auto-Enrich: a pipeline for Streamlined Enrichment Analysis

José Miguel da Silva Ribeiro,
Master in Bioinformatics and Computational Biology,
Department of Biology | Department of Computer Science
Faculty of Sciences of the University of Porto
2025

auto-Enrich: a pipeline for Streamlined Enrichment Analysis

José Miguel da Silva Ribeiro,

Dissertation carried out as part of the Master in Bioinformatics and Computational Biology,

Department of Biology | Department of Computer Science

2025

Supervisor
Cristina Alexandra Gonçalves Paula Vieira, Associated Researcher, Institute for Research and Innovation in Health (i3S)
cgvieira@i3s.up.pt

Co-supervisors
Jorge Manuel de Sousa Basto Vieira, Principal Investigator, Institute for Research and Innovation in Health (i3S)
jbvieira@i3s.up.pt
Hugo López Fernández, SING Research Group, Escola Superior de Enxeñaría Informática (ESEI), University of Vigo
hfernandez@uvigo.gal

Agradecimentos

Á minha orientadora, Doutora Cristina, pela orientação, paciência e dedicação ao longo deste trabalho. Da mesma forma, ao coorientador, Doutor Jorge, pela vossa disponibilidade constante, pelos valiosos conselhos e pela confiança que sempre depositaram em mim foram fundamentais para o desenvolvimento desta tese. Mais do que orientadores, foram verdadeiros mentores, sempre prontos a guiar-me com rigor, mas também com sensatez e compreensão. A vossa contribuição foi essencial, para além deste projeto, para o meu crescimento académico e profissional. Muito obrigado por tudo.

Ao coorientador Doutor Hugo, pela disponibilidade ao longo deste trabalho. Mesmo nos momentos em que a contribuição direta não foi necessária, a sua abertura demonstrada para colaborar foi importante para o bom andamento do projeto.

Á minha família por apoiarem o trajeto que escolhi seguir, pelos incentivos aos meus estudos e percurso académico, e por alimentarem a minha ambição de procurar aquilo que realmente gosto de fazer. Espero que se sintam tão orgulhosos desta conquista como eu.

Á Inês, por ser o meu porto seguro. Por estar sempre ao meu lado, mesmo estando distante. Por me fazer acreditar em mim, mesmo quando eu duvidava. Por nunca desistir de mim, mesmo nos momentos em que tudo parecia mais difícil. E por me lembrares quem sou, quando o mundo tenta que eu me esqueça. Obrigado.

Finalmente, aos meus amigos, por estarem presentes e acreditarem em mim, por partilharem risos, desafios e conquistas, e por tornarem este percurso mais leve e significativo.

Resumo

A análise de enriquecimento é uma técnica fundamental em bioinformática, utilizada para identificar processos biológicos, vias metabólicas ou funções estatisticamente associadas a conjuntos de genes provenientes de experiências de alto rendimento. Apesar da existência de várias ferramentas e metodologias, como a Análise de Sobre-Representação (ORA), a Pontuação de Classes Funcionais (FCS) e a Análise de Vias Baseada em Topologia (TPA), a área continua a carecer de normalização. A variabilidade no tratamento dos dados de entrada, nas bases de dados usadas, nos métodos estatísticos e nos formatos de saída resulta frequentemente em resultados inconsistentes e não reprodutíveis. Para colmatar estas limitações, desenvolvemos o auto-Enrich, uma pipeline modular e containerizada que integra várias ferramentas populares de análise de enriquecimento, g:Profiler, PANTHER e GSEA, num enquadramento unificado e reprodutível. Construído maioritariamente em Bash e disponibilizado via Docker, o auto-Enrich simplifica os fluxos de trabalho ao automatizar as etapas de pré-processamento, análise de enriquecimento e pós-processamento. Aplica de forma consistente o mapeamento de identificadores genéticos, filtragem e correções estatísticas em todas as ferramentas utilizadas. O pipeline é altamente personalizável, permitindo aos utilizadores selecionar e configurar os módulos através de ficheiros de configuração dedicados. Os resultados são organizados em diretórios estruturados, facilitando a comparação e interpretação entre ferramentas. O auto-Enrich inclui ainda verificações automáticas, relatórios de erro e protocolos prontos a usar com conjuntos de dados de exemplo, apoiando tanto utilizadores iniciantes como avançados. A aplicação do auto-Enrich a dados biológicos reais demonstrou a sua capacidade de produzir resultados coerentes e biologicamente relevantes. Ao unificar a execução das ferramentas, reduzir a intervenção manual e garantir um tratamento de dados consistente, o auto-Enrich melhora a reprodutibilidade, escalabilidade e usabilidade da análise de enriquecimento. Em última análise, o auto-Enrich representa uma solução padronizada e extensível para a análise de enriquecimento de conjuntos génicos, promovendo resultados transparentes e interpretáveis em transcriptómica e noutras áreas da investigação ómica.

Palavras-chave: Análises de enriquecimento, g:Profiler, PANTHER, GSEA, pipeline, Docker

Abstract

Enrichment analysis is a key technique in bioinformatics, used to identify biological processes, pathways, or functions statistically associated with gene sets derived from high-throughput experiments. Despite the availability of many tools and methods, such as Over-Representation Analysis (ORA), Functional Class Scoring (FCS), and Topology-based Pathway Analysis (TPA), the field still lacks standardization. Variability in input handling, database sources, statistical methods, and output formats often results in inconsistent, non-reproducible outcomes. To address these issues, we developed auto-Enrich, a modular and containerized pipeline that integrates multiple popular enrichment analysis tools, g:Profiler, PANTHER, and GSEA, within a unified, reproducible framework. Built using mainly Bash and deployed via Docker, auto-Enrich simplifies enrichment workflows by automating pre-processing, enrichment analysis, and post-processing steps. It applies consistent gene identifier mapping, filtering, and statistical corrections across all tools. The pipeline is highly customizable, allowing users to select and configure modules through dedicated configuration files. Outputs are organized into structured directories, facilitating comparison and interpretation across tools. auto-Enrich also includes sanity checks, error reporting, and ready-to-use protocols with example datasets to support both novice and experienced users. Application of auto-Enrich to real biological datasets demonstrated its capability to deliver coherent and biologically meaningful results across tools. By unifying tool execution, reducing manual intervention, and enforcing consistent data handling, auto-Enrich improves reproducibility, scalability, and usability in enrichment analysis. Ultimately, auto-Enrich offers a standardized and extensible solution for gene set enrichment, supporting transparent and interpretable results in transcriptomics and other omics applications.

Keywords: Enrichment analysis, g:Profiler, PANTHER, GSEA, pipeline, Docker

Table of Contents

List of Tables

List of Figures

List of Abbreviations

APIs – Application Programming Interfaces

auto-Enrich – A pipeline for streamlined enrichment analysis

BH-FDR – Benjamini-Hochberg False Discovery Rate

CDS – Coding DNA Sequence

CSV – Comma-Separated Values

DAVID – Database for Annotation, Visualization and Integrated Discovery

DEGs – Differentially Expressed Genes

EASE – Expression Analysis Systematic Explorer

ES – Enrichment Score

FCS – Functional Class Scoring

FDR – False Discovery Rate

FGSEA – Fast Gene Set Enrichment Analysis

GCT – Gene Cluster Text file

GO – Gene Ontology

GO:BP – Gene Ontology Biological Process

GO:CC – Gene Ontology Cellular Component

GO:MF – Gene Ontology Molecular Function

GSEA – Gene Set Enrichment Analysis

GMT – Gene Matrix Transposed file

GUIs – Graphical User Interfaces

GWAS – Genome-Wide Association Studies

HGNC – HUGO Gene Nomenclature Committee

IDs – Identifiers

KEGG – Kyoto Encyclopedia of Genes and Genomes

MSigDB – Molecular Signatures Database

NES – Normalized Enrichment Score

ORA – Over-Representation Analysis

PANTHER – Protein ANalysis THrough Evolutionary Relationships

PEA – Pathway Enrichment Analysis

PSEA – Protein Set Enrichment Analysis

RNK – Ranked list file used in GSEA Preranked

SCI – Spinal Cord Injury

SNPs – Single Nucleotide Polymorphisms

TSV – Tab-Separated Values

TPA – Topology-Based Pathways Analysis

UniProtKB – Universal Protein Knowledgebase

1. Introduction

Enrichment analysis, also known as functional enrichment analysis or pathway enrichment analysis (PEA), is a powerful bioinformatics technique designed to interpret large lists of genes or gene products that are typically derived from high-throughput genomic, proteomic, or bioinformatics scanning approaches [1],[2]. It's an essential tool in omics research to reduce data complexity, improve interpretation, facilitate a deeper understanding of biological systems, by identifying which biological pathways, gene functions, or other annotations are statistically overrepresented (enriched)/ underrepresented (depleted) in a set of genes or genomic regions of interest compared to a background set or the entire genome [3]–[5]. The underlying principle of PEA is that if a particular biological process is affected in a study, then the genes involved in that process are more likely to be found together in a list of "interesting" genes (e.g., differentially expressed genes) than expected by random chance. This allows researchers to move from studying individual genes to understanding the broader biological context of their findings [2].

