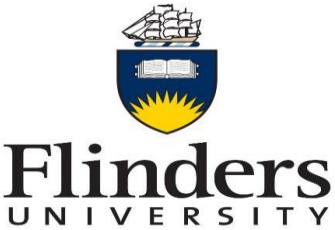
A Workflow for Automated Pairwise Differential Gene Analysis with Real Time Hierarchical Visualization in Shiny R

A thesis submitted for the fulfilment of the degree of

Master of Biotechnology

By

Rohan Khatri



Flinders University

College of Medicine and Public Health

Bedford Park, South Australia

June, 2025

Table of Contents

List of Figures

List Of Tables

List of Abbreviations

Abbreviation	Explanation
DNA	Deoxy Ribonucleic Acid
RNA	Ribonucleic acid
NCBI	National Centre for Biotechnology Information
DDBJ	DNA Data Bank of Japan
PIR	Protein Information Resource
CATH	Class, Architecture, Topology, and Homology
PDB	Protein Data Bank
GEO	Gene Expression Omnibus
KEGG	Kyoto Encyclopedia of Genes and Genomes
STRING	Search Tool for the Retrieval of Interacting Genes/Proteins
NGS	Next Generation Sequencing
BLAST	Basic local Alignment Search Tool
MEGA	Molecular Evolutionary Genetics Analysis
PHYLIP	PHYLogeny Inference Package
RAxML	andomized Axelerated Maximum Likelihood
FASTA	Fast Alignment
SAM	Sequence Alignment Map
BAM	Binary Alignment Map
BED	Browser Extensible Data
GTF	Gene Transfer Format
GO	Gene Ontology
MF	Molecular Function
BP	Biological Process
CC	Cellular Components
ANCOM-BC	Analysis of Composition of Microbiomes with Bias Correction
DEG	Differential Gene Expression
AGRF	Australian Genomics Research Facility
Log	logarithm

MM	Methyl mercaptan
%	Percentage
Seq	Sequence

ABSTRACT

Due to the advancements in sequencing technologies, the amount of transcriptomics and metagenomics data has grown exponentially. As a result, advanced automated bioinformatics workflows are needed for data analysis and interpretation. Dedicated tools for differential gene expression analysis are available for data analysis. However, they are often insufficient for interpreting hierarchical annotated datasets, especially datasets generated from the SEED subsystems. To address this gap, this thesis designs and implements a new analytical bioinformatics pipeline to explore and understand differentially abundant and/or differentially expressed features in a hierarchical dataset, such as data generated from microbial taxonomy analysis (i.e., taxonomic hierarchy), or shotgun DNA/RNA function counts (i.e., functional hierarchy). The primary goal of the research was to build an analysis workflow that can perform the differential gene expression/abundance automatically. The pipeline begins with raw hierarchical gene-annotated data derived from the SEED subsystem hierarchy (level 1 to level 4), along with sample grouping metadata. A Custom R coding script automatically generates all possible pairwise comparisons between sample groups defined in the metadata. DESeq2 is used to perform differential gene expression analysis across all generated pairwise comparisons, and results are stored in a dedicated output directory. These results contain a long list of differentially expressed genes, including log2 fold change values, p values, and adjusted p values, which are difficult to interpret. To address this challenge, a novel pipeline provides a dynamic, shiny dashboard for hierarchical data exploration with real-time filtering options. This novel pipeline generates multiple visualisation outputs, such as bar plots and volcano plots, with real-time filtering options. Users can filter genes across levels 1 to 4, and view volcano plots and an interactive data table that updates according to their selection. Eighteen test DNA samples (six groups, three replicates each) were used for pipeline testing. Fifteen different pairwise comparisons generated by the novel pipeline and DESeq2's result were stored in a dedicated output directory. The real-time filtering options across the hierarchical level reveal a few essential patterns of gene expression, such as " Amino Acids and Derivatives", " Carbohydrates" and " Cofactors, vitamins and Pigments", which are highly expressed in all pairwise comparisons. For validation, this pipeline was tested against the tongue biofilm meta transcriptome-halitosis associated dataset from a study published in npj Biofilms and Microbiomes and got 90% similar results with the existing study's result. In conclusion, this project provides a scalable, reusable and novel bioinformatics pipeline for the

exploration and interpretation of transcriptomics and metagenomics data with a user-friendly shiny dashboard for dynamic filtering and visualisations across hierarchical levels.

Declaration

I, Rohan Khatri, declare that the work in this Thesis entitled “A Workflow for Automated Pairwise Differential Gene Analysis with Real Time Hierarchical Visualization in Shiny R.” is my own original research that has been carried out by me in the Department of Science and Engineering, Flinders University. does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university. Research within will not be submitted for any other future degree or diploma without the permission of Flinders University. To the best of my knowledge and belief, does not contain any material previously published or written by another person except where due reference is made in the text.”

ROHAN KHATRI

02 JUNE 2025

Acknowledgement

I would like to express my sincere gratitude to my supervisor Professor Jim Mitchell for providing me this golden opportunity to work under him where I got chance to learn deep about the scientific research and writing. Thank you for your precious advice, encouragement, support and guidance throughout the period. I am deeply grateful to my co-supervisor Dr. Nicholas Falk for his invaluable assistance and guidance during the whole project.

I would like to extend my sincere thanks to all the faculty members of Master of Biotechnology for providing me vast understanding in various aspects of biotechnology and science since the first year of my master's degree. I also would like to thank Assoc. Prof. Alistair Standish and Dr. Liu Fei who always supported me with their valuable advice and feedback.

At last, but not the least, I want to thank my family and all the friends for their warm and kind support throughout the period.

CHAPTER 1: INTRODUCTION

1 Introduction

DNA is present in all living organisms and carries the genetic information and instructions for cell growth, function and development. DNA is often referred to as the "blueprint of life" (Saridakis, 2021) and serves as the hereditary material that is passed from one generation to the next. The first double helix structure of DNA was proposed by James Watson and Francis Crick in 1953 (Watson & Crick, 1953). RNA (Ribonucleic acid) is a single-stranded nucleic acid that plays an essential role in protein synthesis and gene regulation (Brosius & Raabe, 2016) and is transcribed from instructions within the chromosomal DNA of cells. Connecting DNA, RNA, and proteins is termed the "central dogma", which was first coined by Crick in 1958 (Cooper, 1981) with a brief explanation given in Figure 1.1 below. DNA, RNA & protein are the building blocks of life; therefore, studying all of these is quintessential in the field of life sciences.

Figure removed due to copyright restriction.

Figure 1.1. Central dogma: Central dogma of life, the process in which DNA is transcribed into RNA, known as transcription. The conversion of RNA into protein is known as translation. In some cases, RNA can be converted back into DNA through a process called reverse transcription (Cooper, 1981).

DNA is made up of 4 bases, adenine, thymine, guanine and cytosine (Watson & Crick, 1953). DNA sequences are organised in chromosomes and contain genetic information. The determination of the order of these base pairs is referred to as DNA sequencing, with many methods developed over time. The timeline of sequencing methods is briefly summarised in Figure 2 (Pereira et al., 2020). The first sequencing method, Sanger sequencing, was published in 1977 by Nobel Prize-winning scientist Frederick Sanger (Mitchelson et al., 2007). The

development of the Polymerase Chain Reaction (PCR) occurred in 1983, followed by the launch of the first automated sequencing system in 1986. After that, various next-generation sequencing platforms were developed. For example, in 2005, Roche launched the 454 platform. The Oxford Nanopore MinION portable long-read sequencer, launched in 2015, and SeqLL, released single-molecule sequencing in 2017, have marked significant innovations in genomics in the last decade (Heather & Chain, 2016; Mitchelson et al., 2007; Pereira et al., 2020).

Figure removed due to copyright restriction.

Figure 1.2. Timeline of DNA sequencing: This figure provides a detailed timeline of the DNA sequencing methodology (Mitchelson et al., 2007; Pereira et al., 2020).

DNA, RNA & proteins can be arranged in sequence formats output from sequencing methods. RNA sequence data is particularly important for gene regulation and expression studies, whereas protein sequences provide deeper information about structure-related functions. Therefore, studying RNA sequences and protein sequences are important in the field of biology.

Biological sequence data is being generated at an unprecedented rate. For example, in the year 2000 the GenBank repository of nucleic acid sequences contained 8,214,400 entries and the SWISS-PROT database contained 88,166 entries. It can be clearly seen that this amount of information doubles every 15 months (Luscombe et al., 2001).This surge in data presents challenges in the field of biology, particularly in bioinformatics, which is applied in the analysis

of this data. Bioinformatics combines fields of biology, computer science, mathematics, and statistics to analyse and interpret data, producing meaningful results from sequence data. It plays a vital role in biology and medicine, where it is used to manage and understand large volumes of data generated by techniques like genome sequencing (DNA-based), proteomics (protein-based), and transcriptomics (RNA-based). At its core, bioinformatics focuses on the development and application of computational tools and methods to analyse and interpret complex biological systems and data, including DNA, RNA, and protein sequences. It is an umbrella field that complements the wider range of biological studies and analyses different types of biological data, including sequence annotation, data visualisation, data mining, data exploration and structuring the biological data (Abdurakhmonov, 2016). Sequence data is catalogued and stored digitally in publicly available (and in some cases private) databases. The following are some examples of publicly available data sources.

Table 1.1 List of important and popular database resources for bioinformatics.(Pathak et al., 2022)

Database	**Application**	**Availability**	**References**
National Centre for Biotechnology Information (NCBI)	It offers access to biomedical and genomic information to boost research activity.	https://www.ncbi.nlm.nih.gov/	Benson, Boguski, Lipman, and Ostell (1990)
GenBank	It is a nucleotide database available at NCBI. It is used for the retrieval of nucleotide sequences.	https://www.ncbi.nlm.nih.gov/genbank/	Benson et al. (2012)
European Nucleotide Archive	It is a nucleotide database available at EBI-EMBL. It provides a detailed record of nucleotide	https://www.ebi.ac.uk/ena/browser/home	Leinonen, Sugawara, and Shumway (2010)

Database	Application	Availability	References
	sequencing data, covering raw sequencing data, information about sequence assembly, and functional annotation.		
DDBJ	It is a nucleotide database available at NIG, Japan. It is used for the retrieval of nucleotide sequences.	https://www.ddbj.nig.ac.jp/index-e.html	Tateno et al. (2002)
Protein Information Resource (PIR)	PIR is an integrated database to support research and scientific studies in genomics, proteomics, and systems biology.	https://proteininformationresource.org/	Wu et al. (2003)
UniProt	UniProt aims to provide a complete, high-quality, and freely available protein sequence and functional information resource to the scientific community.	https://www.uniprot.org/	Apweiler et al. (2004)
Pfam	It is a broad set of protein families database used for domain analysis.	https://pfam.xfam.org/	Bateman et al. (2002)

Database	Application	Availability	References
CATH (Class, Architecture, Topology, and Homology)	The CATH database offers information on the evolutionary relationships of protein domains as a free, publicly accessible online database resource.	https://www.cathdb.info/	Orengo et al. (1997)
SCOP	The purpose of the SCOP database is to provide a detailed and systematic explanation of the structural and evolutionary relationships between proteins deposited in the RCSB Protein Data Bank.	http://scop.mrc-lmb.cam.ac.uk/	Murzin, Brenner, Hubbard, and Chothia (1995)
Protein Data Bank (PDB)	It is a structural database that contains the 3D structure of macromolecules. It is crucial for research in the area of structural bioinformatics, drug discovery and protein structure prediction, etc.	https://www.rcsb.org/	Berman et al. (2000)

Database	Application	Availability	References
PubChem	It is the largest database of chemical information. Here, we search chemical molecules and retrieve their structure for molecular docking/virtual screening.	https://pubchem.ncbi.nlm.nih.gov/	Bolton, Wang, Thiessen, and Bryant (2008)
ZINC	It is a database that contains commercially available molecules for computational screening. We can search and retrieve analogs for any molecule based on similarity. It plays a key role in the discovery of lead molecules for drug development.	https://zinc.docking.org/	Sterling and Irwin (2015)
GEO	GEO is a database for functional genomics. It contains gene expression/microarray data. Here, we can download gene expression profiles submitted by the scientific community	https://www.ncbi.nlm.nih.gov/geo/	Edgar, Domrachev, and Lash (2002)

Database	Application	Availability	References
	throughout the world for further investigation as per need.		
Sequence Read Archive	It is the largest publicly available repository of high-throughput sequencing data. Here, we can download NGS data submitted by the scientific community throughout the world for further investigation as per need.	https://www.ncbi.nlm.nih.gov/sra	Leinonen et al. (2010)
The Arabidopsis Information Resource	It is a database of the model plant *Arabidopsis thaliana* provides genetic and genomic information to the scientific community.	https://www.arabidopsis.org/	Rhee et al. (2003)
Rice Genome Annotation Project	It provides sequence and annotation data for the rice genome.	http://rice.plantbiology.msu.edu/	Kawahara et al. (2013)
Gramene	It is a curated, open-source, integrated database platform for research in the area of	https://www.gramene.org/	Ware et al. (2002)

Database	Application	Availability	References
	comparative functional genomics of crop plant species.		
Kyoto Encyclopedia of Genes and Genomes (KEGG)	Database of gene/genome sequencing information for the understanding function of biological system	https://www.genome.jp/kegg/	Kanehisa and Goto (2000)
Search Tool for the Retrieval of Interacting Genes/Proteins (STRING)	It is a database resource for known and predicted protein–protein interactions derived from experimental, computational methods, and text mining.	https://string-db.org/	Szklarczyk et al. (2019)
BioModel	It is a repository of mathematical models. It provides a wide variety of current physiologically and pharmaceutically applicable mechanistic models based on literature in standard file formats.	https://www.ebi.ac.uk/biomodels/	Le Novere et al. (2006)

1.1 Brief History & Advancement of bioinformatics

Bioinformatics originated decades before DNA sequencing became a feasible technology. Margaret O. Dayhoff is known as the mother of Bioinformatics because she developed the early computer programs able to determine the peptide structure for x-ray crystallography and published the book "Atlas of Protein Sequence and Structure". This book gave a new direction to the field of bioinformatics (Diniz & Canduri, 2017). The first dynamic programming algorithm for pairwise alignment for protein sequence was developed by Needleman and Wunsch in 1978 (Pathak et al., 2022), which was fundamental in identifying variations in protein sequences that inform on genetically based diseases. After 10 years in 1980, the game-changing software for multiple sequence alignment was introduced, named CLUSTAL. In 1988, the Notational Centre for Biotechnology Information (NCBI) was established. As genomic research moved from the protein space to the DNA space, the human genome project entered the field in 1990, which was revolutionary for managing a large dataset and assembling small DNA sequences into larger chromosomes computationally (Sawicki et al., 1993). Between the years 2000 and 2010, because of advancements in both biology and computing, high-throughput bioinformatics techniques were applied and could be utilised on next-generation sequencing (NGS) data (Pathak et al., 2022). In today's era, artificial intelligence and machine learning techniques are being used to produce faster and more efficient results.

1.2 Aims of bioinformatics

There are 3 crucial aims of bioinformatics. First and foremost, it simplifies biological data into a clear and concise form that allows researchers to access existing information or contribute new data to existing databases. The second aim is to develop tools that aid in analysing this data (Luscombe et al., 2001).The third aim is developing new software and algorithms. Apart from this, there is an improvement of the content and usefulness of the already available database to document the data in a more organised way (Pathak et al., 2022).

1.3 Needs of Bioinformatics

Due to advancements in technology, such as sequencing methods including those from Illumina, PacBio, and MiniOn platforms, researchers generate large amounts of annotated biological data. The analysis and management of these large datasets requires not only bioinformatic tools, but also analysis pipelines that integrate and visualise the data products of these tools so that users can explore large biological datasets beyond their immediate or central hypotheses (Pathak et al., 2022).

1.4 Application of Bioinformatics in Biological Research.

Figure removed due to copyright restriction.

Figure 1.3. Application of bioinformatics in biological research: A brief overview about how bioinformatics can be helpful in various fields to solve biological problems (SILVA & ALVES).

Bioinformatics has multiple applications as listed above in Figure 1.3. It has great potential to solve biological research problems. Starting with the Human Genome Project, bioinformatics has become an integral tool not only for scientists but also for industry, playing a significant role in the fields of life science, chemical science, physical science, agriculture, cancer research, healthcare, and many other disciplines. Some critical applications are discussed below.

1.4.1 Sequence analysis

Sequence comparison analysis was revolutionised with the development of the basic local alignment search tool (BLAST) in 1990, which is useful for finding similarity between two sequences, including nucleic acid or proteins (Altschul et al., 1990). It is one of the significant applications of bioinformatics in extracting information about gene identification, diseases, genetic variation, and mutations (Mount, 2004). Bioinformatics also plays an essential role in metagenomics (Environmental or community of genomics) by analysing and annotating

sequences to existing databases and providing the information about the composition and function of whole microbial communities, for example, present in human and environmental samples (Handelsman, 2004). In transcriptomics, bioinformatics provides information about differential gene expression. Protein sequence analysis is also helpful for understanding the structure and function of proteins with the help of publicly available databases (Punta et al., 2012).

1.4.2 Phylogenetic analysis

Phylogenetic analysis is another important research area in the field of bioinformatics, involving the study of evolutionary relationships among genes and organisms using sequence data (Challa & Neelapu, 2019). There are multiple tools available, such as MEGA, PHYLIP and RAxML, which utilise sequence data and help in phylogenetic tree construction for understanding the evolutionary relationship between genes and organisms based on sequence similarity (Godini & Fallahi, 2019). It also supports the identification of significant regions of interest within the sequences and provides an essential guide in vaccine and drug design (Pathak et al., 2022).

1.4.3 Prediction of protein structure

Predicting protein structures is crucial to understanding protein functions and interactions. In recent years, due to the overwhelming volume of data, automated or semi-automated methods in bioinformatics for protein structure prediction from amino acid sequence have come to light (Pavlopoulou & Michalopoulos, 2011). The 2024 Nobel Prize in Chemistry was awarded to David Baker, Demis Hassabis, and John Jumper for their work on protein structure prediction and computational protein design.

1.5. Different bioinformatics file types

Producing logical and organised data outputs is vital for parsing and analysing biological sequence data. Over the years, many file types have been developed, but several stand out and are utilised widely as standards.

1.5.1 Sequence file formats.

1.5.2 FASTA

It is a simple file format to provide information about DNA, RNA, or protein sequences. It is most used for storing and sharing of sequencing data in bioinformatics. This file starts with the '>' SIGN followed by the sequence of amino acids or DNA base pairs (Roughan, 2022).

1.5.3 FASTQ

The FASTQ format is complementary to the FASTA format, but the 'Q' stands for quality in this format. Therefore, this file format provides the same information as a FASTA file but also includes the quality of the sequence read as provided by the sequencing platform. It generally has four lines where the first line starts with the '@' sign and is a sequence identifier. The second line contains the raw sequence data. The third line has the "+" sign for separation of data, and the fourth line has a quality score for each score; this line is identical to the second line in length (Roughan, 2022).

