DEG Pipeline Assistant: An Interactive and Reproducible Pipeline for RNA-seq Differential Expression and Pathway Analysis

Ishaan Banwait, Gurkamal Deol

Abstract

We present an interactive, reproducible pipeline for the analysis of normalized RNA-seq data, enabling identification of differentially expressed genes (DEGs), mapping to human orthologs, and downstream functional enrichment analyses. The pipeline integrates quality control, statistical analysis, and visualization, supporting both command-line and graphical user interface (GUI) workflows. Outputs include publication-ready figures and comprehensive result tables, facilitating transparent and efficient transcriptomic analysis.

Introduction

RNA sequencing (RNA-seq) has become essential for transcriptome profiling, offering precise measurement of gene expression across thousands of genes simultaneously (Kukurba & Montgomery, 2015). Yet, post-sequencing analysis remains a bottleneck due to its technical complexity. We introduce a modular Python pipeline that simplifies this process while ensuring reproducibility, usability, and extensibility.

The pipeline supports both command-line usage via Click (Ronacher, 2017) and a graphical interface built with Streamlit (Streamlit Inc., 2023), making it accessible to researchers of varying computational expertise. The command-line interface (CLI) is designed to run locally on the user's machine, offering speed, control, and data privacy. Meanwhile, the graphical user interface (GUI) can be deployed to a server, enabling remote access through a browser without requiring Python expertise.

The pipeline supports flexible experimental designs, robust quality control, and automated result visualizations. This ease of use and reproducibility is particularly valuable in preclinical research settings, where fast and reliable insights can accelerate decision-making and shorten the timeline for therapeutic development—ultimately helping teams bring therapies to market with greater confidence.

Methods

Pipeline Architecture

The pipeline takes normalized RNA-seq data as input and performs a complete analysis workflow, generating various visualizations and identifying biologically significant patterns. Its

modular design allows researchers to execute the entire pipeline or select specific components based on their research needs.

The analysis process follows a logical progression:

1. Metadata Construction: Metadata tables are auto-generated from normalized counts.

2. Quality Control: Boxplots, PCA, and correlation heatmaps assess sample consistency.

3. DEG Processing: DESeq2 result files are filtered and categorized using user-defined thresholds.

4. Visualization: MA plots, volcano plots, and heatmaps reveal global and gene-level expression patterns.

5. Ortholog Mapping: Gene IDs from the model organism are translated to their human orthologs using the g:Profiler API (Raudvere et al., 2019), enabling downstream interpretation in the context of human biology.

6. Functional Enrichment: GO and KEGG enrichment analysis is performed using gseapy, a Python interface to Enrichr (Fang et al., 2021).

Technical Implementation

- Programming Language: Python 3.13+

- Core Libraries:
    - Pandas for data manipulation (McKinney, 2010)
    - numpy for numerical operations (Harris et al., 2020)
    - matplotlib (Hunter, 2007) and seaborn (Waskom, 2021) for plotting
    - scikit-learn for PCA (Pedregosa et al., 2011)
    - click for CLI integration (Ronacher, 2017)
    - Streamlit for GUI (Streamlit Inc., 2023)
    - gseapy for gene set enrichment analysis (Fang et al., 2021)
    - mygene for Ensembl-to-gene symbol mapping (Xin et al., 2016)
    - openai for natural language handling (OpenAI, 2023)

Each of the layers described below is built upon the previous to create a clean user experience. The web interface organizes parameter collection and leverages AI to generate terminal commands, allowing the CLI to access the core pipeline and execute the appropriate analysis steps.

Core Analysis Module

The primary logic resides in *DEGpipeline.py*. Here, the fundamental analysis functions are contained, including data processing, quality control, differential expression analysis, visualization, ortholog mapping, and enrichment analysis. Data is loaded and processed

using pandas DataFrames. Normalized count data is visualized via boxplots and correlation heatmaps to identify batch effects or outliers. PCA is performed using standardized gene expression to evaluate variance across samples.

Statistical filtering of DEGs is based on user-defined thresholds for log2 fold change and adjusted p-value, using standard pandas operations. Results are visualized with volcano and MA plots. Top DEGs are selected for heatmap visualization, and gene identifiers are mapped to human orthologs using the g:Profiler API (Raudvere et al., 2019). GO and KEGG enrichment analysis are performed using the enrichr interface via gseapy (Fang et al., 2021). Gene set enrichment analysis (GSEA) identifies functional groups of genes, allowing researchers to understand the effects of the conditions being studied. Instead of simply being given a number of genes are differentially expressed in an organism treated with a novel drug, they are told which pathways in the body are affected, allowing them to understand the biological interpretation of the data. To perform the enrichment, Ensembl gene IDs from significant DEGs are first converted to gene symbols using the mygene library (Xin et al., 2016). Then, functional annotations are extracted using gene set enrichment analysis. Results are saved in CSV format and visualized as bar plots.

The pipeline generates a rich set of visualizations to aid in data interpretation:

- Quality Control: Boxplots of normalized counts, sample correlation heatmaps, and PCA plots help identify outliers and assess sample relationships.
- Differential Expression: MA plots and volcano plots visualize the distribution and significance of differentially expressed genes.
- Gene Expression Patterns: Heatmaps of top differentially expressed genes reveal expression patterns across samples and conditions.
- Pathway Analysis: Bar plots of enriched GO terms and KEGG pathways highlight biological processes and pathways affected by differentially expressed genes.

