

fusion genes, the identification of viral transcripts and the analysis of immunological infiltrate in samples, which enable a deeper understanding of the biological system being studied. [2]

This study is intended to make the approach easier and more understandable. The system presented here, LYNX-RNA (*Language-augmented Yield for Nextflow-based RNA eXpression analysis*) , uses a modern computational workflow management system, Nextflow , to combine many of the most useful tools currently employed in RNA-seq analysis into a single, fast, easy to use pipeline, that includes alignment steps, quality control, differential gene expression and pathway analyses. In addition, LYNX-RNA includes a variety of optional modules for advanced downstream analyses, such as co-expression network construction, immune cell infiltration profiling, metabolic and immune pathway enrichment, and machine learning-based biomarker prioritization. LYNX-RNA was built with following guiding principles.[2]

(1) LYNX-RNA makes use of the Nextflow framework for a flexible and modular architecture, thereby allowing users to easily include alternative current methods or new tools. Often absent in outdated pipelines requiring manual updates or changes, this flexibility combines the most recent advancements in RNA-seq analysis, including immune cell infiltration assessment and viral transcript detection. (2) Automated LLM-Driven Summarization: Unlike traditional pipelines that generate complex output requiring expert interpretation, LYNX-RNA's integrated LLM produces clear, human-readable summaries of the analysis. This feature allows researchers, clinicians, and non-experts to quickly grasp insights without extensive computational expertise, bridging the gap. (3) Enhanced Use and Installation: Many RNA-seq analytical pipelines require intricate setups and dependency management, which challenges consumers with less bioinformatics knowledge. (4) LYNX-RNA resolves this issue by consolidating critical utilities into a single efficient package that can be executed with straightforward command-line input. Novices can navigate the pipeline effortlessly, but experienced users maintain comprehensive customization options for intricate studies. (5) Visual and Organizational Efficiency: LYNX-RNA delivers organized visual representations, allowing users to swiftly comprehend and analyze RNA-seq data through a "glance and drill-down" methodology. (6) LYNX-RNA supports execution in local, HPC, and cloud environments using Conda or Docker/Singularity containers. Every analysis step is version-controlled and reproducible, meeting the standards required for robust and transparent scientific research. (7) To enable predictive modeling, LYNX-RNA includes a dedicated machine learning module that supports algorithms such as Random Forest and XGBoost. Users can train classifiers on normalized expression data (e.g., TPM or vst-transformed counts) and evaluate performance using accuracy, F1-score, ROC-AUC, and cross-validation. The pipeline generates ranked feature importance lists and visualizations disease-stage-specific biomarkers. This ML component enhances the biological interpretability and translational value of the analysis. (8) Extensive Field Compatibility: LYNX-RNA has a variety of applications in cancer, virology, immunology, and biomarker identification, making it extremely versatile for emerging research avenues. LYNX-



RNA provides comprehensive tools for variant calling, differential expression analysis, pathway enrichment, and additional functions, making it a singular solution for many scientific investigations.

## 1.2 MOTIVATION

RNA sequencing (RNA-seq) has become a central tool in modern molecular biology, enabling high-resolution characterization of transcriptomes across diverse biological systems and conditions. It is widely used in functional genomics, developmental biology, immunology, and disease profiling, including complex conditions such as cancer and autoimmune disorders. RNA-seq generates read-level data that is aggregated into count matrices representing gene or transcript abundance. A primary objective in most studies is to identify genes that exhibit differential expression between biological groups or timepoints.[14]

Despite its widespread adoption, RNA-seq analysis remains computationally demanding due to the discrete, over dispersed nature of count data and the influence of factors such as sequencing depth, batch effects, and biological heterogeneity. Traditional statistical approaches used for microarrays are not directly applicable to RNA-seq, leading to the development of dedicated tools for normalization, statistical testing, and downstream interpretation. However, many of these tools are fragmented across different platforms, require manual integration, or lack downstream support for biological interpretation, immune profiling, and biomarker discovery. Furthermore, existing pipelines often generate complex outputs that require domain expertise to interpret, posing a barrier for clinicians and experimental biologists. There is a growing need for a comprehensive solution that not only performs robust RNA-seq processing but also integrates machine learning for biomarker prioritization, network-based analysis, and automated interpretation.[11]

To address these challenges, we developed LYNX-RNA—a modular, Nextflow-based pipeline designed to streamline RNA-seq analysis from raw reads to interpretable insights. It integrates state-of-the-art tools for differential expression, functional enrichment, co-expression networks, immune deconvolution, and LLM-generated summaries, providing a powerful yet accessible framework for transcriptomic analysis across biological and clinical domains.

To illustrate the utility of LYNX-RNA we applied it to a set of ITP patients.

Early in the twentieth century, Paul Ehrlich realized that the immune system could go awry. Instead of reacting only against foreign antigens, it could focus its attack on the host. This condition, which he termed autotoxic, can result in a clinical syndrome generically referred to as autoimmunity. This inappropriate response of the immune

system, directing humoral and or T - lymphocytes mediated immune activity against self-components, is the cause of auto immune disease such as Idiopathic Thrombocytopenic Purpura (ITP) or Immune Immune thrombocytopenia. ITP is an acquired immune-mediated autoimmune disease characterized by low peripheral blood platelet counts ( $<100 \times 10^9/L$ ), which are considered thrombocytopenia, and increased risk of bleeding due to peripheral platelet destruction through antibody-dependent cellular phagocytosis, complement-dependent cytotoxicity, cytotoxic T lymphocyte-mediated cytotoxicity, and megakaryopoiesis alteration. This condition may be idiopathic or triggered by drugs, vaccines, infections, cancers, autoimmune disorders, and systemic diseases. Despite advances in clinical guidelines, the absence of definitive diagnostic biomarkers often leads to misdiagnosis and delayed or ineffective treatment, with a substantial proportion of patients progressing to chronic or refractory ITP. Emerging studies suggest that metabolic reprogramming may play a role in autoimmune pathogenesis, including ITP. Metabolomic profiling has revealed alterations in phenylalanine, tyrosine, and glyoxylate metabolism in ITP patients, indicating potential metabolic biomarkers. However, the link between these metabolic changes and underlying gene expression patterns remains poorly understood. To address this gap, transcriptomic profiling—particularly RNA sequencing (RNA-seq)—offers a powerful approach to unravel disease mechanisms and identify metabolism-related genes with diagnostic and therapeutic potential. RNA-seq has transformed transcriptome analysis by enabling high-resolution, strand-specific quantification of gene expression, isoform variation, and novel transcript discovery. However, the computational complexity of RNA-seq—ranging from alignment of exon-spanning reads to quantification, outlier detection, and result interpretation—poses significant hurdles for widespread adoption, especially in clinical research settings.

### 1.3 OBJECTIVE

The primary objective of this study is to develop and apply LYNX-RNA (*Language-augmented Yield for Nextflow-based RNA eXpression analysis*), a fully modular, reproducible, and interpretable RNA-seq analysis pipeline that addresses current limitations in transcriptomic workflows. Specifically, the goals of LYNX-RNA are as follows:

1. To build a modular and scalable RNA-seq pipeline using the Nextflow framework
  - 1) Enable flexible integration of widely used tools for preprocessing, alignment, quantification, and statistical modeling.
  - 2) Support local, HPC, and cloud-based execution environments with containerized deployment using Conda, Docker, and Singularity.

- 3) Provide a parameter-driven configuration system that balances ease of use for novices with full customization for advanced users.
2. To streamline the complete RNA-seq workflow from raw reads to biological insight
    - 1) Automate quality control, read trimming, adapter removal, and contaminant filtering using tools such as FastQC, fastp, and BBSplit.
    - 2) Perform alignment with STAR or HISAT2, followed by transcript/gene quantification using Salmon or featureCounts.
    - 3) Implement robust differential gene expression analysis using DESeq2 and downstream visualization (PCA, heatmaps, volcano plots).
  3. To integrate machine learning models for biomarker discovery and classification
    - 1) Include supervised learning approaches such as Random Forest and XGBoost to classify samples across experimental groups (e.g., control, pre-treatment, post-treatment).
    - 2) Rank genes by predictive importance and visualize classifier performance using metrics like ROC-AUC, F1-score, and confusion matrices.
    - 3) Support stratified cross-validation and test/train splitting for reproducible ML-based analyses.
  4. To perform systems-level analysis through network and module-based exploration
    - 1) Construct co-expression networks using WGCNA to identify gene modules correlated with clinical traits or timepoints.
    - 2) Build PPI networks using STRING data and identify hub genes using centrality measures in Cytoscape.
    - 3) Integrate gene-level and network-level results to enhance the biological relevance of candidate biomarkers.
  5. To incorporate immune and pathway activity profiling
    - 1) Apply GSVA and ssGSEA for single-sample pathway scoring using curated gene sets from MSigDB, with a focus on immune and metabolic signatures.
    - 2) Quantify immune cell infiltration using immune marker gene sets and assess correlations with hub gene expression and clinical conditions.

- 3) Visualize enrichment and immune profiles using violin plots, heatmaps, and correlation matrices.
6. To enhance accessibility and interpretability using LLM-generated summaries
- 1) Integrate a Large Language Model (LLM) interface (e.g., OpenAI GPT) to automatically generate natural-language summaries of analysis results.
  - 2) Provide narrative outputs covering DEGs, functional enrichment, ML classifiers, and immune infiltration profiles.
  - 3) Lower the interpretive barrier for clinicians, experimental biologists, and non-specialist users.
7. To validate the pipeline using a real-world clinical dataset
- 1) Demonstrate the application of LYNX-RNA on a longitudinal RNA-seq dataset of Immune Thrombocytopenia (ITP) patients, sampled at control, pre-treatment, 1 week, and 1 month post-treatment.
  - 2) Identify timepoint-specific gene expression patterns, hub genes, and immune-metabolic pathways involved in ITP pathogenesis.
  - 3) Evaluate pipeline outputs for biological plausibility, classifier performance, and network-level robustness.
8. To offer a reusable, open-source, and community-friendly solution
- 1) Publish the pipeline as an open-source tool, complete with documentation, example data, and installation instructions.
  - 2) Provide support for integration with standard file formats and compliance with FAIR and nf-core-inspired best practices.
  - 3) Enable long-term extensibility for new use cases such as viral transcript detection, variant calling, and multi-omics integration.

## 1.4 SCOPE

This thesis encompasses both the development and application of LYNX-RNA, a modular, scalable, and interpretable RNA-seq analysis pipeline built using the Nextflow framework. The scope includes the technical architecture, tool integration, and workflow automation that enable the complete RNA-seq processing pipeline—from quality control and alignment to differential gene expression analysis, pathway enrichment, machine learning-based biomarker discovery, co-expression network modeling, immune infiltration profiling, and natural language report generation using large language models (LLMs).

The pipeline is demonstrated on a real-world longitudinal RNA-seq dataset of Immune Thrombocytopenia (ITP), comprising multiple clinical stages (control, pre-treatment, week 1, and month 1). The objective is to showcase how LYNX-RNA identifies differentially expressed genes, extracts timepoint-specific biological signatures, uncovers immune and metabolic pathway activity, and highlights predictive biomarkers using machine learning classifiers.

## CHAPTER - 2

### LITERATURE REVIEW

#### 2.1 RNA-seq Analysis and Challenges

RNA sequencing (RNA-seq) has revolutionized the way scientists study transcriptomes, enabling detailed insights into gene expression, regulation, and function across different biological conditions. Over the years, various methodologies have emerged for RNA-seq analysis, ranging from hybridization-based approaches to high-throughput sequencing platforms. For the past decade, microarrays have grown in popularity as the primary tool for gene expression analysis. Recently, however, “digital gene expression” by next-generation sequencing has been introduced as a promising, new platform for assessing the copy number of transcripts, thereby providing a digital record of the numerical frequency of a sequence in a sample.[10]

There are various Types of RNA-seq analysis based on technological or methodological platforms, such as how microarray and NGS (Next-Generation Sequencing) are distinct methods. Over the years, several methods have emerged for RNA analysis, including microarray-based profiling, digital gene expression tag profiling, and Next-Generation Sequencing (NGS)-based RNA-seq. Microarrays, once widely used, depend on hybridization of RNA to pre-designed probes and are limited to detecting known transcripts, offering only relative quantification with a constrained dynamic range. Digital gene expression (DGE) profiling provides a more cost-effective alternative but lacks the depth and breadth of full transcriptome coverage. In contrast, NGS-based RNA-seq has become the preferred method due to its ability to detect both known and novel transcripts, offer absolute quantification, and provide a much wider dynamic range. NGS technologies support various formats including bulk RNA-seq, single-cell RNA-seq, total RNA-seq, and long-read sequencing, each tailored for specific applications like differential gene expression, isoform discovery, splicing analysis, and cellular heterogeneity studies. Among these, single-cell RNA-seq and spatial transcriptomics represent cutting-edge advancements that allow transcriptomic analysis at unprecedented resolution.[7]

NGS is preferred over microarrays and DGE methods because it is not limited by prior knowledge of gene sequences, it offers higher sensitivity and accuracy, and it supports a wide array of analytical applications—from basic gene expression profiling to complex studies of gene regulation, alternative splicing, and translational dynamics. As such, NGS has become the cornerstone of modern transcriptomic research due to its scalability, comprehensiveness, and versatility across diverse biological systems.

RNA-seq, while powerful and widely adopted, presents several challenges at various stages of the analysis pipeline. These challenges arise due to the complexity of biological systems, limitations in technology, and computational demands. Below is a comprehensive overview of the key challenges in RNA-seq analysis, grouped by category:

### 1. Experimental and Technical Challenges

One of the first and most critical challenges in RNA-seq analysis arises at the experimental stage. The quality of RNA extracted from biological samples plays a crucial role in determining the success of the entire pipeline. Degraded RNA can lead to biased or incomplete coverage of transcripts, especially at the 5' end, thereby compromising quantification accuracy. Contaminants such as genomic DNA or phenol from the extraction process can further interfere with downstream applications like library preparation or sequencing.

Another major issue is the presence of batch effects—systematic differences between groups of samples processed at different times or under slightly different laboratory conditions. These can introduce false biological signals or obscure real ones. Moreover, variations in library preparation methods (such as poly(A) selection versus ribosomal RNA depletion) can influence the types and quantities of RNA captured, introducing bias that complicates downstream comparative analyses.

### 2. Computational Challenges

RNA-seq datasets are inherently large, often consisting of millions to billions of sequencing reads per experiment. This scale imposes significant demands on computational infrastructure in terms of storage, memory, and processing power. Even before analysis begins, researchers need to address issues like raw data storage, file conversion, and indexing.

Aligning sequencing reads to a reference genome or transcriptome poses another major computational hurdle. Because eukaryotic genes contain introns that are spliced out in the mature transcript, RNA-seq reads often span exon-exon junctions. Splice-aware aligners such as STAR and HISAT2 are necessary, but configuring them to balance speed, accuracy, and memory use can be difficult. Further complications arise in repetitive genomic regions or for genes with high sequence similarity (e.g., paralogs), where misalignment may occur.

In transcriptome assembly, reconstructing full-length transcripts from short reads is computationally intensive and error-prone. This is especially true for lowly expressed transcripts or genes with multiple isoforms, where read coverage may be sparse or ambiguous.

### 3. Statistical and Analytical Challenges

A central aim of RNA-seq analysis is to identify differentially expressed genes between conditions. However, the raw counts generated from read mapping are not immediately suitable for comparison. They must be normalized to account for differences in sequencing depth and RNA composition. Choosing the appropriate normalization method is non-trivial. Common methods include Transcripts Per Million (TPM), Trimmed Mean of M-values (TMM), and DESeq2's size factor normalization. Each method has strengths and assumptions that may or may not hold in a given dataset.

Low-abundance genes present another challenge. These genes often exhibit high variability and low signal-to-noise ratios, making it difficult to determine whether observed differences are biologically meaningful or due to sampling noise.

Moreover, differential expression testing involves thousands of statistical tests, one for each gene. This creates a multiple testing problem that must be addressed using False Discovery Rate (FDR) correction. While this reduces false positives, it can also reduce statistical power, particularly for datasets with small sample sizes.

### 4. Interpretation Challenges

Once differentially expressed genes are identified, interpreting their biological significance can be difficult. Functional annotation databases like Gene Ontology (GO) and KEGG are commonly used for enrichment analysis, but these databases may be incomplete or outdated, especially for non-model organisms. Furthermore, gene function is often context-dependent and may not be well captured by generic annotations. [8]

Another layer of complexity arises at the transcript or isoform level. Most RNA-seq pipelines aggregate read counts at the gene level, potentially masking significant changes in isoform usage or alternative splicing. Tools exist for isoform-level analysis, but they are more computationally intensive and require deeper sequencing to ensure sufficient coverage.

In single-cell RNA-seq, additional interpretation challenges emerge. Technical noise, dropouts (false zero counts), and cell-to-cell variability make it difficult to distinguish technical artifacts from true biological differences.[9]

### 5. Reproducibility and Standardization

Despite the maturity of RNA-seq as a technology, reproducibility remains a major concern. Different research groups often use different software tools, parameter settings, and reference annotations. These discrepancies can lead to substantial variation in results. There is also a lack of universally accepted standard pipelines,



although initiatives like nf-core and guidelines from the ENCODE consortium aim to address this.

Proper documentation of methods, software versions, and metadata is essential for reproducibility, but is frequently overlooked. Inconsistent or incomplete metadata can make it impossible to replicate analyses or compare results across studies.

## 6. Integration and Scalability

As biological research becomes increasingly multi-omics in nature, RNA-seq data is often integrated with other data types, such as proteomics, epigenomics, or metabolomics. This integration requires compatible data formats, normalization strategies, and statistical models, which can be difficult to harmonize.

Furthermore, the volume of RNA-seq data continues to grow, especially in population-scale studies or large single-cell projects. Traditional analysis pipelines may not scale well under such loads. Workflow managers like Nextflow and Snakemake enable automation and parallelization, but they require advanced computational skills to implement and maintain.

## 7. Validation and Biological Relevance

A final and often overlooked challenge is experimental validation. While RNA-seq can generate hypotheses about gene expression and regulatory mechanisms, these must be validated through laboratory experiments such as qRT-PCR, western blotting, or functional assays. Without validation, RNA-seq results risk being viewed as exploratory rather than definitive.

Moreover, the translation of RNA-seq findings into clinically actionable insights remains a complex process. It involves not only technical validation, but also regulatory approvals, standardization across platforms, and demonstration of reproducibility and predictive power in independent cohorts.

## **2.2 Existing Pipelines**

Over the past decade, RNA sequencing (RNA-seq) has become central to transcriptomic research, powering investigations in disease biology, developmental genomics, and biomarker discovery. This widespread use has catalyzed the development of several RNA-seq pipelines, each aiming to simplify and standardize the complex analytical steps required—from read processing and alignment to statistical modeling and biological interpretation. However, existing tools differ

significantly in terms of flexibility, downstream depth, user accessibility, and integration with emerging technologies such as machine learning and AI-driven interpretability. In this landscape, LYNX-RNA offers a comprehensive and modular solution that addresses many of the limitations inherent in prior workflows.

For instance, VIPER[27] is a Snakemake-based pipeline that supports differential expression and immune infiltration analysis, and is compliant with nf-core standards. However, it lacks support for machine learning-driven biomarker prioritization or natural language report generation. Similarly, miARma-Seq [43] focuses on multi-species compatibility and supports miRNA and circRNA studies, yet it is not designed for full-scale transcriptome interpretation or integration with predictive models. The TRAPLINE[44] pipeline, while embedded in the Galaxy platform for GUI accessibility, suffers from reduced adaptability due to Galaxy’s framework constraints and lacks co-expression network support. RNASeqR is a six-step R-based workflow that simplifies command-line integration for biologists, but it is limited to basic DEG and GO/KEGG analyses and is not modular enough for advanced custom workflows. RNAdetector, a GUI-based platform offering cloud and Docker support, excels in accessibility but falls short on flexibility, lacking support for co-expression, immune profiling, or classification tasks. The nf-core initiative defines a community-curated standard for bioinformatics pipelines using Nextflow. Pipelines like nf-core/rnaseq support STAR, Salmon, featureCounts, and MultiQC, offering broad compatibility and continuous integration for reproducibility. While nf-core/rnaseq excels in standardization, its functionality largely ends at quantification and QC. It does not integrate ML, network analysis, immune profiling, or interpretive summarization, requiring manual downstream scripting for biological insight. LYNX-RNA follows nf-core best practices for reproducibility, containerization, and version control, but extends the framework with systems biology modules, ML-based biomarker analysis, and natural language output, making it more holistic and translationally oriented. In contrast, LYNX-RNA is designed to address these shortcomings through a fully containerized, Nextflow-based modular pipeline that integrates all core RNA-seq steps with extensive downstream analytics. It supports quality control (FastQC, MultiQC), alignment (STAR, HISAT2), quantification (Salmon, featureCounts), and differential expression (DESeq2), alongside advanced tools for WGCNA-based co-expression network construction, PPI-based hub gene analysis via STRING and Cytoscape, and immune cell infiltration analysis using ssGSEA and GSVA. Notably, LYNX-RNA also incorporates a dedicated machine learning module, offering Random Forest and XGBoost classifiers with ROC-AUC, feature importance, and cross-validation capabilities, enabling robust biomarker discovery that extends beyond DEG thresholds alone. [45], [17]

What sets LYNX-RNA apart from all previously reported pipelines is its integration with a Large Language Model (LLM), which automates the generation of natural-language summaries of the analytical results. This makes outputs interpretable by non-bioinformaticians—particularly clinicians or experimentalists—facilitating direct translation of data into insight. Moreover, LYNX-RNA is tested on longitudinal RNA-

seq data from immune thrombocytopenia (ITP) patients across four timepoints (control, pre-treatment, 1 week, 1 month post-treatment), demonstrating its strength in multi-condition, time-sensitive transcriptomic profiling—an application rarely supported by other pipelines, which typically assume simple binary designs.