1.5.4 Alignment file format

1.5.5 SAM

SAM stands for the Sequence Alignment Map. This file consists of one header section and one alignment section. In this file each alignment line has 11 mandatory field such as query name, flag, pos, etc (Li et al., 2009; Roughan, 2022).

1.5.6 BAM

BAM represent the Binary alignment map format, which is the binary representation of the SAM file format (Li et al., 2009; Roughan, 2022).

1.5.7 Other file formats in bioinformatics.

BED represent the Browser Extensible Data file format, which includes information about the sequence and can be visualised in a genomic browser. The BED file format provides three essential pieces of information: chromosome names, start positions, and end positions. The PDB file format stores information about protein structures. GTF (Gene Transfer Format) stores information about gene regulation (Roughan, 2022).

1.6 Function Annotation and the Subsystem in Bioinformatics

Function annotation is the process of assigning biological information to specific genes, including their function, identity, and/or location in chromosomes (Berardini et al., 2004). Large datasets require a function annotation to analyse and extract meaningful information from the sequences, as without annotation, the generated sequences are raw data, and no information can be obtained from them. Therefore, function annotation is necessary in biological and bioinformatics research because it is essential for identifying genes, especially those associated with their role in disease or consequential outcomes in the study system. Function annotation is also crucial for the interpretation of metagenomic data. It plays an

essential role in understanding the biological pathway, such as which genes are involved in which parts of the biological pathway.

1.6.1 Function annotation tools in bioinformatics.

Function annotation in bioinformatics is often dependent on a well-established database. The following are some of the most important tools.

1.6.2 KEGG (Kyoto Encyclopaedia of Genes and Genomes)

KEGG is an important database for the biological interpretation of genome sequences and other high-throughput data. More than 4000 complete genomes are annotated within the KEGG database (Kanehisa et al., 2016). In 1995, the KEGG project started with the human genome project (Ogata et al., 1999). The main advantage of KEGG is that it provides the links between the sets of genes from genomes and the functions of cells and organisms (Kanehisa et al., 2016). This database is based on the three main components: 1) KEGG PATHWAY, as it has all the information of high-level functions, and it has a collection of graphical diagrams of biological pathways (Ogata et al., 1999), 2) KEGG GENES has a collection of complete sets of sequenced genomes, 3) KEGG ORTHOLOGY, which provides the links between the genes and functions.

1.6.3 Gene Ontology (GO)

Gene ontology is a well-structured bioinformatics database that provides structured, standard, and controlled vocabularies and classifications that cover several domains of biology, and it is freely available for the community to use for annotation of genes, gene products and sequences (Consortium, 2004). It is subdivided into three non-overlapping ontologies, such as molecular functions (MF), Biological processes (BP), and Cellular components (CC) (Du Plessis et al., 2011). In this database, each annotation has a source of entry. Source can be literature referenced, database references, or computational evidence (Du Plessis et al., 2011).

1.7 Hierarchical Subsystems in Bioinformatics.

In a hierarchical subsystem organisation, high-throughput biological data is organised in a hierarchy that organises the biological function into nested categories. For example, data are organised at different levels, such as broader categories (level 1) to more specific categories (level 2,3,4 and 5). This format allows researchers to interpret their data on multiple levels, from broader function (ex, level 1 in figure 4) to specific gene role (ex, level 3 in figure 4). Another common example is taxonomic systems, where a single phylum (akin to a level 1) can contain multiple nested classes, orders, families, etc. This subsystem hierarchy organisation is crucial for managing large datasets, especially for metagenomics and transcriptomics data,

which assumes that biological functions fall under broad metabolic pathways, and can then be nested into more specific branching functions, which may represent particular enzyme products that catalyse reactions. This subsystem helps simplify data interpretation and computation, and also provides insight into the subsystem's extensive interactions and synergies.

Figure removed due to copyright restriction.

Figure 1.4- Representation of subsystems structure, how each subsystem works and generates the data in a hierarchical format. For example, data is arranged at different levels from the border level to the specific level (L1 to L4) (Silva et al., 2016).

1.8 Current Tools for Hierarchical Functional Profiling

1.8.1 Super focus

Super focus is a bioinformatic tool helpful for functional annotation and profiling of metagenomic data. This is faster than other tools because it features three improvements over other metagenome annotation tools. Firstly, it features a clustered version of the SEED database, which reduces the total search space. Second, it identifies the genera or functions present in the metagenomic sample using FOCUS. Lastly, alignment is performed by RAPsearch2, which is 100 times faster than BLASTX. Therefore, SUPERFOCUS can be up to 1000x quicker than other tools (Silva et al., 2017). The input data can be in the form of FASTA or FASTQ sequences generated by sequencing platforms. Workflows consist of one pre-processing step followed by five computational steps. In the preprocessing stage, it creates a reduced database by clustering the sequences from the SEED database. In the next step, FOCUS identifies the genera from the samples. After that, input sequences are aligned against

a reduced database using RAPsearch2 (default), DIAMOND and BLASTx. In the last step, the alignment output is filtered by some fixed default criteria, and function annotation is generated for all subsystem levels (Silva et al., 2016; Silva et al., 2017).

1.9. Differential Expression/ Abundance Analysis in Omics Studies

DNA and RNA sequencing are strong tools to learn about gene abundance and gene expression. Transcriptomics is the study of RNA that is important for understanding gene expression, especially in biomarker or in the detection of early complex diseases (Dong & Chen, 2013). The transcriptome is the complete set of RNA molecules, a term first proposed by Charles Auffray in 1996 and 1997 (McGettigan, 2013). Studying transcriptomic data is a challenging and time-consuming process because it contains vast amounts of information, particularly when the goal of researchers is hypothesis generation, rather than hypothesis testing. In transcriptomics, differential gene expression analysis identifies genes that are statistically significantly upregulated or downregulated under a specific condition, as inferred from their mRNA count data, which can be annotated to the SEED Subsystem, for example. Differential expression analysis can be useful in various biological fields, including cancer research, neuroscience, and environmental studies. There are several tools available to do analysis; however, DESeq2, EdgeR, and Limma are some of the most widely applied algorithms. All three are Bioconductor packages and can run in the R programming language (Elahimanesh & Najafi, 2024). For differential abundance analysis, ALDx2 and ANCOM-II are the most common packages used; however, differential expression and differential abundance analysis can be interchangeable, as long as the data fit the assumptions of the statistical tests used. Those two tools have been applied to identify abundant microbes in a microbiome study between treatments and sample groups, for example (Nearing et al., 2022). However, all the above-mentioned packages have limitations; all the mentioned tools cannot provide analysis across all hierarchical count data (i.e., multiple hierarchical levels) because they only provide analysis for one level intuitively, and require multiple iterations to perform statistical tests across multiple levels and/or sample groups. For a better understanding of hierarchical count data, a new analysis workflow is essential for providing analysis for all hierarchical levels instead of one, particularly for less experienced users.

1.10 Differential gene expression Tools.

Differential gene expression analysis is a critical aspect in transcriptomics research because it provides identification of the genes that show statistically significant changes in expression between two biological conditions, such as treatment and control (Rapaport et al., 2013).

1.10.1 DESeq2

DESeq2 is an R Bioconductor package that applies a negative binomial distribution for differential gene expression analysis. DESeq2 also utilises the shrinkage estimation for dispersion and fold changes to improve the stability and interpretability of gene count data. DESeq2 used the median-of-ratio method for gene-specific normalisation (M. I. Love et al., 2014). It also utilises a generalised linear model framework for differential expression. This package is also capable of providing some visual outputs, such as heat maps and volcano plots, which play an essential role in data interpretation (M. Love et al., 2014).

1.10.2 EdgeR

EdgeR is an R Bioconductor package designed to analyse differential expression of replicated count data (Robinson et al., 2010). It also employs the negative binomial distribution for modelling the count data. EdgeR estimates the gene-wise dispersions by conditional maximum likelihood. It uses Fisher's exact test for analysing differential expression, which is highly useful, especially for small samples.

1.10.3 Limma

Limma is an R package applied to analyse differential expression for microarray data. Limma uses the voom transformation model, which converts the raw count into log-count per million, by automatically calculating the library size and normalisation factor from itself. This is a powerful tool dealing with large datasets and is well-suited for highly replicated experiments.

1.10.4 Why DESeq2 is used for this study

Several statistical tools have been developed for differential gene expression analysis, including DESeq2, EdgeR and limma. Deseq2 and EdgeR are both based on the negative binomial distribution, making them well-suited for RNA-sequence data. On the other hand, limma was developed initially for microarray data. A comparative study suggested that DESeq2 is most suitable for large datasets (Soneson & Delorenzi, 2013). DESeq2 was selected because of its statistical robustness, flexibility and widespread availability. It can generate pairwise comparisons based on provided metadata, which is easily manipulated and parsed by the algorithm. Apart from this, DESeq2 was most suitable for providing an existing code framework for the visualisation of the results.

1.11 Bioinformatics pipeline

Bioinformatics pipelines are chains of computational processes, tools or coding scripts that are specially designed for analysing specific biological data. Pipelines are arranged in a step-by-step manner to produce automated, faster, and reproducible results by filtering and processing the given raw data. Different pipelines are designed based on three key dimensions (1) syntax: pipeline uses the implicit or explicit syntax. (2) Design paradigm: pipeline uses the configuration, convention or class-based design. (3) Interface: it is a command line-based or a graphical workbench interface (Leipzig, 2017). Pipelines originate from the simple writing of code to then utilise modern software and workflow management systems.

1.12 Modern pipeline frameworks.

1.12.1 Implicit convention frameworks.

These pipelines rely on a full coding script, such as Python or R, for both input and output of the results. It also depends on some workflow management systems like Snakemake and Nextflow (Leipzig, 2017).

1.12.2 Explicit framework.

Explicit frameworks simplify the pipeline creation. For example, Bpipe is a tool which helps run and manage bioinformatics pipelines (Sadedin et al., 2012).

1.12.3 Configuration frameworks

This framework depends on the “fixed task” in the workflow.

1.12.4 Class-based framework

This pipeline contains many thousands of lines of code to handle and manage the large workflows (Leipzig, 2017). For example, the Genome Analysis Toolkit (GATK) is a Java library for variant analysis.

1.13 Purpose of bioinformatics pipeline

The primary purpose of bioinformatic pipelines is to ensure automation, reproducibility, scalability and ease of use.

Figure removed due to copyright restriction.

Figure 1.5: Example of bioinformatics pipeline/workflow, how bioinformatics pipeline workflow can generate (Choudhri et al., 2018).

1.14. Examples of Bioinformatics Pipelines

1.14.1 Cyrille2

Cyrille2 is a simple genome annotation pipeline that uses a graphical user interface (GUI). The interface provides management and monitoring of pipeline execution, including creating, adopting, starting, and stopping the pipeline. The output of this pipeline includes gene prediction, homology searches, and protein domain analysis. This workflow starts with the DNA sequences. After that, in the second layer application, a programming interface allows users to access the three databases for sequence comparison. The biological database and end-user interface connect to external computer software, such as Sun Grid Engine (SGE). This pipeline system needs to store different types of information; therefore, it consists of four different databases to store each type of data separately (Fiers et al., 2008). Although this pipeline is flexible and automated, it has some challenges, including data storage, as it relies on different databases and external computer software. Therefore, customisation is quite challenging for this pipeline, and it has limited support for multi-omics data.

Figure removed due to copyright restriction.

Figure 1.6. Cyrille2 pipeline design, how it is organised in 4 layers (Fiers et al., 2008).

1.14.2 MAP-RSeq: For analysis of RNA sequencing data.

MAP-RSeq is a bioinformatic pipeline used for processing RNA sequence data. This pipeline, developed at the Mayo Clinic, utilises the well-developed coding scripts in Python, Java, R and Perl, which are available in two versions. The first version is single-threaded, and it can run on a virtual machine. Another version is relatively straightforward to install and run on an external cluster environment, such as SGE, which is specifically designed to require minimal effort on Linux. This pipeline consists of six major modules, including read alignment, sequence quality assessment, gene expression analysis, exon expression count, detection of expressed single nucleotide variants (SNVs), and final report output. Read alignment is done by the TopHat software, which generates the binary alignment file (BAM). Then, quality assessment of reads is done using the RSeQC software. For gene expression count, HTSeq software is used. The final report comes out in HTML format (Kalari et al., 2014).

Figure removed due to copyright restriction.

Figure 1.7: Flow chart for MAP-RSeq bioinformatic pipeline (Kalari et al., 2014).

1.14.3 Tuxedo suite Bioinformatic pipeline.

The Tuxedo suite pipeline is useful for RNA-sequence analysis and transcript quantification. In this pipeline, TopHat software is also used for reading alignment and mapping. These alignments are then utilised by Cufflinks, which perform transcriptome assembly and provides information on transcript abundance. For differential gene expression, this pipeline relies on the cuffdiff tool. The final differential expressed gene analysis comes out in a list (Diniz & Canduri, 2017)

Figure removed due to copyright restriction.

Figure 1.8: Workflow of the Tuxedo suite bioinformatic pipeline (Diniz & Canduri, 2017).

1.14.4 RNA Flow Bioinformatic pipeline

The RNA flow bioinformatic pipeline is a comprehensive and automated bioinformatic pipeline that specialists and non-specialists in bioinformatics can use. It is a portable, scalable, and automated Nextflow RNA-sequence pipeline used to detect differentially expressed genes in data generated from RNA sequencing. For input, two parameters are required: the first is the RNA sequences (either single or paired end), and the second is the reference genome with matching annotation in GTF format. Quality reports of raw reads can be generated by running the FASTQC tool. Then, differential gene expression is performed by DESeq2. The final output is available in FASTP, BAM, CSV, or Excel format. The summary report is accessible through the multipleQC report for this pipeline (Lataretu & Hölzer, 2020).

Figure removed due to copyright restriction.

Figure 1.9: workflow for RNAflow pipeline (Lataretu & Hölzer, 2020).

1.15 Limitation and gap

In summary, analysis tools such as DESeq2, edgeR, and limma are available to help detect differentially expressed genes (DEGs) in experiments with contrasting treatments and controls. However, all these tools have some limitations because they typically produce a long list of genes that are differentially expressed, such as those that are upregulated and downregulated. These outputs typically require more interpretation to extract biological insight and draw logical conclusions (Liu et al., 2021). This situation becomes worse when datasets have thousands of genes. Apart from this, one fundamental aspect is that genes are organised in a hierarchy and are poorly characterised at the individual level, but are well characterised at the functional level of subsystems or pathways. For example, indications of wide upregulation of genes involved in carbohydrate metabolism are stronger evidence than interpreting thousands of genes separately. Therefore, there was more attention on functional annotation. Most of the tools available for differentially expressed gene analysis do not provide visualisation and often focus only on a particular subsystem level, lacking hierarchical support for functional annotations. Functional databases, such as SEED, provide a hierarchical (multi-level) gene annotation structure, ranging from broad categories to particular gene functions. Existing workflows either ignore hierarchy organisation completely or focus solely on one level at a

time. Apart from these, most traditional pipelines require a command line interface and script writing, which can be a daunting task to less experienced researchers. Tools that enable scientists to perform real-time visualisation with user-friendly dashboards would be beneficial for clear data understanding and hypothesis generation from large datasets that often go unexplored once the primary aims of a study are fulfilled.

1.16 Research objectives and hypothesis.

The main objective of this study is to develop a novel analysis pipeline that can automatically perform differential gene expression analysis between all possible pairwise comparisons with hierarchical functional filtering and real-time visualisation on count data. This pipeline is specifically developed to analyse and interpret large datasets without excessive script writing by end users. Aims include:

- To automate all the pairwise comparisons based on metadata.
- To visualise differentially expressed genes on user friendly shiny R dashboard
- To enable real-time filtering across hierarchical functional subsystems

1.17 New Approach to bridge the gap

To meet these needs, our goal is to develop a new analysis workflow that explores and understands differentially abundant and/or differentially expressed features in a hierarchical dataset, such as data generated from microbial taxonomy analysis (i.e., taxonomic hierarchy) or shotgun DNA/RNA function counts (i.e., functional hierarchy). This could be applied to the dynamics of microbial community functions and/or taxa across different samples representing treatment (in the case of an experiment) or environmental conditions. It requires bioinformatic tools such as DESeq2 to perform differential gene expression or differential abundance analysis. The project aims to analyse data, particularly those organised from broad functional categories down to specific gene functions in the case of the SEED annotated functional omics data. For example, data from SEED-annotated output is organised in 4 different subsystem levels (L1 to L4), which can be generated by the tool, SUPER-FOCUS. The project aims to analyse the functional capabilities utilising all 4 levels as opposed to counts from a single layer. The overall goal is to develop a comprehensive analysis workflow in R that identifies differentially expressed genes across layers using DESeq2, thereby improving data exploration and consensus.

The initial stage of the project involves preparing and integrating functional data from L1 to L4, along with metadata. This metadata includes information such as the environmental

conditions, location, treatment group, and number of samples taken from the experiment or study. This data format must first be manipulated so that it is suitable for performing in DESeq2 or any other desired differential expression package. This data must be carefully cleaned and formatted in a way that is easily accessible from tools such as SUPER-FOCUS. This initial step is important for providing a suitable format and dataset for further analysis.

The second stage involves analysis using DEseq2 and other algorithms, a statistical tool specifically designed to handle and analyse gene functional annotated count data and identify differentially abundant or expressed features, which include specific taxa or genes within the experiment (as defined in the metadata). This differentially expressed data can be visualised by volcano plots, MApplot, and R packages including ggplot2 and heatmap. This will offer a clear view for users of the functional dynamics across the different samples using the subsystem level.

After that, the project will move to the novel approach of functional network reconstruction. After having results from DESeq2, a network is built to visualise how the identified functions can interact and relate across sample groups. The end goal is to have a ranked list of differentially expressed Subsystems (for example, at Level 1) between sample groups, based on how many functions (i.e., at Level 4) within those subsystems are differentially expressed. This involves summarising functions at Level 4 based on the higher levels to which they belong, and it will also produce graphics and lists of genes/functions ordered by their significance values (p-value or log2 fold change values in the case of DEseq2).

The goal is to provide users with a more comprehensive understanding of the differences between their groups in omics studies and to enable them to explore their large omics datasets more effectively, generating hypotheses and ideas that may not be intuitively apparent. This will help avoid bias and improve upon current analyses that utilise only one level of a hierarchy in the functional data structure. In conclusion, a literature review highlights the key aspects of bioinformatics and its role in modern biological research, particularly in the application of data analysis, such as in metagenomics and transcriptomics. This review focuses on the bioinformatics pipeline and workflows for data analysis, providing an overview of the workflow structure and how these workflows assist researchers in analysing and handling data to produce meaningful results. Additionally, it reviews the challenges associated with workflows and proposes a new solution and method to address these challenges. For example, despite these advances, some challenges remain, as highlighted by the fact that the analysis

workflow needs to be improved for hierarchical count data, as existing tools and workflows currently focus solely on a single layer rather than all hierarchical levels. This review bridges the gap in handling and analysing hierarchical and multi-level data sets, providing a more comprehensive understanding of data than what is currently achievable with single-layer data using existing tools like DESeq2. For the future direction, artificial intelligence and machine learning models can be implemented to automate workflows.