Command-Line Interface

A flexible command-line interface (pipeline_CLI.py) was built using the Click framework to allow selective execution of the core pipeline with customizable parameters. Example command:

```
python pipeline_CLI.py --norm-file "path/to/normalized_data.csv" --pw-data
"path/to/pairwise_data" --base-dir "output_directory" --num-replicates 3
--conditions "Control" --conditions "Treatment1" --conditions "Treatment2"
run-all --model-organism "drerio" --pw-interest "Treatment1_vs_Control"
--pw-interest "Treatment2_vs_Control" --log2fc-threshold 1.0 --padj-threshold
0.05 --enrich-sig-cutoff 0.05 --skip-boxplots
```

[See full CLI options in the repository.]

AI Integration

An AI assistant (ai_assistant.py) uses OpenAI's API to extract pipeline parameters from natural language from the user, and generates the appropriately valid command lines for the CLI. This component is designed to reduce user burden by translating intuitive input (e.g., "zebrafish") into exact identifiers (e.g. “drerio”). Parameters are formatted and validated before generating valid CLI commands.

The integration of AI is intelligently regulated within the script to be more cost-efficient. Instead of the user speaking back-and-forth with the large language model (LLM), incurring many costly interactions, the parameters are collected by hardcoded prompts and then passed once to the LLM in a single API call at the end. The hardcoded prompting function and AI-driven language transformation function will recursively call each other until the LLM recognizes all the parameters and passes them one last time to construct the final commands (mutual recursion).

Web Interface

The Streamlit-based interface (app.py) simplifies pipeline usage by allowing users to upload files and set parameters in a browser. It communicates with the AI assistant to automatically generate CLI commands, execute analyses, and display results in real time. Output files are categorized and made downloadable from the GUI.

Usage

The web interface prompts users for the following parameters:
Setup parameters:

- The normalized data file containing the adjusted number of sequence reads mapped to each gene.
- The pairwise data files from differential expression analysis of the normalized data
- A list of experimental conditions being tested (eg. treated, control…)
- The number of sample replicates per condition

It subsequently prompts for the analysis parameters:

- The model organism used in the experiment, for ortholog mapping
- Which pairwise comparisons of interest to focus downstream analysis on
- Log2 fold change threshold (default provided)
- Adjusted p-value threshold (default provided)
- Enrichment significance cutoff (default provided)
- A selection of analysis steps to skip (optional)

The commands will run and generate results to a standardized directory structure with subfolders for data, results, and plots.

- Figures: Boxplots, correlation heatmaps, PCA plots, MA plots, volcano plots, DEG heatmaps, GO/KEGG barplots (all PNG).

- Tables: Metadata, processed DEG tables, ortholog mapping, enrichment results (all CSV).

Customization

Users can skip specific steps (e.g., QC plots, enrichment) via CLI flags or GUI options. All user-selected parameters and executed workflow steps are automatically recorded to ensure reproducibility and enable exact reruns of the analysis.

Results

We validated the pipeline on a multi-condition zebrafish RNA-seq dataset. Thousands of DEGs were identified across pairwise comparisons. Ortholog mapping successfully translated model organism genes to human identifiers, and enrichment results highlighted biologically interpretable pathways. Representative visualizations are provided in Figures 1–8, including volcano plots, PCA, and enrichment bar plots.