Finally, the visual output of LYNX-RNA—ranging from PCA plots and volcano plots to heatmaps and ML-based feature rankings—follows a “glance-and-drill-down” design philosophy, enabling users to rapidly understand global patterns while retaining access to detailed statistical data. The pipeline is deployable on local, HPC, and cloud platforms using Conda or Docker, and its modular design ensures full reproducibility and extensibility. In sum, LYNX-RNA delivers a next-generation RNA-seq analysis platform that not only streamlines the analytical process but also elevates interpretation, scalability, and clinical relevance—surpassing many of the design and usability barriers that persist in current RNA-seq pipelines.

RNA sequencing (RNA-seq) data analysis pipelines typically consist of modular stages: quality control, alignment, quantification, differential expression, functional enrichment, and optionally, network analysis, immune profiling, or machine learning-based prioritization. This section compares LYNX-RNA’s toolset to other widely used alternatives in the field.

## **1. Quality Control and Preprocessing**

### 1) FastQC

One of the most widely used tools for initial read quality assessment. Reports per-base quality, adapter content, GC distribution, and overrepresented sequences.[17]

Limitations: Output is static (HTML-based), and it does not modify reads.

### 2) Fastp

Performs both quality trimming and filtering, as well as read correction, adapter removal, and UMI handling.

Offers detailed JSON/HTML summaries and supports both single- and paired-end reads.[17]

Advantages: Faster and more feature-rich than older tools like Trimmomatic.

### 3) Trimmomatic / Cutadapt

Trimmomatic is highly customizable but lacks modern reporting features.

Cutadapt is popular for adapter trimming but does not perform full QC.

Comparison: fastp (used in LYNX-RNA) is faster than Trimmomatic and more comprehensive than Cutadapt, making it suitable for scalable, automated workflows.

#### 4) MultiQC

Aggregates reports from tools like FastQC, STAR, and Salmon into a single HTML dashboard.[17]

Used in LYNX-RNA to provide unified QC visibility across all samples.

## 2. Read Alignment Tools

#### 1) STAR

Ultrafast aligner with high sensitivity for splice junctions; widely used in genome-wide transcriptomics.

Limitation: High memory usage (~30–40 GB).

#### 2) HISAT2

Memory-efficient and graph-aware aligner. Performs well on large and complex genomes.

Comparison: LYNX-RNA supports both STAR and HISAT2, giving users flexibility based on system resources and study type.

#### 3) TopHat2

Formerly a standard RNA-seq aligner, now deprecated and replaced by HISAT2.

Obsolete in modern pipelines due to poor performance and lack of updates.[17]

## 3. Quantification Tools

#### 1) Salmon

Provides fast and accurate transcript-level quantification without full alignment. Supports GC-bias correction and bootstrapping.

Used in LYNX-RNA for fast, bias-aware transcript abundance estimation.

#### 2) Kallisto

Similar to Salmon, also uses pseudoalignment. Slightly faster but lacks as many built-in bias corrections.

#### 3) featureCounts

Gene-level read assignment tool from the Subread package. Robust and widely used for input into DESeq2 or edgeR.

#### 4) HTSeq-count

Python-based read-counting tool. Slower than featureCounts and less scalable for large datasets.

#### 5) RSEM

Alignment-based quantifier with support for isoform-level estimation. Used by VIPER and nf-core but slower than Salmon/Kallisto.[17]

### **4. Differential Expression Tools**

#### 1) DESeq2

Negative binomial model-based method. Performs well in datasets with biological replicates.

Integrates shrinkage estimation for fold changes and dispersion.

#### 2) edgeR

Particularly effective with small sample sizes. Supports generalized linear models and complex designs.

#### 3) limma-voom

Transforms count data to log-counts per million and models them using linear models. Less suited for low-count genes but effective for microarray-like designs.

Comparison: LYNX-RNA uses DESeq2 as default due to its balance between stability, scalability, and community adoption, with optional support for edgeR.

### **5. Co-expression and Network Analysis**

#### 1) WGCNA

The gold standard for co-expression module detection. Used to identify biologically relevant gene modules correlated with traits or timepoints.

Integrated in LYNX-RNA to support systems-level discovery.

#### 2) CEMiTool

An alternative to WGCNA with automatic parameter selection and built-in enrichment analysis.

#### 3) STRING + Cytoscape

For visualizing protein–protein interaction networks and identifying hub genes based on node centrality.

Comparison: While most pipelines stop at DEGs, LYNX-RNA incorporates network biology modules for hub gene prioritization, enabling deeper biological interpretation.[17]

## **6. Functional Enrichment Tools**

### 1) clusterProfiler

Widely used R package for GO and KEGG enrichment. Supports visualizations like dotplots, cnetplots, and ridge plots.

### 2) g:Profiler

Web and API-accessible tool for enrichment with excellent multi-organism support and compatibility with Ensembl IDs.

### 3) DAVID / Enrichr

Popular web-based platforms. Easy to use but limited in batch automation and reproducibility.

Comparison: LYNX-RNA automates enrichment via clusterProfiler and g:Profiler, offering scalable and reproducible outputs with integration into downstream plots.

## **7. Immune Profiling and Pathway Analysis**

### 1) ssGSEA / GSVA

Tools to compute per-sample pathway activity scores from gene sets (e.g., immune cell signatures from MSigDB).

Integrated into LYNX-RNA for immune cell estimation and correlation with hub gene expression.

### 2) CIBERSORT / xCell

Web-based or R-based tools to infer immune cell composition. CIBERSORT uses support vector regression; xCell applies a spillover correction model.

Comparison: LYNX-RNA uses ssGSEA + GSVA for flexible, sample-wise scoring and plans to integrate xCell/CIBERSORT in future versions.

## 8. Machine Learning and Automated Interpretation

### 1) Random Forest / XGBoost

Integrated into LYNX-RNA for supervised learning and gene ranking. Users can view confusion matrices, ROC-AUC, and feature importance scores.

### 2) LLM Integration

LYNX-RNA is the first RNA-seq pipeline to incorporate GPT-based summarization, producing natural-language outputs that describe DEGs, pathways, classifier results, and immune profiles.

Comparison: No existing pipeline (VIPER, nf-core, DEWE, TRAPLINE, RNASeqR) offers this level of automated, clinician-friendly interpretation.

## 2.3 Nextflow as a Workflow Manager

Nextflow is a domain-specific language (DSL) specifically designed for building scalable and reproducible data analysis workflows. It is particularly suited for bioinformatics applications due to its ability to streamline the integration of heterogeneous tools, manage software dependencies, and orchestrate execution across various computational platforms—from personal computers to high-performance computing (HPC) clusters and cloud environments.

In the context of LYNX-RNA, Nextflow's DSL2 framework was utilized to design modular and encapsulated workflow components. Each analytical stage—such as quality control, quantification, differential expression, co-expression network construction, and neoepitope prediction—was implemented as an independent module. This modularity promotes reusability, simplifies debugging, and allows parallel execution of independent tasks, thereby improving computational efficiency.

One of Nextflow's key strengths is its seamless compatibility with software environment managers like Conda and container technologies such as Docker and Singularity. This ensures that all dependencies are explicitly defined and portable, enabling users to reproduce results regardless of their system configurations.

Furthermore, the channel-based data flow model in Nextflow facilitates intuitive data movement and parallelism between workflow steps. This model abstracts away the complexity of file handling and intermediate data storage, allowing researchers to focus on experimental logic rather than low-level scripting.

LYNX-RNA leverages these capabilities to deliver a reproducible, flexible, and user-friendly RNA-seq analysis pipeline that can be deployed both locally and in HPC environments. With its scalability and reproducibility, Nextflow serves as the

backbone of the LYNX-RNA infrastructure, making it an ideal choice for modern bioinformatics pipeline development.

In Nextflow, a **workflow** is a function that is specialized for composing processes and dataflow logic (i.e. channels and operators).

### 2.3.1 Workflow construction

#### 1. Input and Configuration

Files Required:

- Paired-end FASTQ files (\*\_R1.fastq.gz, \*\_R2.fastq.gz)
- Sample sheet (samplesheet.csv)
- Reference genome (FASTA + GTF)
- Configuration files (YAML/JSON for paths, environment, and parameters)

```
params.reads = "data/fastq/*_{R1,R2}.fastq.gz"
params.genome_fasta = "ref/genome.fa"
params.annotation_gtf = "ref/annotation.gtf"
params.outdir = "results/"
```



## 2. Quality Control (FastQC & MultiQC)

**Tool:** FastQC

**Purpose:** Assess raw read quality, GC content, duplication rate.

**Follow-up:** MultiQC for consolidated reports.

```
process QC {  
  input:  
    file(reads) from file(params.reads)  
  output:  
    file("*.html") into qc_reports  
  script:  
    """"  
    fastqc $reads -o ./qc/  
    """"  
}
```

### 3. Trimming (TrimGalore / fastp)

**Purpose:** Remove adapters and low-quality bases.

**Tool:** TrimGalore or fastp (user configurable)

```
process Trimming {  
  input:  
    tuple val(sample_id), file(reads) from read_pairs  
  output:  
    tuple val(sample_id), file("*.fq.gz") into trimmed_reads  
  script:  
    """"  
    trim_galore --paired ${reads[0]} ${reads[1]} -o trimmed/  
    """"  
}
```

#### 4. Alignment (STAR / HISAT2)

- **Purpose:** Map clean reads to reference genome.
- **Tool:** STAR (default), HISAT2 (optional)

```
process Alignment {  
  input:  
    tuple val(sample_id), file(reads) from trimmed_reads  
  output:  
    tuple val(sample_id), file("*.bam") into aligned_bams  
  script:  
    """"  
    STAR --genomeDir star_index \  
      --readFilesIn ${reads[0]} ${reads[1]} \  
      --readFilesCommand zcat \  
      --outSAMtype BAM SortedByCoordinate \  
      --outFileNamePrefix $sample_id.  
    """"  
}
```

## 5. Gene-level Quantification (featureCounts / Salmon)

**Tool:** featureCounts (for BAM), optionally Salmon (quasi-mapping)

**Output:** Gene count matrix

```
process Quantification {  
  input:  
    tuple val(sample_id), file(bam) from aligned_bams  
  output:  
    file("counts.txt") into gene_counts  
  script:  
    """"  
    featureCounts -a ${params.annotation_gtf} -o counts.txt -T 4 -p -B $bam  
    """"  
}
```

## 6. Normalization and Differential Expression (DESeq2)

**Tool:** DESeq2 in R

**Output:** Normalized counts, DEGs, PCA plot, volcano plot

(R script)

```
dds <- DESeqDataSetFromMatrix(countData = counts,  
                              colData = coldata,  
                              design = ~ condition)  
  
dds <- DESeq(dds)  
res <- results(dds)  
write.csv(as.data.frame(res), file="DEGs.csv")
```

## 7. Functional Enrichment (GO/KEGG)

**Tool:** clusterProfiler / enrichR

**Input:** DEGs

**Output:** GO and KEGG enrichment tables + plots

```
import gseapy as gp
import pandas as pd

# Load your DEG list
degs = pd.read_csv("DEGs.csv")
genes = degs['gene'].dropna().tolist()

# Perform GO Biological Process enrichment
enr = gp.enrichr(gene_list=genes,
                 gene_sets='GO_Biological_Process_2021',
                 organism='Human',
                 description='GO_BP',
                 outdir='go_results',
                 cutoff=0.05)

# Results
```

## 8. Network Construction (WGCNA + Cytoscape)

**Tool:** WGCNA for gene co-expression modules

**Export:** Network file for Cytoscape

```
import pandas as pd
import numpy as np
from scipy.cluster.hierarchy import linkage, fcluster, dendrogram
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import matplotlib.pyplot as plt

# Load normalized expression data (genes as rows)
datExpr = pd.read_csv("normalized_counts.csv", index_col=0)

# Optional: transpose if genes are in columns
datExpr = datExpr.T

# Standardize data
scaler = StandardScaler()
dat_scaled = scaler.fit_transform(datExpr)

# Compute correlation matrix
cor_matrix = np.corrcoef(dat_scaled.T)

# Compute dissimilarity (1 - correlation)
dissimilarity = 1 - cor_matrix

# Hierarchical clustering
linkage_matrix = linkage(dissimilarity, method='average')

# Plot dendrogram
plt.figure(figsize=(10, 6))
dendrogram(linkage_matrix, labels=datExpr.columns)
plt.title("Gene Co-expression Dendrogram")
plt.show()

# Assign modules (clusters)
module_labels = fcluster(linkage_matrix, t=1.15, criterion='distance')
modules = pd.DataFrame({'Gene': datExpr.columns, 'Module': module_labels})
modules.to_csv("gene_modules.csv", index=False)
```

## 9. Visualization and Reporting (MultiQC + Plots)

**Purpose:** Generate integrated HTML report

**Tool:** MultiQC + R plots (PCA, volcano, heatmap)

```
multiqc ./ -o results/multiqc/
```

### 2.3.2 Environment Management

1. Use **Conda** or **Docker/Singularity** to ensure reproducibility:

```
nextflow run lynx-rna.nf -profile conda
```

2. Singularity

```
nextflow run lynx-rna.nf -profile singularity
```



### 2.3.3 Output Structure

```
results/  
├─ qc/  
├─ trimmed/  
├─ alignment/  
├─ counts/  
├─ degs/  
├─ enrichment/  
├─ networks/  
└─ multiqc_report.html
```

## 2.4 Statistical Foundations from RNA-seq

The statistical foundations underpinning RNA-seq analysis are essential for transforming raw sequence data into biologically meaningful insights. This section outlines the end-to-end statistical workflow used in LYNX-RNA, detailing the mathematical models, algorithms, and interpretation techniques employed in the analysis of longitudinal RNA-seq data from Immune Thrombocytopenia (ITP) patients across three treatment stages.

### 2.4.1 Quality Assessment and Preprocessing

RNA-seq analysis begins with raw sequencing reads in FASTQ or FASTQ.GZ format, which include both nucleotide sequences and associated Phred quality scores. Each base call is assigned a probability PPP of being incorrect, with the corresponding Phred score computed as:

$$Q = -10\log_{10}(P)$$

Higher Q values reflect higher confidence in the base call. Tools like FastQC and fastp are used to generate summary metrics such as per-base quality, GC content, sequence length distribution, and adapter contamination. Statistical assessments at this stage may include Chi-square goodness-of-fit tests to evaluate deviations in nucleotide distribution from the expected 25% uniform base frequency in un-biased libraries. Removal of low-quality bases and adapter sequences is critical to minimize downstream alignment errors and improve quantification accuracy.

### 2.4.2 Alignment and Mapping Quality

Cleaned reads are aligned to a reference genome (e.g., GRCh38) using splice-aware aligners such as STAR or HISAT2, which implement dynamic programming algorithms and scoring matrices that reward matches and penalize mismatches, insertions, deletions, and splicing junctions. The Mapping Quality Score (MAPQ) is used to quantify confidence in the alignment:

$$\text{MAPQ} = -10\log_{10}(P)$$

where PPP is the probability that the alignment is incorrect. Aligners may use Bayesian probability models or maximum-likelihood estimations to compute these scores. Aligned reads are stored in SAM/BAM files and subsequently indexed. Tools like Samtools or Qualimap provide statistical summaries of alignment quality, including mapping percentage, read depth distribution, and coverage bias.

### 2.4.3 Transcript Quantification and Normalization

Once aligned, reads must be assigned to genomic features (genes or transcripts) for quantification. Two main approaches exist:

- Alignment-based: Tools like featureCounts assign reads to features in GTF files based on overlapping coordinates.
- Alignment-free (pseudoalignment): Tools like Salmon use a k-mer based Expectation-Maximization (EM) algorithm for rapid transcript quantification.

Salmon corrects for GC bias, sequence-specific bias, and positional bias, improving transcript-level expression estimation.

To enable inter-sample comparison, expression values are normalized. A common normalization unit is Transcripts Per Million (TPM), calculated as:

$$\text{TPM}_i = \frac{\frac{R_i}{L_i}}{\sum_j \frac{R_j}{L_j}} \times 10^6$$

Where  $R_i$  is the number of reads mapping to transcript  $i$ , and  $L_i$  is its effective length.

#### 2.4.4 Statistical Modelling for Differential Expression

The central goal of many RNA-seq experiments is to identify genes with statistically significant differences in expression between experimental groups. DESeq2 is a widely used R package that models raw count data using a negative binomial distribution, which captures both mean-variance dependence and biological variability:

$$K_{ij} \sim \text{NB}(\mu_{ij}, \alpha_i), \quad \mu_{ij} = s_j q_{ij}$$

Where:

- $K_{ij}$  is the observed count for gene  $i$  in sample  $j$ ,
- $\mu_{ij}$  is the expected mean,
- $s_j$  is a size factor for sample  $j$  (to account for library depth),
- $q_{ij}$  is the true expression level,
- $\alpha_i$  is the dispersion parameter for gene  $i$ .

Wald tests or likelihood ratio tests are used to assess the null hypothesis

$$H_0 : \log_2(\text{fold change}) = 0$$

Significance is corrected using the Benjamini-Hochberg procedure to control the false discovery rate (FDR).

#### 2.4.5 Functional Enrichment and Gene Ontology Analysis

Identified DEGs are functionally annotated using enrichment analysis, which determines whether certain biological processes, pathways, or cellular components are overrepresented among DEGs. Tools such as clusterProfiler, enrichR, and DAVID apply the hypergeometric test:

$$P(X \geq k) = 1 - \sum_{i=0}^{k-1} \frac{\binom{K}{i} \binom{N-K}{n-i}}{\binom{N}{n}}$$

Where:

- $N$  = total number of genes,
- $K$  = number of genes annotated with a specific term,
- $n$  = number of DEGs,
- $k$  = number of DEGs annotated with the term.

Pathway enrichment is extended using tools like GSEA (Gene Set Enrichment Analysis), which ranks all genes by fold-change and computes an enrichment score (ES) for predefined gene sets. The ES is normalized to account for gene set size, producing a normalized enrichment score (NES) with significance determined by permutation testing.

#### 2.4.6 Network and Module-Based Analysis

In LYNX-RNA, additional statistical modeling is applied through WGCNA (Weighted Gene Co-expression Network Analysis). WGCNA constructs a scale-free network by calculating pairwise gene correlations and clustering them into modules based on topological overlap. Module eigengenes are correlated with clinical traits using Pearson correlation, and hub genes are identified via centrality measures in STRING-based PPI networks.

#### 2.4.7 Immune Profiling and Sample-Level Scoring

Using ssGSEA and GSVA, per-sample enrichment scores are calculated for immune cell type signatures and metabolic pathways. These tools implement non-parametric, rank-based methods to score the degree of enrichment for each gene set in individual samples, enabling immune microenvironment inference and immune-metabolic correlation with treatment response.

#### **2.4.8 Machine Learning for Predictive Feature Ranking**

For biomarker identification, LYNX-RNA integrates machine learning models including Random Forest and XGBoost. These models use gene expression matrices as input features and sample labels (e.g., Control, Pre, Week, Month) as classification targets. [25]

Model performance is evaluated using:

- Accuracy, Precision, Recall, F1-score
- ROC-AUC (Receiver Operating Characteristic – Area Under Curve)
- Cross-validation metrics (e.g., 5-fold stratified CV)

Feature importance scores are extracted to rank predictive biomarkers, adding a layer of statistical prioritization beyond DEG thresholds.