CHAPTER 2: MATERIALS AND METHODS

2. MATERIALS AND METHODS

Day by day, biological datasets are becoming increasingly complex due to the rapid development of sequencing technologies (Stephens et al., 2015). As a result, advanced computational methods are required to analyse these datasets, especially for differential gene expression analysis. There are multiple existing tools available on the market that perform data analysis; however, they have limitations in the context of interpreting functional subsystems or pathways. This work aims to bridge the gap by developing a novel integrated pipeline that performs automated differential gene expression analysis while simultaneously enabling real-time visualisation and exploration of gene expression data organised within hierarchical functional subsystems. By combining computational methods with an interpretable biological framework, these novel methods aim to enhance the utility of transcriptomics datasets in genomic research and hypothesis generation.

2.1 Data Source and Processing

For building and testing the workflow, 18 test DNA sequence samples (6 groups, 3 replicates each) were used. This data originates from a previous study, and the nature of the samples is ambiguous to the development of the workflow; it merely provides a count-by-sample data output for developing the code. The DNA samples were submitted to the Australian Genome Research Facility (AGRF) for whole-genome shotgun sequencing. Library preparation was performed using the IDT xGen cfDNA & FFPE DNA Prep Kit, where samples were sheared to 550 bp and sequenced on an Illumina MiSeq (600 cycles). Sequence reads were then analysed on the Flinders University HPC Cluster (Flinders University, 2021).

Raw sequence reads (300 bp paired-end) were processed through Trimnami (Roach, 2023), using Fastp (Chen, 2023) and Prinseq++ (Cantu et al., 2019) for quality filtering, trimming, and adapter removal using the default parameters provided. Functional assignment of reads to the SEED Subsystems hierarchical classification structure (Overbeek et al., 2005) was done using SUPER-FOCUS (Silva et al., 2016) with mmseqs2 used for alignment. The resulting functional count table was used for developing the workflow.

This pipeline was developed using R Studio (Version 2024.04.2+764 (2024.04.2+764)) on macOS (version Sonoma 14.5, chip M3). Figure 2.1 illustrates the entire workflow of the bioinformatics pipeline. The primary input was gene-annotated count data, provided in the L4_function_Subsystems.csv file. It consists of 21,675 rows (genes) and 19 columns (sample IDs), representing gene expression data organised according to function subsystems (Overbeek

et al., 2005). Each row provides the information about the gene name or function described in the #Name column. Each row also has a semicolon-separated string which indicates genes are organised in a multilevel subsystem hierarchy (e.g., L1; L2; L3; L4). The remaining columns represent the count data for different samples, indicating how many hits to the database were detected for that function. The unique approach of this dataset was its functional hierarchy; for example, the gene name in #Name column **"Amino Acids and Derivatives; Alanine, serine, and glycine; Alanine biosynthesis; Alanine_racemase, _biosynthetic_(EC_5.1.1.1)"** could be parsed into

Level 1- Broad category - **Amino Acids and Derivatives**

Level 2 – Subsystem group- **Alanine, serine, and glycine**

Level 3-Pathway - **Alanine biosynthesis**

Level 4 – Enzyme Function- **Alanine_racemase, _biosynthetic_(EC_5.1.1.1).**

This structure enables the organisation of gene expression at various biological levels. As a result, researchers could explore not only the differentially expressed individual genes but also aggregate changes across entire functional subsystems.

Before the analysis, the first column was used to assign row names, to retain only numeric data. The second key input was a metadata file. Here, it was the metadata.csv file, containing information about sample names and grouping (Treatment vs control). Here, it was a sample name and different locations where the samples originated from (SITE). Each sample's name in the metadata matches with sample names in CountData (matrix), which is a critical requirement for DESeq2 to perform differential expression analysis accurately. Upon importing the metadata, it was preprocessed to set the sample names as raw names. This ensured that DESeq2's requirement for column names matched between countData(matrix) and metadata. Prior to DESeq2 model creation, a critical validation step was implemented to ensure that sample names were properly aligned between countData(matrix) and metadata. A mismatch at this stage reports an error. The following R command was used [**all(colnames(countData) %in% rownames(metadata))].** If this returns "true" it confirms all the alignment was correct and verified. To meet DESeq2's requirement for statistical modelling, countData (matrix) was explicitly converted into an integer format. The SITE variable was explicitly converted into a factor to define distinct experimental groups for statistical testing. This enable DESeq2 to perform all pairwise comparison between this SITE factor automatically.

2.1 DESeq2 Modeling

Differential gene expression analysis was a principal component of this workflow. Differential gene expression analysis was performed using the DESeq2 package version 1.44.0 from Bioconductor, an essential tool for using the negative binomial distribution. DESeq2 was chosen because of its robust ability to handle multiple biological replicates and complex experimental designs. Apart from this DESeq2 does not require any preprocessing step to provide statistically rigorous results even in small sample designs. Furthermore, the negative binomial model used in DESeq2 includes a gene-specific dispersion that accurately models this variability (M. Love et al., 2014; M. I. Love et al., 2014).

2.1.1 Construction of DESeq2 dataset

A DESeq2 dataset was created by using the **DESeqDataSetFromMatrix () function** in R programming command (M. Love et al., 2014; M. I. Love et al., 2014). In this, countData = int_matrix, which was derived from countData (matrix) and colData (sample metadata) = metadata (metadata.csv file). In this experiment, the design formula was **design = ~ SITE)** which specified that differential gene expression analysis would be assessed with respect to the sample information present under the SITE grouping factor. Prior to implementing DESeq2, no normalisation was performed on the dataset as the DESeq2 analysis included a normalisation step.

2.1.2 DESeq2 Model

Once the DESeq2 dataset was defined, the DESeq2 model was implemented using the DESeq (dds) R programming command. This command performed the sequence of integrated steps required for appropriate differential gene expression analysis. This single command initiates the multi-step process in the pipeline, such as size factor normalisation, dispersion estimation, and generalised linear model fitting under a negative binomial framework (M. Love et al., 2014; M. I. Love et al., 2014).

2.1.3 Automatic pairwise comparison.

```
R

Generate all pairwise comparisons automatically

SITE <- levels(metadata$SITE)

comparisons <- combn(SITE, 2, simplify = FALSE)
```

```
R.
Loop through each pairwise comparison (significant results only)
for (comp in comparisons) {
 site1 <- comp[1]  # control
 site2 <- comp[2]  # treatment
 # Run DESeq2 contrast
 res <- results(dds, contrast = c("SITE", site2, site1))
```

Here, site1 was treated as control group while site2 was treated as a treatment group. The results () R programming function was used to generate the differentially expression testing result. The contrast vector specified the pairwise comparison between site1 and site2. The result () function extracted the result table with log2 fold changes, p values and adjusted p values.

```
R.
Define output directory
output_dir <- "~/Documents/new_DESeq2_results/"
dir.create(output_dir, showWarnings = FALSE, recursive = TRUE)
```

A dedicated output directory was created by using the dir.create () R programming function to store the result of all the pairwise differentially expressed gene expression analysis files. And DESeq2 results were converted to a standard data frame for further manipulation. To identify meaningful biological insights, genes were filtered based on an adjusted p-value. The base R programming command was used to retain only genes where p-adj was less than 0.05. By this function, all the NA values were excluded to avoid inconsistent results. Additional columns were added to the DESeq2 results output, providing information about which samples belonged to the treatment group and which ones belonged to the control group. Direction of regulation was also implemented for providing information about which genes were upregulated (i.e., higher expressed) in treatment vs. in control. The result of each pairwise comparison was directly saved in the created output directory by providing custom paths. The output file is saved in .csv format by default with the comparison group names in the file name for easy access.

2.2Shiny App workflow

In this study, the shiny app workflow was designed to dynamically explore and visualise results (Chang et al., 2012). This workflow was created using the R Shiny package ('shiny' version 1.10.0). The shiny app workflow was utilised to facilitate real-time filtering of differentially expressed genes. All the necessary libraries were loaded before workflow development. The workflow accepts the custom .csv file generated by DESeq2, containing hierarchical annotated gene counts and metadata. This allows users to interactively explore expression changes across multiple hierarchical levels, such as taxonomy or SEED-annotated subsystems. This approach bridges the gap between complex statistical and biological interpretation, mainly when biological hierarchical data (i.e., genes, taxa) is analysed.

2.2.1 Purpose of workflow

The primary purpose of developing a shiny app was to bridge the gap between complex statistical and biological interpretations, especially when genes are part of subsystems. Traditional DESeq2 models provide the results in a statistical table format. To analyse and interpret those tables requires profound programming skills. This workflow addresses all these problems by integrating hierarchy-based filtering, a Log2fold changes summary, tables for differentially expressed genes, and bar and volcano plots for data visualisation, all within a dynamic, interactive Shiny R application.

Data preprocessing within the Shiny app workflow

In the preprocessing step, the first column of the uploaded file was separated by the semicolons (;) using the R programming command strsplit(). This command splits the first column into 4 levels and merges additional annotated text beyond the fourth semicolon into the 4th level. The resulting levels were stored as separate columns named Level 1, Level 2, Level 3, and Level 4, which represent the Subsystems organisation scheme. This methodology can also be adapted to taxonomic strings, with taxa ranks separated by semicolons. The dataset was then suitable for interactive visualisation.

2.2.2 Interactive Interface overview

The user interface was created using the fluidpage () and slidelayout() R programming commands. The Interface was divided into two main parts.

- A sidebar panel containing input controls and a summary of results.

- A main panel displaying the interactive plots and summary tables.

The prominent functions of the sidebar panel are the fileInput () R programming command, which accepts and parses the .csv file. SelectInput () and renderUI () ensure that results are dynamically updated based on the user's selection. For example, Level 2 relies on the Level 1 user's selection; simultaneously, the same logic applies for all levels. Also, the dropdown levels were added for selecting hierarchy levels. verbatimTextOutput() R programming command provides several key results, such as the total number of genes and the number of upregulated and downregulated genes in control and treatment designated groups. For the main panel, plotOutput(), tableOutput and DTOutput() produce the visualisations including a bar plot, volcano plot, and results table within the interface. Displaying dynamic summaries such as the number of selected genes based on positive and negative log2fold change values. The central panel displays the summary table, log2fold change bar plot.

2.2.3 Visualisation Outputs.

Data visualisation is a critical part of bioinformatic pipelines that are designed for less experienced users to explore hierarchical data. Our shiny app generates multiple visualisation outputs with dynamic filtering options for exploratory data analysis, incorporating necessary R package libraries such as ggplot2 and DT. The primary visualisation outputs provided by this workflow are volcano plots, bar plots (M. I. Love et al., 2014; Rutter et al., 2019), and an interactive filtering panel and summary table with searching and filtering options. Volcano plots give precise information about significantly upregulated and downregulated genes. In Volcano plots, the X-axis represents the log2 fold change, which provides information about the direction of gene expression between treatment and controls. The Y-axis represents the negative log10 of the adjusted p-value, which provides information about the significance of each gene and allows visual identification of genes of statistical significance (adj $p < 0.05$), contrasted against non-significant outputs. Genes were classified into three different categories based on the define threshold. For upregulated genes, adjusted p-value < 0.05 and log2 foldchange > 1 (Indicated in red). For downregulated adjusted p-value < 0.05 and log2 foldchange < -1 (Indicated in blue). For non-significant genes, such as all other genes, are indicated in grey. To enhance the clarity and avoid datapoint overlap, geom_point (alpha = 0.7, size = 1.5) was used within the ggplot2 code. Overall, volcano plots are constructed using ggplot2 and reformed immediately based on the user's hierarchical selection within the interface.

Bar plots were also constructed using the ggplot2 library package. The workflow creates the bar plot based on the level 1 Subsystem category (i.e., a Broad Category). In this graph, the X-axis represents the number of genes differentially expressed, while the Y-axis represents the functional categories' names for level 1. Genes with positive log $fold_2$ change values (log fold2 change > 0) are shown in blue, illustrating the upregulation in treatment, whereas genes with negative log $fold_2$ change value(log fold2 change < 0) are shown in red, illustrating the downregulation in treatment. Each bar in the plot represents the gene count for a specific level 1 category. In order to emphasise the contrast and direction, negative values reflect towards the left side, while positive values reflect towards the right side.

The complete list of filtered differentially expressed genes is presented in a table using the DTOutput R programming command. This table features key elements, including column sorting and real-time searching. Overall, this pipeline allows users to search and visualise prioritised significantly differentially expressed genes at any given Subsystem level dynamically

2.3 Benchmark and validation.

To check the accuracy and scalability, benchmarking and validation were essential. This section includes a validation analysis, where we run a previously published dataset through our pipeline. To check the accuracy and validation, we used the tongue biofilm metatranscriptome-halitosis associated dataset from the study (Carda-Diéguez et al., 2022) published in npj Biofilms and Microbiomes.

In this original study, the authors tested and compared the RNA-seq of tongue coating samples from 83 different individuals with or without intra-oral halitosis. The original study identified differential gene expression and microbial taxonomic patterns related to volatile sulphur compounds in the oral environment of subjects. We downloaded the pre-processed annotated gene expression table provided in the supplementary data for the manuscript. After running our automated pipeline, the results were compared with the existing results of this study to ensure that the DESeq2 results generated by our pipeline were accurate.

2.4 Method Summary

This chapter provides an extensive summary of the material and methods used to create an automated and interactive bioinformatics pipeline for differential gene expression analysis. As illustrated in Figure 2.1, it starts with the acquisition of gene-annotated count data and metadata, which can originate from DNA or RNA-seq. The DESeq2 package was used to

perform differential gene expression analysis. For automated pairwise comparisons across multiple groups/factors in metadata, custom R coding scripts were applied that can save DESeq2 results in a dedicated directory. As illustrated in Figure 2.2, the shiny app workflow side panel includes a section for uploading the desired file containing the group comparison of interest, which was generated in the previous step. Another R coding script facilitates the visualisation and outputs that can support dynamic filtering and real-time visual summaries. Volcano and bar plot visualisation offer quick insight into the biological pattern of genes and can guide the user through multiple hierarchical levels, which may represent Subsystem SEED functions or taxonomic ranks.

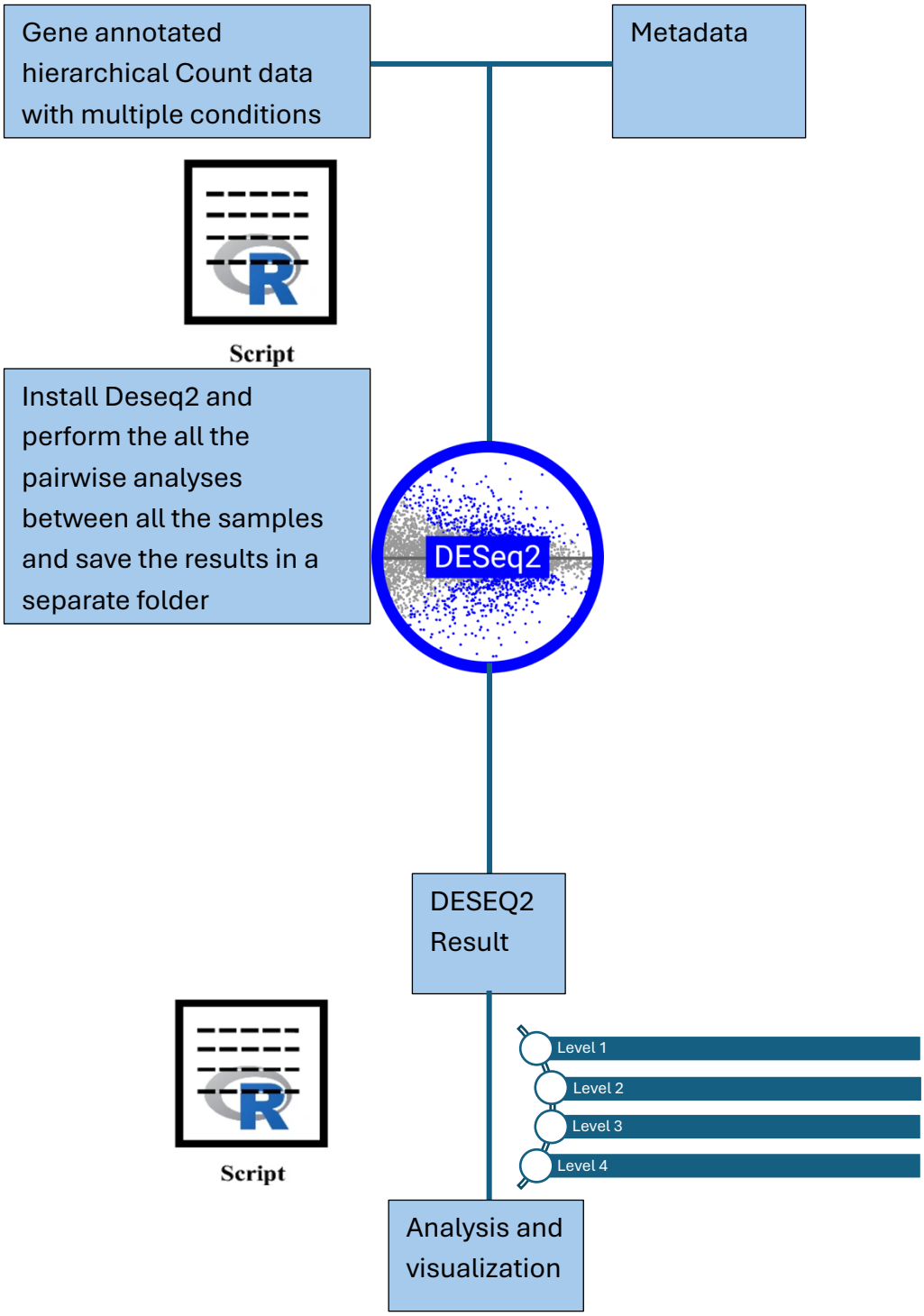


Figure 2.1 Schematic workflow of the novel pipeline.

The given flow chart illustrates the complete structure of the bioinformatics pipeline used in the study. The pipeline begins with the input of gene-annotated count data, along with supporting metadata, which includes all labelling, such as different sample identifiers and group classifications. This input is processed using an R-based coding script, which installs the

DESeq2 package and generates output for differentially expressed genes. The resulting output consists of differentially expressed genes categorised across hierarchical functional levels. The final output is integrated into a Shiny application that facilitates dynamic filtering and visualisation of differential expression results across hierarchical functional annotations.