Discussion

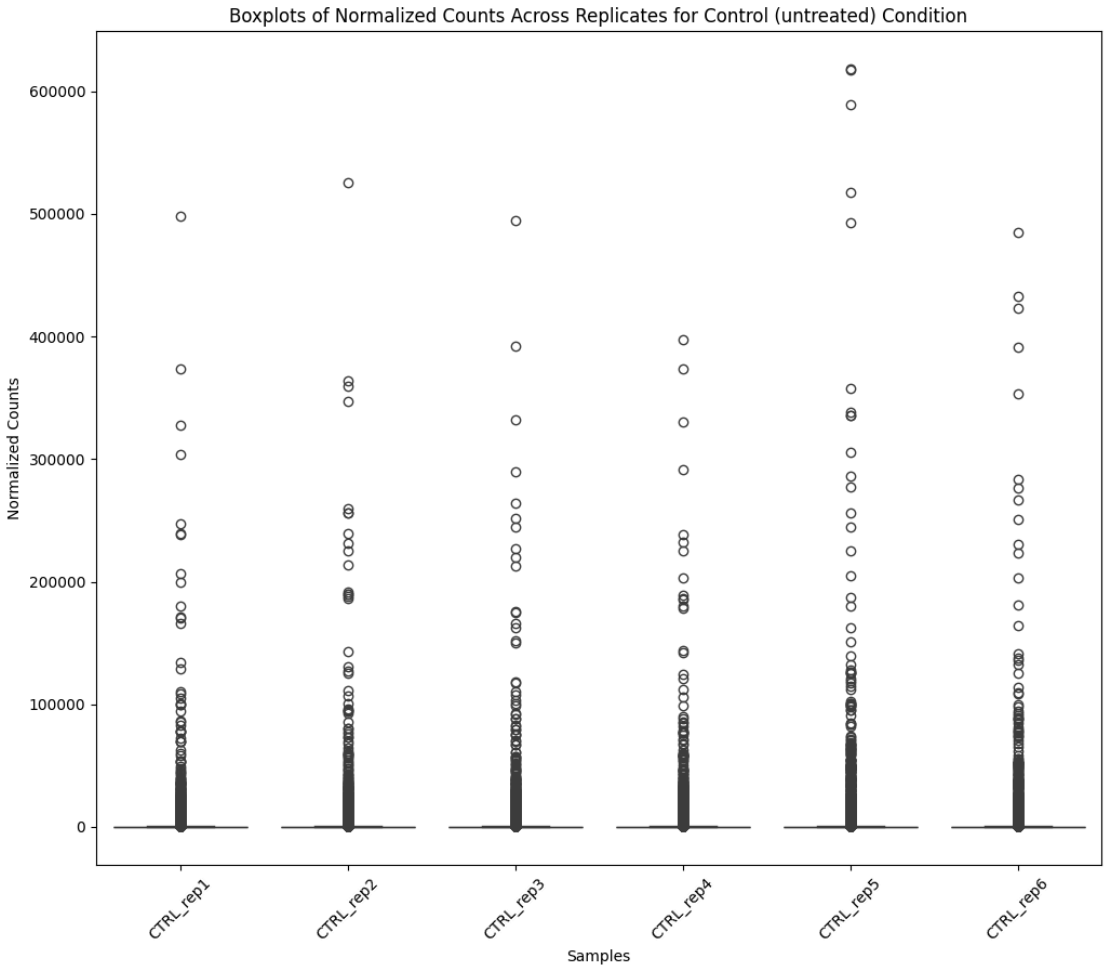
This pipeline addresses critical challenges in RNA-seq analysis: reproducibility, user accessibility, and interpretability. By combining standard statistical methods with flexible interfaces and AI-driven input parsing, the DEG Pipeline Assistant allows researchers to gain insights faster and with fewer barriers. The modular structure allows easy adaptation to new organisms, thresholds, or downstream analyses. This technology is particularly impactful in preclinical settings for companies developing novel therapeutics, where rapid iteration and high confidence in data interpretation are essential to bringing healthcare solutions to market.

Acknowledgements

This work was carried out at **Canurta Therapeutics**, while both authors were affiliated.

Figures

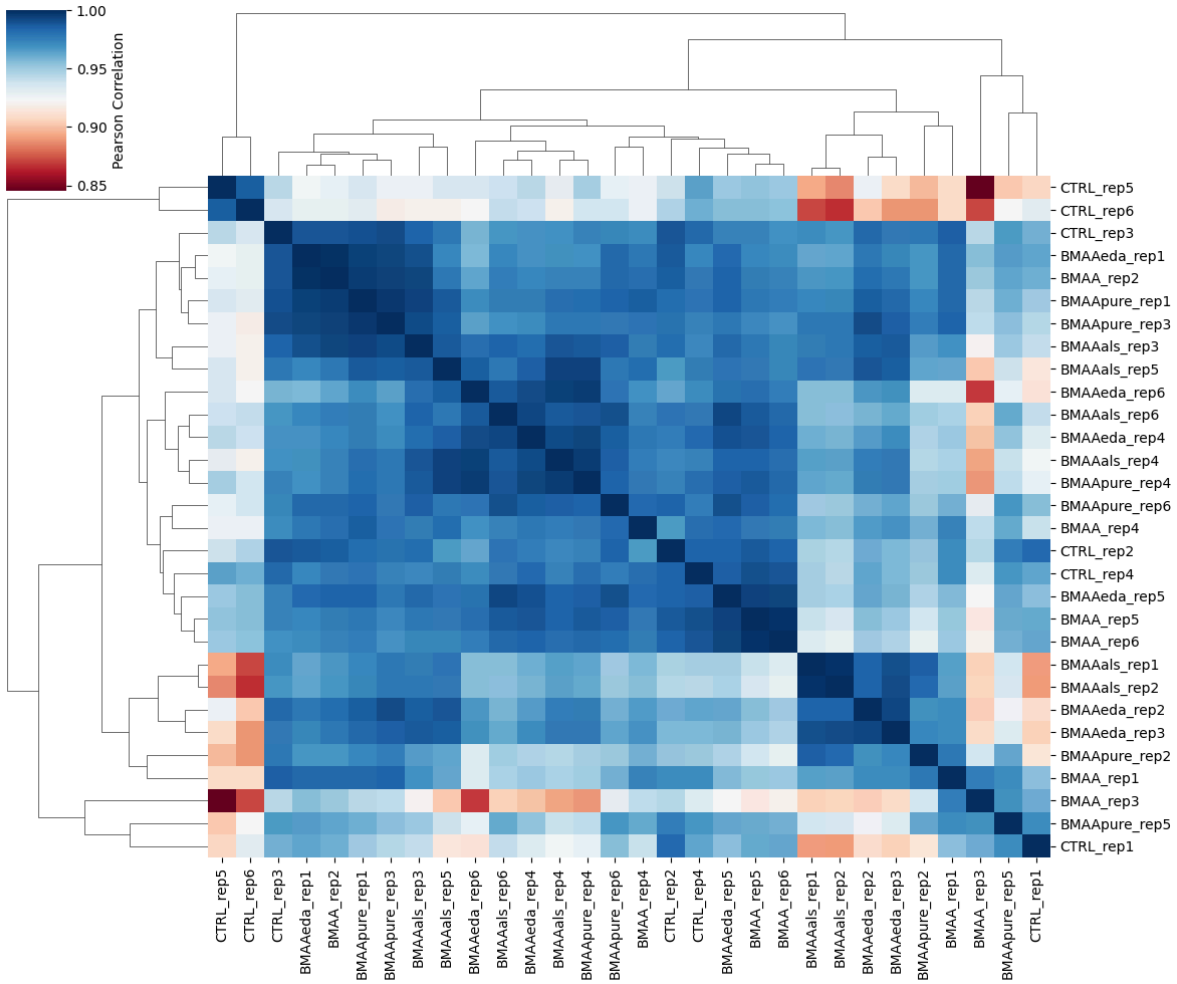
Figure 1. Boxplots of Normalized Gene Expression Counts (Control Condition)