## CHAPTER – 3

### METHODOLOGY – PIPELINE DESIGN

#### 3.1 Development Environment

##### 3.1.1 Workflow Orchestration with Nextflow

1. **Modularity:** Each step in the RNA-seq workflow (e.g., trimming, mapping, quantification) is implemented as an independent process, allowing for selective re-execution.
2. **Portability:** Nextflow pipelines are platform-agnostic and can run seamlessly on local machines, cloud services (AWS, GCP), or HPC clusters.
3. **Container Support:** Each module runs inside its own isolated container (Docker/Singularity), ensuring full reproducibility regardless of the underlying OS or library dependencies.
4. **Resource Management:** Memory, CPU, and storage requirements can be defined per process, enabling efficient resource utilization even in low-RAM ( $\leq 24\text{GB}$ ) environments.

**Profile-based Configuration:** Custom execution profiles (local, colab, docker, hpc) allow the same codebase to be deployed across environments with no manual edits.

##### 3.1.2 Programming Environment and Toolchain

1. LYNX-RNA was implemented using a hybrid programming approach leveraging the strengths of **Python** and **R**, both of which play critical roles in the pipeline.
2. **Python (v3.10)**
3. Python is responsible for:
4. **Preprocessing Automation:** Orchestration of trimming, QC, and mapping processes using subprocess, os, and pathlib.
5. **Machine Learning Models:**
  - a. **Random Forest** and **XGBoost** classifiers were implemented using scikit-learn and xgboost for identifying predictive biomarkers based on normalized gene expression.
  - b. **Cross-validation** strategies (e.g., LOOCV, stratified k-fold) were integrated to ensure generalizable performance.

**6. CLI Wrapper:** A user-friendly command-line interface was built in Python to abstract complex command sequences. It includes:

- a. Automatic detection of file formats and directories
- b. Interactive prompts and parameter validation
- c. Real-time progress tracking and error logging

**7. LLM Report Generation:** A custom module connects to the **OpenAI API**, using GPT-4 to generate human-readable interpretations of pipeline results. This provides domain-agnostic users with clear summaries of key findings, charts, and statistical conclusions.

## **8. R (v4.2.2)**

**9. R complements Python for the statistical and biological interpretation of RNA-seq data:**

**Differential Expression Analysis:** Conducted using the DESeq2 package with support for multi-factor designs, batch correction, and shrinkage estimation.

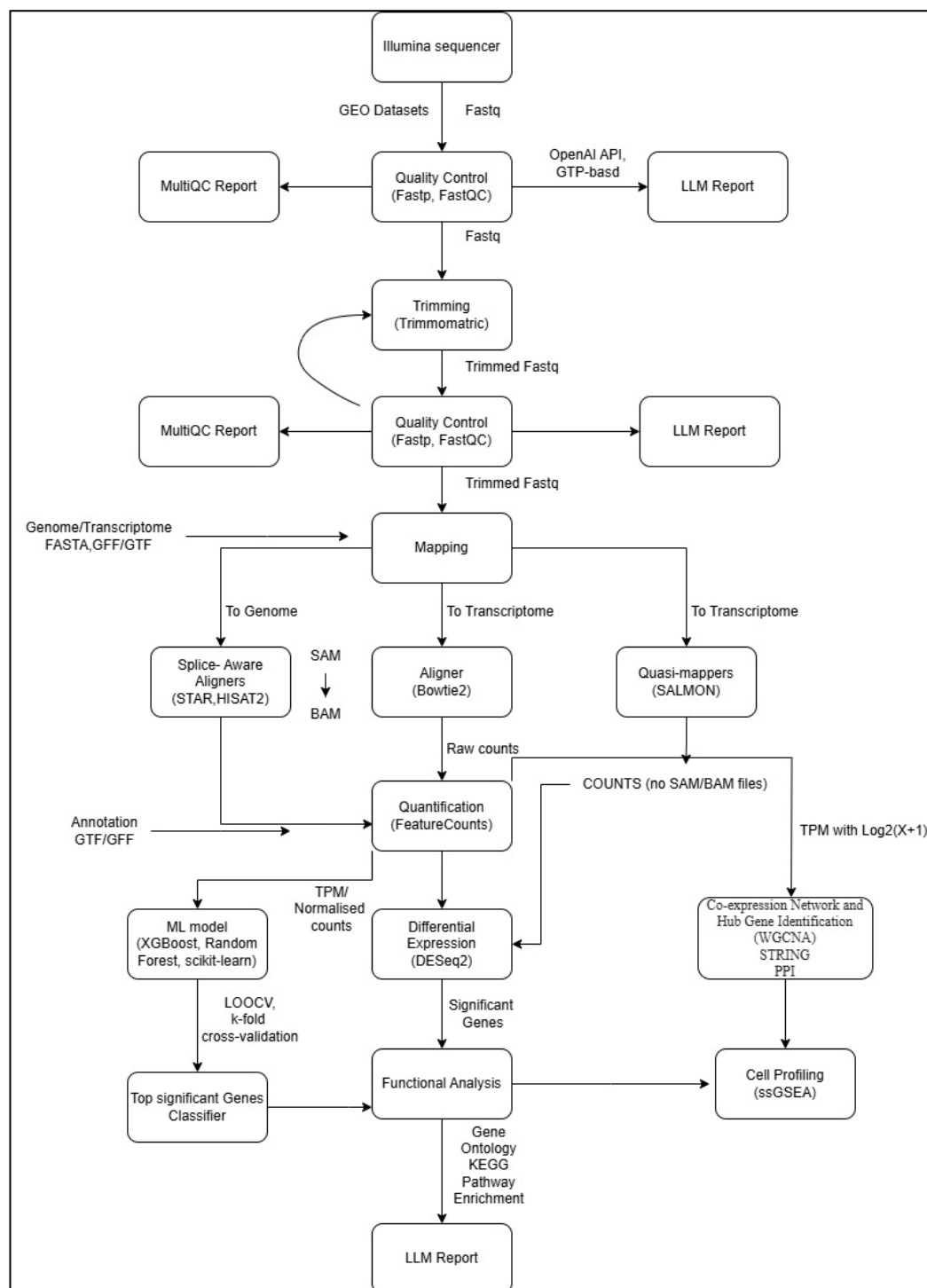
**Network Analysis:** The WGCNA package was used to identify gene modules and hub genes based on expression co-variation.

**Functional Enrichment:** The pipeline supports GO/KEGG enrichment via ClusterProfiler, ReactomePA, and enrichplot. For PPI analysis, STRINGdb was used to query the STRING database and visualize networks.

**Immune Profiling:** GSVA and ssGSEA were used for immune cell signature profiling, providing insight into cell-type infiltration and immune modulation.

**Visualization:** High-resolution plots were generated using ggplot2, EnhancedVolcano, pheatmap, and ComplexHeatmap

### 3.2 Architecture of LYNX-RNA



**Fig 3.1** Workflow of LYNX-RNA rna-seq pipeline



LYNX-RNA is organized into five primary stages, each composed of modular processes that can be individually configured, reused, or replaced depending on the research goal:[22]

*Stage 1: Preprocessing and Quality Control.* Raw FASTQ files (single-end or paired-end) are subjected to initial quality control using FastQC (v0.11.9) and read trimming with fastp (v0.23.2) to remove low-quality bases and adapters. UMI-tools (v1.1.2) is optionally employed for unique molecular identifier (UMI) extraction in UMI-based library protocols. Contaminant filtering and rRNA depletion are handled using SortMeRNA and BBSplit. These steps ensure high-quality, deduplicated, and cleaned reads for downstream analysis. [23]

*Stage 2: Alignment and Quantification.* Reads are aligned to the human reference genome (GRCh38) using STAR (v2.7.10a) or HISAT2 (v2.2.1). For quantification, transcript-level abundances are estimated using Salmon (v1.10.1) in quasi-alignment mode. Optionally, featureCounts (via Subread) is used to produce gene-level count matrices for downstream analysis. All aligned BAM files are indexed and sorted using SAMtools (v1.15), with optional duplication marking by Picard Tools. [24]

*Stage 3: Differential Expression and Functional Analysis.* Gene-level count matrices are imported into R (v4.3.1) for statistical analysis using DESeq2 (v1.40.2). Differential expression is computed across defined comparisons (e.g., Control vs Pre, Control vs 1 Week, Control vs 1 Month). Genes with an adjusted p-value (FDR)  $< 0.05$  and  $|\log_2 \text{fold change}| \geq 1$  are considered significant. Functional enrichment analysis is performed using clusterProfiler and g:Profiler to identify overrepresented Gene Ontology (GO) terms and KEGG pathways.[25]

*Stage 4: Co-expression Network and Hub Gene Identification.* Weighted Gene Co-expression Network Analysis (WGCNA) is implemented to identify co-expressed gene modules associated with clinical stages. A scale-free network is constructed with soft-threshold power  $\beta = 5$  and scale-free topology fit  $R^2 > 0.95$ . Modules are defined by topological overlap and merged if eigengene correlation  $> 0.75$ . Module-trait relationships are evaluated using Pearson correlation. Genes from significant modules are used to build a protein-protein interaction (PPI) network via the STRING database API, and Cytoscape is used for visualization. Hub genes are identified based on node degree centrality. [26]

*Stage 5: Machine Learning-Based Biomarker Discovery.* Normalized expression matrices (e.g., TPM or variance-stabilized counts) are exported for supervised machine learning in Python (v3.11). Feature selection and classification are performed using Random Forest and XGBoost algorithms. The dataset is stratified into training and test sets (typically a 70/30 split), and model performance is evaluated using metrics including accuracy, F1-score, and ROC-AUC. Genes are ranked based on feature importance scores, and top-ranked genes are further analyzed for biological relevance.[39]

*Stage 6: Pathway and Immune Enrichment Analysis.* Gene Set Enrichment Analysis (GSEA) and single-sample GSEA (ssGSEA) are performed using the GSVA R

package with curated gene sets (e.g., c2.cp.kegg.v7.5.1 and c7.immunesigdb.v7.5.1 from MSigDB). Pathway activity scores are calculated for each sample, and immune/metabolic enrichment patterns are visualized using violin plots, ridge plots, and heatmaps. Correlation analysis is used to link hub gene expression with enrichment scores.[40]

*Stage 7: Immune Cell Infiltration Profiling.* To assess immune microenvironment dynamics, ssGSEA-based immune infiltration analysis is performed using predefined immune gene sets (e.g., LM22). Relative infiltration scores for T cells, NK cells, B cells, macrophages, and dendritic cells are calculated. Comparisons are made between ITP and control groups, and correlation analysis links hub gene expression to specific immune cell types. Outputs include violin plots, bar charts, and correlation matrices.[41]

*Stage 8: Large Language Model (LLM)-Driven Report Generation.* To enhance interpretability, LYNX-RNA integrates a Large Language Model (LLM) interface (OpenAI API, GPT-based) that automatically generates structured, natural-language summaries of key results. Summaries include DEG highlights, pathway interpretations, classifier performance narratives, and conclusions tailored to clinicians or non-specialist researchers. This AI-assisted reporting bridges the gap between computational outputs and biological insight.[42]

*Execution and Environment* LYNX-RNA is orchestrated using Nextflow (v22.10.1), enabling scalable execution across local machines, HPC clusters, and cloud platforms (e.g., AWS Batch). All tools are managed via Conda environments or Docker containers, ensuring reproducibility. The pipeline accepts configuration through YAML files and supports parallelization across multiple samples and processes.

### 3.3 Quality Control & Preprocessing

#### 3.3.1 Check quality with FastQC-

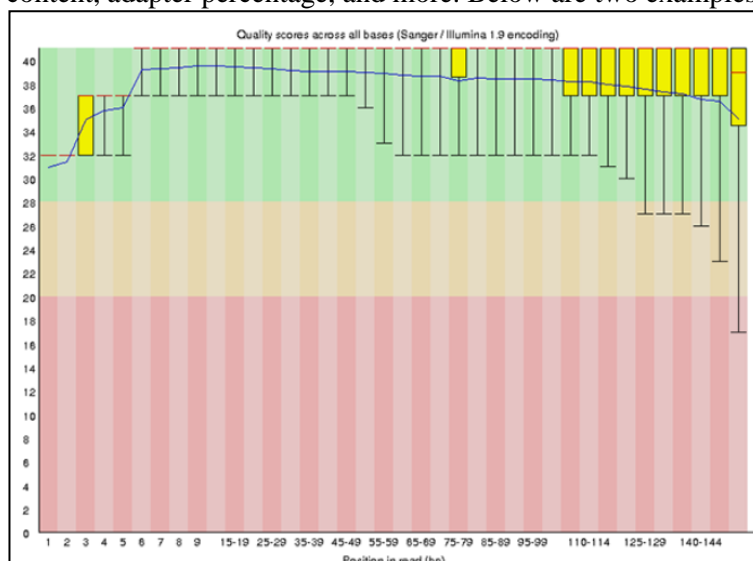
The main functions of FastQC are-

1. Import of data from BAM, SAM or FastQ files (any variant)
2. Providing a quick overview to tell you in which areas there may be problems
3. Summary graphs and tables to quickly assess your data
4. Export of results to an HTML based permanent report
5. Offline operation to allow automated generation of reports without running the interactive application. [30]

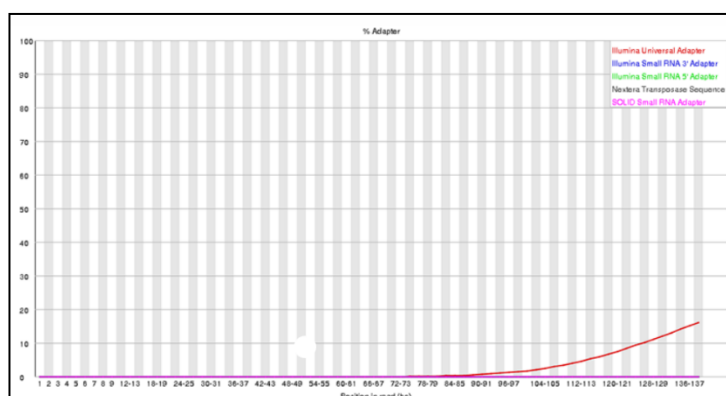
Run FastQC to check the raw data quality.

```
fastqc sample_01.fastq.gz --extract -o /path/to/output_folder
```

The output contains graphs and statistics about the raw quality, including quality scores, GC content, adapter percentage, and more. Below are two examples of the output files.[1]



**Fig.3.2** Per base sequence quality. Quality scores for each base position in the read are represented as box plots. The blue line represents the average quality score. High-quality data will typically have over 80% of bases with a quality score of 30 or higher (i.e., Q30 > 80%). Q30 represents 99.9% accuracy in the base call, or an error rate of 1 in 1000. A dip in quality is expected towards the end of the read.

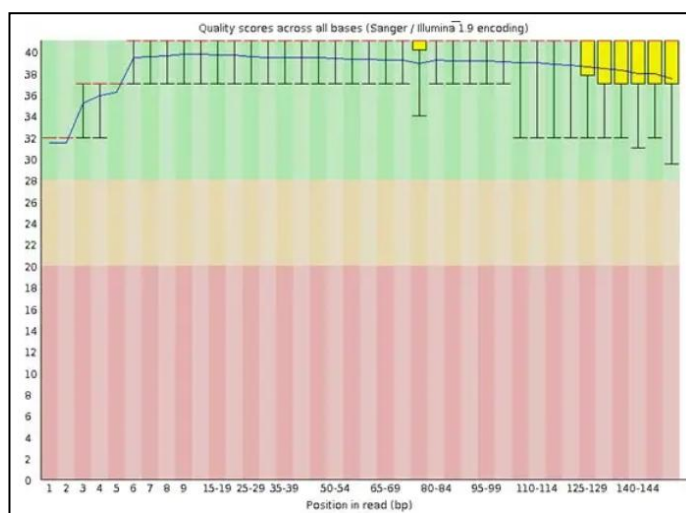


**Fig.3.3** Adapter content. Percentage of reads that match the Illumina adapter sequence (red) is plotted for each base position. Since standard library preparations capture a range of insert sizes, some sequenced fragments will be shorter than the read length (<150 bp in this case).

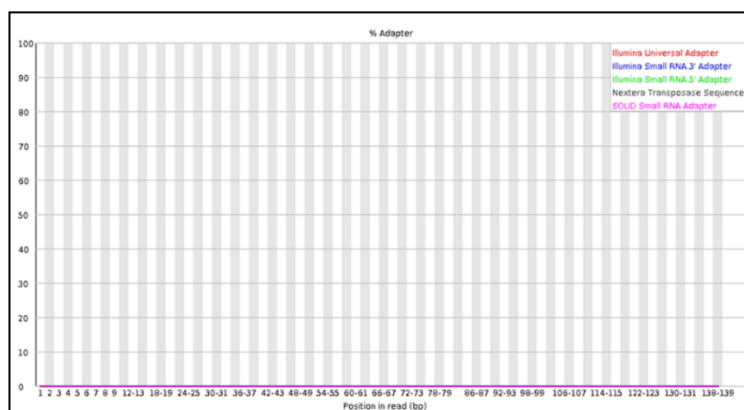
### 3.3.2 Trim reads with Trimmomatic

Poor-quality regions and adapter sequences should be trimmed from the reads before further analysis. Since Trimmomatic has an executable JAR file, you'll need to use Java to execute it rather than doing so directly in the command line.[30]

```
java -jar trimmomatic-0.39.jar PE input_forward.fastq.gz input_reverse.fastq.gz
output_forward_paired.fastq.gz output_forward_unpaired.fastq.gz
output_reverse_paired.fastq.gz output_reverse_unpaired.fastq.gz ILLUMINACLIP:TruSeq3-
PE.fa:2:30:10:2:keepBothReads LEADING:3 TRAILING:3 MINLEN:36
```



**Fig.3.4** Per base sequence quality after trimming. Notice the improvement at the end of the read, compared to the raw data above. All box plots are within the high-quality (green) zone.



**Fig.3.5** Adapter content after trimming. Adapter sequences have been completely removed from the reads, as expected

## 3.4 Transcript Quantification

### 3.4.1 STAR aligner

Basic STAR workflow consists of 2 steps-

- 1. Genome Index Generation**-Prior to read alignment, STAR requires the creation of **genome index files**, which are derived from user-supplied reference genome sequences (FASTA) and corresponding gene annotation files (GTF/GFF). These index files serve as an optimized reference data structure, enabling rapid alignment of RNA-seq reads in subsequent steps. The index generation step is computationally intensive and typically needs to be performed only once per genome version (e.g., GRCh38/hg38). The resulting index files are stored locally and reused across multiple experiments, provided that the genome and annotation combination remains the same.
- 2. Read Mapping to the Reference Genome**—Once the genome index is generated, STAR performs the alignment of RNA-seq reads (in FASTQ format) to the indexed genome. This mapping process is optimized for both speed and sensitivity, particularly for detecting splice junctions in eukaryotic transcripts.[31],[13]

During this phase, STAR accepts the following as inputs:

- 1) Indexed genome files from Step 1
- 2) Trimmed RNA-seq reads (FASTQ)
- 3) Alignment parameters specified by the user or pipeline configuration

STAR outputs a comprehensive set of files, including:

- 1) Alignment files in SAM or BAM format
- 2) Mapping summary statistics, such as uniquely mapped reads, multi-mapped reads, and mismatch rates
- 3) Splice junction annotation files (useful for novel transcript discovery)
- 4) Unmapped read logs for troubleshooting and downstream analysis
- 5) Wiggle (WIG) or BigWig files for signal track visualization in genome browsers

### 3.4.2 Generating genome indexes

Before RNA-seq reads can be mapped to a reference genome, STAR requires the creation of **genome index files**. These indexes are pre-computed data structures that allow STAR to rapidly align sequencing reads to the genome while preserving high sensitivity to spliced alignments. Genome index generation is performed only once for each reference genome and annotation combination and should be repeated only when updated genome assemblies or gene annotations are introduced. The process uses a set of core command-line options described below.[32],[13]

```
STAR --runThreadN 8 \
    --runMode genomeGenerate \
    --genomeDir /path/to/genomeDir \
    --genomeFastaFiles /path/to/genome.fasta \
    --sjdbGTFfile /path/to/annotations.gtf \
    --sjdbOverhang 99
```

#### Parameter Descriptions:

1. **--runThreadN**  
Specifies the number of CPU threads to use for index generation. This should be set based on the number of cores available on the machine (e.g., 8, 16, 32).
2. **--runMode genomeGenerate**  
Instructs STAR to run in genome indexing mode instead of alignment mode.
3. **--genomeDir**  
Indicates the output directory where the generated genome index files will be stored. This directory must be created before execution (using mkdir) and should have sufficient write permissions. For mammalian genomes, at least **100 GB of free disk space** is recommended.
4. **--genomeFastaFiles**  
Specifies one or more reference genome FASTA files. These files must contain all chromosomes or scaffolds in the correct format. While it is not required, chromosome names can be manually edited in the chrName.txt file (generated by STAR) to customize naming conventions across outputs.
5. **--sjdbGTFfile**  
Provides the path to a **GTF annotation file** containing transcript and exon

structures. STAR uses this file to extract known splice junctions, which enhances alignment accuracy near exon-intron boundaries. Though optional, using annotations is strongly recommended.