1.1. Background

Before the advent of enrichment analysis, researchers faced a challenging and daunting task when attempting to interpret the large lists of genes generated by high-throughput genomic and proteomic technologies [2]. Early approaches to understanding these extensive gene lists were characterized by selecting only a handful of high-scoring genes from the larger output, rather than examining the entire list, formulating interpretations that were often subjective and anecdotal, since they lacked the statistical rigor and systematic nature that enrichment methods provide [2],[6],[7]. It is it difficult to investigate data in a gene-by-gene manner from "overwhelmingly large lists of important, but isolated genes detached of biological context", because of the sheer number of associations made [8],[9]. The development of bioinformatics methods for enrichment analysis aimed to overcome these challenges by systematically dissecting large gene lists to summarize the most enriched and pertinent biology, moving from an individual gene-oriented view to a group-based analysis [2]. In the late 1990s, the exponential growth of biological data, particularly with the advent of genome sequencing, led to the improvement of gene annotation, available in different databases, such as Gene Ontology (GO), Ensembl, NCBI Gene, UniProt, Reactome and Kyoto Encyclopedia of Genes and Genomes (KEGG) Pathways. The use of different strategies to organize and

classify biological objects in databases, imply to cross-database results, to identify genes involved in a particular biological function, such as, for instance, identifying genes involved in epithelial sheet formation or DNA repair proteins [10]. In this work, a Docker pipeline is presented to perform such comparative approach.

1.1.1. Gene Annotations and Ontology

The lack of common understanding and interoperability among genomic databases led to the formation of the Gene Ontology (GO) Consortium [10],[11], one of the first projects to address the challenge of integrating vast amounts of biological information and, thus, standardize the description of gene product functions across diverse organisms. Three extensive ontologies to describe gene products, namely Molecular Function (GO:MF), Biological Process (GO:BP) and Cellular Component (GO:CC), have been proposed to represent fundamental information common to all living forms. The common understanding of term meanings cross-database has provided the scientific community resources with standard vocabulary on annotated gene products, along with rules and methods for consistent annotation [10],[12]. The GO Consortium, was founded in 1998, starting with a few model organism databases and expanding over time [11]. As biology and technology evolved, the GO project incorporated more precise relationships between terms, developed methods to infer annotations based on evolutionary data, and adopted formal logic systems to ensure consistency and scalability [13],[14]. In recent years, the Consortium introduced a more expressive framework for modelling gene functions in biological context, strengthened quality control efforts to refine and update annotations, and improved tools for visualization and usability [15],[16]. Overall, the Gene Ontology's structured vocabularies and comprehensive annotations have had a profound impact on enrichment analysis being widely used allowing researchers to identify biologically meaningful themes within large gene sets (e.g. lists of genes identified from a microarray experiment or a cancer study [17]), developing semantic standards for biological data [15].

1.1.2. Pathways and biological processes

Even before Gene Ontology, one of the first functional classification systems was created in 1993 by Monica Riley for *Escherichia coli*. This system served as a prototype for Gene Ontology, and was applied in databases like EcoCyc (based on the metabolism and biochemistry of *E.coli*) and MetaCyc (based on pathways and enzymes from various organisms, with a microbial focus and structured according to an ontology and that

include information on transport and genetic regulatory networks) [10],[12],[18]. These databases applied classification systems that supported searching for genes by physiological rules [19]. Databases such as KEGG (Kyoto Encyclopedia of Genes and Genomes), created in 1995, before the formal launch of Gene Ontology, had the initial goal of linking genomic information to higher-order functional information, computerizing current knowledge about cellular processes and standardizing gene annotations [20],[21]. Moreover, other databases such as Reactome, first announced in 2003, explicitly integrate GO terms into its annotations and data models, thus facilitating interoperability with other bioinformatics resources such as UniProt, Ensembl and Entrez Gene. [22]. Nowadays, these databases are some of the most influential and widely used, together with Gene Ontology, for enrichment analyses [23]. Important ontology-driven structure databases, like PANTHER Pathways and WikiPathwyas, that use controlled vocabulary and standards for data modelling, entry and display, are freely available infrastructure for community curation [24],[25].

1.1.3. First bioinformatics tools and methods

The early 2000s marked the emergence of bioinformatics tools designed to interpret large gene lists derived from microarray and proteomics experiments. These tools sought to map gene-level results onto biological themes using statistical enrichment techniques [26]–[28]. Pioneering tools such as EASE (Expression Analysis Systematic Explorer), GeneMerge, and Onto-Express exemplify the first generation of enrichment analysis methods. These first approaches are collectively known as Over-Representation Analysis (ORA) and assess whether functional categories (e.g., GO terms or pathways) are enriched in a list of differentially expressed genes more than expected by chance [26],[28],[29]. These method required a threshold-defined list of significant genes and consistently relied on statistical tests rooted in the hypergeometric distribution, most notably Fisher's Exact Test, the earliest and most widely adopted method for threshold-dependent enrichment analysis [26],[28],[29]. EASE, released in April 2003, applied both the one-tailed Fisher's Exact Test and a modified version called the EASE score, which penalized categories supported by few genes to reduce false positives. EASE also incorporated multiple testing corrections, including Bonferroni-type and bootstrap methods [28]. Similarly, GeneMerge (2003) used the hypergeometric distribution to score category Over-Representation and applied a modified Bonferroni correction to address the multiple comparisons problem [29]. Onto-Express, part of the Onto-Tools suite introduced in 2003, mapped gene lists to functional profiles using Gene Ontology and

applied a tiered statistical strategy: chi-squared test by default, Fisher's Exact Test when expected values fell below five, and the binomial test as an approximation of the hypergeometric distribution for typical microarray scenarios [26]. Threshold-based methods, while effective, had key limitations, chiefly their binary gene treatment and sensitivity to cutoff choices [6].

These limitations led to a second generation of ranking-based or Functional Class Scoring (FCS) methods, where gene lists were evaluated in a ranked fashion [5]. The most prominent example is Gene Set Enrichment Analysis (GSEA), introduced conceptually by Mootha et al. (2003) [30] and formalized by Subramanian et al. (2005) [4], which employs a running-sum statistic based on the Kolmogorov-Smirnov statistic to assign an enrichment score to a gene set [31]. Later innovations included Random Set (RS) methods[32] and Biological Module-Centric Classification, such as DAVID (Database for Annotation, Visualization and Integrated Discovery) gene functional clustering approach using kappa statistics and fuzzy partitioning [27].

The third and "last" generation of pathway enrichment analysis methods, the Topology-based Pathways Analysis (TPA), began to emerge in the mid-2000s. These methods marked a significant shift by incorporating the topological structure of biological pathways, going beyond simple gene set membership which largely ignored gene position, interaction types, and signal propagation within pathways [33]. Early implementations like Pathway-Express and TAPPA introduced the concept of "impact analysis," laying the groundwork for this approach [34]. In the early 20s, numerous tools such as SPIA, TopoGSA, and TopologyGSA integrating both enrichment statistics and network perturbation effects have been developed [33]. This field expanded rapidly, and in 2013 more than 30 topology-based tools were available [35]. Methods like DEGraph and CePa, further solidified this approach as a critical advancement [34]. Today, topology-based methods are recognized alongside over-representation analysis and functional class scoring as one of the three major generations of PEA, offering improved biological insight by accounting for pathway structure and gene interactions [36].

1.2. Pathway Enrichment Analysis techniques

A typical enrichment analysis starts with an input list of genes or genomic regions identified as relevant from high-throughput experiment (e.g. microarray, RNA-seq, proteomics, ChIP-seq) [2]. These genes or regions are then mapped to annotations in

biological databases, such as Gene Ontology, KEGG pathways, or other functional classification systems. For each annotation term (e.g. a specific GO term or a pathway), a statistical test is performed to determine whether the number of genes from the input list associated with that term is statistically significant overrepresented/underrepresented than expected based on the background gene population (e.g., the entire genome or the genes measured in the experiment). Common statistical methods include the Chi-square test, Fisher's exact test, Binomial probability, and Hypergeometric distribution [4],[11],[27],[37]. Since many annotation terms are tested simultaneously, corrections for multiple hypothesis testing (e.g. Bonferroni, Benjamini-Hochberg) are often applied to control the false-positive rate [2],[9],[38],[39]. The results are typically presented as a list of annotation terms ranked by their enrichment/depletion significance (e.g., *P*-values or adjusted *P*-values), indicating the biological processes or functions that are most likely to be relevant to the study [2].

Different omics technologies, such as transcriptomics, proteomics, metabolomics, genome-wide association studies (GWAS), and epigenomics, generate datasets with distinct characteristics that require tailored enrichment analysis approaches. Traditionally, Over-Representation Analysis (ORA) and Gene Set Enrichment Analysis (GSEA) are among the most widely adopted methods for enrichment analysis, commonly used to interpret high-throughput biological data [2],[5],[23],[40]. Their widespread use stems from distinct advantages and historical precedence in the field. Besides this, different approaches are used in proteomics data, characterized by high variability, requiring methods like protein set enrichment analysis (PSEA) and PSEA-Quant that consider protein abundance and variation [5]. Metabolomics being sparser and more ambiguous data need methods like Metabolomics Pathway Analysis (MetPA) that use compound annotations and network topology, or Mummichog that predicts pathways based on spectral features without prior compound annotation [5]. GWAS focuses on single nucleotide polymorphisms (SNPs), which need to be mapped to genes before enrichment analysis, methods like SNP-GSEA and regression with summary statistics enrichment analysis (RSS-E) are typically used, considering the uneven distribution of SNPs [5]. Epigenomics data, such as DNA methylation profiles, can be biased towards long genes and multigene mapping of CpG sites. Methods like ebGSEA and GOmeth address these issues by ranking genes by methylation change or by using empirical analysis with CpG sites, respectively [5].

1.2.1. Over-Representation Analysis (ORA)

Over-Representation Analysis (ORA) also known as overlap-based methods or singular enrichment analysis, typically involves taking a user-selected list of genes (e.g. differentially expressed genes) into a more concise set of impacted biological functions or processes [2],[41],[42]. This approach shifts the analysis from an individual gene-oriented view to a relevant gene-group-based analysis, increasing the likelihood of identifying biological processes that are most pertinent to a study. The core principle is to identify pathways or functional categories where the number of selected molecules is higher than what would be expected by random chance [41]–[43].