Boxplots showing the distribution of normalized RNA-seq counts for each biological replicate in the control (untreated) condition.

These plots assess the consistency of gene expression across replicates. Uniform distribution across samples indicates consistent library sizes and proper normalization. Any sample with a distinctly different distribution may indicate a technical issue.

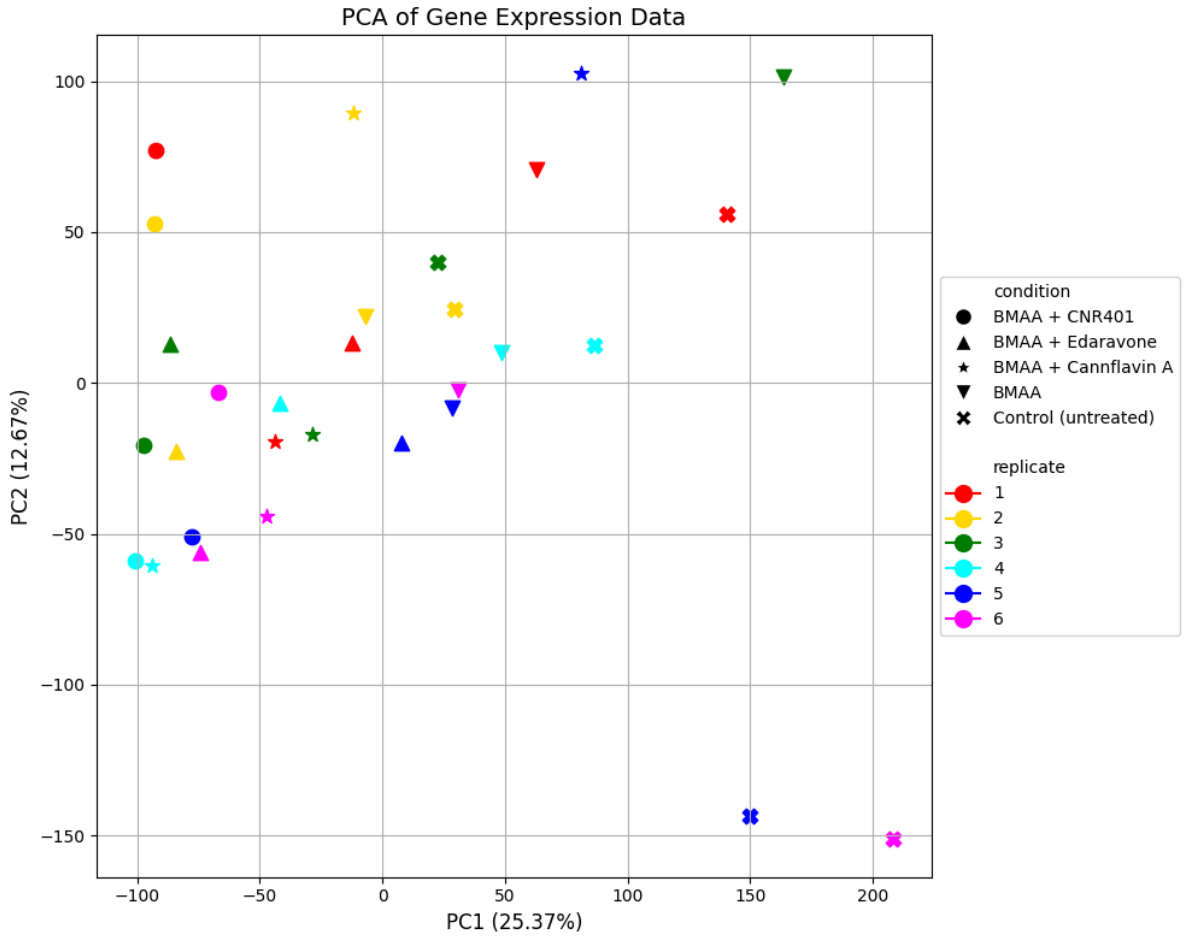
Figure 2. Correlation heatmap of all sample expression profiles.



Heatmap showing pairwise Pearson correlations of gene expression between all samples, clustered hierarchically.