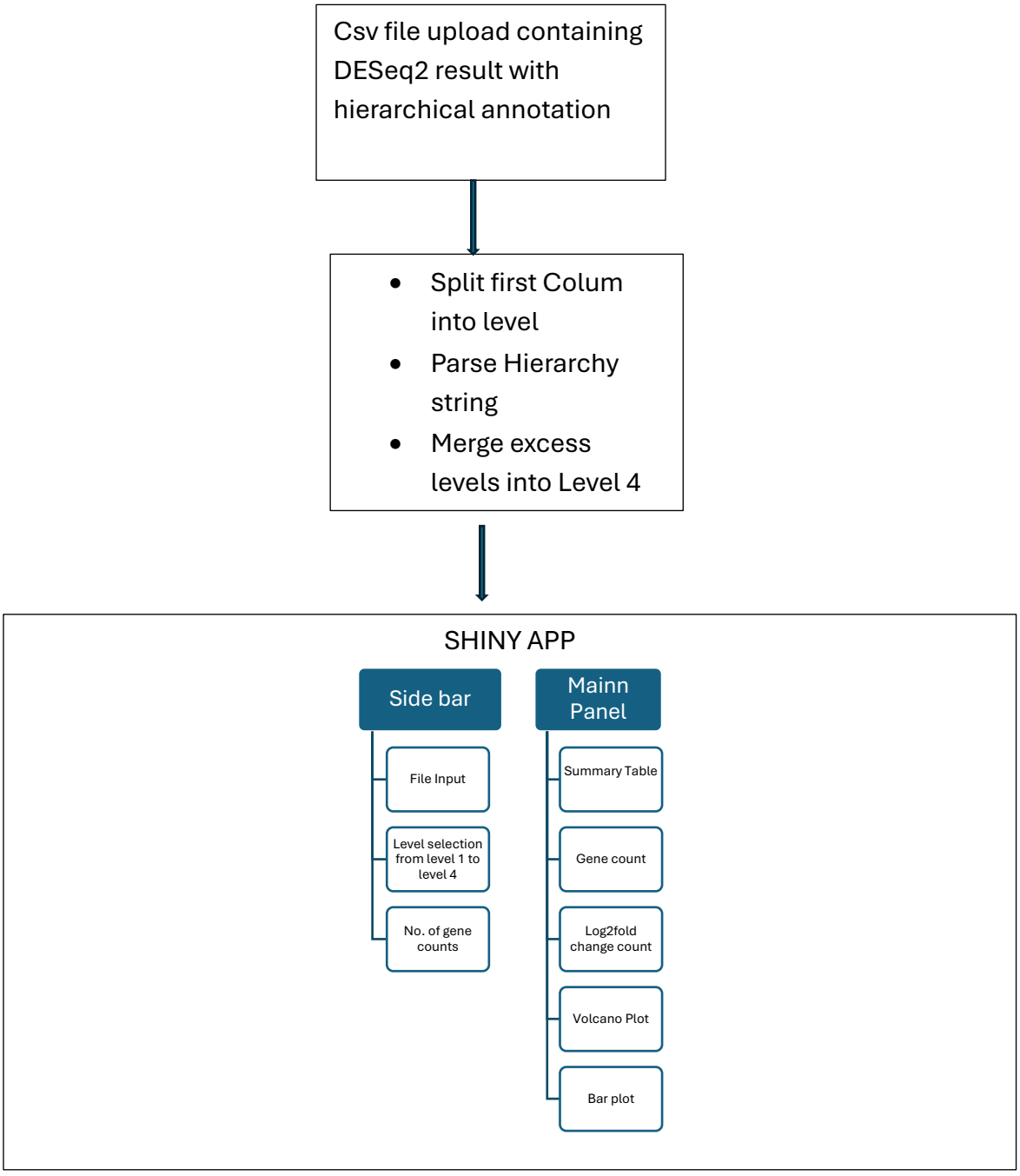


Figure 2.2 Workflow of shiny app for gene exploration

Figure 2.2 illustrates the complete workflow of the Shiny app developed in this study. The workflow is designed to visualise gene expression across multiple hierarchical levels. The workflow begins with the upload of a .csv file containing DESeq2 results with hierarchical annotations of genes. Next, the file is passed to a data processing module, which splits the first column into 4 levels (in the case of Subsystem SEED annotations). After the data is explorable through a dynamic filtering system, which provides options across levels 1 to 4. The output also displays a summary in a table format, allowing for sorting and searching through the results. A summary of the totals of upregulated and downregulated genes is visualised in a bar plot format. The application was built using R code and is deployable either locally or cloud-based via Shiny host platforms. This makes it more portable, reusable, and accessible without requiring programming skills.

CHAPTER 3: RESULT

3.1 Summary of Differentially Expressed Genes in the Testing DNA Dataset.

To understand the differences between the samples, differential gene expression analysis was performed using DESeq2 (M. Love et al., 2014). The DESeq2 dataset was created by the R script outlined in section 2.1.2. The provided DESeq2 R script in section 2.1.3 automatically identified and generated all the possible pairwise combinations of all six sample groups defined in the metadata file. With six sample groups analysed, a total of 15 pairwise comparisons are generated, along with the total numbers of differentially expressed genes listed in Table 3.1. Each sample provided in the metadata file was treated as a control (reference) and compared with each other for the respective comparisons. All the results were stored in the dedicated directory, created by section 2.1.3. All the DESeq2 results were filtered with an adjusted p-value (p-value <0.05) to get significantly differentially expressed genes between all the pairwise comparisons.

Table 3.1. Summary of all the possible pairwise comparisons generated between defined samples in the metadata file

NO	Treatment	Control (Reference)	Comparison	DEGs
1	PGMW5	Seep2	PGMW5 vs Seep2	6272
2	PGMW5	CT_Portal	PGMW5 vs CT_Portal	5734
3	PGMW5	Seep1	PGMW5 vs Seep1	4953
4	Seep1	CT_Portal	Seep1 vs CT_Portal	5953
5	Seep2	CT_Portal	Seep2 vs CT_Portal	5574
6	Seep2	Seep1	Seep2 vs Seep1	4889
7	VLF1	Seep2	VLF1 vs Seep2	6805
8	VLF1	Seep1	VLF1 vs Seep1	6046
9	VLF1	PGMW5	VLF1 vs PGMW5	5312
10	VLF1	CT_Portal	VLF1 vs CT_Portal	5087
11	VLF2	Seep1	VLF2 vs Seep1	6512
12	VLF2	PGMW5	VLF2 vs PGMW5	6474

13	VLF2	Seep2	VLF2 vs Seep2	6270
14	VLF2	CT_Portal	VLF2 vs CT_Portal	5474
15	VLF2	VLF1	VLF2 vs VLF1	4676

Table 3.1 illustrates the 15 pairwise comparisons along with the numbers of differentially expressed genes generated by methods 2.1.3. Across all comparisons, the number of significantly differentially expressed genes varied from 4,676 to 6,805 between the sample groups. These variable numbers indicate the biological differences among the pairwise comparisons.

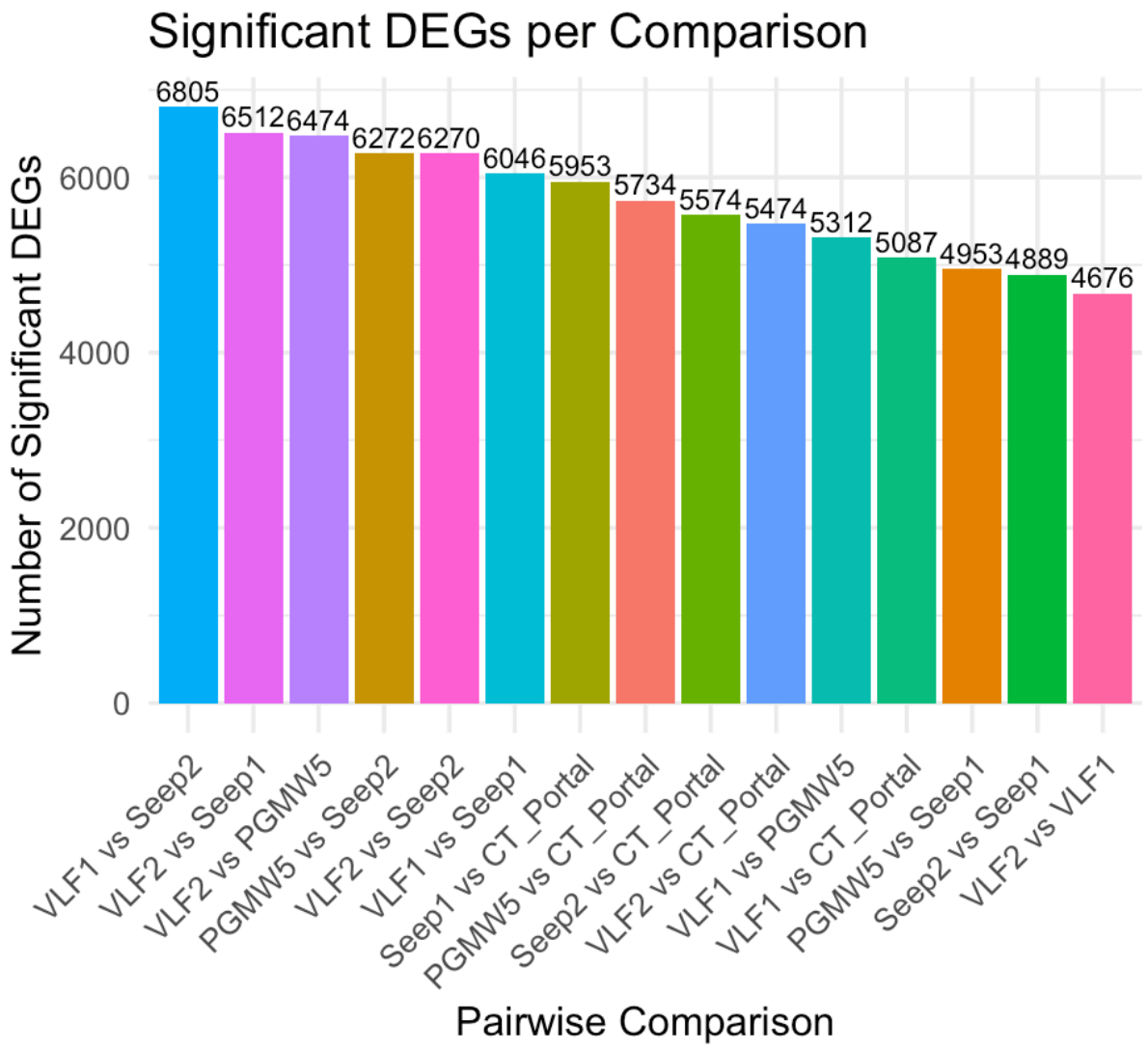


Figure 3.1. Bar plot showing the numbers of significantly differentially expressed genes across all 15 pairwise comparisons of six sample groups. The Y-axis represents the number of significantly differentially expressed genes, which were generated by filtering the DESeq2 result with an adjusted p-value of < 0.05. In contrast, the X-axis represents the name of the pairwise comparison, which was generated by a custom R programming coding script from section 2.1.3. The highest numbers of significantly DEGs were noticed in site VLF1 vs seep2, while the lowest numbers of significantly DEGs were noticed in site VLF2 vs VLF1. This figure provides a clear visualisation and comparison of the differentially expressed genes in all pairwise comparisons.

To visualise the overall trend, Figure 3.1 (bar graph) was generated to compare the number of significantly differentially expressed genes across all the comparisons. Figure 3.1 and Table 3.1 provide information about which comparison has the highest number of genes and which comparison has the lowest number of genes. Table 3.1 and Figure 1 clearly illustrate that comparison sites such as VLF1 vs seep2, VLF2 vs seep1 and VLF2 vs PGMW5 indicated the high DEG counts, while sites such as seep2 vs seep1 and VLF2 and VLF1 indicated the lower numbers of DEG compared to all other comparisons.

3.2 Key pairwise comparison.

According to section 3.1, we have noticed that some key pairwise comparisons showed some biologically significant patterns in gene expression change. These comparisons were selected based on the number of significantly differentially expressed genes. Some key pairwise comparisons are illustrated below.

3.2.1 VLF1 VS Seep2

Among all 15 comparisons, this comparison showed the highest numbers (6805) of significantly differentially expressed genes. In this case, as shown in Table 3.1.1, VLF1 was treated as the treatment, while seep2 was treated as the control (reference). Figure 3.2 provides a precise distribution of significantly differentially expressed genes based on a log2-fold change for level 1 functional categories. Figure 3.2 indicated that “Amino acids and derivatives”, “carbohydrates”, and “Cofactors, Vitamins, Prosthetic Groups, Pigments” were highly upregulated categories in treatment (VLF1), with DEG counts of 648, 718 and 524, respectively. This suggested increased metabolic activity. These categories also showed the downregulated genes. This indicated the balancing differential regulation within level 1’s subsystem. Some categories, such as DNA metabolism, phosphorus metabolism, and potassium

metabolism, showed bidirectional regulation. Figure 2 illustrates the functional regulation for level 1, where metabolic functions, such as amino acids and their derivatives, and carbohydrates showed the most significant upregulation in the treatment group.

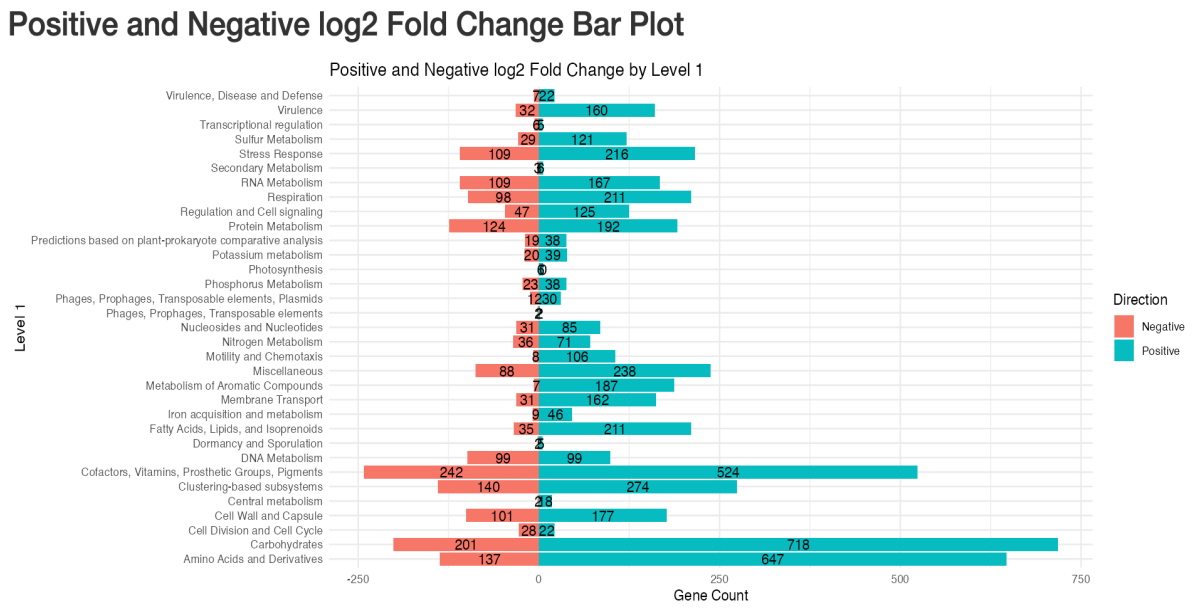


Figure 3.2. Distribution of differentially expressed genes based on log2 fold change for level 1 functional categories for VLF1 VS Seep2. Figure 2 illustrates the number of differentially expressed genes across the level 1 subsystem based on hierarchical annotated data. In this graph, the X-axis represents the number of genes, while the Y-axis represents the functional categories. Genes with positive log2 fold change value(log2fold change > 0) are shown in blue, illustrating the upregulation in treatment (VLF1), whereas genes with negative log2fold change value(log2fold change < 0) are shown in red, illustrating the downregulation in treatment (VLF1). Each bar represents the number of genes for a specific category. To emphasise the contrast and direction, negative values are reflected towards the left side, while positive values are reflected towards the right side. Gene-annotated hierarchical count data were analysed using a custom R coding-based Shiny application, which utilised DESeq2 to generate log2 fold changes between treatment and control conditions. This figure provides a clear visualisation and comparison of differentially expressed upregulated and downregulated genes in the dataset. This figure helps identify functionally enriched pathways or categories for prioritising targets for future investigation. Factional categories such as carbohydrates, amino Acids, and Derivatives are highly upregulated in treatment (VLF1), while cofactors, vitamins, prosthetic groups, and pigments are downregulated in treatment (VLF1).

3.2.2 VLF2 VS Seep1

This pairwise comparison generated the second-highest number of differentially expressed genes (6512). For this, VLF2 was treated as the treatment, and Seep1 was treated as the control (reference), as described in Table 3.1. Figure 3.3 provides a precise distribution of significantly differentially expressed genes based on a log2-fold change for level 1 functional categories. According to the figure, it was a clear indication that amino acids and derivatives, carbohydrate, clustering-based subsystem, cofactors, vitamins, prosthetic groups and pigments were highly upregulated in treatment(VLF2).

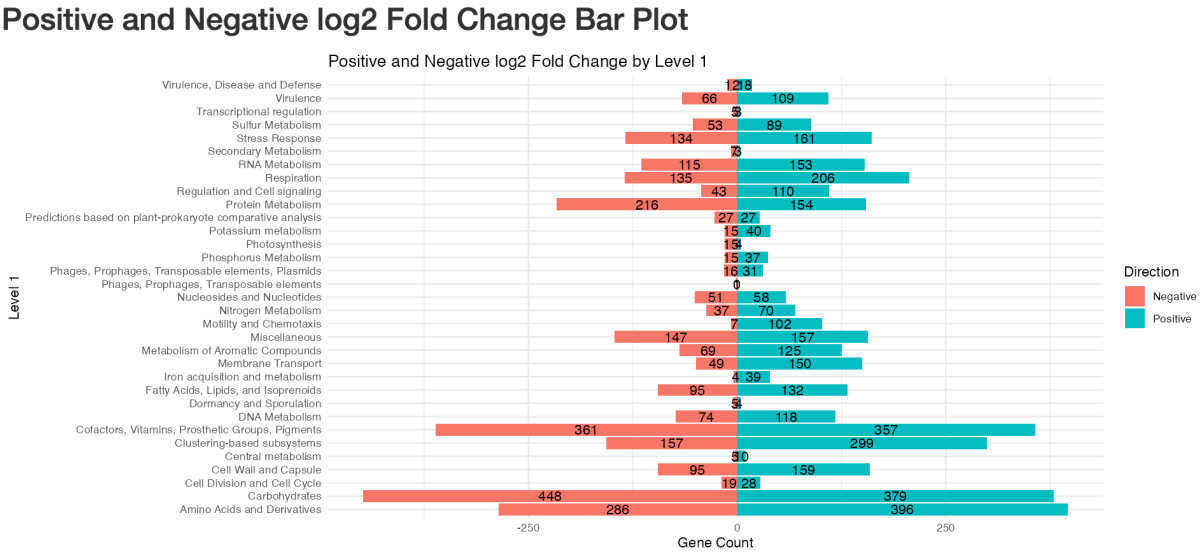


Figure 3.3. Distribution of differentially expressed genes based on log2 fold change for level 1 functional categories for VLF2VS seep1. Figure 3.3 illustrates the numbers of differentially expressed genes across the level 1 subsystem based on hierarchical annotated data. In this graph, the X-axis represents the number of genes, while the Y-axis represents the functional categories. Genes with positive log2 fold change value (log2fold change > 0) are shown in blue, illustrating the upregulation in treatment (VLF2), whereas genes with negative log2 fold change value (log2fold change< 0) are shown in red, illustrating the downregulation in treatment (VLF2). Each bar represents the number of genes for a specific category. To emphasise the contrast and direction, negative values are reflected towards the left side, while positive values are reflected towards the right side. Gene-annotated hierarchical count data were analysed using a custom R coding-based Shiny application, which utilised DESeq2 to generate log2 fold changes between treatment and control conditions. This figure provides a clear visualisation and comparison of differentially expressed upregulated and downregulated

genes in the dataset. This figure helps identify functionally enriched pathways or categories for prioritising targets for future investigation. Factional categories such as carbohydrates, amino Acids, and Derivatives are highly upregulated in treatment(VLF2), while carbohydrate pigments are downregulated in treatment(VLF2).