ORA methods require three essential inputs to perform a meaningful statistical evaluation. First, a preselected list of "interesting" genes, often those identified as differentially expressed (DEGs) or statistically significant based on criteria such as *P*-value thresholds (e.g. *P*-value < 0.05) or fold-change cutoffs [41],[42],[44],[45]. Second, a curated collection of molecules (e.g. genes, metabolites) grouped into functional categories, such as molecular functions, biological processes or signaling and biochemical pathways [41],[46], typically sourced from databases like Gene Ontology (GO) [16], KEGG [47], Reactome [48], BioCyc [49], DisGeNet [50], MSigBD [4] and WikiPathways [51]. The third is a background or reference set, representing the full set of molecules that could realistically be measured in the experiment. ORA assesses whether the proportion of DEGs in a given functional category is statistically higher than what would be expected based on that category's proportion in the entire background set. This comparison determines whether the observed enrichment is likely due to biological signal rather than random chance [41],[45]. ORA methods determine significance by comparing ratio of significantly expressed genes within a specific pathway to the total number of genes in that pathway, and contrasting this with the overall proportion of significantly expressed genes across all pathways in the database [45]. This comparison typically involves testing the proportion of DEGs in a functional gene set against a discrete probability distribution [2],[41],[44]. Common statistical approaches include the Fisher's Exact Test, which uses the hypergeometric distribution to assess whether the fraction of DEGs in a pathway is greater than expected relative to the background set [52]; direct application of the hypergeometric distribution itself to calculate the probability of observing a given number of genes from a pathway in the DEGs list by chance [41],[43],[46]; and tests like the chi-square and binomial probability, with the latter suited for large background populations and the former more appropriate for smaller datasets [2],[46].

Numerous pathways and gene sets can be tested simultaneously [9],[39]. For example, a single analysis can involve hundreds to thousands of parallel tests, one for each gene set in a library (e.g., MSigDB v2025.1.Hs contains 35134 sets [53]). This substantially increases the probability of making at least one Type I error (false positive across the entire set of tests), known as the experiment-wise error rate or family-wise error rate [38],[54]–[56]. Therefore, to reduce the chance of false positives, multiple testing corrections approaches such as Benjamini-Hochberg False Discovery Rate (BH-FDR) or the Bonferroni Correction must be used [9],[38]. The first methodology, FDR, introduced by Benjamini and Hochberg, controls the expected proportion of false positives among all rejected null hypotheses [54],[56]. The Bonferroni Correction is a simple and widely used method where the significance level is divided by the total number of tests (N). Each individual test's *P*-value must then be less than this adjusted threshold to be considered significant [54],[56]. While easy to implement, the Bonferroni method is often considered overly conservative, especially when the tests are not independent (e.g., due to linkage disequilibrium among SNPs in genetic studies or correlations among genes in a set), that can lead to a loss of statistical power, increasing the risk of false negatives [56],[57]. Considering this, the BH-FDR is often preferred in 'omics' studies, where the number of tests is very large, because it provides a notable gain in statistical power compared to family-wise error rate control (like Bonferroni Correction), especially when many true associations are anticipated [56],[58]. However, these methods often assume independence among tests, which is typically not true for biological pathways due to overlapping genes and crosstalk, potentially leading to inaccurate but still useful FDR estimates [42].

ORA is often favoured for its simplicity and speed compared to more complex methods, such as Functional Class Scoring approaches, for example, Gene Set Enrichment Analysis [39]. A basic ORA can involve simply pasting a list of gene symbols into a web-based tool and obtaining immediate results. This ease of use has contributed to its widespread adoption and high citation rate in scientific literature. However, despite the emergence of more sophisticated techniques, ORA continues to be recommended for exploratory analyses involving simple gene lists or when genes are classified into binary groups, such as gene expression modules or genes carrying variants associated with disease [39],[44],[59]. This simplicity is accompanied by a number of inherent limitations. For example, it ignores the magnitude of gene expression changes, treating all genes equally. ORA focuses solely on the number of genes rather than incorporating quantitative data such as fold-changes, probe intensities, or the individual gene significance. This can result in the loss of potentially meaningful data, particularly for

genes that are marginally less significant [37],[59]. Another key limitation is ORA sensitivity to arbitrary cutoffs. The method relies on a user-defined threshold to identify differentially expressed genes, making its outcomes highly dependent on this selection. Genes just below the threshold may be excluded, which reduces sensitivity and risks overlooking subtle but biologically relevant associations, something that more nuanced methods like FCS and pathway topology-based approaches handle more effectively [9],[41],[59]. Moreover, ORA assumes that genes function independently. This is a flawed assumption in biological contexts, where gene products often operate within complex, interrelated pathways. By relying on a competitive null hypothesis, ORA fails to account for the inherent correlation structure among genes, potentially leading to biased or inaccurate significance estimates [9],[39],[44]. ORA presumes that pathways are independent of one another, which does not reflect biological reality. Many genes participate in multiple pathways, creating overlaps and interconnections. This is especially evident in the hierarchical structure of Gene Ontology (GO) terms. Consequently, when one pathway is significantly enriched, others that share genes with it may also appear significant, complicating interpretation [37].

Despite these challenges, Over-Representation Analysis remains one of the most widely implemented approaches in popular tools and web services today due to its simplicity and ease of use [60].

1.2.2. Gene Set Enrichment Analysis (GSEA)

Gene Set Enrichment Analysis (GSEA) is a widely used method that belongs to the category of functional class scoring approaches. These methods evaluate differential expression at the level of gene sets rather than individual genes, by first computing a differential expression score for each gene and then aggregating these scores to assess enrichment for a given gene set [61]**.** Unlike methods that rely on predefined thresholds for significance, GSEA aims to uncover biological functions or pathways that show coordinated shifts in gene expression, thus providing a systems-level view of gene activity under different conditions [2],[4],[31].

GSEA operates on a ranked list of all genes from an experiment, typically sorted by a metric reflecting differential expression, such as the signal-to-noise ratio or t-statistic [4],[62]. Because it uses ranked data, GSEA does not require an arbitrary cutoff to define "significant" genes, allowing it to capture subtle but coordinated changes that may be missed by threshold-dependent approaches [2],[63]. The method evaluates predefined

gene sets, often sourced from databases such as Gene Ontology, KEGG, or prior experimental studies. A commonly used collection is the Molecular Signatures Database (MSigDB), which provides curated gene sets designed for enrichment analysis [4],[42]. For each gene set, GSEA calculates an Enrichment Score (ES) using a modified Kolmogorov-Smirnov statistic that quantifies whether the genes in the set are preferentially located at the top or bottom of the ranked list [4]. This is done through a running-sum statistic, which increases when a gene from the set is encountered and decreases otherwise. The resulting score is then normalized to account for differences in gene set size, yielding the Normalized Enrichment Score (NES) [4]. To determine statistical significance, GSEA uses permutation testing. In phenotype permutation, sample labels are randomly shuffled, and the NES is recalculated to generate a null distribution. Alternatively, gene set permutation compares the observed NES to the distribution obtained from random gene sets of equivalent size, which is particularly useful when sample sizes are small or unbalanced. *P*-values derived from these tests are corrected for multiple hypothesis testing, typically using False Discovery Rate procedures. GSEA reports *Q*-values, which reflect the proportion of false positives and are estimated using the median of the null distribution, in contrast to other approaches that may use extreme values [4].

A key advantage of GSEA is that it evaluates the entire ranked list of genes, incorporating information from all measured genes, not just those that pass arbitrary significance or fold-change thresholds [37],[63]. This comprehensive approach reduces the risk of discarding potentially relevant genes that fall just below traditional cutoffs, thus minimizing information loss [37]. By avoiding predefined thresholds, GSEA also limits subjective biases in gene selection and ensures that subtle but coordinated changes contribute meaningfully to the analysis [2]. Another strength of GSEA is its ability to account for gene dependencies and correlations within pathways, a reflection of true biological interactions. Through phenotype-based permutation testing, GSEA preserves the underlying correlation structure between genes during significance estimation [37],[64]. In addition to ranked gene lists, GSEA can also be applied to unranked data, although it is particularly effective when ranking metrics are available genome-wide, such as in RNA-seq experiments [42]. Furthermore, GSEA is capable of detecting weak, one-sided deregulation signals and can perform robustly even when gene sets include both upregulated and downregulated genes, depending on the statistical approach used [31],[46].

Despite its many strengths, GSEA has several important limitations that span statistical, interpretative, and practical dimensions. Its reliance on a modified

Kolmogorov-Smirnov statistic limits sensitivity and statistical power, as the null distribution remains largely unaffected by increasing sample size, making factors like gene set size and data dimensionality more influential [46],[64]. Consequently, other factors, such as the number of measured attributes or the size of the gene set, play a more critical role in shaping statistical power [65]. The empirical estimation of the null distribution through permutation tests introduces potential biases, often producing non-uniform or even zero *P*-values, which undermine reliability [39]. GSEA default approach to calculating FDR *Q*-values, using the median of the *P*-value distribution, can suppress strong signals and reduce power unpredictably [61]. While GSEA accounts for gene-gene correlations via a subject-sampling null hypothesis, this doesn't fully resolve issues arising from complex, noisy, and context-dependent gene expression patterns, often leading to inflated false positives [9],[64]. Additionally, GSEA struggles with gene sets containing both up- and down-regulated genes, as it isn't optimized for capturing two-sided deregulation [31]. The method's sensitivity can lead to the detection of secondary or overlapping pathways, which may not reflect true biological signals. From a data standpoint, GSEA requirement for a single summarized metric per gene can oversimplify complex datasets, potentially discarding valuable biological inferences [2]. Lastly, reproducibility remains a concern due to inconsistent reporting of critical analytical settings—such as ranking metrics, gene weighting, and permutation strategies [9]. While faster implementations like Fast Gene Set Enrichment Analysis (FGSEA) [66] exist, the original GSEA framework can be computationally intensive, particularly when deep permutations or large gene set collections are involved. [39].