Control samples show generally high pairwise correlation values (blue tones), indicating reproducibility across replicates. A few samples show slightly lower correlations, suggesting potential outliers or subtle technical variation. This heatmap is useful for identifying batch effects and verifying the consistency of sample groups.

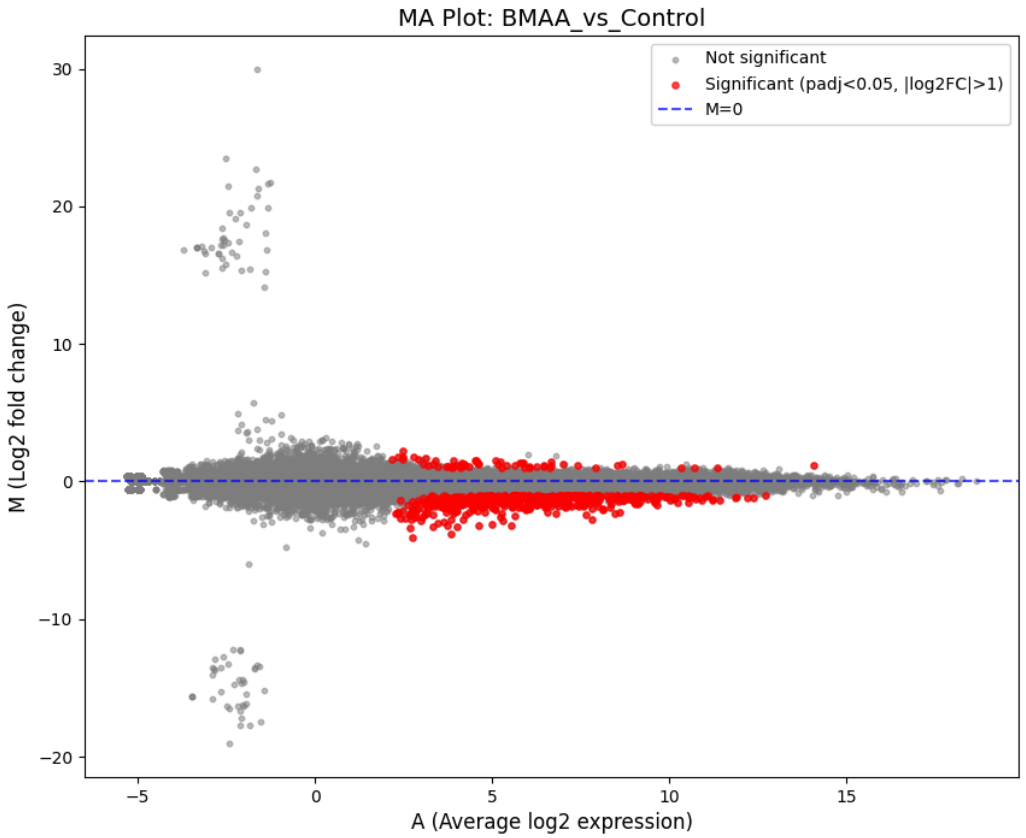
Figure 3. Principal Component Analysis (PCA) of Gene Expression



PCA plot of all RNA-seq samples using normalized gene expression values. Points are colored by replicate and shaped by condition.

Control (untreated) samples form a distinct cluster, separating from BMAA-treated samples across principal components. This separation suggests treatment-specific transcriptomic signatures. Variance percentages for PC1 and PC2 are shown on axes.