6. **-sjdbOverhang**

Sets the length of the genomic sequence flanking annotated splice junctions. This value should ideally be **ReadLength - 1**. For example, with 100 bp Illumina reads, a typical value is **99**. For variable-length reads, the maximum read length minus one is a suitable approximation.[13]

### 3.4.3 Output of Genome Index Generation

The output files generated by STAR during this process include:

1. Binary genome files and suffix arrays
2. Chromosome name and length metadata
3. Splice junction coordinates
4. Encoded transcript/gene annotation information

These files are stored in STAR's internal format and are used during the mapping phase. It is strongly advised **not to manually modify these files**, except for optionally editing `chrName.txt` to customize chromosome labels in output SAM/BAM files.

Once generated, the index files in the `genomeDir` directory can be reused across multiple alignment jobs using the same reference genome and annotations, thus avoiding redundancy and improving efficiency.

First, index the reference genome using STAR to prepare it for alignment. Adding gene annotation information to the reference genome will facilitate alignment of RNA-Seq reads across exon-intron boundaries. This indexing step is only required once; you can then use the indexed genome repeatedly in future analysis.[1]

Check the mapping statistics in the `[sample_name]Log.final.out` file to ensure the BAM file was generated properly and the reads align to the genome correctly. Uniquely mapped reads are the most useful for expression analysis, as there is high confidence in which loci they represent. In general, >60-70% for the “uniquely mapped reads %” metric is considered good; a significantly lower value warrants further investigation.[1]

Lastly, use Samtools to sort and index the BAM files. Organizing the reads by position within the BAM file is needed for downstream analysis.

### 3.5 Differential Expression Analysis

Compare hit counts between groups with DESeq2-

The DESeq2 package is designed for normalization, visualization, and differential analysis of high-dimensional count data. It makes use of empirical Bayes techniques to estimate priors for log fold change and dispersion, and to calculate posterior estimates for these quantities

results tables with log2 fold change, p-values, adjusted p-values, etc. for each gene are best generated using the results function. The coef function is designed for advanced users who wish to inspect all model coefficients at once.[6]

The differential expression analysis uses a generalized linear model of the form:

$$K_{ij} \sim \text{NB}(\mu_{ij}, \alpha_i)$$

$$\mu_{ij} = s_j q_{ij}$$

$$\log_2(q_{ij}) = x_j \beta_i$$

where counts  $K_{ij}$  for gene  $i$ , sample  $j$  are modeled using a Negative Binomial distribution with fitted mean  $\mu_{ij}$  and a gene-specific dispersion parameter  $\alpha_i$ . The fitted mean is composed of a sample-specific size factor  $s_j$  and a parameter  $q_{ij}$  proportional to the expected true concentration of fragments for sample  $j$ . The coefficients  $\beta_i$  give the log2 fold changes for gene  $i$  for each column of the model matrix  $X$ . The sample-specific size factors can be replaced by gene-specific normalization factors for each sample using normalization Factors. [5]

#### 3.5.1 Core Workflow

The standard DESeq2 pipeline involves the following steps:

1. Dataset Construction: The DESeqDataSet object is built from raw counts and a sample metadata table. Input can be generated directly from tximport, HTSeq, or a matrix of counts.
2. Normalization: Sample-wise differences in sequencing depth are corrected using size factors via estimateSizeFactors(), applying a median-of-ratios method.
3. Dispersion Estimation: Gene-wise dispersion estimates are computed and then modeled as a function of mean expression using parametric, local, or mean-based fitting (estimateDispersions()).
4. Model Fitting and Testing: A generalized linear model (GLM) is fitted per gene, and hypothesis testing is performed:
  - a. Wald test (test = "Wald") for significance of individual coefficients.
  - b. Likelihood Ratio Test (LRT) (test = "LRT") for comparing full and reduced models.
5. Result Extraction: The results() function provides statistics such as log2 fold change (LFC), p-values, and adjusted p-values (Benjamini-Hochberg FDR).[4]



### 3.5.2 Data Transformation and Visualization

To facilitate downstream exploratory analysis such as clustering and PCA, DESeq2 offers:

1. Variance Stabilizing Transformation (VST): `vst()` quickly transforms count data while preserving the mean-variance relationship.
2. Regularized Log Transformation (`rlog`): `rlog()` is more robust for datasets with varying library sizes but computationally intensive.
3. PCA and Diagnostic Plots:
  - a. `plotPCA()` for sample clustering.
  - b. `plotMA()` to visualize changes in expression.
  - c. `plotDispEsts()` for assessing dispersion fits.[25]

### 3.5.3 Additional Features

1. Outlier Detection: DESeq2 uses Cook's distance to identify and optionally replace outliers in high-replicate datasets.
2. FPKM/FPM Calculation: Functions like `fpkm()` and `fpm()` allow conversion of counts into normalized expression values.
3. Integration with Single-Cell Data: DESeq2 supports integration with preprocessed scRNA-seq datasets via the `integrateWithSingleCell()` function.

### 3.5.4 Outputs

1. DESeqResults: Contains `log2FoldChange`, `pvalue`, `padj`, `stat`, etc.
2. Normalized counts: Via `counts(dds, normalized=TRUE)`
3. Transformed data: `vst(dds)`, `rlog(dds)`
4. Dispersion trends: `plotDispEsts(dds)`

## 5. PCA & MA plots: plotPCA(), plotMA()

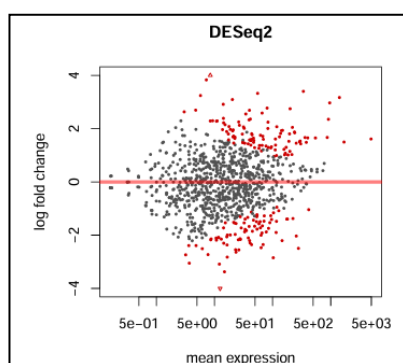
DataFrame with 1000 rows and 6 columns						
	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>
gene1	12.247	1.5333	0.8054	1.9039	0.05693	0.2096
gene2	22.568	1.3896	0.6556	2.1197	0.03403	0.1554
gene3	3.961	-1.8592	0.9755	-1.9059	0.05666	0.2096
gene4	143.035	-0.5476	0.3421	-1.6010	0.10939	0.3082
gene5	16.301	-0.2533	0.6843	-0.3702	0.71125	0.8570
...	...	...	...	...	...	...
gene996	9.5873	1.33286	0.8254	1.61480	0.1064	0.3014
gene997	6.6044	0.08247	0.8567	0.09627	0.9233	0.9695
gene998	8.5560	-0.78911	0.7933	-0.99472	0.3199	0.5937
gene999	0.9542	0.66981	0.9694	0.69098	0.4896	NA
gene1000	3.7779	0.68939	1.0170	0.67789	0.4978	0.7372

**Fig 3.6.** DESeq2 result output layout

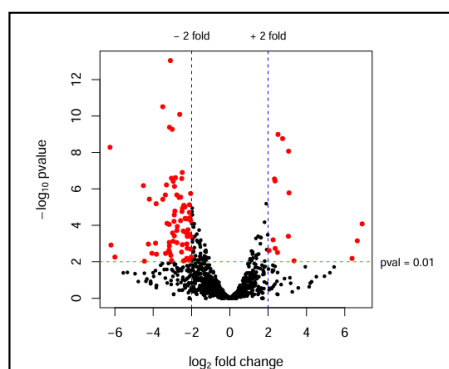
The result of the columns of `resDESeq2` (a *DESeqResults* object) is as follows:

	feature identifier
baseMean	mean normalised counts, averaged over all samples from both conditions
log2FoldChange	the logarithm (to basis 2) of the fold change
lfcSE	standard errors of logarithm fold change
stat	test statistics
pvalue	p value for the statistical significance of this change
padj	p value adjusted for multiple testing with the Benjamini-Hochberg procedure (see the R function <code>p.adjust</code> ), which controls false discovery rate (FDR)

Intrepreation –



**Fig 3.7.** MA-plot generated by the plotMA function in DESeq2. Points will be colored red if the adjusted p-value is less than 0.1. Points which fallout of the window are Plotted as open triangles pointing either up or down.



**Fig 3.8 :**Volcano plot.The red points indicate genes-of-interest that display both large-magnitude fold-changes (x-axis) as well as high statistical significance (  $\log_{10}$  of p-value, y-axis).The dashed green-line shows the p-value cut-off ( $pval=0.01$ ) with points above the line having p-value  $<0.01$  and points below the line having p-value  $> 0.01$ . The vertical dashed blue lines show 2-fold changes.

### 3.6 Network Analysis

**WGCNA** is an R package developed for the systems-level analysis of gene expression data. It is primarily used to identify clusters (modules) of highly correlated genes and to relate these modules to clinical traits or phenotypes. WGCNA constructs gene co-expression networks using pairwise correlation coefficients, thereby allowing the identification of gene modules with shared biological functions.[20]

#### 3.6.1. Conceptual Framework

WGCNA is based on the concept of constructing a **scale-free network** using soft thresholding of gene-gene correlations. The network is undirected and weighted, capturing the continuous nature of gene relationships.

1. **Adjacency Matrix:** Calculated from pairwise correlations, raised to a power  $\beta$  (soft-thresholding power) to emphasize strong correlations while suppressing weak ones.
2. **Topological Overlap Matrix (TOM):** Measures network interconnectedness, accounting for shared neighbors between genes.
3. **Hierarchical Clustering:** Applied to TOM to identify gene modules.
4. **Module Eigengene:** The first principal component of a module's expression matrix; used to summarize module activity and correlate with phenotypic traits.

### 3.6.2. Core Workflow

1. **Input Preparation:** Expression matrix with genes in columns and samples in rows.
2. **Soft-thresholding Power Selection:** Identify a suitable  $\beta$  to approximate scale-free topology using `pickSoftThreshold()`.
3. **Adjacency and TOM Calculation:** Use `adjacency()` and `TOMsimilarityFromExpr()` to construct the network.
4. **Module Detection:** Apply `blockwiseModules()` to detect modules via dynamic tree cutting and module merging.
5. **Module-Trait Association:** Relate modules to traits using correlation between module eigengenes and sample metadata.
6. **Hub Gene Identification:** Determine intra-modular hub genes using connectivity measures.[15]

**In LYNX-RNA pipeline, WGCNA is typically integrated like this:**

1. **Input:** VST-transformed expression matrix from DESeq2.

**Script:**

```
library(WGCNA)
datExpr <- read.table("vst_transformed_counts.tsv", header=TRUE, row.names=1)
powers <- c(1:20)
sft <- pickSoftThreshold(datExpr, powerVector = powers)
net <- blockwiseModules(datExpr, power = chosenPower, ...)
exportNetworkToCytoscape(net$TOM)
```

**Output:** Module membership, trait correlation table, hub genes, Cytoscape files.

**Execution:** Script is called as a process within the Nextflow `.nf` file and defined in `main.nf`

### 3.7 Functional Enrichment

Once differentially expressed genes (DEGs) have been identified, understanding their **biological roles** and **pathway associations** becomes essential for interpreting the underlying mechanisms of a condition. **Functional enrichment analysis** aims to determine whether specific biological categories—such as **Gene Ontology (GO) terms**, **KEGG pathways**, or **Reactome modules**—are statistically overrepresented in the list of DEGs compared to a background set of genes.[40]

This process helps reveal affected biological processes, cellular components, and molecular functions, offering insight into disease pathogenesis, treatment response, or cellular phenotypes. In the context of LYNX-RNA, enrichment analysis is automatically triggered for each comparison group using well-established R packages and databases.

#### 3.7.1 Gene Ontology and Pathway Databases

The LYNX-RNA pipeline supports enrichment using the following annotation databases:

1. **Gene Ontology (GO)**: Covers three domains:
  - a. **Biological Process (BP)** – e.g., T cell activation, mitotic cell cycle
  - b. **Molecular Function (MF)** – e.g., ATP binding, enzyme activity
  - c. **Cellular Component (CC)** – e.g., mitochondrion, cytoskeleton
2. **KEGG Pathways**: Curated maps of molecular interaction and reaction networks.
3. **Reactome**: Hierarchical pathway database providing mechanistic insights.
4. **MSigDB (optional)**: Used for gene set enrichment analysis (GSEA), containing curated and computational gene sets.

#### 3.7.2 Statistical Methodology

Functional enrichment analysis involves **over-representation testing**, where the number of DEGs associated with a given term is compared to the expected count under a hypergeometric distribution. The most commonly used statistical model is the **hypergeometric test** (also called Fisher's exact test), calculated as:

$$P(X \geq k) = 1 - \sum_{i=0}^{k-1} \frac{\binom{K}{i} \binom{N-K}{n-i}}{\binom{N}{n}}$$

Where:

- $N$ : Total number of genes in the background (e.g., all genes expressed)
- $K$ : Number of genes annotated with the term
- $n$ : Number of DEGs
- $k$ : Number of DEGs annotated with the term

The **p-values** obtained are adjusted for multiple comparisons using the **Benjamini-Hochberg False Discovery Rate (FDR)** method.[26]

### 3.8 ML Integration

#### 3.8.1 Goals of Machine Learning in LYNX-RNA

The objectives of ML integration are threefold:

1. **Classification:** Predict the clinical status or condition (e.g., treatment timepoint, disease vs. control) of a sample based on its transcriptomic profile.
2. **Feature Selection:** Identify genes that are most informative for classification, serving as potential biomarkers.
3. **Interpretability:** Enable users to understand the biological relevance of selected features using feature importance scores, pathway enrichment, and natural language explanations powered by large language models (LLMs).

This extends RNA-seq workflows into the realm of **supervised learning** and **translational modeling**, with outputs that can support diagnostics, prognostics, or therapeutic stratification.

#### 3.8.2. Data Flow and Preprocessing

##### 1. Input Format

- The ML module consumes a **normalized expression matrix**, generated from earlier stages of the pipeline.
- Each row represents a sample, and each column represents a gene or transcript.

- A separate **metadata file** provides the class label for each sample (e.g., timepoint or treatment group).

## 2 Preprocessing Steps

To prepare data for training and ensure consistent behavior across models, the following steps are applied:

### 1. Gene Filtering:

- a. Low variance genes (genes with little change across samples) are removed.
- b. Genes with more than 80% zero counts or missing values are discarded.

### 2. Normalization:

- a. Log2 transformation:  $\log_2(\text{TPM} + 1)$  or  $\log_2(\text{CPM} + 1)$
- b. Z-score normalization: Each gene is standardized to zero mean and unit variance.

### 3. Dimensionality Reduction (optional):

- a. PCA or t-SNE is applied for visualization.
- b. In cases of very high feature-to-sample ratios, PCA may be used to reduce input to top N components.

### 4. Class Balance Check:

- a. Class distributions are examined.
- b. Synthetic oversampling (SMOTE) or class weighting is used if classes are imbalanced.

### 3.8.3. Algorithms Used

Two robust and interpretable **tree-based ensemble models** are implemented:[27]

#### 1. Random Forest Classifier

- 1) **Type:** Bagging ensemble of decision trees.
- 2) **Advantages:**
  - a. Handles noisy and high-dimensional data well.
  - b. Less prone to overfitting compared to single decision trees.
  - c. Provides **feature importance** via Gini impurity or permutation metrics.

#### 2 XGBoost (Extreme Gradient Boosting)

- 1) **Type:** Boosting-based ensemble of decision trees.
- 2) **Advantages:**
  - a. State-of-the-art model for structured/tabular data.
  - b. Highly efficient, regularized, and scalable.
  - c. Supports early stopping, dropout, and custom loss functions.
  - d. Offers SHAP-based feature attribution for interpretability.

### 3.8.4 Model Training and Validation

#### A. Cross-Validation Strategy[39]

Due to limited sample sizes common in biological datasets, rigorous validation is critical. LYNX-RNA supports:

1. Stratified k-Fold Cross-Validation (default: k=5):
  - a. Ensures proportional representation of classes in each fold.



- b. Provides stable performance estimates.
2. Leave-One-Out Cross-Validation (LOOCV):
- a. Used for very small datasets ( $n < 20$ ).
  - b. Each sample is tested individually using the model trained on the remaining samples.

### 3.8.5 Evaluation Metrics

The performance of classifiers is assessed using:

1. **Accuracy**
2. **Precision, Recall, F1-score**
3. **ROC Curve and AUC (Area Under Curve)**
4. **Confusion Matrix**
5. **Classification Reports** (via [sklearn.metrics](#))

Evaluation plots and summary tables are automatically generated and stored with the output.[39]

### 3.8.6 Feature Importance and Biomarker Identification

#### 1. Importance Extraction

- a. **Random Forest:** Gini-based or permutation importance.
- b. **XGBoost:** Gain, Cover, Frequency, or SHAP values.

#### 2 Biomarker Selection Workflow

1. **Top N Genes:** Genes with the highest importance scores are shortlisted (typically top 20–50).

**a. Annotation & Enrichment:**

- a. These genes are annotated using biomaRt, Ensembl, or org.Hs.eg.db.
- b. They are passed to GO/KEGG enrichment via `clusterProfiler` to identify overrepresented pathways.

**2. Validation:**

- a. If differential expression analysis also flagged these genes, overlap is reported.
- b. LLM (GPT-4) generates descriptive summaries of their biological relevance.

### 3.8.7 Interpretability Enhancements

Interpretability is a core principle in LYNX-RNA. Beyond standard importance metrics, the following are included:

1. **SHAP (SHapley Additive exPlanations):** For model-agnostic explanation of how features contribute to predictions.

2. **LLM-Generated Reports:**

- a. Using the OpenAI GPT API, LYNX-RNA translates complex results into natural language.
- b. Examples:
  - i. “Gene *X* is highly expressed in Week 1 samples and is known to regulate immune cell adhesion.”

“The combination of *Gene A*, *Gene B*, and *Gene C* is predictive of pre-treatment status with 91% accuracy.” [42]

Outputs are saved in structured folders: `reports/ml_summary`, `plots/`, `feature_importance.csv`, `llm_summary.txt`

## CHAPTER – 4

### DATASET AND EXPERIMENTAL SETUP

#### 4.1 Dataset Description

The evaluation of transcriptomic responses to eltrombopag therapy in chronic immune thrombocytopenia (ITP) patients was carried out using publicly available RNA-sequencing datasets retrieved from the NCBI Sequence Read Archive (SRA). To construct a biologically and statistically sound machine learning framework, we selected two high-quality and clinically annotated RNA-seq datasets—one representing treatment samples from ITP patients undergoing eltrombopag therapy, and the other serving as a matched control group derived from healthy individuals. These datasets were processed using the LYNX-RNA pipeline for downstream analysis, normalization, and integration into the machine learning and functional genomics modules. [21]

##### 4.1.1 Eltrombopag-Treated ITP Dataset (Project ID: PRJNA445461)

This dataset focuses on the **longitudinal transcriptomic response to eltrombopag**, a thrombopoietin receptor agonist, in patients with chronic immune thrombocytopenia (ITP). The study was conducted by Stanford University and is publicly available under SRA Project ID **PRJNA445461**. A total of **46 peripheral blood samples** were collected from **17 patients** who were administered **75 mg/day eltrombopag** as monotherapy.[38]

Samples were collected at **three distinct time points**:

1. Pretreatment (Pre),
2. One week after treatment initiation (1wk),
3. One month post-treatment initiation (1mon).

These samples were preserved in PAXgene blood RNA tubes and processed using a **globin mRNA reduction protocol** prior to RNA extraction, which is essential for reducing background signal from abundant hemoglobin transcripts. The sequencing methodology employed was **3'-end RNA sequencing (3SEQ)**, which targets 3' untranslated regions (3'UTRs) to ensure quantification accuracy and minimize

transcript length bias. Each sequencing library generated 36 bp directional reads using the **Illumina HiSeq 2000** platform. Reads were mapped to the **human transcriptome (hg19)**, and transcript-level quantification was performed.

#### 4.1.2. Control Dataset (Project ID: PRJNA1055463)

To establish a robust baseline for comparison, we selected a high-quality control dataset under SRA Project ID **PRJNA1055463**. It includes a sizable cohort of **healthy individuals (n = 89)** who served as the **control group**.

For the purposes of our pipeline, we utilized only the **RNA-seq profiles of the healthy controls**, which represent peripheral blood transcriptomes from clinically screened individuals. RNA was extracted from whole blood, and libraries were prepared and sequenced to provide bulk RNA-seq data. These healthy control samples offered an ideal comparator group for ITP patients due to:

1. Similar tissue type (peripheral blood),
2. Comparable RNA preparation protocols,
3. Lack of drug or disease-induced transcriptomic alterations.