Despite its limitations, GSEA remains highly relevant and continues to be a standard method due to its conceptual simplicity, biological interpretability, and broad software support. It is still implemented in most popular tools and web services today [60].

1.2.3. Topology-Based Pathway Analysis (TPA)

Unlike the other methods, which treat pathways as simple lists or sets of genes, Topology-based Pathway Approaches (TPA) integrate complex gene interactions and structure within pathways [33]. Their fundamental goal is to incorporate the knowledge embedded in pathway diagrams about how various genes interact and regulate each other [34]. Therefore, such approaches aim to provide a more precise assessment of statistical relevance and decipher biological causal relations between pathway components [67]. By incorporating pathway topology, they offer improved sensitivity and

specificity in detecting significantly impacted pathways and provide a "systems view" of biological questions, reducing the complexity of omics data [36]. These methods typically model biological pathways as graphs, where nodes represent genes, proteins, or metabolites, and edges symbolize interactions. They process experimental data (e.g., gene expression) and pathway graph information to compute a score that quantifies changes in a pathway. The scoring can be statistical or a method-specific metric, resulting in a ranked list of pathways [34]. For this, two main approaches are employed: node-level scoring approaches, also known as hierarchically aggregated models, and pathway-level multivariate modelling, or multivariate scoring [34]. The first approach, hierarchically aggregated models, typically model biological pathways as graphs, where nodes represent genes, proteins, or metabolites, and edges symbolize interactions. They compute individual scores for genes or nodes by considering their expression values in the context of their position in the pathway graph and then aggregate those to obtain a pathway-level score, followed by significance assessment [34]. The second approach, pathway-level multivariate modelling, models the entire pathway or gene set as a multivariate distribution informed by the pathway's topology, and test whether these distributions significantly differ across conditions [34].

These methods generally rely on two major types of input: experimental data and pathway topology information. The experimental data usually comes from high-throughput technologies such as RNA sequencing, microarrays, or proteomics. This data can be provided in various forms. One common input is a complete expression matrix, containing normalized or transformed values (such as log fold-changes or *P*-values) for all genes or metabolites measured in the study [34]. Alternatively, some approaches use lists of differentially expressed genes (DEGs) or biomolecules, which are identified based on thresholds for statistical significance or fold change. In some cases, continuous metrics like log fold-changes or *P*-values may be used directly, without pre-selecting DEGs, allowing the analysis to consider the full spectrum of expression changes [34],[36]. For metabolomics-based pathway analysis, input may include concentration tables or lists of key metabolites instead of gene expression values. The second critical input is pathway topology, which provides the structural context needed for the analysis [34]. These structures are derived from curated pathway databases that offer standardized representations of biological processes.

Topology-based approaches in enrichment analysis offer notable advantages but come with important limitations. Their key strength lies in leveraging the structural complexity of biological pathways, which enables more sensitive and specific detection

of relevant biological signals compared to non-topology-based methods [36]. By incorporating pathway interconnections, gene positions, and signal flow directionality, tools like SPIA and NetGSA provide a richer, more nuanced understanding of pathway activity and potential causal mechanisms [34]. Methods such as NetGSA also stand out for their ability to reconstruct incomplete networks using Bayesian inference, integrate multiple omics data types, and handle multi-condition comparisons, offering flexibility across varied experimental designs [67]. Some tools, like TPEA, have demonstrated superior stability across datasets and can detect novel pathways missed by conventional methods [68]. Visualization capabilities further aid in the biological interpretation of complex networks [31]. However, these benefits come with trade-offs. There is a lack of systematic benchmarking, and method performance can vary based on pathway overlap, data type, and assumptions [69]. Many tools rely on the presence of differentially expressed genes, limiting their use in scenarios with subtle expression changes or post-translational regulation [36]. Topology requirements, such as acyclic structures or identical networks across conditions, can conflict with biological reality, leading to biased or oversimplified models. Assumptions like linear correlation or biomolecule independence are often violated, compromising statistical validity [67]. Computational demands can be high, especially for large-scale datasets, and some methods struggle with outdated or incomplete pathway definitions [69]. Graph modelling challenges, such as handling cycles or complex hierarchies, further complicate analysis, while the lack of standardized outputs and potential bias under the null hypothesis hinder result interpretation and comparison [34]. Overall, while topology-based approaches enrich the biological relevance of pathway analysis, their complexity, assumptions, and practical constraints must be carefully weighed against the specific goals and context of each study [35].

Although topology-based enrichment methods give detailed biological inferences, they are underutilized by modern PEA tools due to complexity, data requirements, limited tool support, and lack of community familiarity [36],[70].

1.3. Overview of Available Tools

The field of pathway enrichment analysis has expanded rapidly since the first comprehensive reviews in 2009. Early reviews by Huang et al. (2009)[2], Khatri et al. (2012)[37] and Mitrea et al. (2013)[34] documented 68, 33, and 22 additional tools respectively, totalling at least 123 unique PEA tools at that time. More recent systematic analyses, such as the review by Xie et al. (2021)[60], have reported over 300 tools and

methods, reflecting continued growth and diversification in the field. The majority of widely used PEA tools today implement two main methodological approaches: Over-Representation Analysis (ORA) and Functional Class Scoring (FCS). Examples of ORA tools include g:Profiler, clusterProfiler, Enrichr, PANTHER, BiNGO, WebGestalt, ClueGO, and DAVID, while FCS methods are exemplified by GSEA and supported by tools such as GSEA (Broad Institute), clusterProfiler, GOrilla, KOBAS, GSVA, WebGestalt, SSGSEA, and GSEA-P [60]. The number of available tools continues to evolve, and different tools support different multiple analytical approaches. Nevertheless, ORA and FCS remain the foundational methodologies in current pathway enrichment analysis.

The wide variety of pathway enrichment tools offers flexibility but also creates a "confusing plethora" of options, making tool selection dependent on several key factors [2],[23]. One of the most critical factors is the choice of underlying databases. Differences in gene and pathway annotation sources can lead to substantially different results across tools [45]. Tools also differ in input and output formats, which affects ease of use and compatibility with downstream workflows. Accurate mapping of gene or metabolite identifiers to annotation databases is essential, mismatches can severely compromise data quality and analysis outcomes [3]. Another key consideration is how users interact with the tool. Some provide Graphical User Interfaces (GUIs) for manual exploration, while others support automation through Application Programming Interfaces (APIs), offering more flexibility in large-scale studies [41],[60]. Species coverage and the breadth of supported functional annotations also influence a tool's suitability for specific biological systems [71]. Finally, integration into broader research workflows is increasingly important. Many tools now offer web services, R or Python packages, or command-line interfaces, enabling seamless integration into automated pipelines and scalable analyses.

This variability in maintenance, combined with differences in underlying methodologies (ORA versus FCS), the choice of gene and pathway annotation sources, species coverage, and output formats, means that results can differ substantially depending on the tool selected. Consequently, this lack of standardization and varying update frequencies force researchers into a challenging situation where choosing an enrichment analysis tool can lead to inconsistent, non-reproducible, and sometimes contradictory results. The differences in methodologies, database versions, and software maintenance create uncertainty and complicate the interpretation of biological data, ultimately hindering reliable insights and slowing scientific progress.

1.4. Current Challenges

Despite its widespread use in high-throughput studies, pathway enrichment analysis remains vulnerable to methodological flaws and human error, undermining the reliability, interpretability, and reproducibility of its results.

Regardless of the algorithm used, the quality of enrichment outcomes is heavily dependent on the quality of input expression data [71]. Poor mapping of user-submitted gene identifiers to annotation databases can severely impair downstream analysis [2]. Identifier conversion across systems often causes information loss. For example, unique compounds may be generalized into broad chemical classes (e.g., specific lipids reduced to a class in KEGG). This issue is compounded by redundant identifiers across databases [3].

A critical, yet common, mistake is failing to use an appropriate background gene list when comparing to the foreground list [9],[59]. Since many genes may be undetectable in a given assay, skipping this step introduces significant bias and inflates false positives. Alarmingly, only 4.1% of published studies using ORA correctly defined or applied a suitable background list[41]. Another widespread oversight is neglecting multiple testing correction. Even when applied, it's often too conservative, reducing sensitivity [39],[72]. Only 54% of surveyed ORA studies used any *P*-value adjustment, increasing the likelihood of false discoveries [9].

Database choice also significantly affects results. Outdated annotations and analytical methods can inflate false positives and negatives [3],[35]. Many databases are incomplete, skewed toward well-studied genes, and slow to incorporate findings from newer high-throughput technologies [37],[39]. These gaps can lead to missed biological associations. Pathway definitions vary widely across databases, and evolve over time, thus, creating inconsistencies in enrichment results [23]. Databases often contain redundant or ambiguous entries for genes and metabolites, reducing analytical precision. Gene set overlap, where genes appear in multiple pathways due to multifunctionality or hierarchy, is largely ignored in most methods, further compromising specificity [35],[72]. Finally, pathway size itself can bias results, with smaller pathways often producing more statistically significant *P*-values [43].

Reproducibility, usability, and interpretability remain major challenges in enrichment analysis. Many studies fail to report key methodological details, such as the tools, software versions, and gene sets used, making it difficult to reproduce results. In methods like GSEA, critical parameters including gene weighting, permutation type, and

ranking method are often omitted [9]. Enrichment analysis is best treated as an exploratory technique, not a definitive statistical test. Results require careful interpretation, and the choice of *P*-value thresholds is often arbitrary. Conclusions depend not just on statistics, but on the investigator's biological knowledge, the choice of databases, algorithmic implementations, and subjective interpretation. Users still play a central role in deciding which enriched terms matter, often dismissing statistically significant findings if they don't align with prior knowledge, introducing bias in the process.