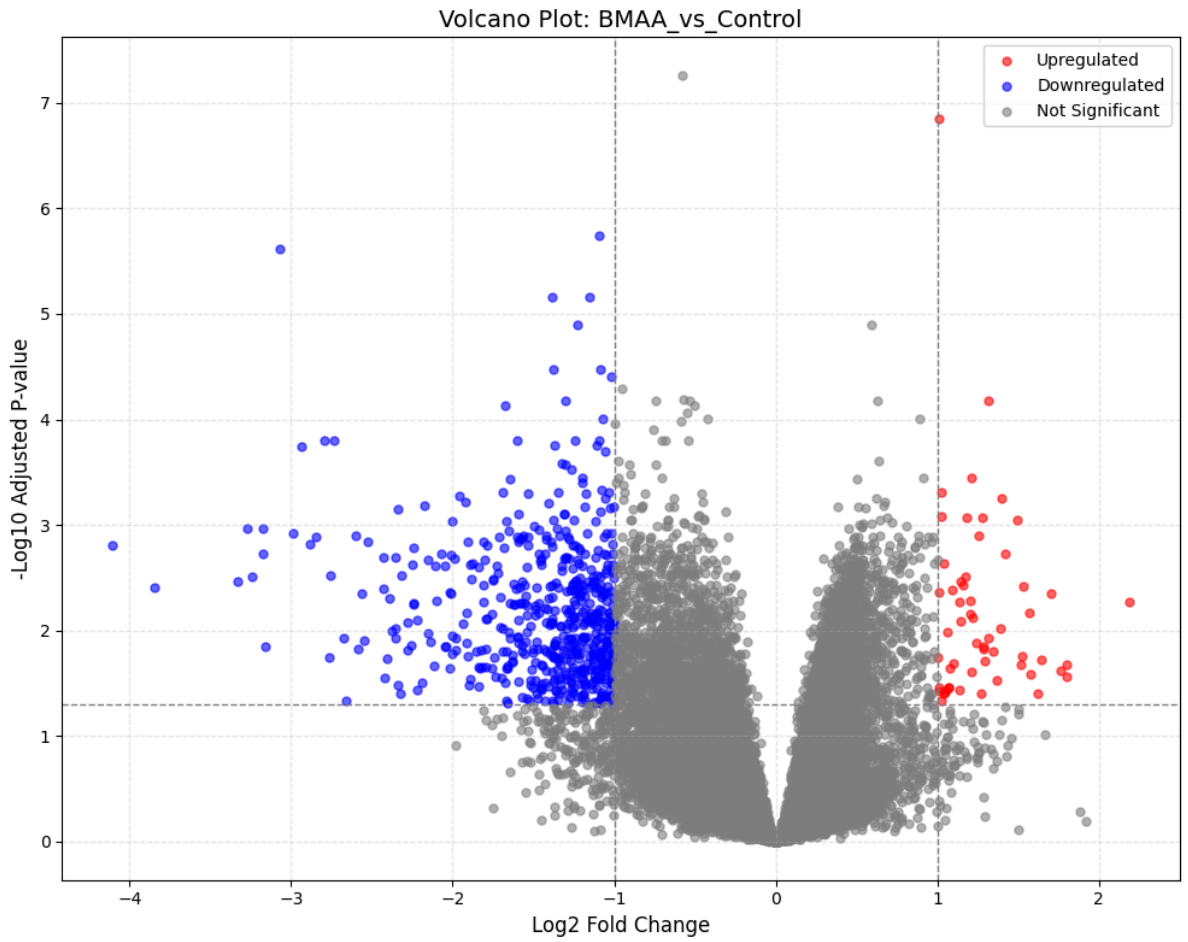
Figure 4. MA Plot: BMAA vs Control



MA plot comparing mean expression (A) to log2 fold change (M) for the BMAA vs Control contrast.

Each dot represents a gene. Red dots are significantly differentially expressed (|log2FC| > 1, FDR < 0.05). The horizontal line at M = 0 represents no change. This plot highlights genes with biologically meaningful changes in expression relative to the control.