3.2.3 VLF2 vs PGMW5

This pairwise comparison generated the third-highest number of differentially expressed genes (6474). For this, VLF2 was treated as the treatment, and PGMW5 was treated as a control (reference), as described in Table 3.1. Figure 3.4 provides a precise distribution of significantly differentially expressed genes based on a log2-fold change for level 1 functional categories. According to the figure, it was a clear indication that amino acids and derivatives, and carbohydrates were highly downregulated in treatment (VLF2), with numbers of 364 and 532, respectively. However, the same categories were upregulated in treatment as well.

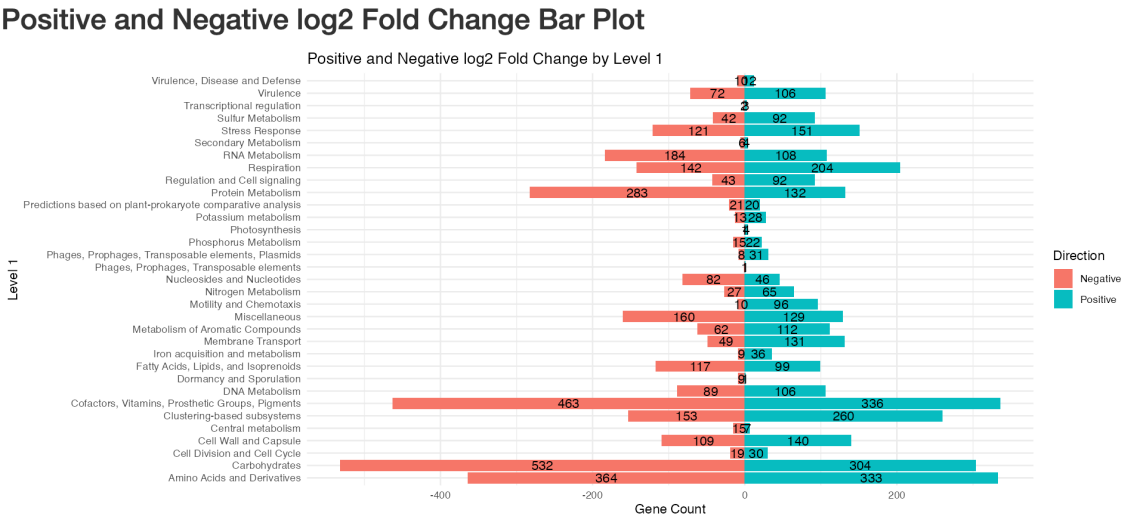


Figure 3.4. Distribution of differentially expressed genes based on log2 fold change for level 1 functional categories for VLF2VS PGMW5. Figure 3.4 illustrates the number of differentially expressed genes across the level 1 subsystem based on hierarchical annotated data. In this graph, the X-axis represents the number of genes, while the Y-axis represents the functional categories. Genes with positive log2 fold change value (log2 fold change > 0) are shown in blue, illustrating the upregulation in treatment (VLF2). In contrast, genes with negative log2 fold change value (log2 fold change < 0) are shown in red, illustrating the downregulation in treatment (VLF2). Each bar represents the number of genes for a specific category. To emphasise the contrast and direction, negative values are reflected towards the left

side, while positive values are reflected towards the right side. Gene-annotated hierarchical count data were analysed using a custom R-coding-based Shiny application, which utilised DESeq2 to generate log2 fold changes between treatment and control conditions. This figure provides a clear visualisation and comparison of differentially expressed upregulated and downregulated genes in the dataset. This figure helps identify functionally enriched pathways or categories for prioritising targets for future investigation. Factional categories such as carbohydrates, amino Acids, and Derivatives were highly downregulated in treatment (VLF2), while the same categories were upregulated in treatment (VLF2).

3.2.4 VLF2 VS VLF1

This pairwise comparison generated the lowest number of differentially expressed genes (4676). For this, VLF2 was treated as the treatment, and VLF1 was treated as the control (reference), as described in Table 3.1. Figure 3.5 provides a precise distribution of significantly differentially expressed genes based on a log2fold change for level 1 functional categories. According to the figure, it was a clear indication that all the categories were highly downregulated in treatment(VLF2).

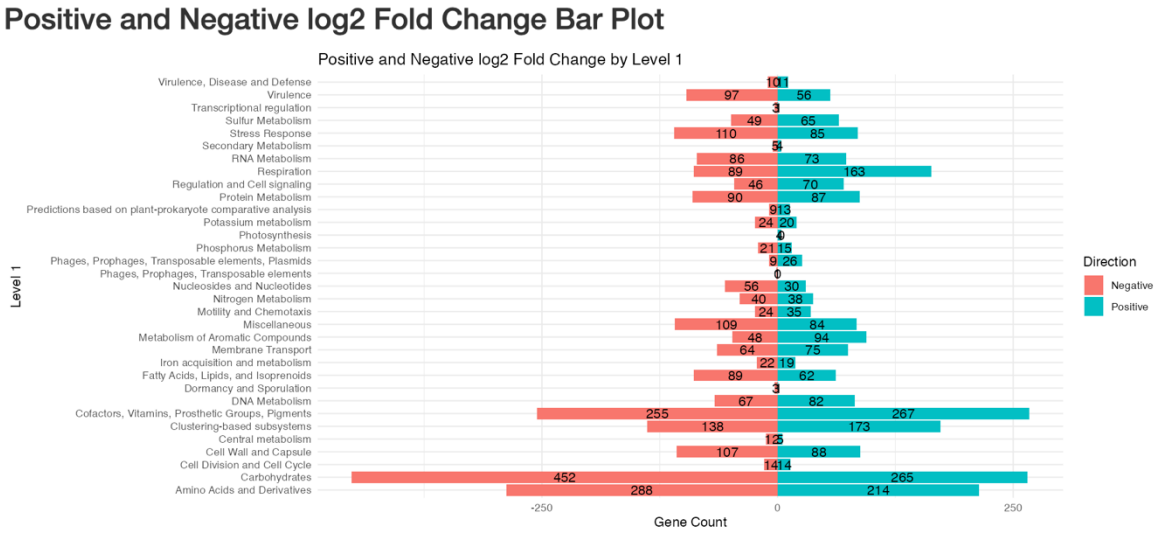


Figure 3.5. Distribution of differentially expressed genes based on log2 fold change for level 1 functional categories for VLF2VS VLF1. This figure 3.5 illustrate the numbers of differentially expressed genes across level1 subsystem based on hierarchical annotated data. In this graph, the X-axis represent the number of genes, while the Y-axis represent the functional

categories' names. Genes with positive log2 fold change value(log2 fold change > 0) are shown in blue, illustrating the upregulation in treatment. In contrast, genes with negative log2 fold change value(log2 fold change < 0) are shown in red, illustrating the downregulation in treatment. Each bar represents the number of genes for a specific category. To emphasise the contrast and direction, negative values are reflected towards the left side, while positive values are reflected towards the right side. Gene-annotated hierarchical count data were analysed using a custom R coding-based Shiny application, which utilised DESeq2 to generate log2 fold changes between treatment and control conditions. This figure provides a clear visualisation and comparison of differentially expressed upregulated and downregulated genes in the dataset. This figure assists in identifying functionally enriched pathways or categories for prioritising targets for future investigation. The majority of categories showed the downregulated trend in treatment(VLF2) across all level 1 categories.

3.3 Hierarchical Insights by workflow

In large-scale transcriptomics and metagenomics data, DESeq2 provides limited insights because it focuses only on one level of the dataset. In contrast, this shiny app workflow overcomes this issue by providing filtering options across the gene-annotated count data (SEED subsystems). This hierarchical filtering option enabled the discovery of biological insights, ranging from broad category-level trends to the specific functions of genes.

Across all the key pairwise comparisons from figures 3.2 to 3.5, it is evident that “Amino Acids and Derivatives” and “Carbohydrates” were consistently dominant categories in differentially expressed gene analysis, either as upregulated or downregulated. It was a strong indication that these two categories were involved in microbial activity.

3.4 Shiny App workflow- walkthrough.

In this study, an improved workflow was created by custom R coding scripts provided in section 2.2. This was the critical part of the novel pipeline because it was the interface for the user to explore and interpret the DESeq2 results with SEED subsystem annotations. This workflow started with the file uploading process.

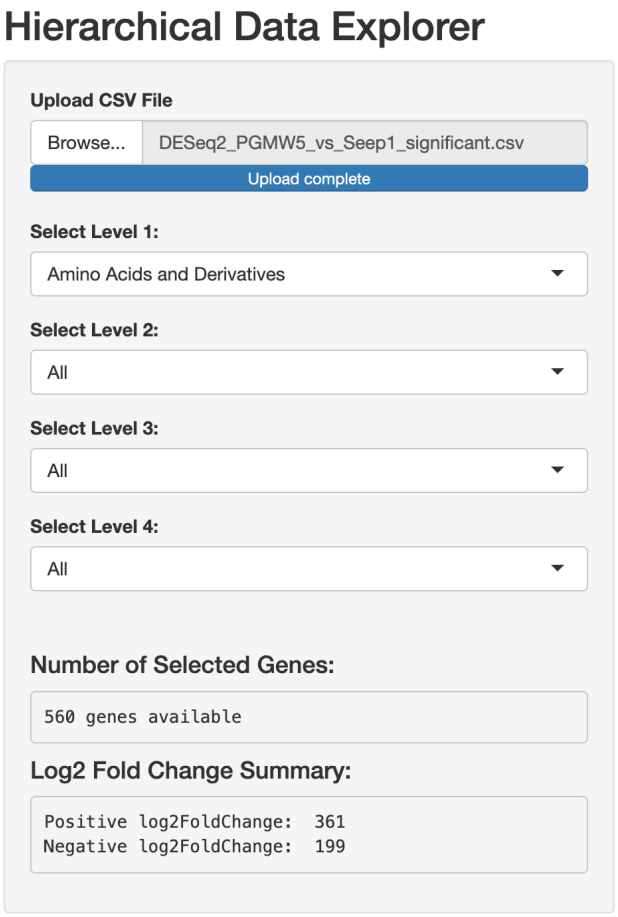


Figure 3.6 Side bar panel for hierarchical data exploration. Figure 3.6 illustrates the sidebar panel of the Shiny application workflow. Here, a dedicated section was provided for data upload, which can only accept data in .csv file format. Additionally, the drop-down options provided across Levels 1 to 4 enable users to select or refine their results according to their level of interest. After selecting the interested level, the summary output, such as the total number of selected genes and numbers of upregulated and downregulated genes based on the log2fold change value, is displayed

The workflow begins by uploading the file. This contains the differentially expressed genes annotated with SEED level 1 through level 4 subsystems, and upon uploading the file, the data processing function processed the data.

3.4.1 Filtering option

One of the most prominent features of this pipeline is the hierarchical filtering options, which were generated by the selectInput() and renderUI() functions. In Figure 3.6, this function is illustrated by the names from level 1 to level 4. As shown in Figure 3.6, for example, users may

select “Amino Acids and Derivatives” as a level 1 broad category. After that, based on the selected level 1, a second dropdown for level 2 was automatically filtered using the renderUI() function and showed relevant subsystem categories within “Amino Acids and Derivatives”. This logic continues to levels 3 and 4.

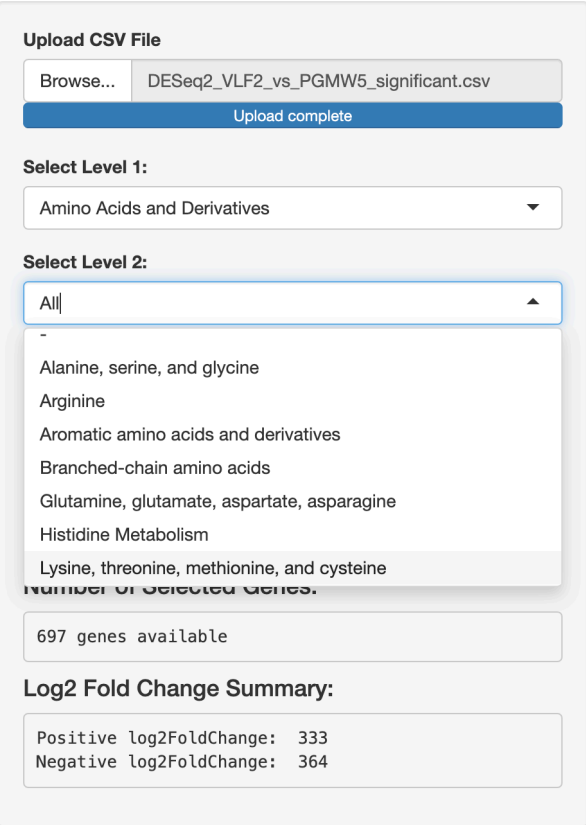


Figure 3.7: How the subsystems filtering options work across the levels. Figure 3.7 illustrates the power of the filtering option across the subsystem levels.

3.4.2 Example of filtering options.

Let’s explore the power of filtering options with an example. According to Figure 3.2, the user has already selected "Amino Acids and Derivatives" as the level 1 category. It is also clearly shown that the relevant subsystem option for the Level 2 category is instantly populated, as shown in Figure 3.9 (A). If the user selected “Branched chain amino acids” as a level 2 category, then the total number of selected genes was updated automatically, along with a summary of the log2 fold change, which provided the direction of gene expression. A similar logic is applied to the entire set of filtering options and across all subsystem levels. This has several advantages, first and foremost is that every change in selection updates the numbers of selected genes, the volcano plot, and the data tables.

3.4.3 Visualisation- Real-time filtering and Update.

According to the user's selected options, the visualisations are updated. As shown in Figures 3.7 and 3.8, the total number of genes and Log2fold change summaries are updated according to the selection. Like visualisation, such as volcano plots and interactive data tables, were changing according to the selection. Figures 3.9 and 3.10 provide a clear illustration of how the volcano plot has altered according to the user's selection. According to Figure 3.9 (A), the user has selected only the level one broad category, and a volcano plot was generated (Figure 3.9 (B)) according to this selection. However, according to Figure 3.10 (A), the user has selected the level 2 category along with the level 1; therefore, the volcano plot is updated according to the selection (Figure 3.10 (B)). This dynamic approach of updating the volcano plot based on the hierarchical user's selection allows the user to enhance the interpretability. Because users can visualise the data or gene expression from broad categories to specific pathways

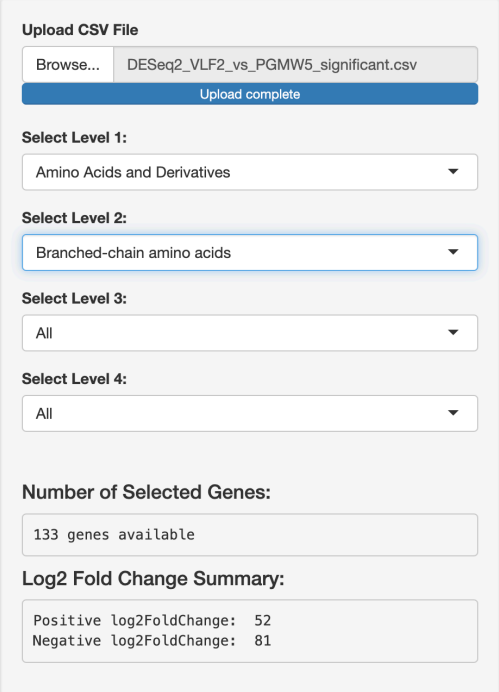
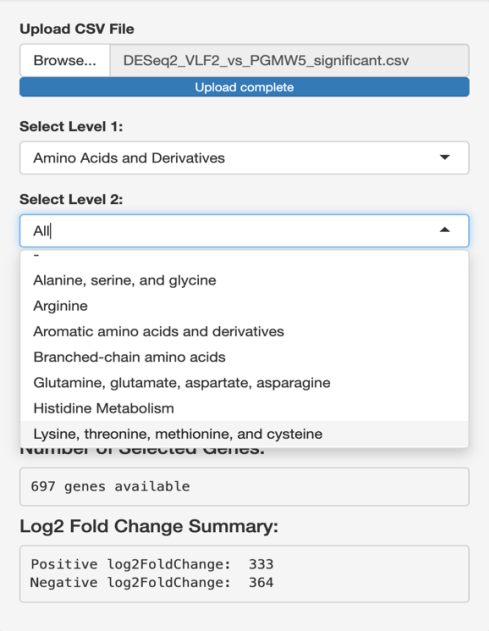
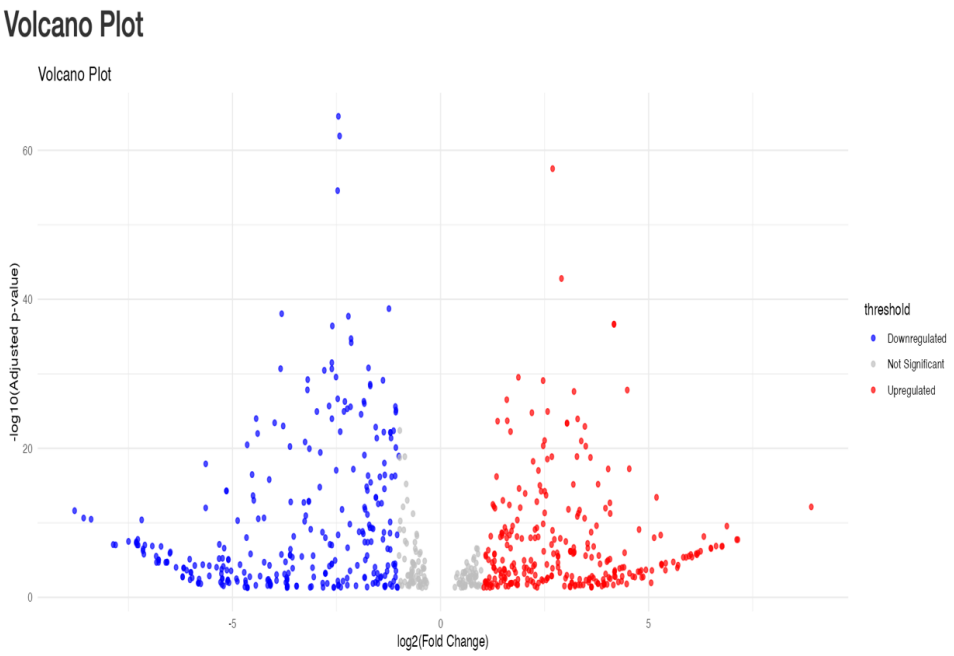


Figure 3.8 Real time filtering and updating. This figure illustrates the how numbers of differentially expressed genes were updating after user's selection. It is clear from figure 3.7 and 3.8 how real time filtering options updating the numbers of gene counts.