Cross-study comparisons are also difficult to perform. Absolute enrichment *P*-values are hard to compare across gene lists of differing sizes, and different tools can yield conflicting results due to variations in implementation [59],[71]. Therefore, using multiple tools and synthesizing a consensus signature is recommended.

The lack of a standardized, gold-standard test set further hinders objective benchmarking. Existing comparisons have been sparse, small-scale, and inconsistent, offering little clarity on which methods perform best [60]. This adds to the confusion users face when selecting enrichment tools.

1.5. Advantages of Pipelines

Pipelines are structured workflows that automate the transformation of raw data into meaningful results. Each step typically performs a specific task, such as data cleaning, transformation, analysis, or visualization, and is executed in a consistent, repeatable manner. By automating these steps, pipelines minimize human error, reduce manual workload, and ensure methodological consistency. Reproducibility is central to scientific research. Pipelines help achieve it by standardizing workflows, so analyses can be reliably repeated and validated across different teams and environments [73],[74]. Unlike manual processes, which are prone to variability, pipelines enforce uniform execution, enhancing result integrity. Automation is another major advantage. Many analytical tasks are repetitive and error-prone when done manually. Pipelines eliminate this burden, freeing researchers to focus on interpretation and insight rather than data handling. Transparency is also improved. Every step in a pipeline is explicitly defined and documented, creating a clear record of how results were generated. This documentation is crucial for verifying methods, enabling others to replicate the analysis, and maintaining scientific rigor. Modern research often deals with massive datasets, from genomics to neuroimaging, and pipelines are built for scalability. They can efficiently handle datasets

of varying size and complexity, adapting to the evolving demands of scientific inquiry. Pipelines also promote collaboration. Teams can divide and integrate tasks within a shared framework, ensuring alignment and workflow continuity across contributors. This shared structure is vital for large, multi-disciplinary projects. Finally, pipelines support version control. Changes can be tracked, compared, and reverted when needed, which is essential when new data or methods are introduced. This maintains analytical integrity and provides a transparent history of the research process.

In sum, pipelines are foundational tools in scientific analysis. They enable consistent, scalable, and transparent workflows; support collaboration and reproducibility; and provide the structure and documentation necessary for credible, efficient research. Their role is indispensable in ensuring that scientific findings are not only robust but also repeatable and trustworthy.

1.6. Containerization with Docker

Docker[1] is an open-source platform that streamlines the building, packaging, and deployment of applications [75],[76]. Since its public release in 2013, Docker has become the leading container technology, allowing applications and their dependencies to be bundled into standardized containers that run in isolation on the host operating system kernel [75]. Developers use Dockerfiles to define build instructions, which are executed with the “docker build” command to create Docker images. These images are stored in registries such as Docker Hub, a public repository where users can share and access pre-built images. Containers launched from these images provide consistent, lightweight environments that can run across development, testing, and production systems without modification [76]. Benefits of Docker include speed, portability, scalability, and rapid deployment [77]. Containers are quick to spin up, maintain performance across environments, and offer predictable, reliable deployment, essential qualities for scientific computing and bioinformatics, where reproducibility and modularization are critical. However, Docker also presents challenges, such as a lack of standardized image distribution and potential inconsistencies caused by version mismatches or modified scripts [78]. In bioinformatics, containerization tools like Nextflow[79], Snakemake[80], and Targets[81] further enhance modular workflow design. Nextflow is a workflow management system that supports portable and reproducible pipelines through Docker integration, isolating each pipeline step to ensure consistent environments across runs

[1] https://www.docker.com/

and platforms [79]. It uses a domain-specific language that's flexible and resource-aware, making it effective for managing complex, high-throughput sequencing workflows. Similarly, Snakemake provides scalable, reproducible workflow management tailored to scientific data analysis [80]. Inspired by Make, it automates dependency handling, supports parallel execution, and integrates smoothly with Docker to run each task in a controlled environment. Targets, built in R, simplifies the creation and maintenance of data analysis pipelines by explicitly defining task relationships and using Docker to isolate each step [81]. For the auto-Enrich pipeline, Docker is a strong choice due to its user-friendly interface, widespread adoption, and integration with workflow tools. Its simple commands and configuration make it accessible to users new to container technologies, and Docker Hub's extensive library of images allows for rapid deployment of preconfigured tools. While Nextflow, Snakemake, and Targets excel at managing workflows, Docker's compatibility with them makes it a versatile and powerful platform. By adopting Docker for auto-Enrich, we gain a flexible, reproducible setup for enrichment analysis. The building of pipeline modules inside a unique Image eliminates the need for manual installation and configuration of each one. Building a Docker image simply involves writing a Dockerfile that defines the base system and installation steps. Once created, the image can be easily expanded and automated into workflows, boosting efficiency and ensuring reproducibility in bioinformatics research. The portability, scalability, speed, resource efficiency and ease of maintenance favorites the choice of Docker to build the auto-Enrich pipeline.

1.7. Motivation

There is a pressing need to establish standardized practices in enrichment analysis to improve reproducibility, usability, and interpretability. Many studies lack critical methodological details making it difficult to reproduce or interpret results (see for instance [59],[71]). The absence of consistent standards hampers cross-study comparisons, as different tools and input sizes yield conflicting or incomparable results. Moreover, the lack of a gold-standard benchmark for evaluating tool performance leaves users with little guidance on which methods to trust.

In response to these challenges, auto-Enrich integrates several widely used enrichment analysis tools, including g:Profiler, PANTHER, and GSEA, to promote more consistent, transparent, and interpretable results. Since no single tool is universally optimal, combining multiple approaches offers a more comprehensive and reliable

analysis [59],[71]. This pipeline acts as a centralized access point for running enrichment analyses across different methods simultaneously, while ensuring uniform handling of inputs, parameters, and outputs. It standardizes gene ID translation and input formatting for each tool and produces harmonized output files that enable clear, side-by-side comparisons. To ensure fair and consistent significance metrics, the Benjamini-Hochberg FDR correction is applied uniformly across all analyses. Built on a modular architecture, auto-Enrich supports the integration of additional tools, databases, or custom components, allowing users to tailor workflows and extend functionality over time. The platform is distributed as a Docker image, ensuring easy deployment across different operating systems without complex setup, making it accessible to both computational and experimental researchers. By unifying multiple enrichment tools within a single, reproducible framework, auto-Enrich streamlines analysis, simplifies benchmarking, and reduces the subjectivity and fragmentation that often hinders biological interpretation. Ultimately, it provides a robust, standardized solution for extracting meaningful insights from gene expression and other high-throughput datasets.

2. Methods

A modular pipeline called auto-Enrich, that offers flexible pre-processing, enrichment analysis, and post-processing functionalities for enrichment analysis has been developed. The pipeline includes modules for Over-Representation Analysis (ORA) and Gene Set Enrichment Analysis (GSEA), leveraging popular bioinformatics tools. The choice of the three selected tools (g:Profiler, PANTHER and GSEA) for auto-Enrich version 1, was based on a systematic review of the literature from January 2024 to June 2025. Enrichment tools were ranked by citation counts on Google Scholar, and the top 10 tools evaluated according to methodological rigor, species and database coverage, software maintenance frequency, and database update regularity.

The pipeline uses a modular structure built mainly in Bash, with some written in Python. This design makes it easy to customize, extend, and maintain. Updates or changes can be made to individual modules without impacting the entire pipeline. Every module is composed of several interconnected components. All modules run inside a dedicated Docker image, ensuring consistent and reproducible environments. This approach reduces issues with software dependencies and configuration conflicts. Modules can be called individually to perform their specific tasks, and their execution is managed by a central run script. The use of Bash and Python with Docker also improves interoperability, allowing the pipeline to run reliably across different systems and environments.

Certain modules rely on specific variables stored in separate configuration files, each named according to the module number (e.g., "config1" for module 1). This keeps parameters organized and easy to track. In addition to these module-specific configuration files, a main pipeline configuration file called "config0" must be set with the list of modules to run, target species name of the analysis and post-processing parameters. The pipeline leverages file detection based on standardized naming conventions, automatically identifying and processing outputs from previous modules without requiring manual input. Internally, state flags help coordinate logic, ensuring that each module runs under the appropriate conditions.

Sanity checks are embedded throughout to verify the presence of the needed files and correct configurations, reducing the likelihood of runtime errors. Errors are actively reported.

During pipeline execution, directories with informative names are created, where modules save the output files, allowing users to efficiently trace modules results. It also

supports running multiple queries in the same setup for enrichment analysis, storing results in individual subdirectories within the module directories.

An auto-Enrich manual has been written that describes input and output formats, explains parameters, and provides usage examples for each module. This documentation gives users quick access to clear information, reducing confusion and errors.

In addition, pre-defined protocols were developed to improve the usability and reliability of auto-Enrich. Each protocol offers a tested workflow for a specific type of enrichment analysis and includes a detailed README with clear instructions. Sample result files show users what to expect and help validate their analyses. These protocols save researchers time by reducing setup and troubleshooting, allowing them to run complex bioinformatics analyses with confidence.

3. Results and Discussion

3.1. Comparison of popular PEA tools

In the January 2024 to June 2025 time period, the 10 most cited enrichment tools (as measured by main article citations on Google Scholar) were: GSEA (Broad Institute), clusterProfiler, Metascape, Enrichr, PANTHER, GOseq, WebGestalt, DAVID, g:Profiler, and KOBAS. Among these, Over-Representation Analysis and Functional Class Scoring remain the most commonly implemented methods, with GSEA continuing to be the leading FCS-based tool. Therefore, these tools were reviewed regarding methods, species coverage, integrated databases (DBs), software release frequency, and database update regularity (Table 1).

Table 1. Top 10 Most popular used Enrichment Analysis Tools (Jan 2024 and June 2025).