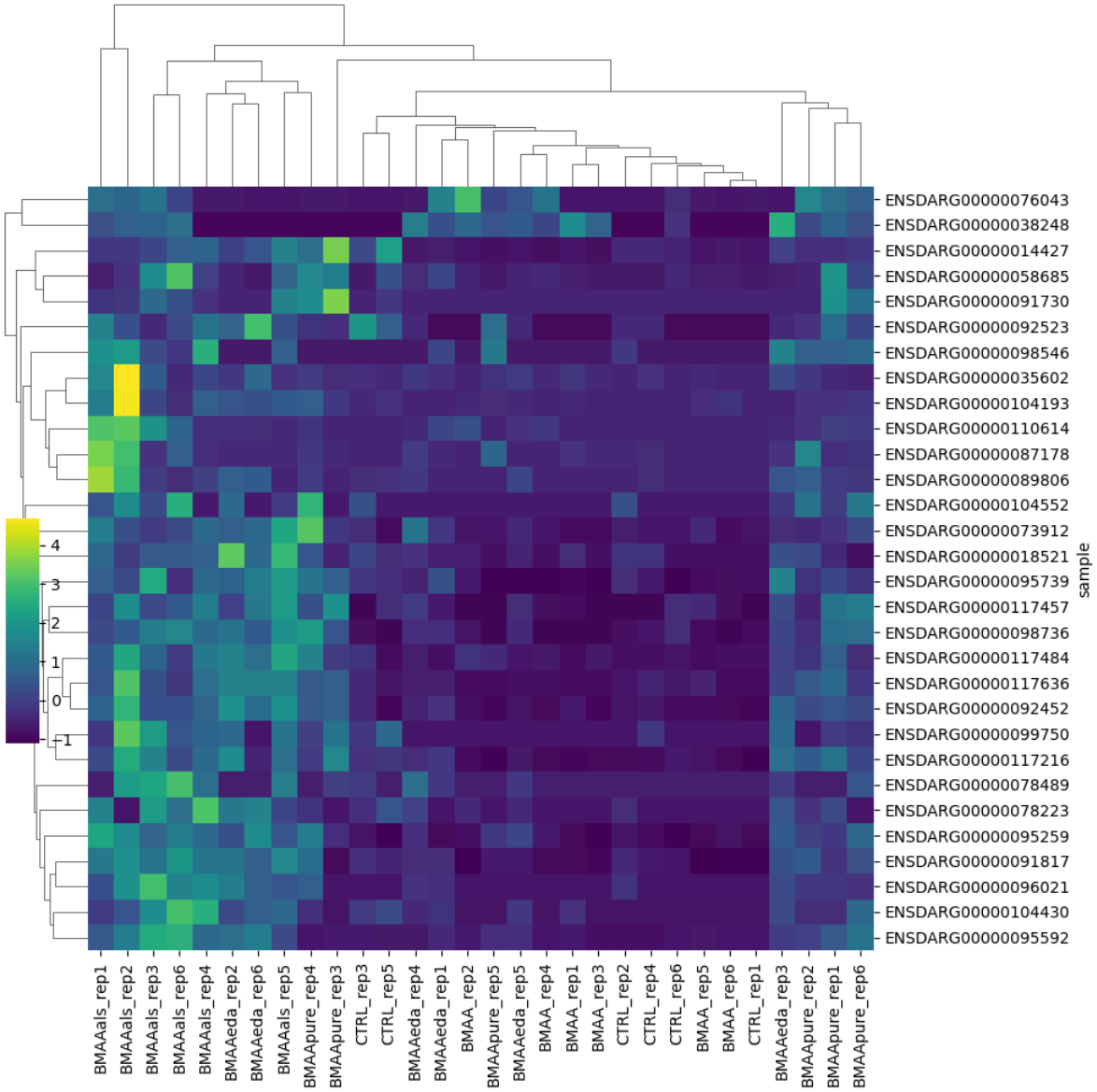
Figure 5. Volcano Plot: BMAA vs Control



Volcano plot showing differentially expressed genes in BMAA-treated samples compared to control.

Genes with statistically significant upregulation are in red, downregulation in blue, and nonsignificant in gray. The plot displays log2 fold change on the x-axis and –log10 adjusted p-value on the y-axis. This visualization captures both effect size and significance.

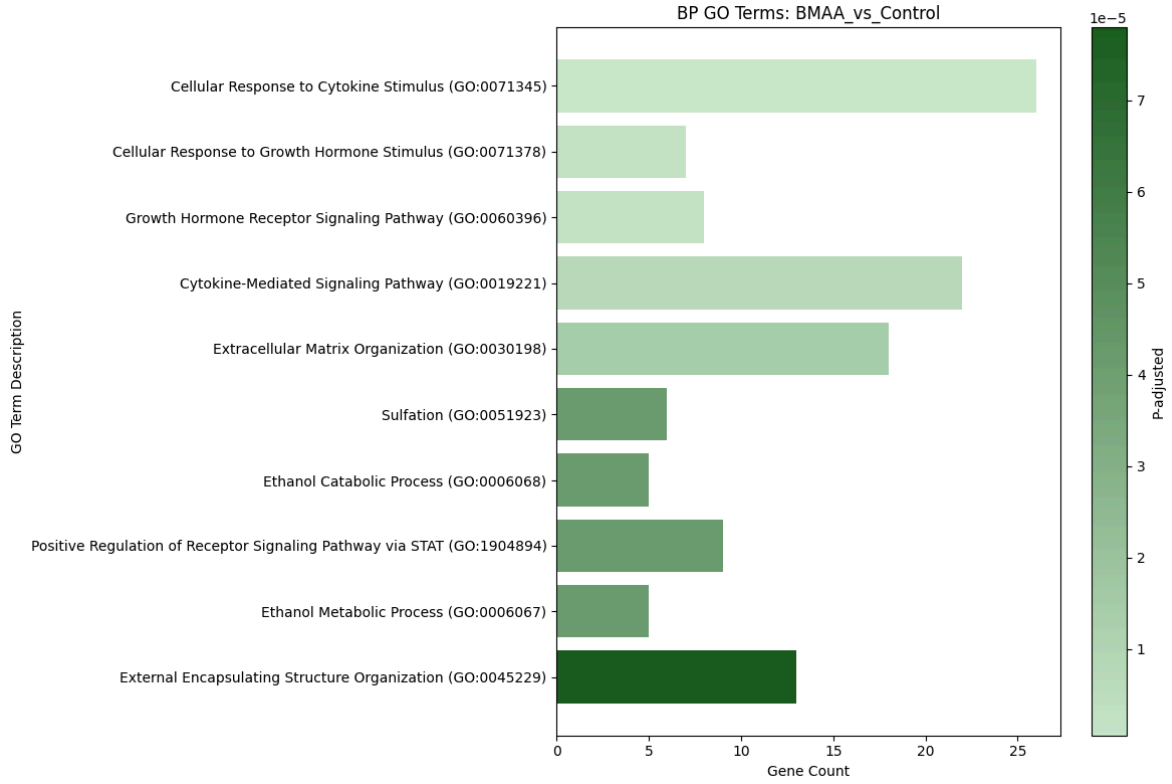
Figure 6. Heatmap of top differentially expressed genes.



Heatmap of the top 30 differentially expressed genes across all samples.

Rows represent genes and columns represent samples, clustered by expression pattern. Control and BMAA-treated samples separate into distinct groups, highlighting genes most influenced by the treatment. Colors indicate normalized expression levels (e.g., z-scores).