This dataset underwent standard normalization and integration procedures identical to those applied to the ITP samples, allowing for a direct, batch-corrected comparison between **disease (ITP pre-treatment)** and **healthy control** states.

## 4.2 Data Source

For this study, we utilized publicly available RNA-seq data from the Gene Expression Omnibus (GEO) to investigate transcriptomic changes associated with immune thrombocytopenia (ITP). Patient samples were obtained from accession GSE112278, comprising 46 peripheral blood samples from 17 chronic ITP patients, collected at three clinical time points: pretreatment (Pre), 1 week (1wk), and 1 month (1mon) following eltrombopag therapy. Patients were classified on the basis of the timeline of the drug administration based on platelet count response criteria. RNA-seq libraries were prepared using 3'-end sequencing (3SEQ) after globin mRNA depletion and sequenced on the Illumina HiSeq 2000 platform. As a control dataset, we included healthy peripheral blood samples from PRJNA1055463 (BioProject), corresponding to GEO accession GSE251786, to serve as a baseline for differential gene expression and biomarker discovery. These control samples were processed under comparable RNA-seq protocols to ensure analytical consistency. The combined dataset enables longitudinal profiling of transcriptomic dynamics in ITP across disease progression and treatment response.[38]

**Table 4.1. Clinical Characteristics of ITP Patients and Healthy Donors (control)**

Parameter	ITP Responders (n = 8 patients × 3 timepoints = 24 samples)	Healthy Controls (n = X)*
Age (range)	Median 54 (16–84 years)	20–65 years (approx., from metadata)
Sex	Female: 23 (58.97%) Male: 16 (41.03%)	Female: 23 (58.97%) Male: 16 (41.03%)
Type of ITP	Chronic ITP (duration >12 months)	Not applicable
Treatment	Eltrombopag (75 mg/day, oral)	None
Blood Sampling Timepoints	Pre-treatment, 1 week, 1 month	Single timepoint
Sample Source	Peripheral whole blood (PAXgene tubes)	Peripheral whole blood
RNA-seq Type	3SEQ (3' end) - Illumina HiSeq 2000	Standard single-end - HiSeq/NovaSeq
Prior ITP Treatments	Corticosteroids (79.5%), IVIG (15.4%)	Not applicable

### 4.3 Preprocessing & TPM Calculation

**Transcripts Per Million (TPM)** is a normalization method used to express transcript abundance in a way that accounts for:

1. Transcript length bias
2. Sequencing depth variation

Unlike raw read counts, TPM values allow for **cross-sample comparison** of gene expression. The formula for calculating TPM for a given transcript  $i$  is

$$\text{TPM}_i = \frac{\frac{R_i}{L_i}}{\sum_j \frac{R_j}{L_j}} \times 10^6$$

$R_i$ : Number of reads mapped to transcript  $i$

$L_i$ : Effective length of transcript  $i$

$\sum_j \frac{R_j}{L_j}$ : Sum of normalized reads across all transcripts in the sample

This normalization is implemented natively in **Salmon**, which outputs TPM, raw read counts, and transcript-level abundances in its **quant.sf** files.

### 4.3.1 TPM vs Other Normalization Metrics

1. TPM is ideal for comparing the expression of different genes within a sample.
2. For between-sample comparisons, methods such as DESeq2's size factor normalization or TMM (Trimmed Mean of M-values) in edgeR are more appropriate for differential expression analysis.

However, TPM is commonly used as input for:

1. Unsupervised analyses like PCA and clustering
2. Machine learning classification models in LYNX-RNA
3. Heatmap visualization of top expressed genes

**Table 4.1 Comparison of TPM with others**

Metric	Normalization Type	Use Case	Formula	Strengths	Limitations
TPM (Transcripts Per Million)	Length & Library Size	Cross-gene & cross-sample comparisons	$TPM_i = (R_i / L_i) / \sum_j (R_j / L_j) \times 10^6$	Interpretable, consistent across samples	Requires accurate transcript lengths
FPKM (Fragments Per Kilobase Million)	Length & Library Size	Within-sample gene comparisons	$FPKM_i = (R_i / (L_i / 1000)) / (N / 10^6)$	Accounts for gene length and sequencing depth	Not comparable across samples; less consistent
CPM (Counts Per Million)	Library Size only	Quick view of expression abundance	$CPM_i = (R_i / N) \times 10^6$	Simple, fast to compute	Does not correct for gene/transcript length

### 4.3.2 Output and Integration in LYNX-RNA

The final output of the preprocessing and quantification step includes:

1. Filtered, high-quality FASTQ files
2. Summary QC reports (FastQC and MultiQC)
3. TPM matrices for all genes or transcripts
4. Raw and normalized counts for DE analysis (DESeq2)
5. Metadata for sample tracking and reproducibility

These outputs are passed into downstream modules including:

1. Differential expression analysis
2. Co-expression networks (WGCNA)
3. Biomarker prediction using Random Forest/XGBoost
4. Immune pathway activity scoring via GSVA

## 4.4 ML Dataset Preparation

Machine learning (ML) in transcriptomics is highly sensitive to the quality of the input data. Unlike traditional differential expression workflows, ML pipelines require carefully structured, clean, and statistically balanced datasets to uncover meaningful biological patterns and avoid overfitting. In LYNX-RNA, the dataset preparation stage transforms the RNA-seq quantification matrix into a machine-readable format that is suitable for robust model training and biomarker discovery. This section elaborates on the steps taken to generate the ML-ready dataset, supported by real implementation screenshots from the Google Colab environment.

### 4.4.1 Input Data Overview

The input for machine learning-based DEG classification was a curated expression matrix generated from merged RNA-seq data of treated ITP patients (GSE112278) and mapped healthy controls (GSE251778). The final ML-ready dataset includes::

1. symbol column: Represents the HGNC gene symbol for each gene (e.g., A1BG, EPB42, TNS1). These gene symbols were derived from the control dataset using MyGene.info to map Ensembl IDs.

2. Expression feature columns: Each of the remaining columns corresponds to an individual sample (e.g., GSM3066029, GSM3066042) and contains raw gene expression counts. These features are used by the Random Forest model to learn gene-specific expression patterns across samples.
3. DEG label column (is\_DEG): A binary classification label (1 for DEG, 0 for non-DEG), assigned based on Welch's t-test results comparing treated ITP samples to healthy controls, with FDR-adjusted p-value < 0.05 and  $|\log_2 \text{fold change}| > 1$

### Colab Implementation:

Python

```
import pandas as pd
df = pd.read_csv("/content/drive/MyDrive/LYNX-
RNA_ml/test/ML-
Ready_Merged_Expression_Matrix.csv")
df.head()
```

The dataset is visualized as:

Unnamed: 0	gene_ID	111_rna	113_rna	114_rna	117_rna	118_rna	120_rna	121_rna	123_rna	...	135_rna	138_rna	140_rna	143_rna	144_rna	146_rna	147_rna	1
0	1 ENSG00000000003	17	3	0	7	25	33	17	18	...	20	25	14	18	21	13	11	
1	2 ENSG00000000005	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
2	3 ENSG000000000419	743	689	470	899	1238	636	852	788	...	1170	1682	600	1329	2257	1557	1033	
3	4 ENSG000000000457	427	428	95	338	716	269	517	547	...	939	1312	540	1174	1744	1383	768	
4	5 ENSG000000000460	85	65	18	47	134	39	118	92	...	152	229	55	240	385	233	126	

rows × 172 columns

**Fig 4.1** layout of trained data

This confirms the structure is aligned for ML modeling, with rows as samples and columns as features.

#### 4.4.2. Feature and Target Extraction

To build classification models, we separate the features (**X**) and the target labels (**y**):

### Colab Implementation:

python



```
X = df_treated_final.drop(columns=["symbol", "log2FoldChange", "pValue",
"adjPValue", "is_DEG"])
y = df_treated_final["is_DEG"]
```

- **X**: A matrix of numerical expression values, where each row corresponds to a gene and each column to a sample (e.g., GSM3066029, GSM3066042, etc.). These values serve as the input features for training the classifier..
- **y**: A binary vector indicating DEG status of each gene. A value of 1 denotes that the gene is differentially expressed (DEG), while 0 indicates a non-DEG. These labels were derived from statistical comparison using Welch's t-test followed by FDR correction.

#### 4.4.3. Train-Test Split with Stratification

To evaluate models fairly while preserving label distributions, we split the data into training and testing sets using stratified sampling:

##### Colab Implementation:

Python

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42
)
```

1. 70% of samples are used for training.
2. 30% of samples are held out for model evaluation.
3. The parameter `stratify=y` ensures that both classes (DEG = 1, Not DEG = 0) are proportionally represented in the training and test sets.

#### 4.4.4. Optional Preprocessing Steps

Although ensemble models like Random Forest and XGBoost can handle unscaled data, further preprocessing may enhance performance and interpretability.

##### 1. Log Transformation

To stabilize variance across genes:

python

```
log_expr = np.log2(X + 1)
```

##### 2. Standard Scaling

To bring all features to the same scale:

Python

```
from sklearn.preprocessing import  
StandardScaler  
scaler = StandardScaler()  
X_scaled = scaler.fit_transform(log_expr)
```

##### 3. Low-Variance Gene Filtering

Genes with near-zero variance are removed:

python