PEA Tool	Method	Integrated Databases	Organisms coverage	Last release and DBs updates	Number of main article citations*
GSEA (Broad Institute)	FCS	MSigDB, which includes GO, KEGG, Reactome …	Human and Mouse	May 2024 (2-3 times per year)	8840 [4]
cluster Profiler	ORA FCS	GO, KEGG, Reactome, DO, MeSH, NCG	Over 20 model species	April 2025 (2 times per year)	6100 [82]
Metascape	ORA FCS	GO, KEGG, Reactome, WikiPathways, MSigDB	40 species	2023 (irregular)	4250 [83]
Enrichr	ORA	Over 300+ libraries, including GO, KEGG,	Up to 6 model species	June 2023 (annually, irregular)	2580 [84]

		Reacomte, WikiPathways…			
PANTHER	ORA	PANTHER Pathways, PANTHER Collections, GO and Reactome	144 species	June 2024 (1-3 times per year)	1070 [85]
GOseq	ORA	GO and KEGG	20 model species	October 2023 (twice per year)	972 [86]
WebGestalt	ORA FCS TPA	GO, KEGG, Reactome, WikiPathways, +11	14 species	May 2024 (irregular)	849 [87]
DAVID	ORA	GO, KEGG, Reactome, PathBank, +5	Over 65 species	2016 (2 times per year)	793 [88]
g:Profiler	ORA	GO, KEGG, Reacome, WikiPathways, CORUM, HPA, +3	Over 800 species	April 2024 (1-4 times per year)	775 [89]
KOBAS	ORA FCS	GO, KEGG, Reactome, WikiPathways, +17	Over 5000 species	2021 (1-2 times per year, irregular)	700 [90]

*According to Google Scholar from 01/2024 until 06/2025.

Most tools rely on widely used annotation and pathway databases such as Gene Ontology (GO), KEGG, Reactome, and WikiPathways, although some also include additional resources like PANTHER Pathways, MeSH, Disease Ontology (DO), CORUM, HPA, and PathBank.

All tools reviewed support at least two model organisms, typically human and mouse, with broader coverage ranging up to over 5000 species (e.g., KOBAS). However, not all tools maintain regular software or database updates. DAVID and KOBAS have not had major software updates since 2016 and 2021, respectively. In contrast, clusterProfiler, Metascape, Enrichr, PANTHER, GOseq, WebGestalt, and g:Profiler have had recent releases. When it comes to database updates, tools like Metascape, Enrichr, and KOBAS show irregular patterns. Conversely, tools such as GSEA, which regularly updates its Molecular Signature Database (MSigDB), along with clusterProfiler, GOseq, PANTHER, DAVID, and g:Profiler, provide more consistent updates to their annotations and pathway datasets.

For auto-Enrich (version 1) pipeline, g:Profiler (g:GOSt), PANTHER ORA, and GSEA (Classic and Preranked) were selected. Nevertheless, the modular design implemented in auto-Enrich, takes into consideration the possibility of incorporating additional methods easily, although all methods will have their own specificities.

PANTHER ORA provides a highly maintained, consistently updated resource for overrepresentation testing, with broad species support and deep integration of high-quality pathway and ontology data. It also offers fast, well-documented APIs and user-friendly programmatic interfaces, facilitating straightforward pipeline integration.

g:Profiler (g:GOSt) provides a flexible, multi-database ORA platform with wide species coverage, rigorous statistical correction, and an active development team ensuring frequent updates. Its REST API and R/Python client libraries further simplify automated workflows.

GSEA, the most cited and widely adopted FCS method, remains the benchmark tool for gene set enrichment analysis of ranked lists. Its consistent updates, robust support for large, curated gene sets (via MSigDB), and proven statistical framework make it indispensable for hypothesis-driven analyses of transcriptomic data.

Collectively, these tools provide methodological diversity (ORA and FCS), consistent updates, strong community validation, and ease of programmatic integration, making them ideal for inclusion in auto-Enrich.

3.2. Building and running the Docker Image

auto-Enrich is a Bash script-based pipeline designed to integrate multiple widely used enrichment analysis tools into a single, modular framework, aimed at analysis automation and results standardization of enrichment analysis workflows. Packaged as a Docker image[2], this pipeline enables users to perform comprehensive enrichment analysis using three of the most popular tools available: g:Profiler, PANTHER, and GSEA (Gene Set Enrichment Analysis). The pipeline includes both pre-processing and post-processing steps to facilitate end-to-end analysis. It is designed to simplify and standardize the execution of a pre-processed gene expression matrix, mapping gene ID information, performing g:Profiler (g:GOSt) enrichment analysis, running the PANTHER statistical overrepresentation test, preparing inputs to perform GSEA in both Classic and Preranked modes, executing GSEA itself, and filtering the resulting enrichment analysis annotations.

The system is organized into seven core modules that use 34 component modules developed during this work, except for gene-id-to-uniprotkb. These modules can be used independently or in combination, providing flexibility and adaptability depending on the user's needs. A detailed explanation of each module, including setup instructions and configuration parameters, is provided in subsequent sections.

The auto-Enrich pipeline can be executed on any computing environment where Docker is installed, and the Docker image is accessible. At the top of the pegi3s Bioinformatics Docker Images Project website[3] there is a small section explaining how to install Docker in both Linux and Windows operating systems. Moreover, instructions are easily found in the Docker guides in the internet, as Docker is so widely used.

Before running the pipeline, users must prepare the necessary input data and configuration files. The pipeline supports two primary input data formats: a simple list of gene identifiers (one Gene ID per line), which can be used to perform over-representation analysis with the g:Profiler and PANTHER modules along with the required pre-processing steps; and a gene expression matrix, which can be used to extract gene lists using the developed pre-processing modules and then used to perform the above analyses, as well as Gene Set Enrichment Analysis using the FPKM gene expression values. At a minimum, two configuration files are required to run the pipeline: 1) a pipeline configuration file, which specifies the modules to be executed, the target species name and post-processing settings; 2) depending on the specific modules being

[2] https://github.com/pegi3s/dockerfiles/tree/master/auto-enrich
[3] https://hub.docker.com/r/pegi3s/auto-enrich

used, up to three additional configuration files may also be necessary, along with one more input file. Once all input and configuration files are properly set up and located within the same directory, the pipeline is ready to be executed. The following sections will provide a detailed explanation of the setup and execution process, including how to configure each module, run the pipeline under various scenarios, and access the resulting output files. Figure 1 shows the pipeline workflow, with all the tasks that can be performed and the modules that make them possible.

After filling in the configuration files and making sure that all input files are present, the user should adapt and run the following Docker command:

```
docker run -v /your/data/dir:/data pegi3s/auto-Enrich
```

Where /your/data/dir is the path to their data directory.

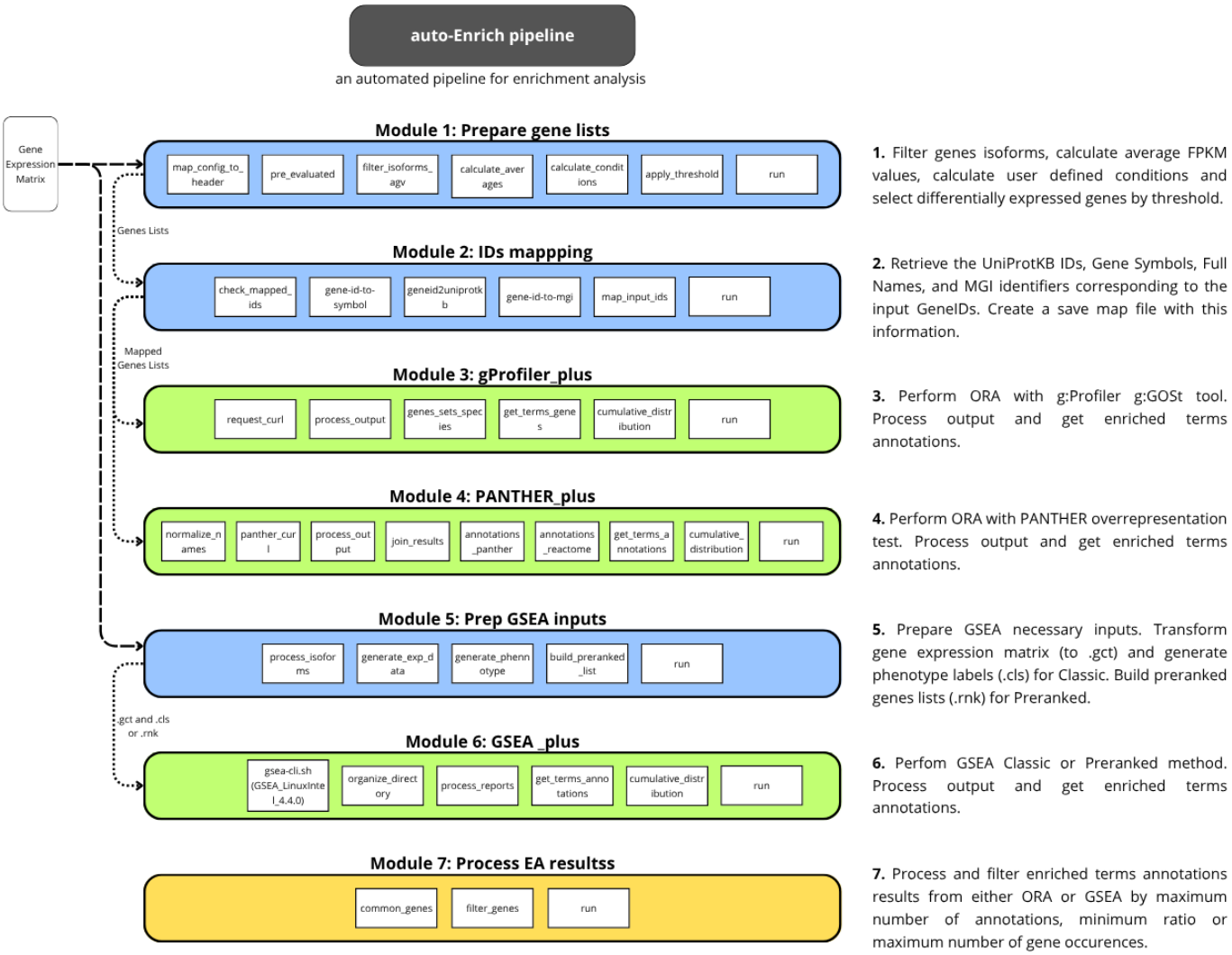

Figure 1. Diagram of the auto-Enrich pipeline Modules. Rectangles in blue correspond to pre-processing modules, in green to Enrichment Analysis modules, and in orange to the post-processing module.