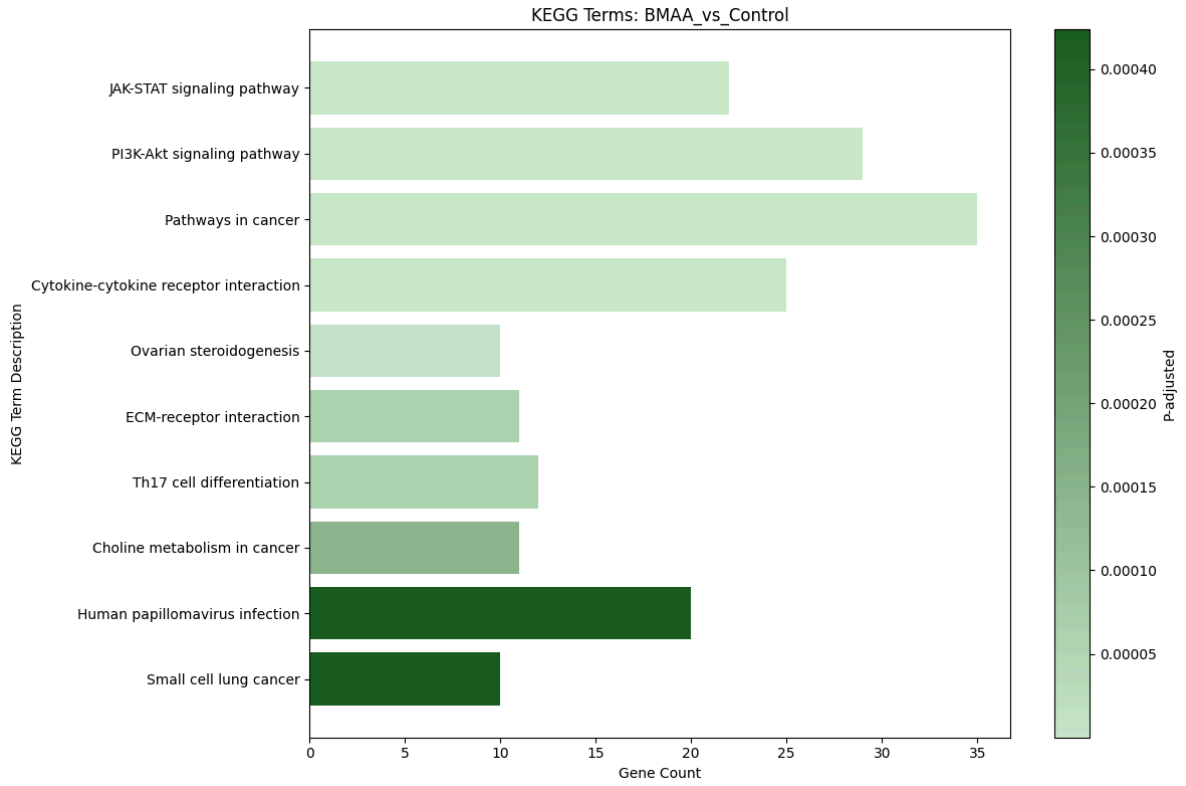
Figure 7. GO Enrichment: BMAA vs Control



Bar plot of enriched Gene Ontology (GO) biological processes for genes differentially expressed between BMAA and control.

Terms such as cytokine signaling and extracellular matrix organization are significantly enriched. Bar lengths represent the number of associated genes, and shading indicates adjusted p-values. These results highlight biological processes altered by BMAA treatment.

Figure 8: KEGG Pathway Enrichment: BMAA vs Control



Bar plot of enriched KEGG pathways for DEGs between BMAA and control conditions.

Enriched pathways include cancer signaling and immune response pathways, providing insight into the functional impact of BMAA exposure. Bars are colored by adjusted p-value, and pathway gene counts are shown on the x-axis.

Note: The zebrafish dataset used in this paper serves as a demonstrative test case. Full results and biological interpretation will be reported in a forthcoming paper focused on the effects of treating BMAA-exposed zebrafish models with novel therapeutics.

Data and Code Availability

Pipeline Source Code: https://github.com/Shaan7071/DEG-pipeline-assistant

This manuscript presents the DEG Pipeline Assistant using a representative RNA-seq dataset for demonstration only. The full dataset, complete results, and biological interpretation will be published in a separate study.

References

Fang, S., Liu, B., Zheng, D., & Huang, Y. (2021). GSEApy: a comprehensive Python package for gene set enrichment analysis. bioRxiv. https://doi.org/10.1101/2021.04.10.439370

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

McKinney, W. (2010). Data structures for statistical computing in Python. Proceedings of the 9th Python in Science Conference, 51–56. https://doi.org/10.25080/Majora-92bf1922-00a

OpenAI. (2023). OpenAI Python Library (Version 1.0.0) [Computer software].
Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Raudvere, U., Kolberg, L., Kuzmin, I., Arak, T., Adler, P., Peterson, H., & Vilo, J. (2019). g:Profiler: a web server for functional enrichment analysis and conversions of gene lists. Nucleic Acids Research, 47(W1), W191–W198. https://doi.org/10.1093/nar/gkz369

Ronacher, A. (2017). Click: Python composable command line interface toolkit. https://click.palletsprojects.com

Streamlit Inc. (2023). Streamlit: The fastest way to build data apps in Python. https://streamlit.io

Waskom, M. L. (2021). Seaborn: Statistical data visualization. Journal of Open Source Software, 6(60), 3021. https://doi.org/10.21105/joss.03021

Xin, J., Mark, A., Afrasiabi, C., Tsueng, G., Juchler, M., Gopal, N., Stupp, G. S., Putman, T. E., Ainscough, B. J., Griffith, O. L., Torkamani, A., Whetzel, P. L., Mungall, C. J., Mooney, S. D., Su, A. I., & Wu, C. (2016). High-performance web services for querying gene and variant annotation. Genome biology, 17(1), 91. https://doi.org/10.1186/s13059-016-0953-9