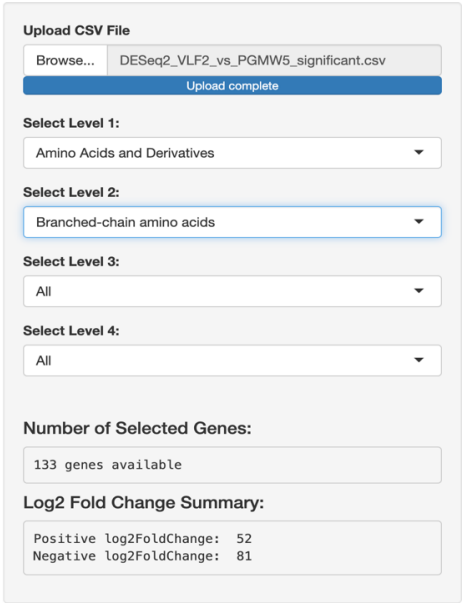
A) Sidebar panel



B) Volcano Plot.

Figure 3.9. Real-time filtering and updating of the volcano plot. Figure 3.9 (A) illustrates the sidebar panel of the workflow, where the user selected "Amino Acids and Derivatives" as the level 1 category. Figure 3.9 (B) demonstrates the volcano plot, which was automatically generated by an R programming script based on the selection. In the volcano plot, the X-axis represents the log2fold change, while the Y-axis represents the -log10 adjusted p-value. The

genes were expressed according to the method in section 2.2.3. This plot was updated according to the user's selection and includes all significant (red and blue dots) and non-significant (grey dots) DEGs in each group for all Amino Acid and Derivatives functions.



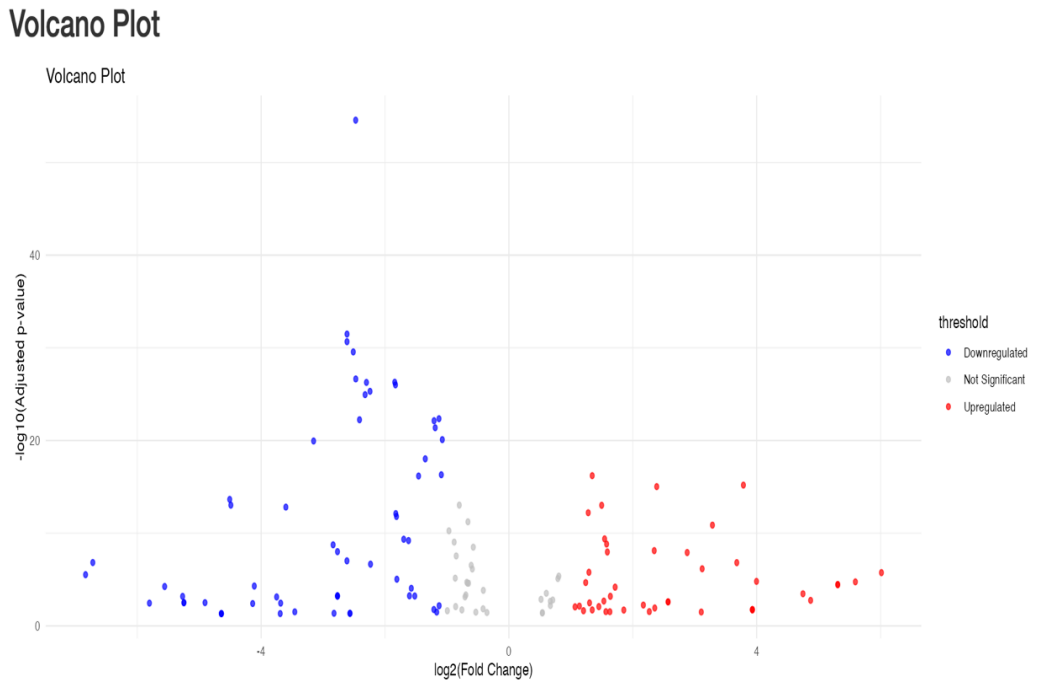
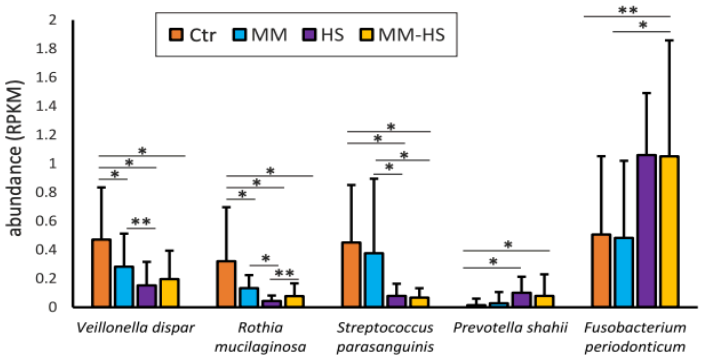


Figure 3.10. Real-time updating of the volcano plot after the user's selection. Figure 3.10 (A) illustrates the sidebar panel of the workflow, in which the user selected "Amino acids and Derivates" as the level 1 category and "Branched chain amino acids" as the level 2 category. Figure 3.10 (B) illustrates the volcano plot, which was automatically generated by an R programming script according to the selection. In the volcano plot, the X-axis represents the log2fold change, while the Y-axis represents the -log10 adjusted p-value. The genes were expressed according to the method in section 2.2.3. This plot was updated according to the user's selection and includes all significant (red and blue dots) and non-significant (grey dots) DEGs in each group for only "branched chain amino acids" functions within "Amino Acid and Derivatives".

Hierarchical analysis plays a crucial role in the functional interpretation of complex and large biological datasets. Instead of examining gene expression or abundance at an individual level alone, hierarchical analysis offers several advantages. First and foremost, it enhances biological insights by allowing researchers to observe how differentially expressed or abundant genes contribute to larger biological pathways or processes. Another benefit is that it can simplify the complex data instead of dealing with thousands of genes; hierarchical analysis reduces the complexity by summarising the analysis at different functional levels. It is also helpful for hypothesis generation and enhancing the visualisation.

3.5 Benchmark and validation

To evaluate the accuracy, we validate our pipeline's results with the existing published results as per Section 2.3. This study was selected because it utilised the same primary differential expression algorithm (DESeq2) and provided raw count data in supplementary data, and was published in a high-tier journal. In this study, the authors utilised 83 individual samples and distributed them into 4 major groups: control (healthy), MM (Methyl mercaptan), HS (Hydrogen Sulphide), and MM-HS. We have downloaded the primary input from this study and converted it into a .csv format. Secondary metadata input was created based on the information in the manuscript's supplementary data. The provided DESEq2 R script in Section 2.1.3 automatically identified and generated all possible pairwise combinations of the samples defined in the metadata file. All the pairwise comparisons are outlined in Table 3.2. The differential gene expression patterns were comparable to the published results, with a 90% similarity rate (Figure 3.11). That indicated that our pipeline produces accurate results.



A

Taxa Tested	HS vs Health Comparison Result	MM vs Health Comparison Result	MM.HS vs Health Comparison Result	MM vs HS Comparison Result	MM.HS vs HS Comparison Result	MM.HS vs MM Comparison Result
Veillonella dispar	Up in health*	Up in health*	Up in health*	Up in MM*	Up in MM.HS	Up in MM
Rothia mucilaginosa	Up in health*	Up in health*	Up in health*	Up in MM**	Up in MM.HS**	Up in MM.HS
Streptococcus parasangunis	Up in health*	Up in health	Up in health*	Up in MM*	Up in MM.HS	Up in MM*
Prevotella Shahii	Up in HS*	Up in MM	Up in MM.HS*	UP IN HS	Up in MM.HS	Up in MM.HS**
Fusobacterium peridonticum	Up in HS	Up in health	Up in MM.HS**	UP IN HS	Up in MM.HS	Up in MM.HS*

B

Figure 3.11 Comparison of results with the existing dataset. Figure 3.11 illustrates the validation of the pipeline by comparing its results with those from existing published studies. A) represented the existing study result, which the authors generated after running DESeq2. The asterisks (*) represent the level of significance for each statistical comparison, with * representing p>0.05 and ** representing p>0.1, as per the authors of the manuscript. B) Results generated by our pipeline. The asterisks (*) represent the level of significance found in our analysis. Black-coloured asterisks indicate that the same level of significance was seen as in the published study, while a red asterisk indicates a different level of significance was found.

Our results had a comparable rate of 90% (27/30 correct significance values), matching the published study.

Table 3.2 All possible pairwise comparisons generated by our pipeline for the published dataset

treatment	control	comparison
HS	Health	HS vs Health
MM	Health	MM vs Health
MM.HS	Health	MM.HS vs Health
MM	HS	MM vs HS
MM.HS	HS	MM.HS vs HS
MM.HS	MM	MM.HS vs MM

Table 3.2 illustrates all possible pairwise comparisons generated through our pipeline. Here, 4 different samples were defined in the metadata; therefore, six different pairwise comparisons were generated.

3.6 Accessibility for all researchers.

This novel pipeline is helpful for every researcher; anyone can utilise this without knowledge of programming skills. It has a simple and easy-to-use layout that leads users through a logical flow, including data import, subsystem filtering, data visualisation, and interpretation. By combining automation, a user-friendly dashboard, and filtering options, this pipeline enhances the analysis of differentially expressed gene expression for gene-annotated hierarchical count data.

CHAPTER 4: DISCUSSION

4.1 Overview of study

The rapid development of sequencing technology has created a huge amount of sequencing data (including gene expression data) that allows researchers to explore complex biological information in greater resolution (Satam et al., 2023). Although data analysis tools are available to perform differential gene expression analysis, they often require manual coding, which can be a time-consuming and frustrating process. Apart from this, all these tools produce long lists of statistically significantly changed genes that may be important but very hard to interpret holistically, especially for those who do not have solid profound knowledge of bioinformatics or an existing hypothesis to test (i.e., they need to explore their data for unknown linkages or results from treatment vs. control studies). This is compounded by a lack of interactive visualisation and real-time filtering for hierarchical biological annotations, such as Subsystem SEED annotated data, as well as taxonomic strings. This study introduces a novel bioinformatics workflow aimed at addressing this gap. The primary objective was to develop a bioinformatics pipeline that automates differential gene expression analysis. The novelty of this pipeline lies in its ability to generate all possible pairwise comparisons based on the provided metadata and output them in a parsable file that can be sorted interactively. The R shiny workflow fulfils this visualisation of SEED annotated hierarchical count data with an interactive filtering option ranging from level 1 to level 4. The workflow comprises several pivotal phases. It begins by generating all possible pairwise comparisons and then filtering the output gene table based on adjusted p-values to obtain statistically significant results. Next, parsing a hierarchical gene's function based on the semicolon. Then, visualisation and interactive exploration of results with dynamic filtering options in the Shiny R environment. With this pipeline, we have analysed 6 different samples in a test dataset (VLF1, VLF2, Seep1, Seep2, PGMW5 and CT_Portal) and this pipeline generated 15 pairwise comparisons. Thousands of differentially expressed genes were illustrated, and comparisons were made to identify which groups had the highest numbers of differentially expressed genes and which comparisons had the lowest. One of the centres of innovation is the real-time interactive filtering data table, bar plots, and volcano plots created by this pipeline, which provide clear visualisations of the distribution of differentially expressed genes.

4.2 Differential expression across the test samples

The result of differential gene expression over 15 pairwise comparisons gave evidence on great diversity of microbial activity as numbers of differentially expressed genes ranging from 6805 to 4676 as shown in table 3.1 and figure 3. 1. The contrasts were designed to for comparison

of gene expression between all the possible pairwise comparisons defined in the metadata. As a result, this purpose provided the biological significance to compare specific enriched functions and metabolic patterns.

Out of 15 pairwise comparisons, the VLF1 vs Seep2 group comparison had the highest number of differentially expressed gene counts. VLF1 served as the treatment while Seep2 served as the control, in this instance. Figure 3.2 demonstrates the distribution of differential gene expression at the SEED level 1 categories, including amino acids and Derivatives (648 genes), carbohydrates (718 genes), and cofactors, vitamins, and pigments (524 genes), which were notably upregulated in the treatment (VLF1). These results indicate that in the treatment group, these pathways were essential for basic metabolic and biosynthetic activity. However, at the same time, the same level 1 categories also illustrated the strong downregulation for the treatment condition. This indicated the balancing regulation. This expression indicated that some genes were activated while others were downregulated to maintain effective metabolism. This mixed expression represents the adaptation in cellular response.

The second-highest number of differentially expressed genes was observed in VLF2 versus Seep1, with 6,512 differentially expressed genes. Figure 3.3 demonstrates the distribution of differential gene expression at SEED level 1 categories. The same pattern of upregulation was observed as categories observed for VLF1 VS Seep2, along with that clustering-based subsystem was observed for upregulation. Clustering-based subsystems with highly upregulated means in treatment means genes were typically poorly characterised and often involved in a shared pattern of expression, even if the exact role of the gene was not fully defined. Highly downregulation was observed for carbohydrate in the treatment due to the switch to an alternative energy source, which means energy is diverted from protein synthesis.

The third most highlighted pairwise comparison was VLF2 vs PGMW5, with 6474 differentially expressed genes. This comparison has also shown the mixed regulation approach. As shown in Figure 3.4, it was clear that Amino acids and their derivatives, as well as carbohydrates, were downregulated. However, the same categories also showed an upregulation trend, revealing a dual regulation pattern. It was indicated that the treatment group may facilitate the increased metabolic activity.

The VLF1 vs. VLF2 comparison showed the lowest number of differentially expressed genes. As shown in Figure 3.5, this comparison reveals a majority of the downregulation trend compared to the controlled condition (VLF2). Lower expression of amino acids and

derivatives, carbohydrates, and other pathways indicated that VLF1 supports the view of stress under the treatment condition.

Across all the comparisons. We have noticed that several conditions, including Amino Acids and Derivatives, Carbohydrates, Cofactors, Vitamins, and Pigments, have shown a repeated contribution in differential gene expression analysis. These standard categories were fundamental for the survival and adaptation of microbes. Apart from this, the observation that both upregulated and downregulated genes are observed for all the comparisons reflects the complex biological pattern. This study can identify and interpret patterns based on adjusted p-values and log2fold change values. Additionally, real-time filtering options within the Shiny workflow facilitate the easy detection of patterns and understanding of biological differences across all pairwise comparisons.

4.3 Hierarchical data explorer

The hierarchical gene annotation count data were generated using the SEED subsystem. This annotation allowed functional trend to organise from border category (level 1) to specific pathway(level 2-4). This approach would be essential for understanding the distribution of differentially expressed genes from the border category to a specific pathway. We have already analyzed the 15 pairwise comparisons in section 3.2 and discussed the common expressed categories in section 4.3. The majority of them showed the bidirectional regulation. Therefore, the SEED subsystem approach helps to get a clearer understanding of the regulation. According to Section 2.1 of the method, the SEED annotated gene count data is parsed into levels 1 to 4. For example, at level 1 for Amino acids and derivatives, the results demonstrate the mixed approach as outlined in section 3.2. It would remain the same for the level 2 category, such as "Branched Chain Amino Acid Biosynthesis". However, with further filtering options at level 3 or level 4, the specific direction of expression is disclosed. For example, at level 4, the Branched-Chain acyl-CoA dehydrogenase (EC 1.3.99.12) enzyme showed an upregulated trend in the treatment. These enzymes play an essential role in the catabolism of branched-chain amino acids. This hierarchical parsing structure enables researchers to understand the role of gene expression not only at the metabolic level but also to comprehend the specific reactions within the pathway.

4.4 Role of filtering options and visualisation

One of the significant contributions of this study was the development and integration of a shiny dashboard for filtering and data visualisation. These options significantly enhance the

accessibility and usability of this pipeline. It allows researchers to study and interpret complex datasets without requiring programming and bioinformatics skills. Traditional approaches rely on statistical differential gene expression tables. From this table, it isn't easy to understand and interpret the results, especially for those with a non-bioinformatics background. Because the raw DESeq2 result table contains the list of expressed genes, log2 fold change, p-value, and adjusted p-value, it lacks complex biological information. To address these issues, a streamlined workflow was created to interpret and organise the DESeq2 output file with SEED hierarchical annotated gene count data, featuring real-time filtering and visualisation. It expressed the genes across multiple levels (from level 1 to level 4). For example, sections 3.5.2 and 3.5.3 briefly illustrate the mechanism of filtering options. The user is interested in Amino acids and derivatives, as selected from the Level 1 dropdown menu. The volcano plot was generated based on level 1 selection. After selecting these options, the workflow automatically populated the level 2 categories. After selecting from the Level 2 list, the number of genes and the volcano plot were updated, as shown in Figure 3.10. This live filtering is not helpful for the immediate visualised understanding, but also enhances the interpretation of gene expression (RNA) or gene abundance for DNA. The volcano plot immediately reflects the upregulated or downregulated genes.

Overall, this streamlined workflow is beneficial for every researcher, regardless of their programming skills, as it is simple to use by uploading a file and selecting options from a drop-down menu in the side panel. It facilitates complex data analysis and interpretation within a few clicks. Additionally, it could be helpful for teaching purposes to demonstrate the gene expression or gene abundance related to specific biological functions. Additionally, it could facilitate comparisons of metabolic responses across different samples. For reproducibility, the workflow ensured the consistent filtering, graph plotting and interpretation. This streamlined workflow could be readily useful for future studies by simply uploading the appropriate input.

4.5 Comparing with existing findings and pipeline.

Due to the ongoing advancement of technology in bioinformatics, the field is growing rapidly. This section compares our novel pipeline with several widely used tools.

4.5.1 Comparing with shiny-seq.

Shiny-seq was established in 2019 to study RNA sequencing data by utilising a wide range of tools and algorithms (Sundararajan et al., 2019). Shiny-seq relies on classical databases, such as KEGG and GO, for functional enrichment analysis, but lacks hierarchical annotated data

analysis. Our pipeline addressed this gap by incorporating multi-level biological interpretation. This allows users to trace the biological information from broader categories to specific pathways, such as from Level 1 to Level 4. Apart from this, shiny-seq does not provide real-time filtering options like our novel pipeline does for visualisation output, for example, filtering dynamically using the drop-down menu from the side panel and immediately updating the volcano plot and numbers of differentially expressed genes (as well as non-significant comparisons, which are displayed as grey dots). Additionally, our pipeline can generate all possible pairwise comparisons as defined in the metadata.