3.3. Modules description, components and parameters

3.3.1. Module 1: Prepare gene lists

This module was designed to help users automatically generate gene lists of differentially expressed genes based on custom conditions and thresholds for over- and under-expression. The goal is to make it easier to identify meaningful gene expression changes without manual pre-processing.

To achieve this, the module is built from six component modules, all coordinated by a central run file that ensures smooth integration into the overall pipeline. Moreover, it allows the future ease integration of additional functions. Each component handles a specific task: mapping configuration settings to input file headers, filtering isoforms to keep only the highest expressed per gene, calculating average FPKM values across samples, testing for expression differences between groups under user-defined conditions, applying expression thresholds to identify differentially expressed genes for over- and under-expression, and creating gene lists of pre-selected genes. This modular approach gives users flexibility while ensuring consistency and reproducibility across analyses.

It takes two inputs: a TSV genes expression matrix file and a configuration file, named config1, with user defined parameters that must be placed in the assigned data directory (/data). The output is a set of gene lists ready for downstream analyses.

Example config1 file:

```
input=”gene_expression_data.tsv”
gene=1
control=2
avg1=2
avg2=”3,4,5”
cond1=”avg1/avg2”
expression_min=2
expression_max=1
select=8
```

Parameters description:

- input= Indicates the name of the input expression matrix TSV file (must be in /data). In the example: “gene_expression_data.tsv”

- gene= Specifies the column index in the input expression matrix that contains the Gene Identifiers (Gene IDs). In the example: column index “1” of the input file.
- control= (Optional) Specifies the column index in the file that contains the pre-calculated average expression values of the control group samples. This value is used as a filter criterion for gene isoforms. In the example: column index “2” of the input file.
- avgN= Specifies the column index in the input expression matrix that contains the pre-calculated average expression values for a group, or the column indexes of the samples expression values of a group. The module supports N different groups, each with N number of samples. In the example: for Group 1, the pre-calculated average expression values are in column index “2” of the input file. For Group 2 the expression values for the samples are in columns indexes “3, 4, 5” of the input file.
- condN= Defines the mathematical formulas used to compare average expression values between two distinct groups. Each set condition must use the previously defined avgN parameters to specify the groups involved. The module supports N different set conditions. In the example: “avg1/avg2” represents the division of Group 1's average expression values by the Group 2’s average expression values.
- expression_min= Sets the minimum threshold that a gene’s calculated condition value must exceed to be considered overexpressed. In the example: “2” means only genes with condition values greater than 2 are marked as overexpressed.
- expression_max= Sets the maximum threshold that a gene’s calculated condition value must fall below to be considered underexpressed. In the example: “1“ means genes with condition values less than 1 are marked as underexpressed.
- select= (Priority) Indicates the index column of the input file that contains the pre-evaluated binary values (0 or 1) that indicate gene selection. In the example: column index “8” marks genes with a value of 1 (selected) or 0 (not selected).

Note 1: Column indexes start at 1 (so the first column index is 1).

Note 2: If both expression_min= and expression_max= are defined, the output will include genes with condition value(s) falling within the specified range. Expression thresholds are applied to every condition defined during the same run.

Note 3: If the select= parameter is defined and not empty, it takes priority over all other defined logic. Only genes marked with 1 in the specified column index will be included in the output gene list.

3.3.2. Module 2: IDs mapping info

This module standardizes a list of Gene Identifiers (Gene IDs) into a unified format to ensure consistent annotation across different biological databases and enrichment tools. The output includes four core fields: the original Gene ID, the corresponding UniProtKB ID, the official Gene Symbol, and the official Full Name. For the species *Mus musculus*, a fifth field, the Mouse Genome Informatics (MGI) ID, is also included.

The primary objective is to normalize gene information, enabling accurate cross-referencing of term-gene annotations from various data sources, such as UniProtKB IDs, Gene Symbols, and MGI. Additionally, it provides users with extended gene information, including full gene names, to facilitate easier identification and interpretation. This harmonization ensures compatibility with downstream tools and datasets that may rely on different identifier standards. The module outputs a tab-separated file (TSV) containing one gene per row with its mapped identifiers and generates a species-specific support file to cache retrieved mappings. This support file prevents redundant queries, significantly improving processing efficiency when working with multiple gene lists.

The module is built from five interconnected components, managed through a central run file that controls their execution within the pipeline. These components handle fetching UniProtKB IDs, retrieving Gene Symbols and Full Names, mapping MGI IDs (if applicable), updating the support file with newly encountered unmapped Gene IDs, and creating the final mapping input file. The resulting mapping file is saved as <species_prefix>_ids_map (e.g., hsapiens_ids_map), ensuring standardized inputs for subsequent enrichment analyses and making gene identification more straightforward and reliable for users.

3.3.3. Module 3: gProfiler_plus

This module's structure and purpose is designed to provide a robust, automated, and user-friendly workflow for performing comprehensive functional enrichment analysis using g:Profiler g:GOSt tool. By interfacing directly with the g:Profiler API[4], it ensures that analyses are based on up-to-date, curated biological databases spanning a broad spectrum of annotation sources, which enhances the biological relevance and depth of the results. Applying False Discovery Rate (FDR) correction controls for multiple testing, increasing confidence in the identified enriched terms.

[4] https://biit.cs.ut.ee/gprofiler/page/apis

The module takes as input a species name and a mapped gene list (previously mapped by Module 2), to guarantee consistency with earlier data processing steps and supports accurate term-gene associations.

It converts the raw JSON results into two CSV files: a short_results.csv file containing key enrichment metrics for easy interpretation and quick reporting, and a long_results.csv file including the full set of enrichment data fields for transparency and detailed analysis. Additionally, it generates a terms_annotations_results.csv file that consolidates the mapping of enriched terms back to input genes, facilitating biological interpretation. The module downloads and locally stores species-specific GMT annotation files from g:Profiler, which it uses to efficiently map enriched terms to gene sets. It organizes the results into a clear directory structure, creating individual subdirectories for each enriched term that contain files with matched input genes in the list and their mapped information (TSV), along list file with the full gene set annotations, making downstream exploration, visualization, and reporting more straightforward and efficient.

The module's internal components are coordinated by a central run script that manages the execution flow across enrichment requests, the output formatting, annotation retrieval, and results' organization. This design improves maintainability, scalability, and clarity of the pipeline, reducing manual effort and minimizing errors, while providing users with rich, organized output that supports rigorous and reproducible functional genomics research.

3.3.4. Module 4: PANTHER_plus

This module performs functional enrichment analysis using the PANTHER database by interfacing with the PANTHER API[5] to run statistical overrepresentation tests across all available gene sets. These include the Gene Ontology subdomains (Biological Process, Molecular Function, and Cellular Component), a curated "GO-SLIM" subset, Reactome and PANTHER Pathways, and the PANTHER Protein Class ontology. The enrichment is conducted with multiple testing correction using the False Discovery Rate (FDR) method.

[5] https://www.pantherdb.org/services/openAPISpec.jsp

The module takes as input a species name and a mapped gene list (previously mapped by Module 2), to guarantee consistency with earlier data processing steps and supports accurate term-gene associations.

The module is composed of eight internal components, all coordinated by a central run script that manages the complete workflow. It begins by sending a structured curl request to the PANTHER API's overrepresentation endpoint and retrieves the enrichment results in raw JSON format. This output is automatically parsed, filtered, and reformatted into two structured CSV files: short_results.csv, containing key enrichment fields for summary analysis, and long_results.csv, providing a more comprehensive view of the full results. This dual-format output allows users quick access to essential results for easy interpretation while retaining the full dataset for in-depth analysis, balancing usability and transparency. To further enhance interpretability, the module downloads species-specific gene set annotation files from the PANTHER FTP server[6] for the SLIM GO terms, PANTHER Pathways, and PANTHER Protein Class, which are cleaned and standardized to retain only essential fields such as UniProtKB IDs and Term IDs. These files are saved using the format <species_prefix>_PTHR19.0_annotations (e.g., hsapiens_PTHR19.0_annotations), improving organization and making them reusable across analyses. This local caching reduces repeated web queries, boosting efficiency and reproducibility. Since Reactome pathways are shared between g:Profiler and PANTHER, the module reuses the Reactome annotation data from the g:Profiler dataset, filters it for Reactome-only terms, and saves it as <species_prefix>_REAC_annotations (e.g., hsapiens_REAC_annotations), avoiding redundancy and maintaining consistency across tools. Additionally, Gene Ontology term annotations from its gene sets (Molecular Function, Cellular Component, and Biological Process) are obtained in real time from the Gene Ontology platform (AmiGO[7]), ensuring the incorporation of the most up-to-date functional data for biological relevance. These annotation files are then used to map enriched terms back to the input genes, allowing identification of which genes are associated with which terms. The results are compiled into a summary file (terms_annotations_results.csv) and organized into a directory structure where each enriched term has a dedicated subdirectory containing a TSV file listing input genes associated with the term and a list file listing all genes annotated to that term, one per line. Altogether, this structured approach enhances interpretability, reduces manual

[6] https://data.pantherdb.org/ftp/sequence_classifications/current_release/PANTHER_Sequence_Classification_files/
[7] https://amigo.geneontology.org/amigo

processing, supports scalability across species and datasets, and ensures data alignment with standardized identifiers used throughout the pipeline.