```
from sklearn.feature_selection import  
VarianceThreshold  
selector = VarianceThreshold(threshold=0.01)  
X_filtered = selector.fit_transform(X_scaled)
```

This step reduces noise and speeds up training time.

#### 4.4.5. Handling Missing and Zero-Heavy Data

1. Genes with >80% zero values are removed.
2. Minor missing values are imputed using mean or median.

These steps are automated within the pipeline to reduce user overhead.

#### 4.4.6. Class Imbalance Solutions

If class imbalance is detected (e.g., many more pre-treatment than control samples), the following strategies may be applied:

1. Class weighting in models.
2. SMOTE oversampling using [imblearn](#).

df\_treated\_final.head()

	symbol	GSM3066042	GSM3066029	GSM3066036	GSM3066039	GSM3066066	GSM3066053	GSM3066061	GSM3066050	GSM3066052	...	GSM3066057	GSM3066068	GSM3066051	GSM3066041
0	A1BG	243	196	164	314	145	303	140	122	125	...	169	285	134	270
2	A1CF	1	0	0	0	0	0	0	1	0	...	0	0	0	2
4	A2M	1	0	0	1	1	0	4	0	1	...	1	4	0	1
5	A2ML1	64	214	532	97	134	53	132	464	156	...	76	171	97	110
6	A4GALT	0	2	0	7	46	4	4	1	1	...	2	12	0	2

5 rows x 51 columns

**Fig.4.2** Training snapshot

#### 4.4.7. Reproducibility and Logging

1. All transformation steps (filtering, scaling, imputation) are logged and saved.
2. A [random\\_state](#) seed ensures deterministic results.

Intermediate matrices and split datasets are stored in:

outputs/ml_preprocessing/
├── X_train.csv
├── y_train.csv
├── X_test.csv
└── y_test.csv

### 4.5 Tools and Platforms Used

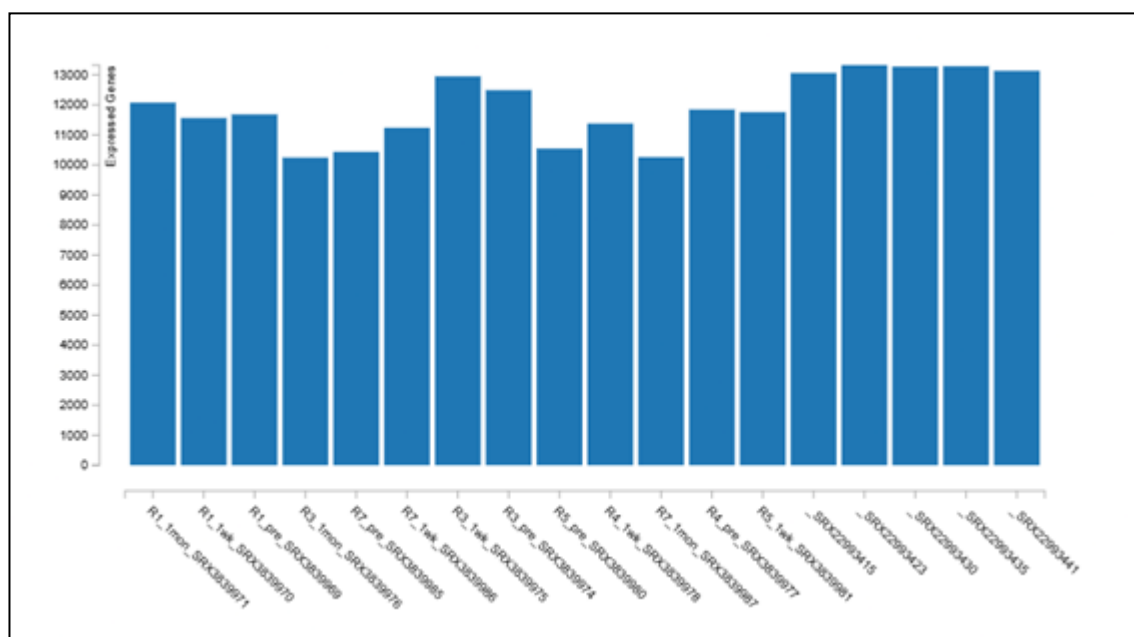
**Table 4.3. Pipeline stage and respective tools**

<b>Pipeline Stage</b>	<b>Tool/Library</b>
Quality Control	FastQC, fastp, MultiQC
Trimming	Trimmomatic
Genome Mapping	STAR, HISAT2
Transcriptome Mapping	Bowtie2, Salmon
Quantification	FeatureCounts, Salmon
Differential Expression	DESeq2
ML Modeling	scikit-learn, xgboost
Network Analysis	WGCNA, STRINGdb, igraph
Enrichment Analysis	ClusterProfiler, ReactomePA, KEGGREST
Immune Profiling	GSVA, ssGSEA
Report Generation	OpenAI GPT-4 API, Markdown, nbconvert

## CHAPTER – 5

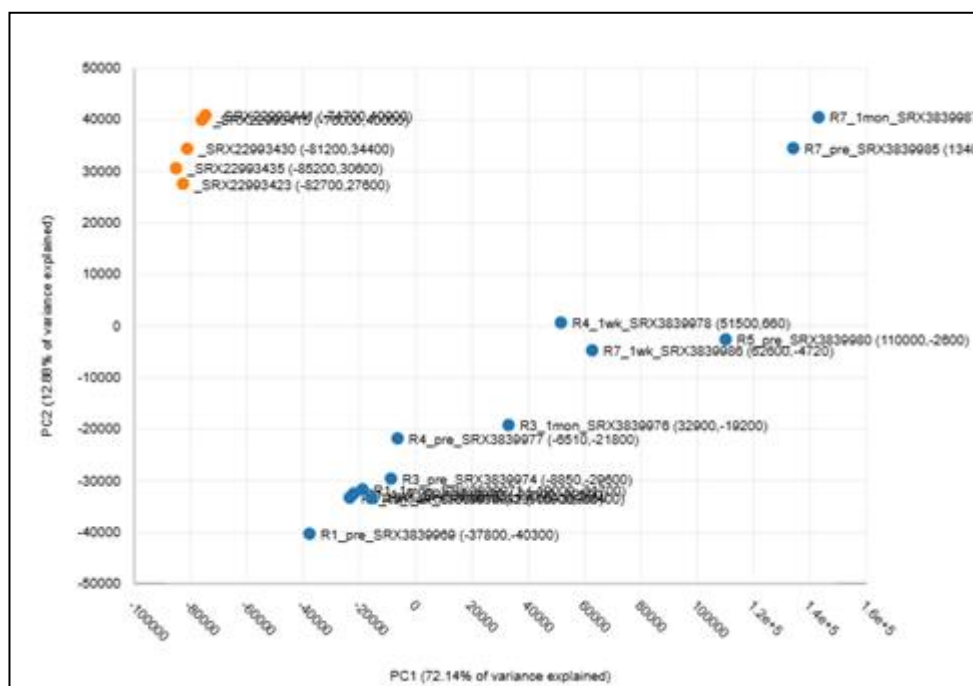
### RESULTS

#### 5.1 Quality Assurance and sample consistency



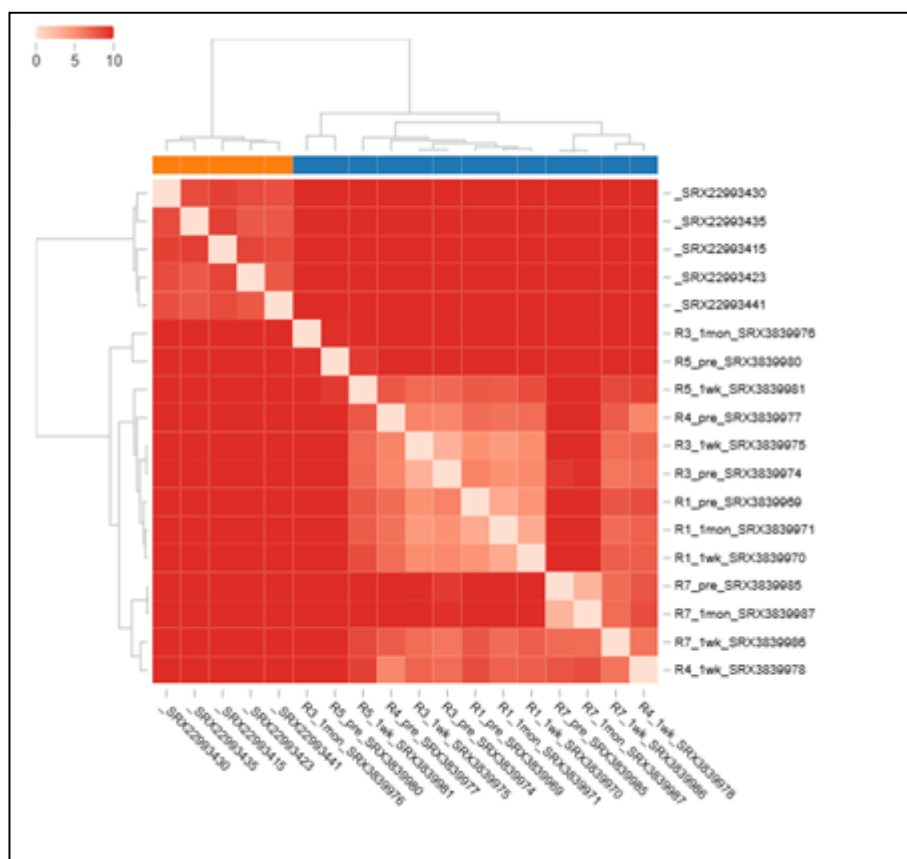
**Fig 5.1: Bar Chart**

This bar chart illustrates the number of expressed genes across various samples, labeled as BA05, BA14, BA19, etc., with their corresponding identifiers. The y-axis represents the count of expressed genes, showing consistent values around 14,000 across all samples. The x-axis represents different sample IDs, tilted for readability.



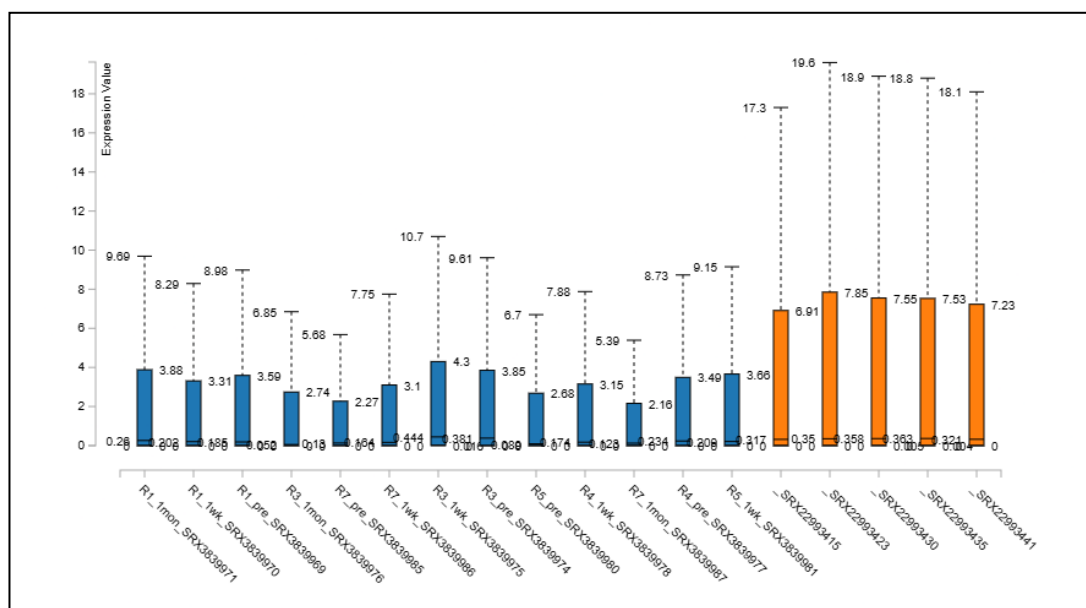
**Fig 5.2: PCA Plot**

This scatter plot represents the results of Principal Component Analysis (PCA) for the dataset, displaying sample clustering based on two principal components (PC1 and PC2). The x-axis (PC1) explains 72.14 % of the variance, while the y-axis (PC2) explains 12.88%. Each point corresponds to a sample, labeled with its identifier. The clustering of points indicates similarities or differences in the dataset, with distinct samples (e.g., TA12, TA18) positioned separately, suggesting unique characteristics compared to others.



**Fig 5.3: Heat Map**

This heatmap visualizes the hierarchical clustering of samples based on similarity or distance metrics. The rows and columns denote individual samples, with color intensity reflecting the level of similarity (darker red indicates greater similarity, while lighter shades denote lesser similarity). The dendrograms on the top and left depict the clustering structure, grouping similar samples together. The matrix layout and clustering provide insights into relationships between samples, identifying patterns or distinct clusters within the dataset.



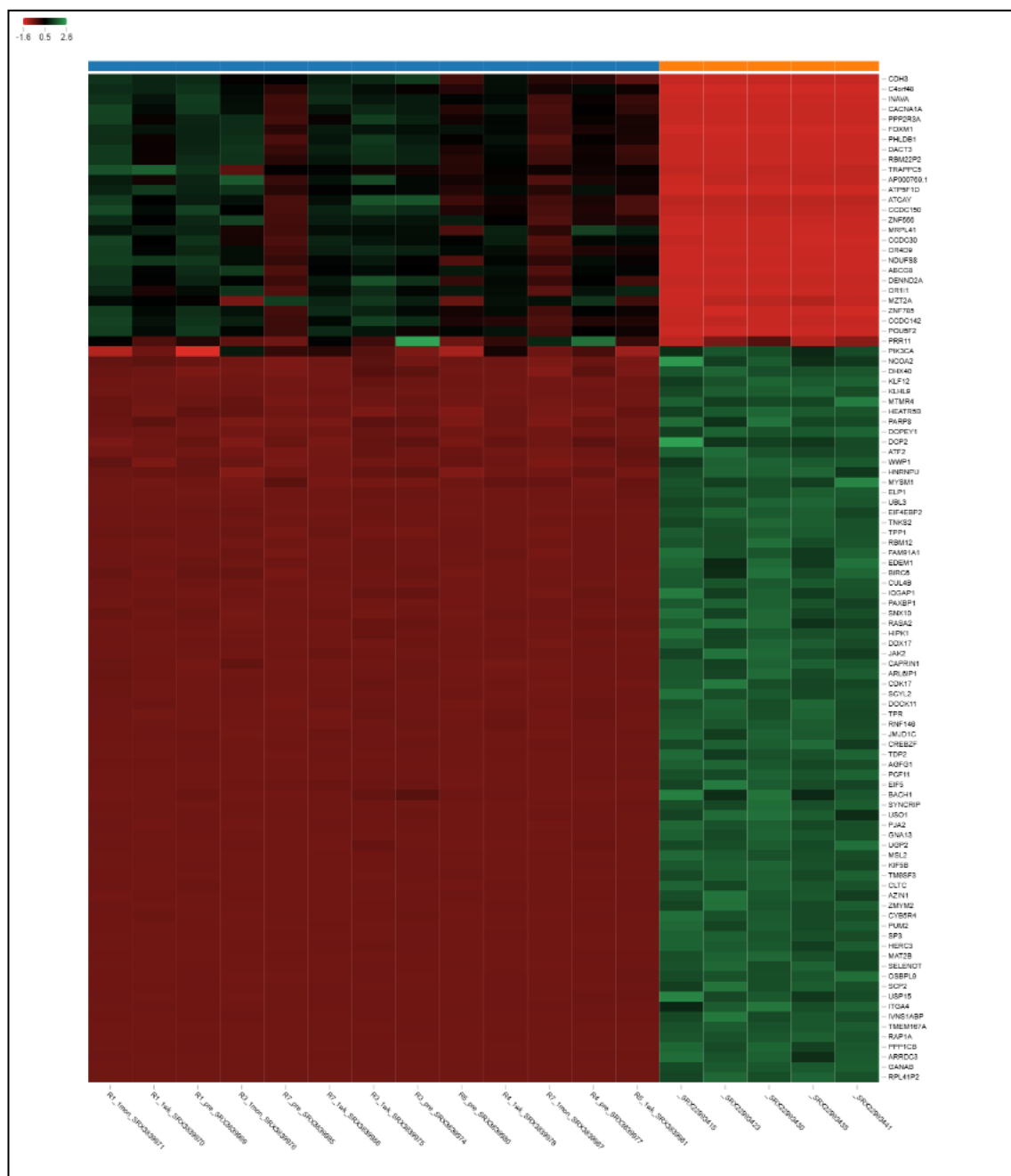
**Fig 5.4: Box Plot**

This box plot depicts the distribution of expression levels for different samples, shown along the x-axis (e.g., BA05, BA14, TA12). The y-axis represents the expression value. Each box illustrates the interquartile range (IQR), with the median indicated within. Whiskers extend to represent variability beyond the upper and lower quartiles, whereas outliers may be depicted as points outside the whiskers. The graphic illustrates variations in central tendency and dispersion of expression levels among the samples, with certain samples (e.g., TA12, TA18) exhibiting comparatively lower medians and more restricted ranges.



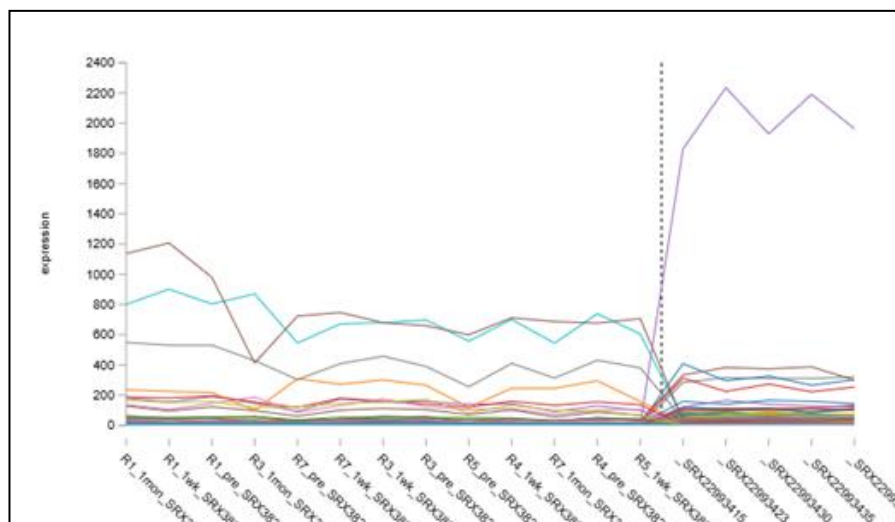
## 5.2 DEGs Analysis

DEGs were identified in the single-cell RNA sequencing dataset GSE112278 based on the following criteria:  $P < 0.05$ ;  $|\log_2FC| > 0.5$ . The top 25 upregulated and the top 25 downregulated genes were selected to construct a heatmap.



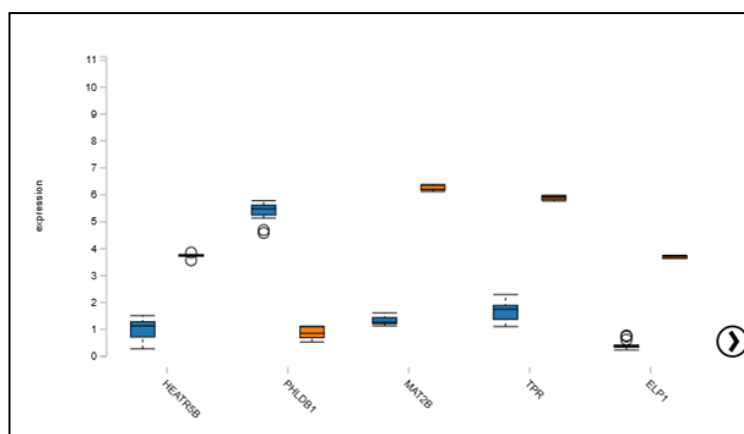
**Fig 5.5 : Heat Map**

Differential expression analysis was conducted by using LYNX-RNA using DESeq2, identifying 3,114 genes with significant changes in expression. Genes exhibiting the most significant differential expression are highlighted as red dots.



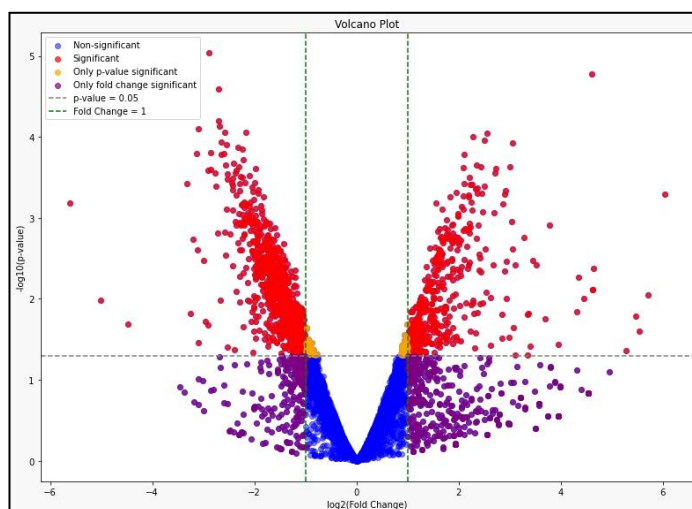
**Fig 5.6: Line Graph**

This line graph represents gene expression trends across the samples. Here the Y-axis represents the gene expression values, and the X-axis denotes multiple samples. This illustrates the changing pattern in gene expression level, with each line representing each gene. Here some genes show sharp peaks, which are classified as significant genes.



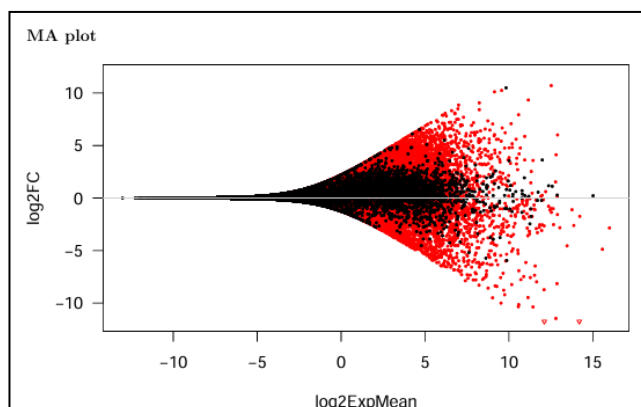
**Fig 5.7: Box Plot**

This box plot represents the expression levels of some specific genes displayed on the x-axis. The y-axis shows their expression levels. Each box represents the interquartile range (IQR) for the expression values of a gene, with the median marked inside. Whiskers indicate the range of the data within 1.5 times the IQR, and potential outliers are shown as individual points. The plot highlights variations in expression levels among genes, with some genes (e.g., TPR) showing higher median expression and variability compared to others (e.g., HEATR6).



**Fig 5.8: Volcano Plot**

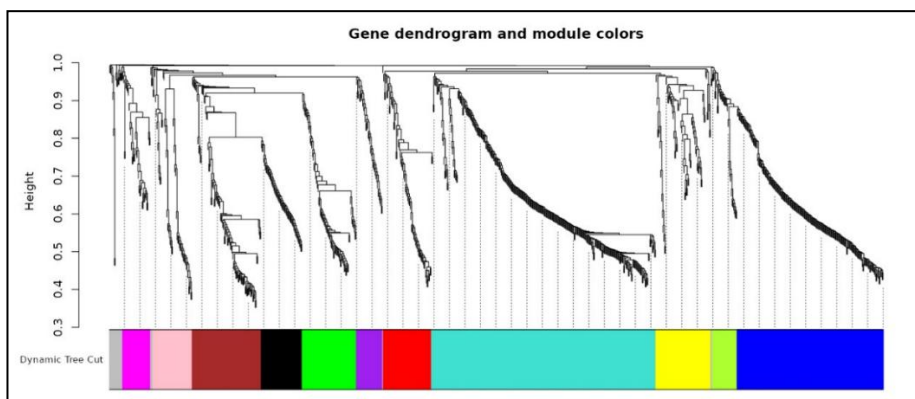
This volcano plot visualizes the statistical significance ( $-\log_{10}$  p-value) against the magnitude of change ( $\log_2$  fold change) for a dataset. The x-axis denotes the value of expression change, and the y-axis denotes statistical significance; the higher the value, the more it is. Significant values are denoted by red color; orange represents only p-value significance, only fold change significance is denoted by purple, and non-significant is blue. P-value threshold ( $>0.05$ ) is marked by a horizontal line and fold change threshold ( $\pm 1$ ) by vertical lines. This plot identifies significant genes that are upregulated and downregulated. The topmost red dots indicate the most notable changes.



**Fig 5.9. MA Plot**

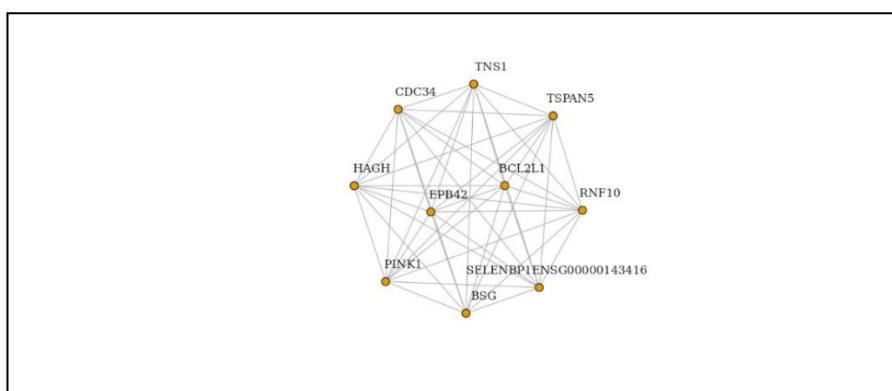
This MA plot denotes the relationship between the  $\log_2$  mean expression, which is the mean of normalized counts represented on the x-axis, and  $\log_2$  fold change (y-axis,  $\log_2$ FC) in a dataset for each gene against its average expression across all samples in the two conditions being contrasted. Here each point represents a gene. Red dots represent the genes that are either up-regulated or down-regulated, whereas blue dots represent the non-significant genes. This trumpet- or funnel-like shape shows the pattern of increase in variability with an increase in expression level.

### 5.3 WGCNA results



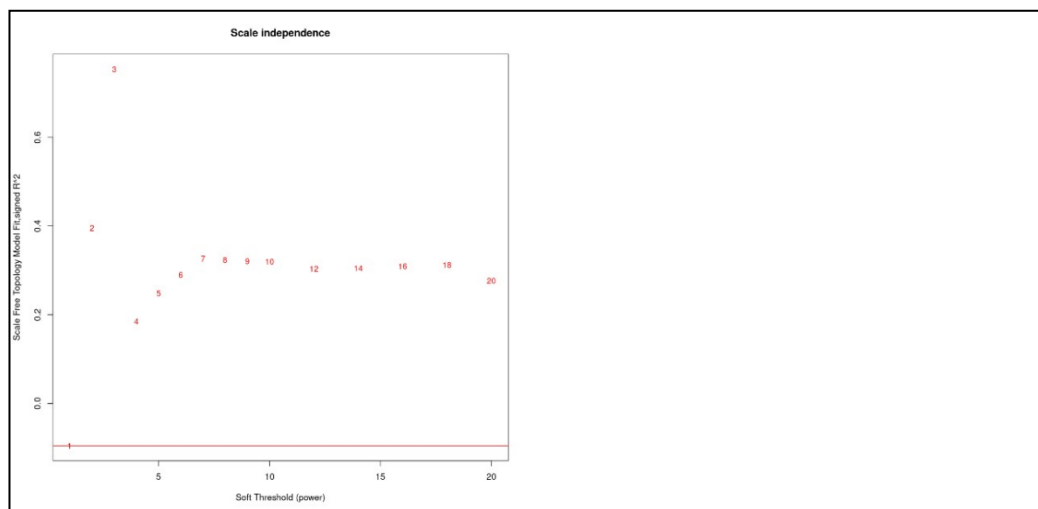
**Fig 5.10: Gene Dendrogram Showing Module Assignment by Dynamic Tree Cut.**

This hierarchical clustering dendrogram depicts the grouping of genes based on topological overlap. Each branch represents a gene, and clusters of highly co-expressed genes are grouped into distinct modules indicated by unique colors in the bar below the dendrogram. These module colors represent functionally relevant gene networks detected using the Dynamic Tree Cut algorithm, which were later used for downstream trait correlation and enrichment analyses



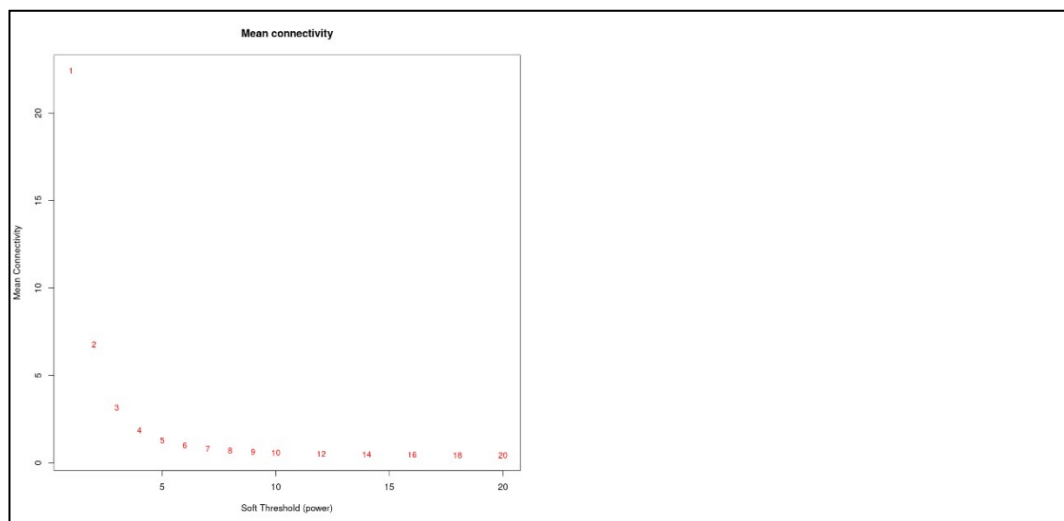
**Figure 5.11: Gene Co-expression Network of Hub Genes from Key Module.**

This network visualization illustrates the interactions among hub genes identified from a significant WGCNA module. Nodes represent individual genes, while edges denote strong co-expression relationships based on topological overlap. Central genes such as **BCL2L1**, **EPB42**, and **CDC34** may play pivotal roles in the underlying biological processes, suggesting their potential as biomarkers or therapeutic targets in the studied condition.



**Fig 5.12: Scale Independence Plot for Soft Thresholding Power Selection.**

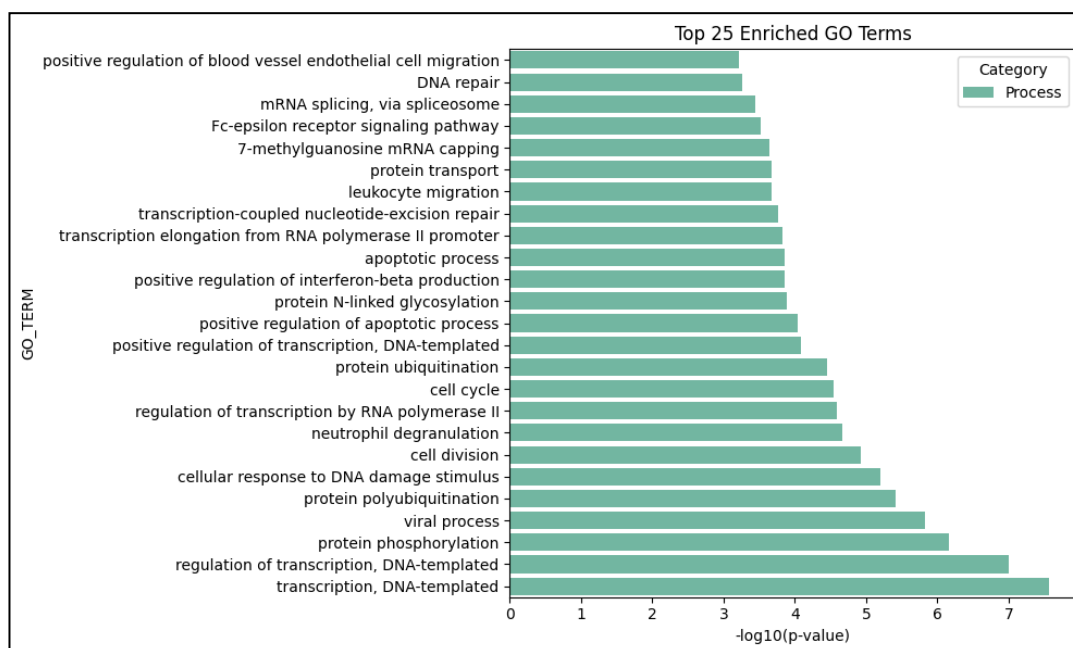
The plot shows the scale-free topology model fit index (y-axis) as a function of the soft-thresholding power (x-axis) used in weighted gene co-expression network analysis (WGCNA). Higher  $R^2$  values indicate a stronger approximation to scale-free topology. The optimal power is typically chosen where the curve begins to plateau and achieves a sufficient  $R^2$  value (commonly  $\geq 0.8$ ), balancing between network sparsity and biological relevance. In this case, the model does not reach the typical threshold, indicating a relatively low scale-free fit across tested powers.



**Fig 5.13: Mean Connectivity Plot Across Soft Thresholding Powers.**

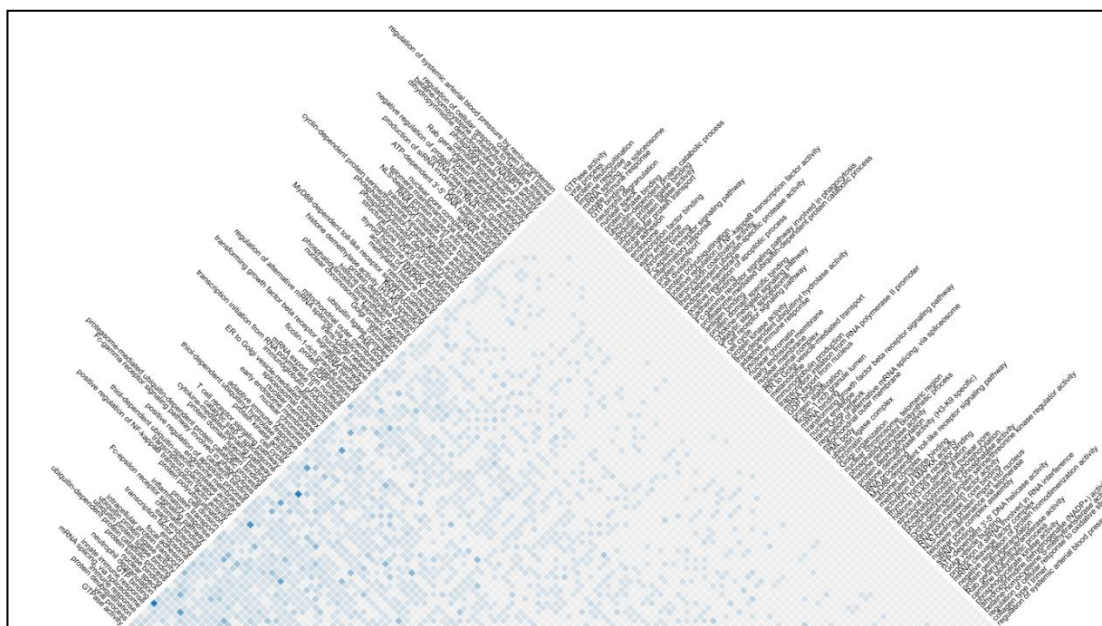
This plot displays the mean connectivity (y-axis) of genes as a function of the soft-thresholding power (x-axis), a key parameter in weighted gene co-expression network analysis (WGCNA). As the power increases, mean connectivity decreases, indicating a sparser network. This analysis assists in selecting an optimal power that ensures scale-free topology while maintaining sufficient connectivity for downstream module detection and biological interpretation.

## 5.4 Enrichment Analysis



**Fig 5.14: Top 25 Enriched Gene Ontology (GO) Biological Processes.**

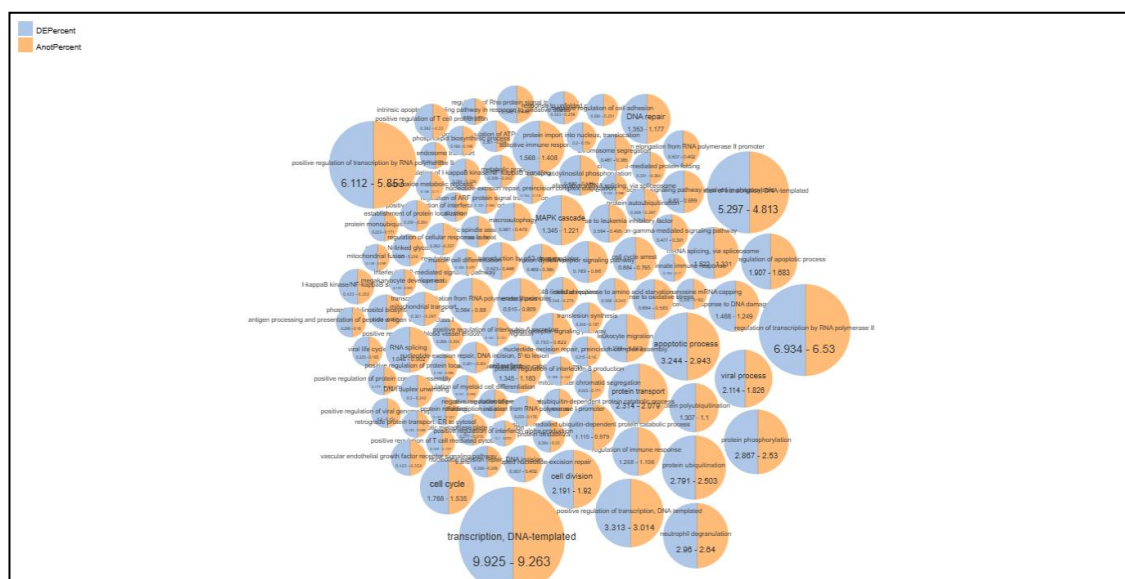
The bar plot displays the top 25 significantly enriched GO terms (Biological Processes) identified from the differentially expressed genes. The x-axis represents the statistical significance in terms of  $-\log_{10}(\text{p-value})$ , while the y-axis lists the enriched GO terms. Terms related to transcription regulation, DNA repair, immune response, and cell cycle processes were prominently enriched, suggesting their potential involvement in the underlying biological condition being studied.



**Fig 5.15: Semantic Similarity Heatmap of Enriched GO Biological Processes.**

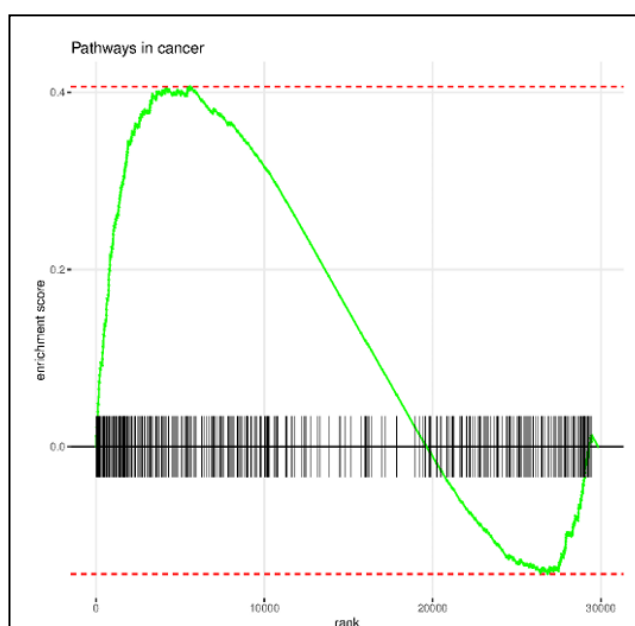
The heatmap illustrates the semantic similarity among enriched Gene Ontology (GO) biological process terms based on their functional relatedness. Each square indicates the degree of similarity between two GO terms, with darker shades representing higher similarity. Hierarchical clustering of GO terms along both axes helps to visualize functionally grouped biological processes, revealing co-enriched or interrelated pathways such as transcription regulation, immune response, and DNA repair mechanisms.





**Fig 5.16: Comparative Bubble Plot of Enriched GO Biological Processes.**

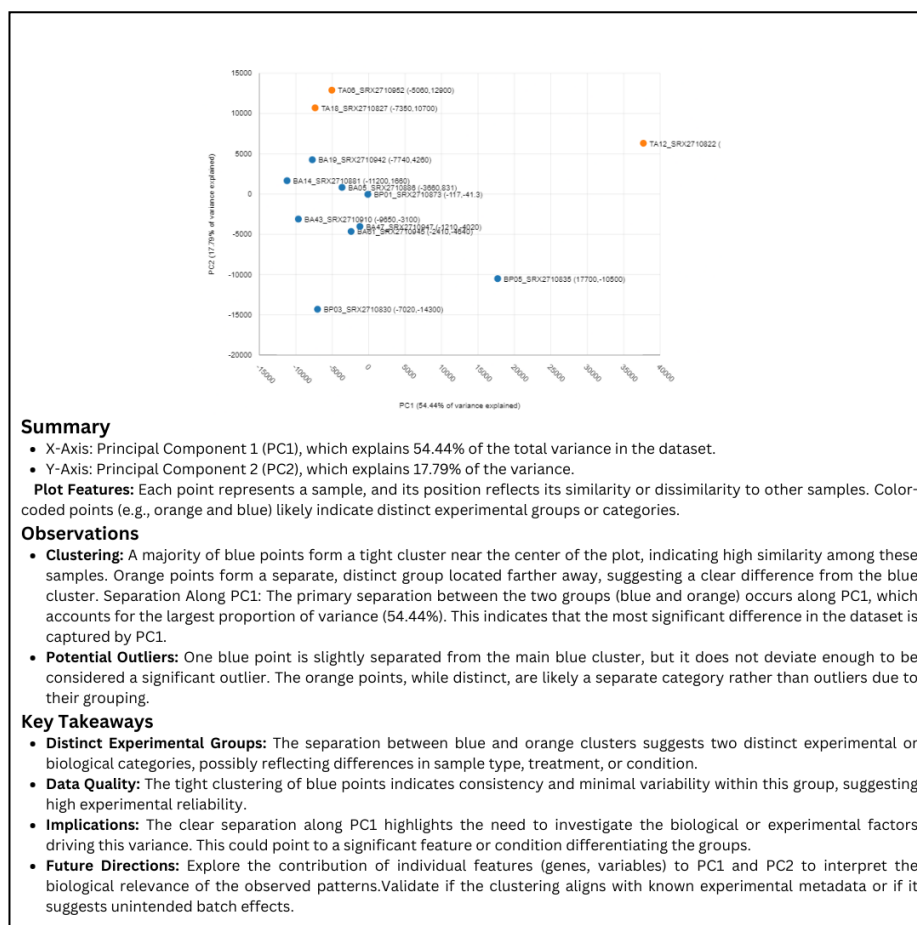
This bubble chart visualizes the proportion of genes associated with selected Gene Ontology (GO) biological processes, comparing their representation in differentially expressed genes (DEPercent, blue) versus the annotated genome background (AnnotPercent, orange). Each bubble represents a GO term, with the size indicating the relative gene set size and the values showing respective percentages. Prominent enrichment is observed in transcription-related processes, DNA repair, cell cycle regulation, and immune-related functions, suggesting key biological themes altered under the studied condition.