4.5.2 Comparing with RNA-flow

RNAflow is a complex, robust and reproducible RNA sequence pipeline, implemented using the Nextflow workflow management system. This pipeline is promising for preprocessing data, alignment, and detecting differentially expressed genes; thus, it provides valuable preprocessing of raw sequence data that falls outside the scope of our project. However, it provides limited downstream analysis (Lataretu & Hölzer, 2020). While our shiny-based pipeline enhances the multi-layer biological interpretability and interactive data exploration with real-time filtering options for visualisation. In the RNAflow pipeline, knowledge of different file formats is necessary because each step requires uploading a new file format to process further. While our pipeline does not require more file formats, it only requires one. CSV file format, which was already generated and stored by our pipeline. Another current issue with RNAflow is that the installation and execution of the bioinformatics workflow operate within the Nextflow framework, whereas our pipeline features a simple, user-friendly dashboard. However, output from RNAflow could be adapted to our pipeline.

4.5.3 Comparing with VIPER

VIPER is a robust bioinformatics pipeline for RNA sequence analysis, which is implemented using the Snakemake workflow management system (Cornwell et al., 2018). This pipeline provides end-to-end RNA sequence analysis by utilising multiple tools, including DESeq2, gene set enrichment analysis, KEGG, and STAR-fusion. It is primarily optimised for human and mouse transcriptomics data. However, it could not provide hierarchical subsystem parsing and real-time filtering across multi-level gene functional level hierarchical count data. Our pipeline fills the gap by incorporating real-time filtering options across all hierarchical levels. VIPER's output is based on the input and configuration, whereas in our pipeline, users can dynamically filter and change the output using a simple interactive dropdown menu. VIPER performs differential gene expression analysis for only one comparison. In contrast, our novel

bioinformatics pipeline can generate all possible pairwise comparisons as defined in the metadata and perform differential gene expression analysis for all pairwise comparisons. The results are stored automatically in a dedicated output directory in the form of a .csv table. This would require several extra steps in VIPER.

4.5.4 Comparing with TRAPLINE

TRAPLINE is an automated RNA sequence analysis pipeline, implemented within the Galaxy environment (Wolfien et al., 2016). It is designed to provide reproducible and automated results. This pipeline can perform preprocessing, quality control, read alignment, differential gene expression analysis, single-nucleotide polymorphism detection, and protein-protein interaction analysis, thus encompassing utilities like RNA-Seq. However, like all other pipelines, it cannot support hierarchical functional annotation analysis, such as data generated from the SEED subsystem. Another significant comparison is accessibility, as TRAPLINE requires the Galax system to run the analysis, which can be challenging during the installation process. In contrast, our pipeline features a user-friendly dashboard that allows for easy analysis and interpretation of results, facilitated by real-time filtering options. Overall, TRAPLINE is highly valuable for detecting single-nucleotide polymorphisms and protein-protein interactions, while our pipeline is crucial for analysing differential gene expression or abundance.

Overall, all the above-mentioned pipelines were proficient for RNA-sequence analysis, starting with raw sequence files; however, they all show gaps in end-user data exploration, which are addressed by our novel bioinformatics workflow. The most critical gap was the inability to perform differential gene expression analysis for functional hierarchical gene count data. Apart from this, limited support for multiple comparisons was also a critical gap. Because tools like DESEq2 provide multiple pairwise comparison analyses, but only after script writing, it may be too complicated for those without profound knowledge in bioinformatics and programming skills. Lack of real-time filtering options was also highlighted as a critical gap. By incorporating real-time filtering options and hierarchical parsing, we were addressing this gap. Also, a Shiny dashboard is provided for data exploration and data interpretation. Our tool is a valuable “next step” after the tools mentioned above, thus it does not serve to replace these existing tools but complements them, as expanded on below.

4.6 Strengths and Limitations.

The development of an automated hierarchical differential gene expression bioinformatic pipeline was a critical requirement to address the aforementioned research gap. It is necessary to create an automated, interpretable and real-time filtering pipeline for sequencing data. This section provides an overview of the strengths and limitations of this novel bioinformatics pipeline.

4.6.1 Strength.

Overall, our pipeline has several strengths. Let's start with the most important one, which was the automated generation of all possible pairwise comparisons according to the metadata file. As per Section 2.1.3 of the method, it utilises the combn() R programming function, which identifies the different samples and generates all possible pairwise comparisons for data analysis. For example, we had 6 different samples defined in the test metadata; therefore, 15 pairwise comparisons were generated. Each generated pair is subjected to DESeq2's result and stored in the dedicated directory. All results were stored after being filtered by adjusted p-value less than 0.05. This automation saves time, ensures consistency (for example, a 90% similarity rate to a previous study in terms of replicating significance level), and minimises human error while covering all necessary biological information by comparing all possible pairwise comparisons. Another strength was a streamlined workflow, which was portable, scalable, reproducible, and easy to use, featuring a real-time filtering option for data exploration. Here, the user needs to upload the .csv file containing DESeq2's results, which were generated by this pipeline. As per Section 2.2.2 of the method, the entire interface was created using the fluidPage () and sliderLayout() R programming commands, with SelectInput() and renderUI() ensuring that results dynamically update and support filtering options. This strength has made this pipeline more accessible to both bioinformaticians and non-bioinformaticians. We also utilise SEED subsystems' functional annotated hierarchical data. This structure is best suited for filtering options, as researchers could explore the data from multiple levels. The user-friendly dashboard, featuring dropdown filtering options, made this pipeline accessible. This shiny dashboard also provides visualisations. This visualisation output provides the immediate illustration of which genes were categorised as upregulated or downregulated, which expands data exploration when hypothesis generation is required.

This pipeline, however, has several limitations. A key limitation of this pipeline was the data input format, as it was dependent on the function-annotated dataset generated from SEED subsystems. Using other data may take manual pre-processing to convert to a semicolon-

separated string of hierarchical data; however, many genomic data types (such as microbial taxonomy) are output in this format by default. Currently, the pipeline is optimised for all possible pairwise comparisons by using DEseq2's model. The crucial limitation was that users had to compare each comparison individually by uploading it one by one to the Shiny app workflow. It cannot compare every pairwise comparison at once. Another limitation is that this pipeline currently provides gene expression or abundance trends at all levels (from level 1 to level 4). However, it does not support functional enrichment analysis.

Despite providing valuable functional insights, the study's reliance on functional annotations has inherent limitations. Functional annotation approaches, such as the SEED subsystem hierarchy, categorise genes by their biological roles but abstract away from their taxonomic origins. Consequently, distinct microbial communities with varying species compositions may exhibit similar functional profiles. This functional redundancy means that even if the species diversity across samples is markedly different, the dominant pathways identified through functional enrichment may appear identical. Such limitations underscore the need to interpret functional analyses in conjunction with taxonomic composition data to gain a comprehensive understanding of microbial community structure and function.

4.7 Future direction

The bioinformatics field is experiencing rapid growth, particularly in the areas of transcriptomics and metagenomics. This section provides an overview of the future direction for extended application development and increased scalability.

4.7.1 Integration with functional enhancement databases.

As discussed in section 4.6, the current pipeline does not support the functional enrichment analysis. Functional enrichment analysis is a statistical method used in bioinformatics that provides information about which specific biological functions, pathways or categories are overrepresented in a set of genes. Therefore, it is crucial to integrate with functional databases such as Gene Ontology (GO) and KEGG. This integration would be helpful to perform the statistical enrichment test. It would provide a better understanding of functional enrichment for hierarchical SEED subsystem-based annotated genes. After adding this into the current pipeline, the user can identify the functional property of differentially expressed genes.

4.7.2 Incorporating a Machine learning model for classification and prediction

Another potential future direction would be to incorporate a machine learning model into the pipeline. As per the current stage, this pipeline can identify and visualise differentially

expressed genes across hierarchical SEED subsystem annotated gene count data. Machine learning models can be built and trained to predict and classify these findings, providing a quicker biological understanding related to controls and treatments (Pirooznia et al., 2008).ML model such as random forest and support vector machines can be trained on the differentially expressed genes' profile to classify samples into their relative environment. It would lead towards biomarker discovery.

4.7.3 Widespread availability.

To increase the accessibility of this pipeline, widespread availability is mandatory. This should be hosted on a platform such as Shinyapps.io, GitHub, or any Docker, so that anyone can utilise this pipeline from any browser.

4.8 Conclusion

This study successfully developed a novel bioinformatics workflow that can perform automated differential gene expression analysis between all possible pairwise comparisons defined in the input metadata file. Additionally, a R shiny workflow was implemented for interactive visualisation of SEED annotated hierarchical count data, featuring interactive filtering options ranging from level 1 to level 4, as well as clear visualisation of hierarchical genomic data. Overall, this study provides a comprehensive solution for exploring transcriptomics and metagenomics data, which can be integrated after pre-processing and gene annotation steps. This pipeline achieves dual strength, solid automated analysis power, and biological interpretability. One of the key achievements of this study was the computerised generation of all possible pairwise comparisons for differential gene expression. This part eliminates manual scripting. Apart from this, the real-time interactive filtering options enable users to trace expression from a broader biological category to a specific pathway or function of the genes. This filter, facilitated by the shiny dashboard, represents the major accessibility by just selecting options from the drop-down menu. Filtering options dynamically re-update the volcano plot and the number of differentially expressed genes according to the user's selection. By addressing critical gaps, such as the need for a real-time filtering option and a lack of hierarchical analysis, this pipeline provides gene expression or abundance analysis and improved visualisation output.

References:

Abdurakhmonov, I. Y. (2016). *Bioinformatics: basics, development, and future*. InTech Rijeka. https://doi.org/http://dx.doi.org/10.5772/61421

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, *215*(3), 403-410. https://doi.org/https://doi.org/10.1016/S0022-2836(05)80360-2

Berardini, T. Z., Mundodi, S., Reiser, L., Huala, E., Garcia-Hernandez, M., Zhang, P., Mueller, L. A., Yoon, J., Doyle, A., & Lander, G. (2004). Functional annotation of the Arabidopsis genome using controlled vocabularies. *Plant physiology*, *135*(2), 745-755. https://doi.org/10.1104/pp.104.040071

Brosius, J., & Raabe, C. A. (2016). What is an RNA? A top layer for RNA classification. *RNA biology*, *13*(2), 140-144. https://doi.org/https://doi.org/10.1080/15476286.2015.1128064

Carda-Diéguez, M., Rosier, B., Lloret, S., Llena, C., & Mira, A. (2022). The tongue biofilm metatranscriptome identifies metabolic pathways associated with the presence or absence of halitosis. *npj Biofilms and Microbiomes*, *8*(1), 100. https://doi.org/doi.org/10.1038/s41522-022-00364-2

Challa, S., & Neelapu, N. R. R. (2019). Phylogenetic trees: applications, construction, and assessment. *Essentials of Bioinformatics, Volume III: In Silico Life Sciences: Agriculture*, 167-192. https://doi.org/78-3-030-19318-8 (eBook)

https://doi.org/10.1007/978-3-030-19318-8

Chang, W., Cheng, J., Allaire, J. J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., & Borges, B. (2012). Shiny: web application framework for R. *(No Title)*.

Choudhri, P., Rani, M., Sangwan, R. S., Kumar, R., Kumar, A., & Chhokar, V. (2018). De novo sequencing, assembly and characterisation of Aloe vera transcriptome and analysis of expression profiles of genes related to saponin and anthraquinone metabolism. *BMC genomics*, *19*, 1-21. https://doi.org/10.1186/s12864-018-4819-2

Consortium, G. O. (2004). The Gene Ontology (GO) database and informatics resource. *Nucleic acids research*, *32*(suppl_1), D258-D261. https://doi.org/https://doi.org/10.1093/nar/gkh036

Cooper, S. (1981). The central dogma of cell biology. *Cell biology international reports*, *5*(6), 539-549. https://doi.org/doi.org/10.1016/S0309-1651(81)80002-1

Cornwell, M., Vangala, M., Taing, L., Herbert, Z., Köster, J., Li, B., Sun, H., Li, T., Zhang, J., & Qiu, X. (2018). VIPER: Visualization Pipeline for RNA-seq, a Snakemake workflow for efficient and complete RNA-seq analysis. *BMC bioinformatics*, *19*, 1-14. https://doi.org/doi.org/10.1186/s12859-018-2139-9

Diniz, W. J. d. S., & Canduri, F. (2017). Bioinformatics: an overview and its applications. *Genet Mol Res*, *16*(1), 10.4238. https://doi.org/DOI http://dx.doi.org/10.4238/gmr16019645

Du Plessis, L., Škunca, N., & Dessimoz, C. (2011). The what, where, how and why of gene ontology—a primer for bioinformaticians. *Briefings in bioinformatics*, *12*(6), 723-735. https://doi.org/https://doi.org/10.1093/bib/bbr002

Fiers, M. W., van der Burgt, A., Datema, E., de Groot, J. C., & van Ham, R. C. (2008). High-throughput bioinformatics with the Cyrille2 pipeline system. *BMC bioinformatics*, *9*, 1-10. https://doi.org/10.1186/1471-2105-9-96

Godini, R., & Fallahi, H. (2019). A brief overview of the concepts, methods and computational tools used in phylogenetic tree construction and gene prediction. *Meta Gene*, *21*, 100586. https://doi.org/https://doi.org/10.1016/j.mgene.2019.100586

Handelsman, J. (2004). Metagenomics: application of genomics to uncultured microorganisms. *Microbiology and molecular biology reviews*, *68*(4), 669-685. https://doi.org/ https://doi.org/10.1128/mmbr.68.4.669-685.2004

Heather, J. M., & Chain, B. (2016). The sequence of sequencers: The history of sequencing DNA. *Genomics*, *107*(1), 1-8. https://doi.org/https://doi.org/10.1016/j.ygeno.2015.11.003

Kalari, K. R., Nair, A. A., Bhavsar, J. D., O'Brien, D. R., Davila, J. I., Bockol, M. A., Nie, J., Tang, X., Baheti, S., & Doughty, J. B. (2014). MAP-RSeq: Mayo analysis pipeline for RNA sequencing. *BMC bioinformatics*, *15*, 1-11. https://doi.org/http://www.biomedcentral.com/1471-2105/15/224

Kanehisa, M., Sato, Y., Kawashima, M., Furumichi, M., & Tanabe, M. (2016). KEGG as a reference resource for gene and protein annotation. *Nucleic acids research*, *44*(D1), D457-D462. https://doi.org/https://doi.org/10.1093/nar/gkv1070

Lataretu, M., & Hölzer, M. (2020). RNAflow: An effective and simple RNA-seq differential gene expression pipeline using nextflow. *Genes*, *11*(12), 1487. https://doi.org/https://doi.org/10.3390/genes11121487

Leipzig, J. (2017). A review of bioinformatic pipeline frameworks. *Briefings in bioinformatics*, *18*(3), 530-536. https://doi.org/doi: 10.1093/bib/bbw020

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., & Subgroup, G. P. D. P. (2009). The sequence alignment/map format and SAMtools. *bioinformatics*, *25*(16), 2078-2079. https://doi.org/10.1093/bioinformatics/btp352

Love, M., Anders, S., & Huber, W. (2014). Differential analysis of count data–the DESeq2 package. *Genome Biol*, *15*(550), 10-1186.

Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome biology*, *15*, 1-21. https://doi.org/DOI 10.1186/s13059-014-0550-8

Luscombe, N. M., Greenbaum, D., & Gerstein, M. (2001). What is bioinformatics? An introduction and overview. *Yearbook of medical informatics*, *10*(01), 83-100. https://doi.org/10.1055/s-0038-1638103

Mitchelson, K. R., Hawkes, D. B., Turakulov, R., & Men, A. E. (2007). Overview: developments in DNA sequencing. *Perspectives in Bioanalysis*, *2*, 3-44. https://doi.org/https://doi.org/10.1016/S1871-0069(06)02001-5

Mount, D. W. (2004). Bioinformatics-sequence and genome analysis. In: Cold spring harbor laboratory press Cold Spring Harbor, NY.

Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., & Kanehisa, M. (1999). KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic acids research*, *27*(1), 29-34. https://doi.org/https://doi.org/10.1093/nar/27.1.29

Overbeek, R., Begley, T., Butler, R. M., Choudhuri, J. V., Chuang, H.-Y., Cohoon, M., de Crécy-Lagard, V., Diaz, N., Disz, T., & Edwards, R. (2005). The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes. *Nucleic acids research*, *33*(17), 5691-5702. https://doi.org/https://doi.org/10.1093/nar/gki866

Pathak, R. K., Singh, D. B., & Singh, R. (2022). Introduction to basics of bioinformatics. In *bioinformatics* (pp. 1-15). Elsevier. https://doi.org/https://doi.org/10.1016/B978-0-323-89775-4.00006-7

Pavlopoulou, A., & Michalopoulos, I. (2011). State-of-the-art bioinformatics protein structure prediction tools. *International journal of molecular medicine*, *28*(3), 295-310. https://doi.org/ https://doi.org/10.3892/ijmm.2011.705

Pereira, R., Oliveira, J., & Sousa, M. (2020). Bioinformatics and computational tools for next-generation sequencing analysis in clinical genetics. *Journal of clinical medicine*, *9*(1), 132. https://doi.org/https://doi.org/10.3390/jcm9010132

Pirooznia, M., Yang, J. Y., Yang, M. Q., & Deng, Y. (2008). A comparative study of different machine learning methods on microarray gene expression data. *BMC genomics*, *9*, 1-13. https://doi.org/doi:10.1186/1471-2164-9-S1-S13

Punta, M., Coggill, P. C., Eberhardt, R. Y., Mistry, J., Tate, J., Boursnell, C., Pang, N., Forslund, K., Ceric, G., & Clements, J. (2012). The Pfam protein families database. *Nucleic acids research*, *40*(D1), D290-D301. https://doi.org/https://doi.org/10.1093/nar/gkr1065

Rapaport, F., Khanin, R., Liang, Y., Pirun, M., Krek, A., Zumbo, P., Mason, C. E., Socci, N. D., & Betel, D. (2013). Comprehensive evaluation of differential gene expression analysis methods for RNA-seq data. *Genome biology*, *14*, 1-13. https://doi.org/doi:10.1186/gb-2013-14-9-r95

Robinson, M. D., McCarthy, D. J., & Smyth, G. K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *bioinformatics*, *26*(1), 139-140. https://doi.org/https://doi.org/10.1093/bioinformatics/btp616

Roughan, J. (2022). Your Essential Guide to Different File Formats in Bioinformatics. *Form Bio, September*, *27*.

Rutter, L., Moran Lauter, A. N., Graham, M. A., & Cook, D. (2019). Visualization methods for differential expression analysis. *BMC bioinformatics*, *20*, 1-31. https://doi.org/https://doi.org/https://doi.org/10.1093/bioinformatics/btp616

Sadedin, S. P., Pope, B., & Oshlack, A. (2012). Bpipe: a tool for running and managing bioinformatics pipelines. *bioinformatics*, *28*(11), 1525-1526. https://doi.org/https://doi.org/10.1093/bioinformatics/bts167

Saridakis, E. (2021). The genetic informational network: How DNA conveys semantic information. *History and Philosophy of the Life Sciences*, *43*(4), 112. https://doi.org/DOI: 10.23880/mjccs-16000307

Satam, H., Joshi, K., Mangrolia, U., Waghoo, S., Zaidi, G., Rawool, S., Thakare, R. P., Banday, S., Mishra, A. K., & Das, G. (2023). Next-generation sequencing technology: current trends and advancements. *Biology*, *12*(7), 997. https://doi.org/doi.org/10.3390/biology12070997

Sawicki, M. P., Samara, G., Hurwitz, M., & Passaro Jr, E. (1993). Human genome project. *The American journal of surgery*, *165*(2), 258-264. https://doi.org/https://doi.org/10.1016/S0002-9610(05)80522-7

Silva, G. G. Z., Green, K. T., Dutilh, B. E., & Edwards, R. A. (2016). SUPER-FOCUS: a tool for agile functional analysis of shotgun metagenomic data. *bioinformatics*, *32*(3), 354-361. https://doi.org/https://doi.org/10.1093/bioinformatics/btv584

SILVA, R. C. C. d., & ALVES, M. C. S. The advances and challenges of bioinformatics applied to health: a review. https://doi.org/DOI: 10.48017/dj.v9i3.2910

Soneson, C., & Delorenzi, M. (2013). A comparison of methods for differential expression analysis of RNA-seq data. *BMC bioinformatics*, *14*, 1-18.

Stephens, Z. D., Lee, S. Y., Faghri, F., Campbell, R. H., Zhai, C., Efron, M. J., Iyer, R., Schatz, M. C., Sinha, S., & Robinson, G. E. (2015). Big data: astronomical or genomical? *PLoS biology*, *13*(7), e1002195.

Sundararajan, Z., Knoll, R., Hombach, P., Becker, M., Schultze, J. L., & Ulas, T. (2019). Shiny-Seq: advanced guided transcriptome analysis. *BMC research notes*, *12*, 1-5. https://doi.org/https://doi.org/10.1186/s13104-019-4471-1

Watson, J. D., & Crick, F. H. (1953). Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, *171*(4356), 737-738.

Wolfien, M., Rimmbach, 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*, 1-11. https://doi.org/DOI 10.1186/s12859-015-0873-9

Appendix.

Appendix A : R coding Script

```
install.packages("BiocManager")
BiocManager::install("DESeq2")
library(DESeq2)
library(tibble)
library(dplyr)
library(ggplot2)
install.packages("pheatmap")
BiocManager::install("apeglm")
library(apeglm)
library(pheatmap)


ainstall.packages("ggrepel")
library(ggrepel)


# Load the dataset
countData <-
read.csv("/Users/rohan/Downloads/L4_function_Subsystems.csv")
metadata <- read.csv("/Users/rohan/Downloads/metadata.csv")
colnames(countData)
rownames(metadata)
rownames(metadata) <- metadata[[1]]
```

```
metadata<-metadata[-1]
rownames(countData)<-countData[[1]]
countData<-countData[-1]

str(countData)

# all columns and row should have same names this result in true
all(colnames(countData) %in% rownames(metadata))




# Convert to matrix
matrix <- as.matrix(countData)
int_matrix <- apply(matrix, c(1, 2), as.integer)

#  Convert SITE column to factor (automatically detect levels)
metadata$SITE <- factor(metadata$SITE, levels =
unique(metadata$SITE))

# Create DESeq2 dataset
dds <- DESeqDataSetFromMatrix(countData = int_matrix,
                              colData = metadata,
                              design = ~ SITE)
dds <- DESeq(dds)

# Generate all pairwise comparisons automatically
SITE <- levels(metadata$SITE)
comparisons <- combn(SITE, 2, simplify = FALSE)

#  Define output directory
output_dir <- "~/Documents/new_DESeq2_results/"
dir.create(output_dir, showWarnings = FALSE, recursive = TRUE)

#  Loop through each pairwise comparison (significant results only)
```

```
for (comp in comparisons) {
  site1 <- comp[1]  # control
  site2 <- comp[2]  # treatment

  # Run DESeq2 contrast
  res <- results(dds, contrast = c("SITE", site2, site1))

  # Convert to dataframe
  res_df <- as.data.frame(res)

  # Filter by adjusted p-value < 0.05 (significant genes only)
  sig_res_df <- subset(res_df, padj < 0.05 & !is.na(padj))

  # Add metadata columns for clarity
  sig_res_df$treatment <- site2
  sig_res_df$control <- site1
  sig_res_df$comparison <- paste(site2, "vs", site1)
  sig_res_df$direction <- ifelse(sig_res_df$log2FoldChange > 0,
paste("Up in", site2),
                                 ifelse(sig_res_df$log2FoldChange <
0, paste("Up in", site1), "No change"))

  # Save only if there are significant results
  if (nrow(sig_res_df) > 0) {
    filename <- paste0(output_dir, "DESeq2_", site2, "_vs_", site1,
"_significant.csv")
    write.csv(sig_res_df, filename, row.names = TRUE)
    print(paste(" Saved significant results:", filename))
  } else {
    print(paste(" No significant results for:", site2, "vs", site1))
  }
}

# Optional: Summary of all comparisons
summary_table <- do.call(rbind, lapply(comparisons, function(comp) {
```

```
  data.frame(
    treatment = comp[2],
    control = comp[1],
    comparison = paste(comp[2], "vs", comp[1])
  )
}))
write.csv(summary_table, paste0(output_dir,
"All_DESeq2_Comparisons_Summary.csv"), row.names = FALSE)
print(" Summary of all comparisons saved!")


Shiny app workflow


library(shiny)
library(dplyr)
library(readr)
library(ggplot2)
library(DT)


# Function to process uploaded file
data_processing <- function(file_path) {
  data <- read_csv(file_path)


  split_data <- strsplit(data[[1]], ";")
  max_splits <- max(sapply(split_data, length))
  split_data <- lapply(split_data, function(x) {
    length(x) <- max_splits
    return(x)
  })


  processed_data <- do.call(rbind, split_data) %>%
as.data.frame(stringsAsFactors = FALSE)
  colnames(processed_data) <- paste0("Level",
seq_len(ncol(processed_data)))


  processed_data <- processed_data %>%
```

```
    mutate(Level4 = apply(processed_data[, 4:ncol(processed_data),
drop = FALSE], 1, function(x) paste(na.omit(x), collapse = ";")))
%>%
    select(Level1, Level2, Level3, Level4)


  final_data <- cbind(processed_data, data[,-1])
  return(final_data)
}


# Shiny App
ui <- fluidPage(
  titlePanel("Hierarchical Data Explorer with Volcano Plot"),
  sidebarLayout(
    sidebarPanel(
      fileInput("file", "Upload CSV File", accept = c(".csv")),
      selectInput("level1", "Select Level 1:", choices = NULL),
      uiOutput("level2_ui"),
      uiOutput("level3_ui"),
      uiOutput("level4_ui"),
      br(),
      h4("Number of Selected Genes:"),
      verbatimTextOutput("gene_count"),
      h4("Log2 Fold Change Summary:"),
      verbatimTextOutput("logfc_summary")
    ),
    mainPanel(
      h3("Summary of Selected Data"),
      DTOutput("summary_table"),
      br(),
      h3("Volcano Plot"),
      plotOutput("volcano_plot"),
      br(),
      h3("Positive and Negative log2 Fold Change Bar Plot"),
      plotOutput("logfc_barplot"),
      br(),
```

```
      h3("Positive and Negative log2 Fold Change Summary by Level
1"),
      tableOutput("level1_summary"),
      br(),
      h3("Bar Plot Based on Level 1 Summary"),
      plotOutput("level1_barplot"),
      br(),
      h3("Overall Summary for Level 1"),
      plotOutput("overall_level1_barplot")
    )
  )
)


server <- function(input, output, session) {
  data_reactive <- reactive({
    req(input$file)
    processed_data <- data_processing(input$file$datapath)
    return(processed_data)
  })


  observe({
    req(data_reactive())
    updateSelectInput(session, "level1", choices =
unique(data_reactive()$Level1))
  })


  output$level2_ui <- renderUI({
    req(input$level1)
    level2_choices <- unique(data_reactive() %>% filter(Level1 ==
input$level1) %>% pull(Level2))
    selectInput("level2", "Select Level 2:", choices = c("All",
level2_choices))
  })


  output$level3_ui <- renderUI({
    req(input$level2)
```

```
    filtered_data <- data_reactive() %>% filter(Level1 ==
input$level1)
    if (input$level2 != "All") {
      filtered_data <- filtered_data %>% filter(Level2 ==
input$level2)
    }
    level3_choices <- unique(filtered_data$Level3)
    selectInput("level3", "Select Level 3:", choices = c("All",
level3_choices))
  })

  output$level4_ui <- renderUI({
    req(input$level3)
    filtered_data <- data_reactive() %>% filter(Level1 ==
input$level1)
    if (input$level2 != "All") {
      filtered_data <- filtered_data %>% filter(Level2 ==
input$level2)
    }
    if (input$level3 != "All") {
      filtered_data <- filtered_data %>% filter(Level3 ==
input$level3)
    }
    level4_choices <- unique(filtered_data$Level4)
    selectInput("level4", "Select Level 4:", choices = c("All",
level4_choices))
  })

  filtered_data_reactive <- reactive({
    filtered_data <- data_reactive()
    if (!is.null(input$level1) && input$level1 != "") {
      filtered_data <- filtered_data %>% filter(Level1 ==
input$level1)
    }
    if (!is.null(input$level2) && input$level2 != "All" &&
input$level2 != "") {
      filtered_data <- filtered_data %>% filter(Level2 ==
input$level2)
```

```
    }

    if (!is.null(input$level3) && input$level3 != "All" &&
input$level3 != "") {

      filtered_data <- filtered_data %>% filter(Level3 ==
input$level3)

    }

    if (!is.null(input$level4) && input$level4 != "All" &&
input$level4 != "") {

      filtered_data <- filtered_data %>% filter(Level4 ==
input$level4)

    }

    return(filtered_data)

  })


  output$gene_count <- renderText({

    filtered_data <- filtered_data_reactive()

    paste(nrow(filtered_data), "genes available")

  })


  output$logfc_summary <- renderText({

    filtered_data <- filtered_data_reactive()

    positive_count <- sum(filtered_data$log2FoldChange > 0, na.rm =
TRUE)

    negative_count <- sum(filtered_data$log2FoldChange < 0, na.rm =
TRUE)

    paste("Positive log2FoldChange: ", positive_count, "

Negative log2FoldChange: ", negative_count)

  })


  output$volcano_plot <- renderPlot({

    data <- filtered_data_reactive()

    data$threshold <- "Not Significant"

    data$threshold[data$padj < 0.05 & data$log2FoldChange > 1] <-
"Upregulated"

    data$threshold[data$padj < 0.05 & data$log2FoldChange < -1] <-
"Downregulated"
```

```
    ggplot(data, aes(x = log2FoldChange, y = -log10(padj), color =
threshold)) +
      geom_point(alpha = 0.7, size = 1.5) +
      scale_color_manual(values = c("Upregulated" = "red",
"Downregulated" = "blue", "Not Significant" = "grey")) +
      theme_minimal() +
      labs(title = "Volcano Plot", x = "log2(Fold Change)", y = "-
log10(Adjusted p-value)")
  })

  output$logfc_barplot <- renderPlot({
    summary_data <- data_reactive() %>%
      group_by(Level1) %>%
      summarise(Positive = sum(log2FoldChange > 0, na.rm = TRUE),
                Negative = sum(log2FoldChange < 0, na.rm = TRUE))
%>%
      tidyr::pivot_longer(cols = c(Positive, Negative), names_to =
"Direction", values_to = "Count")

    ggplot(summary_data, aes(x = Level1, y = ifelse(Direction ==
"Negative", -Count, Count), fill = Direction)) +
      geom_bar(stat = "identity") +
      geom_text(aes(label = abs(Count)), position =
position_stack(vjust = 0.5)) +
      coord_flip() +
      labs(title = "Positive and Negative log2 Fold Change by Level
1", x = "Level 1", y = "Gene Count") +
      theme_minimal()
  })

  output$level1_summary <- renderTable({
    data_reactive() %>%
      group_by(Level1) %>%
      summarise(Genes_Present = n())
  })

  output$overall_level1_barplot <- renderPlot({
```

```
    summary_data <- data_reactive() %>%
      group_by(Level1) %>%
      summarise(Positive = sum(log2FoldChange > 0, na.rm = TRUE),
                Negative = sum(log2FoldChange < 0, na.rm = TRUE))
%>%
      tidyr::pivot_longer(cols = c(Positive, Negative), names_to =
"Direction", values_to = "Count")


    ggplot(summary_data, aes(x = Level1, y = ifelse(Direction ==
"Negative", -Count, Count), fill = Direction)) +
      geom_bar(stat = "identity") +
      geom_text(aes(label = abs(Count)), position =
position_stack(vjust = 0.5)) +
      coord_flip() +
      labs(title = "Overall Positive and Negative log2 Fold Change
by Level 1", x = "Level 1", y = "Gene Count") +
      theme_minimal()
  })


  output$summary_table <- renderDT({
    datatable(filtered_data_reactive(), options = list(pageLength =
10, autoWidth = TRUE))
  })
}


shinyApp(ui, server)
```

Appendix B. Supplementary result data table for tongue biofilm metatranscriptome-halitosis associated dataset, used for validation and benchmarking .

HS VS HEALTH

Taxonomy	baseMean	log2FoldCha	lfcSE	stat	pvalue	padj	treatment	control	comparison	direction
Streptococcus parasanguinis	9450.77131	-3.2943372	0.76033079	-4.3327684	1.47E-05	0.00073917	HS	Health	HS vs Health	Up in Health
Veillonella dispar	11190.5041	-2.5145687	0.82818226	-3.0362504	0.0023954	0.03164454	HS	Health	HS vs Health	Up in Health
Fusobacterium periodonticum	29382.0668	1.15740796	0.86824106	1.3330491	0.18251569	0.48735573	HS	Health	HS vs Health	Up in HS
Rothia mucilaginosa	4798.07472	-3.9700588	0.88928098	-4.4643469	8.03E-06	0.00061614	HS	Health	HS vs Health	Up in Health
Prevotella shahii	1838.17699	3.25388427	1.2862444	2.52975584	0.01141419	0.0992757	HS	Health	HS vs Health	Up in HS

MM VS HEALTH

Texonomy	baseMean	log2FoldCha	lfcSE	stat	pvalue	padj	treatment	control	comparison	direction
Streptococcus parasanguinis	9450.77131	-0.5943947	0.78365586	-0.7584895	0.448158	1	MM	Health	MM vs Health	Up in Health
Veillonella dispar	11190.5041	-1.6870412	0.85365369	-1.9762595	0.0481254	0.74541571	MM	Health	MM vs Health	Up in Health
Fusobacterium periodonticum	29382.0668	-0.1667378	0.8949673	-0.1863061	0.85220475	1	MM	Health	MM vs Health	Up in Health
Rothia mucilaginosa	4798.07472	-2.426247	0.91652689	-2.6472185	0.00811569	0.44230505	MM	Health	MM vs Health	Up in Health
Prevotella shahii	1838.17699	1.22199883	1.32589678	0.92163949	0.35671665	1	MM	Health	MM vs Health	Up in MM

MM VS HS

Texonomy	baseMean	log2FoldCha	lfcSE	stat	pvalue	padj	treatment	control	comparison	direction
Streptococcus parasanguinis	9450.77131	2.6999425	0.78372173	3.44502698	0.000571	0.03920883	MM	HS	MM vs HS	Up in MM
Veillonella dispar	11190.5041	0.82752751	0.85367652	0.96936895	0.33236114	0.61852365	MM	HS	MM vs HS	Up in MM
Fusobacterium periodonticum	29382.0668	-1.3241458	0.89495453	-1.4795677	0.13898866	0.45447086	MM	HS	MM vs HS	Up in HS
Rothia mucilaginosa	4798.07472	1.54381179	0.91667779	1.68413789	0.09215502	0.3757347	MM	HS	MM vs HS	Up in MM
Prevotella shahii	1838.17699	-2.0318854	1.3252925	-1.53316	0.12523644	0.43726621	MM	HS	MM vs HS	Up in HS

MM-HS VS HEALTH

Texonomy	baseMean	log2FoldCha	lfcSE	stat	pvalue	padj	treatment	control	comparison	direction
Streptococcus parasanguinis	9450.77131	-3.0482855	0.81290315	-3.7498753	0.00017692	0.00484751	MM.HS	Health	MM.HS vs He	Up in Health
Veillonella dispar	11190.5041	-2.0488598	0.88538478	-2.3140897	0.0206628	0.10079417	MM.HS	Health	MM.HS vs He	Up in Health
Fusobacterium periodonticum	29382.0668	1.73235253	0.92818836	1.86638037	0.06198817	0.19678783	MM.HS	Health	MM.HS vs He	Up in MM.HS
Rothia mucilaginosa	4798.07472	-2.2700212	0.95061737	-2.3879442	0.01694291	0.09158332	MM.HS	Health	MM.HS vs He	Up in Health
Prevotella shahii	1838.17699	3.65598398	1.37498945	2.65891785	0.00783921	0.06271366	MM.HS	Health	MM.HS vs He	Up in MM.HS

MM-HA VS HS

Texonomy	baseMean	log2FoldCha	lfcSE	stat	pvalue	padj	treatment	control	comparison	direction
Streptococcus parasanguinis	9450.77131	0.24605179	0.81296669	0.30265912	0.76214965	1	MM.HS	HS	MM.HS vs HS	Up in MM.HS
Veillonella dispar	11190.5041	0.46570896	0.88540683	0.52598302	0.59890001	1	MM.HS	HS	MM.HS vs HS	Up in MM.HS
Fusobacterium periodonticum	29382.0668	0.57494457	0.9281761	0.61943479	0.53562997	1	MM.HS	HS	MM.HS vs HS	Up in MM.HS
Rothia mucilaginosa	4798.07472	1.70003751	0.95076291	1.78807722	0.07376355	0.62599337	MM.HS	HS	MM.HS vs HS	Up in MM.HS
Prevotella shahii	1838.17699	0.4020997	1.37440691	0.29256234	0.7698567	1	MM.HS	HS	MM.HS vs HS	Up in MM.HS

MM-HA VS MM

Texonomy	baseMean	log2FoldCha	lfcSE	stat	pvalue	padj	treatment	control	comparison	direction
Streptococcus parasanguinis	9450.77131	-2.4538907	0.83482242	-2.9394164	0.00328831	0.05293874	MM.HS	MM	MM.HS vs MM	Up in MM
Veillonella dispar	11190.5041	-0.3618186	0.90927668	-0.3979191	0.69068983	0.82794229	MM.HS	MM	MM.HS vs MM	Up in MM
Fusobacterium periodonticum	29382.0668	1.89909036	0.95322341	1.99228255	0.04634007	0.28885308	MM.HS	MM	MM.HS vs MM	Up in MM.HS
Rothia mucilaginosa	4798.07472	0.15622572	0.97629457	0.16001904	0.87286608	0.94223349	MM.HS	MM	MM.HS vs MM	Up in MM.HS
Prevotella shahii	1838.17699	2.43398515	1.41158504	1.72429225	0.0846551	0.3272307	MM.HS	MM	MM.HS vs MM	Up in MM.HS