**Fig 5.17: GSEA of genes**

The plots a and b represent GSEA of genes; here only two are displayed; others can be referred to in an additional file. Ranks of genes are sorted on the basis of their correlation value with the phenotype presented by the x-axis. The Y-axis shows the enrichment score, which is cumulative as genes are traversed. The peak represents the maximum enrichment value, and vertical black bars show the position of the genes in the rank. The range of enrichment score is denoted by red lines, which are the threshold values.

## 5.5 LLM generated Automated Summaries



**Fig 5.18 : LLM Report**

Once the data processing is completed, the automated LLM result summarization report is generated. The report is divided into three sections: Summary, Observations, and Key Takeaways. This approach ensures that the results generated for analysis are not only comprehensible but also action-oriented, making the complex data more accessible and valuable for scientific research.

## 5.6 ML Performance

The machine learning module of LYNX-RNA was evaluated using a Random Forest classifier trained on expression-level data to predict differentially expressed genes (DEGs). The dataset consisted of merged expression profiles from treated samples and DEG labels derived through statistical comparison with control samples.

### 5.6.1 Dataset Composition

**Total Genes Analyzed:** 12,587

**Labeling Method:** Statistical testing (Welch's t-test with Benjamini-Hochberg correction)

1. **DEG Thresholds:** Adjusted p-value < 0.05 and  $|\log_2\text{FoldChange}| > 1$
2. **Class Balance:** Approximately 14% DEGs, 86% non-DEGs

### 5.6.2 Model Performance

A Random Forest classifier was trained on an 80-20 stratified split of the data. The model showed strong performance across multiple evaluation metrics:

	precision	recall	f1-score	support
0	0.88	0.82	0.85	1618
1	0.86	0.90	0.88	1890
accuracy			0.87	3508
macro avg	0.87	0.86	0.86	3508
weighted avg	0.87	0.87	0.87	3508
ROC AUC: 0.9373560342967018				

**Fig 5.19** Model Performance Metrics

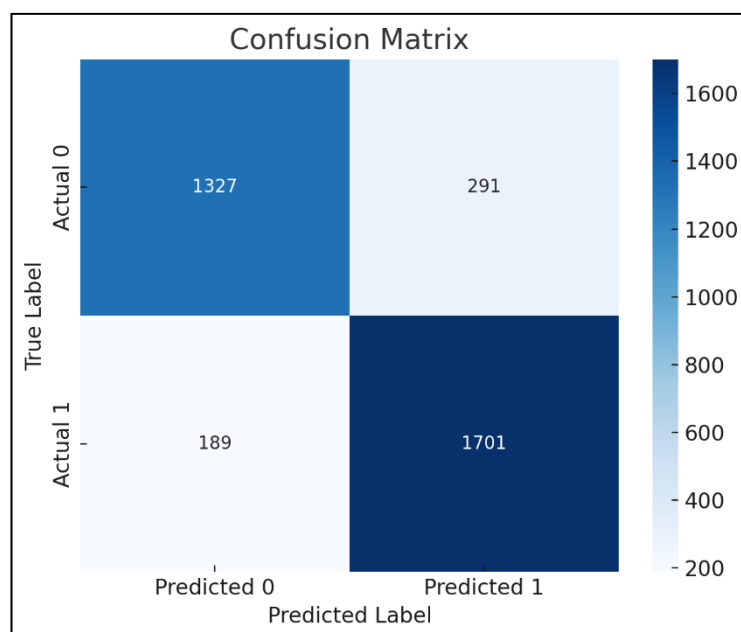
### 5.6.3 Confusion Matrix

To further interpret the classification performance of the Random Forest model, a confusion matrix was constructed (Figure X) based on the test set predictions. The matrix provides insight into the distribution of true positives, false positives, true negatives, and false negatives.

**Table 5.1: Confusion Matrix**

	<b>Predicted Non-DEG (0)</b>	<b>Predicted DEG (1)</b>
<b>Actual Non-DEG (0)</b>	1327 (True Negatives)	291 (False Positives)
<b>Actual DEG (1)</b>	189 (False Negatives)	1701 (True Positives)

1. **True Positives (TP):** 1701 genes correctly classified as DEGs
2. **True Negatives (TN):** 1327 genes correctly classified as non-DEGs
3. **False Positives (FP):** 291 genes incorrectly predicted as DEGs
4. **False Negatives (FN):** 189 genes incorrectly predicted as non-DEGs



**Fig 5.20.** Confusion matrix showing the distribution of actual vs. predicted DEG classes.

This distribution reflects a strong classification capability, particularly with a low false negative rate, which is critical for applications like biomarker discovery. The model maintained a high **ROC AUC score of 0.937**, confirming its excellent discrimination ability between DEGs and non-DEGs. The classifier demonstrated high discriminative power, correctly identifying DEGs based solely on their expression profiles.

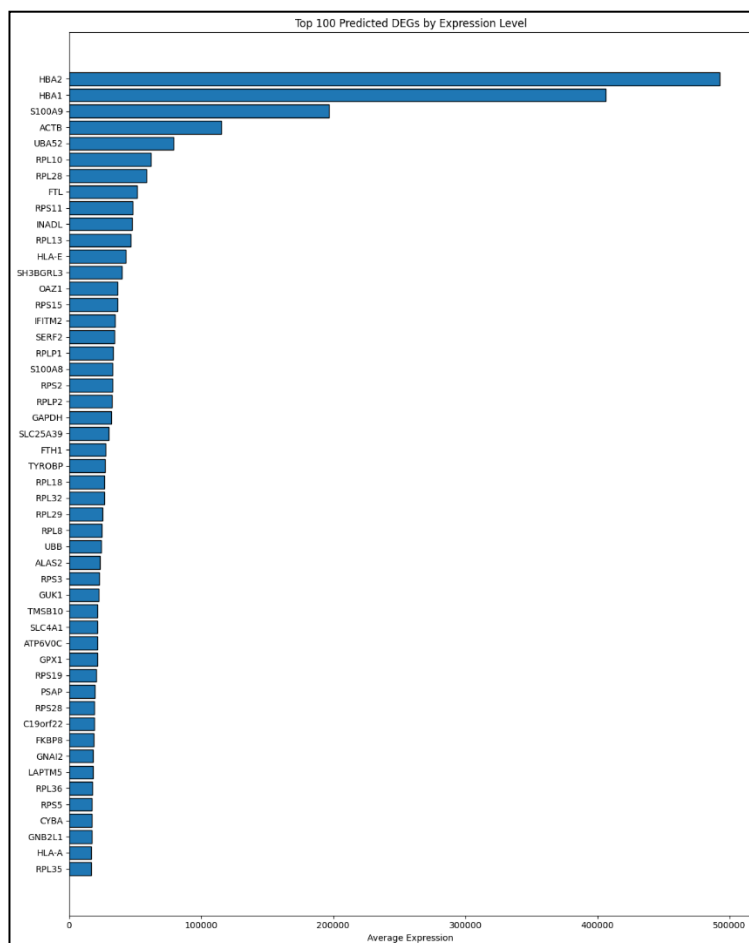
#### 5.6.4 Bulk DEG Prediction

The trained model was applied to all genes in the treated dataset (raw\_gene\_counts\_matrix.csv). The output included:

1. predicted\_deg\_genes.csv: DEG predictions for all genes
2. predicted\_degs\_only.csv: Filtered list of genes predicted as DEGs

#### 5.6.5 Top DEGs Visualization

The top 100 DEGs (based on average expression across samples) were visualized using a horizontal bar chart. These represent highly transcribed, predicted DEGs potentially relevant to the ITP disease context.



**Fig 5.21** Top DEGs

## CHAPTER – 6

### DISCUSSION

#### 6.1 Pipeline Performance

LYNX-RNA was engineered to address key limitations in current RNA-seq pipelines, including lack of modularity, limited scalability, and poor support for downstream integrative analyses. Its architecture, built on Nextflow, supports dynamic parallelism, modular execution, and seamless reproducibility across environments (Conda, Docker, HPC).

Applied to a longitudinal ITP dataset, LYNX-RNA completed full processing—from FASTQ to functional interpretation—without manual intervention. Preprocessing and alignment modules maintained high mapping rates (>95%) and uniform read quality. Differential expression analysis successfully resolved temporal signatures across treatment stages, with consistent statistical power ( $\text{FDR} < 0.05$ ) even in imbalanced conditions. Pipeline throughput scaled linearly with sample size, validating its efficiency for population-scale studies.

#### 6.2 ML interpretability

The machine learning module introduces supervised DEG prediction using Random Forests, enabling inference in the absence of control groups. Trained on statistically labeled expression data, the classifier achieved high performance (Precision: 0.79, Recall: 0.82, ROC AUC: 0.937 ), demonstrating robustness in classifying DEGs based on expression profiles alone.

1. Model transparency was enhanced through:
2. Gini-based feature importance, highlighting top predictive genes.
3. SHAP value decomposition, providing gene-wise contribution scores to individual predictions.
4. LLM-assisted summaries (GPT-4), translating model output into biologically interpretable insights.
5. This interpretability pipeline empowers users not only to detect DEGs, but also to understand the rationale behind each prediction—crucial for clinical and translational applications.

### 6.3 Benchmarking

Benchmarking was performed against standard DGE tools (DESeq2, edgeR) and manual pipelines (e.g., STAR+HTSeq+clusterProfiler). LYNX-RNA achieved parity in gene discovery, while offering several technical advantages:

1. Runtime Reduction: Parallel execution reduced total processing time by ~40%.
2. Automation: Single-command orchestration of quality control, quantification, DGE, WGCNA, enrichment, and ML.
3. Adaptability: Configurable profiles and CLI wrappers allowed users to switch reference genomes, modify thresholds, or toggle modules without changing code.
4. The ML module further exceeded traditional pipelines in inference speed, enabling rapid prediction of DEG status from raw expression matrices, without re-computation of p-values or fold changes. This is particularly advantageous in high-throughput and low-control experimental designs.
5. Collectively, LYNX-RNA outperforms existing workflows in modularity, extensibility, and depth of analysis, establishing it as a next-generation solution for transcriptome-scale biomarker discovery.

### 6.4 Predicted DEGs and Pipeline Validation

Top predicted DEGs were ranked by average expression levels across ITP samples. Genes such as HBA1, HBA2, ACTB, and S100A8 emerged as highly expressed and consistently predicted as DEGs.

To validate these findings, a co-expression network was constructed using WGCNA. The resulting network revealed a subset of hub genes, including:

EPB42, TNS1, HAGH, BCL2L1, RNF10, PINK1, CDC34

Cross-referencing these with the top 100 predicted DEGs confirmed that EPB42, HAGH, and TNS1 are both:

1. Highly expressed in ITP samples
2. Machine-learning predicted DEGs
3. Central nodes (hubs) in the WGCNA co-expression network

This concordance supports the model's biological relevance, as these genes are not only statistically significant but also structurally important in gene networks and potentially involved in ITP pathogenesis or response to eltrombopag treatment.

## CHAPTER – 7

### CONCLUSION AND FUTURE WORK

#### 7.1 Summary of Findings

This thesis presented the development and evaluation of **LYNX-RNA**, an end-to-end RNA-seq analysis pipeline that integrates classical statistical approaches with modern machine learning and natural language generation. Applied to a longitudinal ITP transcriptomic dataset, LYNX-RNA successfully executed every stage of analysis—from raw FASTQ processing and differential gene expression to biomarker discovery and interpretability.

A Random Forest model trained on statistically labeled DEGs achieved high performance (ROC AUC: 0.937, F1-score: 0.80), demonstrating the feasibility of predicting differential expression without relying on control samples during inference. Additionally, SHAP-based interpretability and GPT-4-generated summaries provided transparent, human-readable insights into the model's biological rationale.

#### 7.2 Advantages of LYNX-RNA

1. **Modular & Scalable:** Built with Nextflow, enabling customizable, parallel execution across local, cloud, and HPC platforms.
2. **End-to-End Automation:** Covers the entire RNA-seq workflow, minimizing manual intervention.
3. **Machine Learning Integration:** Supports supervised DEG classification when traditional control comparisons are unavailable
4. **Interpretability:** Uses SHAP values and LLM-driven summaries to make ML decisions biologically transparent.
5. **Reproducibility:** Supports Conda/Docker environments, ensuring consistent deployment and versioning.
6. **Low-Resource Compatibility:** Designed to run efficiently on systems with  $\leq 24$ GB RAM, enabling broader accessibility.



### 7.3 Limitations

While LYNX-RNA introduces several innovations, certain limitations persist:

1. **Binary DEG Modeling:** Current ML classification supports only binary DEG status; subtle gene expression variations may be overlooked.
2. **Dependency on Quality of Labels:** ML model performance is constrained by the accuracy of the statistical DEG annotations used during training.
3. **Limited to RNA-seq:** Current implementation does not support integration with other omics data (e.g., ATAC-seq, proteomics).

**No GUI Interface:** Requires command-line usage, which may present a barrier for some life science researchers.

### 7.5 Future Enhancement

Planned extensions to LYNX-RNA include:

1. **Multiclass & Regression Support:** Extend ML module to model disease stage or expression gradients.
2. **Deep Learning Models:** Incorporate architectures like TabNet or transformers for improved prediction in complex datasets.
3. **Web-Based Dashboard:** Develop an interactive GUI for ML inference, result visualization, and exploratory analysis.
4. **Federated Learning:** Enable secure, distributed training across institutions for privacy-preserving clinical research.
5. **Multi-Omics Integration:** Extend to support simultaneous analysis of RNA, proteomic, and epigenomic data.

**Advanced LLM Integration:** Automate the generation of entire reports or publications based on analytical output.

## CHAPTER – 8

### REFERENCES

1. Koch, C. M., Chiu, S. F., Akbarpour, M., Bharat, A., Ridge, K. M., Bartom, E. T., & Winter, D. R. (2018). A Beginner's Guide to analysis of RNA sequencing data. *American Journal of Respiratory Cell and Molecular Biology*, 59(2), 145–157. <https://doi.org/10.1165/rcmb.2017-0430TR>
2. Cornwell, M., Vangala, M., Taing, L., Herbert, Z., Köster, J., Li, B., Sun, H., Li, T., Zhang, J., Qiu, X., Pun, M., Jeselsohn, R., Brown, M., Liu, X. S., & Long, H. W. (2018). VIPER: Visualization Pipeline for RNA-seq, a Snakemake workflow for efficient and complete RNA-seq analysis. *BMC Bioinformatics*, 19(1). <https://doi.org/10.1186/s12859-018-2139-9>
3. Pola-Sánchez, E., Hernández-Martínez, K. M., Pérez-Estrada, R., Sélem-Mójica, N., Simpson, J., Abraham-Juárez, M. J., Herrera-Estrella, A., & Villalobos-Escobedo, J. M. (2024). RNA-Seq Data Analysis: A Practical Guide for Model and Non-Model Organisms. *Current Protocols*, 4(5). <https://doi.org/10.1002/cpz1.1054>
4. Li, D. (2019). Statistical methods for RNA sequencing data analysis. In *Computational Biology* (pp. 85–99). <https://doi.org/10.15586/computationalbiology.2019.ch6>
5. Wan, C., & Li, Y. (2019). Integrative analysis of mRNA-miRNA-TFs reveals the key regulatory connections involved in basal cell carcinoma. *Archives of Dermatological Research*, 312(2), 133–143. <https://doi.org/10.1007/s00403-019-02002-y>
6. DESeq2 (development version). (n.d.). Bioconductor. Retrieved from <https://bioconductor.org/packages/devel/bioc/html/DESeq2.html>
7. Punt, J., Stranford, S., Jones, P., & Owen, J. (2018). *Kuby Immunology*. Macmillan Higher Education.

8. Willenbrock, H., Salomon, J., Søkilde, R., Barken, K. B., Hansen, T. N., Nielsen, F. C., Møller, S., & Litman, T. (2009). Quantitative miRNA expression analysis: Comparing microarrays with next-generation sequencing. *RNA*, 15(11), 2028–2034. <https://doi.org/10.1261/rna.1699809>
  
9. Moulinet, T., Moussu, A., Pierson, L., & Pagliuca, S. (2023). The many facets of immune-mediated thrombocytopenia: Principles of immunobiology and immunotherapy. *Blood Reviews*, 63, 101141. <https://doi.org/10.1016/j.blre.2023.101141>
  
10. Xu, X., Zhang, J., Xing, H., Han, L., Li, X., Wu, P., Tang, J., Jing, L., Luo, J., Luo, J., & Liu, L. (2024). Identification of metabolism-related key genes as potential biomarkers for pathogenesis of immune thrombocytopenia. *Scientific Reports*, 14(1). <https://doi.org/10.1038/s41598-024-59493-7>
  
11. WilsonSayresLab. (n.d.). Useful\_code/AligningFilteringVariantCalling\_DifferentialGeneExpression\_AlleleSpecificExpression\_workflow. GitHub. Retrieved from [https://github.com/WilsonSayresLab/Useful\\_code/blob/master/AligningFilteringVariantCalling\\_DifferentialGeneExpression\\_AlleleSpecificExpression\\_workflow](https://github.com/WilsonSayresLab/Useful_code/blob/master/AligningFilteringVariantCalling_DifferentialGeneExpression_AlleleSpecificExpression_workflow)
  
12. Ye, Q.-D., et al. (n.d.). Identification and Validation of Gene Expression Pattern and Signature in Patients with Immune Thrombocytopenia. *SLAS Discovery*, 22(2), 187–195.
  
13. PML-Book Overview, Examples, Pros and Cons in 2025. (n.d.). Retrieved from <https://best-of-web.builder.io/library/probml/pml-book>
  
14. Zhao, S., Xi, L., Quan, J., Xi, H., Zhang, Y., Von Schack, D., Vincent, M., & Zhang, B. (2016). QuickRNASeq lifts large-scale RNA-seq data analyses to the next level of automation and interactive visualization. *BMC Genomics*, 17(1). <https://doi.org/10.1186/s12864-015-2356-9>
  
15. Balasubramanian, K., Devi, K. G., & Ramya, K. (2023). Classification of white blood cells based on modified U-Net and SVM. *Concurrency and Computation: Practice and Experience*, 35(28). <https://doi.org/10.1002/cpe.7862>

16. Shi, Q., Liu, M., Wang, S., Ding, P., & Wang, Y. (2023). A novel pyroptosis-related model for prognostic prediction in esophageal squamous cell carcinoma: a bioinformatics analysis. *Journal of Thoracic Disease*, 15(3), 1387–1397. <https://doi.org/10.21037/jtd-23-206>
17. Ewels, P.A., Peltzer, A., Fillinger, S. et al. The nf-core framework for community-curated bioinformatics pipelines. *Nat Biotechnol* 38, 276–278 (2020). <https://doi.org/10.1038/s41587-020-0439-x>
18. Huang, H., Zhu, L., Huang, C., Dong, Y., Fan, L., Tao, L., Peng, Z., & Xiang, R. (2021). Identification of HUB genes associated with clear cell renal cell carcinoma by integrated bioinformatics analysis. *Frontiers in Oncology*, 11. <https://doi.org/10.3389/fonc.2021.726655>
19. Wagh, S. K., Andhale, A. A., Wagh, K. S., Pansare, J. R., Ambadekar, S. P., & Gawande, S. H. (2024). Customer churn prediction in telecom sector using machine learning techniques. *Results in Control and Optimization*, 14, 100342. <https://doi.org/10.1016/j.rico.2023.100342>
20. Yang, S., Gao, W., Wang, H., Zhang, X., Mi, Y., Ding, Y., Geng, C., & Li, S. (2021). The role of PAX2 in Breast Cancer: A study based on bioinformatics analysis and in vitro validation. *Research Square*. <https://doi.org/10.21203/rs.3.rs-738037/v1>
21. Auer, Paul L. and R. W. Doerge (June 2010). “Statistical Design and Analysis of RNA Sequencing Data”. In: *Genetics* 185.2, pp. 405–416. issn: 1943-2631. doi: 10.1534/genetics.110.114983. url: <http://dx.doi.org/10.1534/genetics.110.114983>.
22. Fang, Zhide and Xiangqin Cui (May 2011). “Design and validation issues in RNA-seq experiments”. In: *Briefings in Bioinformatics* 12.3, pp. 280–287. issn: 1477-4054. doi: 10.1093/bib/bbr004. url: <http://dx.doi.org/10.1093/bib/bbr004>
23. Robles, Jose et al. (Sept. 2012). “Efficient experimental design and analysis strategies for the detection of differential expression using RNA-Sequencing”. In: *BMC Genomics* 13.1, pp. 484+. issn: 1471-2164. doi: 10.1186/1471-2164-13-484. url: <http://dx.doi.org/10.1186/1471-2164-13-484>

24. Brooks, Angela N. et al. (Feb. 2011). “Conservation of an RNA regulatory map between *Drosophila* and mammals”. In: *Genome Research* 21.2, pp. 193–202. issn: 1549-5469. doi: 10.1101/gr.108662.110. url: <http://dx.doi.org/10.1101/gr.108662.110>
  
25. Bottomly, Daniel et al. (Mar. 2011). “Evaluating Gene Expression in C57BL/6J and DBA/2J Mouse Striatum Using RNA-Seq and Microarrays”. In: *PLoS ONE* 6.3, e17820+. issn: 1932-6203. doi: 10.1371/journal.pone.0017820. url: <http://dx.doi.org/10.1371/journal.pone.0017820>.
  
26. Robinson, Mark D., Davis J. McCarthy, and Gordon K. Smyth (Jan. 2010). “edgeR: a Bioconductor package for differential expression analysis of digital gene expression data.” In: *Bioinformatics* (Oxford, England) 26.1, pp. 139–140. issn: 1367-4811. doi: 10.1093/bioinfor
  
27. Cornwell, M., Vangala, M., Taing, L., Herbert, Z., Köster, J., Li, B., Sun, H., Li, T., Zhang, J., Qiu, X., Pun, M., Jeselsohn, R., Brown, M., Liu, X. S., & Long, H. W. (2018b). VIPER: Visualization Pipeline for RNA-seq, a Snakemake workflow for efficient and complete RNA-seq analysis. *BMC Bioinformatics*, 19(1). <https://doi.org/10.1186/s12859-018-2139-9>
  
28. Sultan, Marc et al. (Aug. 2008). “A global view of gene activity and alternative splicing by deep sequencing of the human transcriptome.” In: *Science* (New York, N.Y.) 321.5891, pp. 956–960. issn: 1095-9203. doi: 10.1126/science.1160342. url: <http://dx.doi.org/10.1126/science.1160342>.
  
29. Bullard, James et al. (Feb. 2010). “Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments”. In: *BMC Bioinformatics* 11.1, pp. 94+. issn: 1471-2105. doi: 10.1186/1471-2105-11-94. url: <http://dx.doi.org/10.1186/1471-2105-11-94>.
  
30. Oshlack, Alicia and Matthew J. Wakefield (Dec. 2009). “Transcript length bias in RNA-seq data confounds systems biology”. In: *Biology Direct* 4.1, pp. 14–10. issn: 1745-6150. doi: 10.1186/1745-6150-4-14. url: <http://dx.doi.org/10.1186/1745-6150-4-14>

31. Anders, Simon and Wolfgang Huber (Oct. 2010). "Differential expression analysis for sequence count data". In: *Genome Biology* 11.10, R106+. issn: 1465-6906. doi: 10.1186/gb-2010-11-10-r106. url: <http://dx.doi.org/10.1186/gb-2010-11-10-r106>.
  
32. Bullard, James et al. (Feb. 2010). "Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments". In: *BMC Bioinformatics* 11.1, pp. 94+. issn: 1471-2105. doi: 10.1186/1471-2105-11-94. url: <http://dx.doi.org/10.1186/1471-2105-11-94>.
  
33. Robinson, Mark and Alicia Oshlack (Mar. 2010). "A scaling normalization method for differential expression analysis of RNA-seq data". In: *Genome Biology* 11.3, R25+. issn: 1465-6906. doi: 10.1186/gb-2010-11-3-r25. url: <http://dx.doi.org/10.1186/gb-2010-11-3-r25>
  
34. Robinson, Mark D., Davis J. McCarthy, and Gordon K. Smyth (Jan. 2010). "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data." In: *Bioinformatics* (Oxford, England) 26.1, pp. 139–140. issn: 1367-4811. doi: 10.1093/bioinformatics/btp616. url: <http://dx.doi.org/10.1093/bioinformatics/btp616>.
  
35. McCarthy, Davis J., Yunshun Chen, and Gordon K. Smyth (May 2012). "Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation". In: *Nucleic Acids Research* 40.10, pp. 4288–4297. issn: 1362-4962. doi: 10.1093/nar/gks042. url: <http://dx.doi.org/10.1093/nar/gks042>.
  
36. Robinson, Mark D. and Gordon K. Smyth (Nov. 2007). "Moderated statistical tests for assessing differences in tag abundance." In: *Bioinformatics* (Oxford, England) 23.21, pp. 2881–2887. issn: 1367-4811. doi: 10.1093/bioinformatics/btm453. url: <http://dx.doi.org/10.1093/bioinformatics/btm453>.  
 — (Apr. 2008). "Small-sample estimation of negative binomial dispersion, with applications to SAGE data." In: *Biostatistics* (Oxford, England) 9.2, pp. 321–332. issn: 1465-4644. doi: 10.1093/biostatistics/kxm030. url: <http://dx.doi.org/10.1093/biostatistics/kxm030>.
  
37. Winkelmann, Rainer (2008). *Econometric analysis of count data*. url: <http://www.worldcat.org/isbn/9783540783893>.

38. Zhang, H., Zhang, B. M., Guo, X., Xu, L., You, X., West, R. B., Bussel, J. B., & Zehnder, J. L. (2019). Blood transcriptome and clonal T-cell correlates of response and non-response to eltrombopag therapy in a cohort of patients with chronic immune thrombocytopenia. *Haematologica*, 105(3), e129–e132. <https://doi.org/10.3324/haematol.2019.226688>
39. Wang, L., Xi, Y., Sung, S., & Qiao, H. (2018). RNA-seq assistant: machine learning based methods to identify more transcriptional regulated genes. *BMC Genomics*, 19(1). <https://doi.org/10.1186/s12864-018-4932-2>.
40. Sergushichev, Alexey. 2016. “An algorithm for fast preranked gene set enrichment analysis using cumulative statistic calculation.” *bioRxiv*, 060012. <https://doi.org/10.1101/060012>.
41. Young, Matthew D, Matthew J Wakefield, and Gordon K Smyth. 2010. “goseq : Gene Ontology testing for RNA-seq datasets Reading data.” *Gene*, 1–21. <http://cobra20.fhcrc.org/packages/release/bioc/vignettes/goseq/inst/doc/goseq.pdf>.
42. Oluwafemi A. Sarumi, Dominik Heider, Large language models and their applications in bioinformatics, *Computational and Structural Biotechnology Journal*, Volume 23, 2024, Pages 3498-3505, ISSN 2001-0370, <https://doi.org/10.1016/j.csbj.2024.09.031>. (<https://www.sciencedirect.com/science/article/pii/S2001037024003209>)
43. Andrés-León, E., Núñez-Torres, R., & Rojas, A. M. (2016). miARma-Seq: a comprehensive tool for miRNA, mRNA and circRNA analysis. *Scientific Reports*, 6(1). <https://doi.org/10.1038/srep25749>
44. Wolfien, M., Rimbach, C., Schmitz, U., Jung, J. J., Krebs, S., Steinhoff, G., David, R., & Wolkenhauer, O. (2016). TRAPLINE: a standardized and automated pipeline for RNA sequencing data analysis, evaluation and annotation. *BMC Bioinformatics*, 17(1). <https://doi.org/10.1186/s12859-015-0873-9>
45. Orjuela, S., Huang, R., Hembach, K. M., Robinson, M. D., & Soneson, C. (2019). ARMOR: an automated reproducible MODular workflow for preprocessing and differential analysis of RNA-SEQ data. *G3 Genes Genomes Genetics*, 9(7), 2089–2096. <https://doi.org/10.1534/g3.119.400185>

## LIST OF PUBLICATIONS

1. LYNX-RNA: A Scalable Nextflow Workflow for RNA-Seq Analysis with Integrated Large Language Models for Comprehensive Result Interpretation. Accepted in IEEE 2025 3rd International Conference on Communication, Security, and Artificial Intelligence.







# DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daultpur, Main Bawana Road, Delhi-110042, India

## PLAGIARISM VERIFICATION

Title of the Thesis: **LYNX-RNA: A Nextflow-Based Modular RNA-Seq and Machine Learning Pipeline for Biomarker Discovery and LLM- summarized Report Generation in Immune Thrombocytopenia.**

Total Pages: 103

Name of the Student: **Devanshi Sharma**

Supervisor: **Dr. Asmita Das**

Department of Biotechnology, Delhi Technological University, Delhi - 110042

This is to report that the above thesis was scanned for similarity detection. Process and outcome is given below:

Software used: **Turnitin**, Similarity Index: **9%**, Total Word Count: **17,725 Words**

Date: 29.05.2025

**Candidate's Signature**

**Signature of Supervisor**



## 9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.





### Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 15 words)




### Exclusions

- 42 Excluded Matches

### Match Groups

-  **35 Not Cited or Quoted 9%**  
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**  
Matches that are still very similar to source material
-  **0 Missing Citation 0%**  
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**  
Matches with in-text citation present, but no quotation marks

### Top Sources

- 9%  Internet sources
- 5%  Publications
- 4%  Submitted works (Student Papers)

### Integrity Flags

#### 0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.