3.3.5. Module 5: Preparation of GSEA inputs

This module was designed to simplify and automate the preparation of input files required for running GSEA using a gene expression matrix. By taking a tab-separated (TSV) gene expression matrix and a user-defined configuration file (named config5) specifying group assignments, gene ID column names, and comparison transformations, it streamlines the conversion of raw data into the correct formats for both GSEA Classic and GSEA Preranked modes. Automating this process reduces manual effort, minimizes errors, and ensures standardized input formatting, which improves reproducibility and consistency across analyses. The use of a flexible configuration file allows users to adapt the module to various experimental designs without modifying the code, making it easily reusable across projects. The module is composed of four interconnected internal components, orchestrated by a central run script that manages the execution flow across expression parsing, isoform filtering, formula evaluation, and file formatting. By supporting both GSEA modes, the module accommodates different analytical needs, whether users are working with raw expression data or pre-ranked gene lists. It also handles group assignments and comparison logic internally, simplifying the management of multi-group studies. Overall, this approach enhances efficiency, scales well for large or repeated analyses, and lowers the barrier for users unfamiliar with GSEA specific input requirements, ultimately enabling more reliable and accessible functional enrichment workflows.

For GSEA Classic, it generates a properly formatted expression dataset file (GCT) and a corresponding phenotype labels file (CLS). Additionally, it provides an optional isoform filtering step, which uses average expression values (either across all samples or within a selected group) to retain only the highest-expressed isoform per gene, preventing arbitrary selection when running GSEA.

For GSEA Preranked, the module supports generating one or more ranked gene lists (RNK files) by applying user-defined mathematical formulas to expression columns specified in the configuration file. These formulas typically represent fold changes between two biological groups, such as “Exp_avg / Control_avg”. The script evaluates each formula for all genes in the dataset, calculates the log2 fold change, and outputs a preranked file (RNK) sorted by decreasing score, suitable for direct input into the GSEA Preranked workflow. Multiple formulas can be defined, enabling the user to generate

ranked lists for various conditions or comparisons in a single run. Column names referenced in the formulas are automatically mapped from the input header, and the log2 transformation ensures interpretability and consistency with GSEA expectations.

To prepare inputs for either GSEA Classic or GSEA Preranked methods, a configuration file, named config5, must be created, configured accordingly to the pretended method and be placed in the assigned data directory (/data), alongside the expression matrix input file.

Example configurations:

To prepare GSEA Classic inputs:

method=”classic”

input=”gene_expression_data.tsv”

gene=”GeneID”

control=”3,4,5”

description=2

number_samples=8

samples=”3,4,5,6,7,8,9,10”

number_groups=2

group_order=”Control 4 Exp 4”

To prepare GSEA Preranked inputs:

method=”preranked”

input=”gene_expression_data.tsv”

gene=”GeneID”

formula1=”Control_avg / Exp_avg”

formula2=”Exp_avg / Control_avg”

Parameters common to both modes:

- method= Specifies the method to prepare the inputs to run GSEA. Options are “classic” or “preranked”. In the examples: “classic” is the selected method to prepare inputs for GSEA Classic. “preranked is the selected method to prepare inputs for GSEA Preranked.
- input= Indicates the name of the input expression matrix TSV file (must be in /data). In the example: “gene_expression_data.tsv”

- gene= Specifies the column name in the input file that contains the Gene Identifiers (Gene IDs or Gene Symbols, see note 1 for more). In the example: column “GeneID” contains the Gene IDs in the provided file.

Additional parameters for GSEA Classic:

- control= (Optional) Specifies the column index in the input file that contains the pre-calculated average expression values for the control group samples. This value is used as a filter criterion for gene isoforms. In the example: column indexes “2, 3, 4” contains the expression value of the control group samples in the provided file.
- description= (Optional) Specifies the column Index in the input file that contains the description of the gene (not mandatory, if empty filled with N/A). In the example: column index “2” contains the Gene description in the provided file.
- number_samples= Indicates the total number of samples (across all groups) to be processed. In the example:”8”
- samples= Specifies the column Indexes of samples, separated by commas inside double quotes. Must go in line with the total number of indicated samples. Column headers must be unique and have no special characters. In the example: column indexes “3, 4, 5, 6, 7, 8, 9, 10” have the expression values across the samples in the provided file
- number_groups= Indicates the total number of distinct groups. In the example: “2” for two distinct groups (control and experimental)
- group_order= Indicates the group name and sample count following the same order as the provided column indexes in “samples”. In the example: 4 control samples in the previously assigned columns (“3, 4, 5, 6”) and 4 experimental samples in the previously assigned columns (“7, 8, 9 , 10”)

Additional parameters for GSEA Preranked:

- formulaN= Specifies a custom mathematical formula to calculate the relative expression between two conditions or sample groups, using the column names that represent average expression values. Multiple formulas can be defined by incrementing N (e.g., formula1, formula2, etc…). Each formula should reference the appropriate column headers from the input file.

Note 1: The gene identifier used in the gene= parameter, either a Gene ID or Gene Symbol, must match the identifier format used in the gene set file for GSEA analysis. Gene sets from the Molecular Signatures Database (MSigDB[8]) may use either NCBI (Entrez) Gene IDs or Gene Symbols. Make sure the content of the column specified in the gene= parameter matches the format used in the gene set file.

[8] https://www.gsea-msigdb.org/gsea/msigdb/index.jsp

Note 2: Column index is 1-based (1st column = 1). The group order in input data file must match samples indicated in samples. The number of distinct groups (number_groups) must match the distinct groups in group_order.

3.3.6. Module 6: GSEA_plus

This module streamlines Gene Set Enrichment Analysis by wrapping the gsea-sli.sh command-line tool of GSEA version 4.4.0 (Broad Institute[9]) and automates result filtering, term-to-gene mapping, and output organization to facilitate downstream analysis. The module is prepared to run either GSEA Classic or GSEA Preranked and handle the raw GSEA results from both.

For GSEA Classic, the required inputs include an expression dataset file in GCT or RES format containing gene expression values, a phenotype labels file in CLS format defining sample groupings and phenotypes, and a gene set matrix file in GMX format describing the gene sets or pathways of interest. These files and other parameters must be set up with the corresponding values in a tab-separated key-value parameter file format, named gsea_parameters, and placed in the assigned data directory (/data).

Example GSEA Classic (tab-separated key-values):

res	expression_dataset.gct
cls	phenotype_labels.cls
gmx	m2.all.v2024.1.Mm.symbols.gmt
collapse	No_Collapse
chip	*
permute	gene_set
nperm	1000
out	results

Specific parameters for GSEA Classic:

- res - Indicates the name of the input expression dataset file (GCT or RES). In the example: “expression_dataset.gct”
- cls - Indicates the name of the input phenotype labels file (CLS), of the set expression dataset file, to define categorical phenotypes (e.g., tumor vs normal). In the example: “phenotype_labels.cls”

[9] https://www.gsea-msigdb.org/gsea/downloads.jsp

For GSEA Preranked, the inputs include a preranked gene list in RNK format, typically containing gene identifiers and a ranking metric such as log2 fold change, and a gene set matrix file in GMX format describing the gene sets or pathways of interest. These files and other parameters must be set up with the corresponding values in a tab-separated key-value parameter file format, named gsea_parameters, and placed in the assigned data directory (/data), alongside the required input files.

Example GSEA Preranked (tab-separated key-values):

rnk	preranked_list.rnk
gmx	m2.all.v2024.1.Mm.symbols.gmt
collapse	No_Collapse
chip	*
permute	gene_set
nperm	1000
out	results

Specific parameters for GSEA Preranked:

- rnk - Indicates the name of the ranked gene list file (RNK format). In the example: preranked_list.rnk

Independently of the GSEA approach used, both share several key parameters recommended to be set up by the user in the parameter configuration file, gsea_parameters, for a smooth GSEA run. These parameters include shared required input files, such as the gene set file (GMT) and situationally the chip annotation file (CHIP), along with parameters related to the GSEA methodology itself.

Recommended common parameters:

- gmx - Indicates the name of the gene set file (GMT format), such as those provided by Molecular Signature Database. In the example: “m2.all.v2024.1.Mm.symbols.gmt” from the MSigDB Mouse collections
- collapse - Defines how gene identifiers are handled. Available values are: **Collapse** (default) uses the chip file to convert to gene symbols; **Remap_Only** remaps identifiers without collapsing data; **No_Collapse** uses gene symbols as-is (no chip file needed). In the example: No_Collapse since no chip annotation file is set.
- chip - (Optional) Indicates the name of the chip annotation file that maps array probe IDs to Gene Symbols. **Required if collapse is set to Collapse or Remap_Only.** Not set in the example.

- permute - Defines the permutation type: **phenotype** (default) shuffles sample labels, recommended for datasets with ≥7 samples per group; **gene_set** randomizes gene sets, used for small datasets (<7 samples per group/phenotype). In the example: “gene_set”
- nperm - Defines the number of permutations for statistical significance (1000 recommended, start with 10 to test setup). Default: 1000. In the example: 1000.
- out - Indicates the name of the output directory to where the analysis results will be saved. Default: results. In the example: results

Note 1: The gene set files (gmx) contain one or more gene sets (terms or pathways). For each gene set, it gives the gene set name and list of features (genes or probes) in that gene set. Features can be downloaded in either NCBI (Entrez) Gene IDs or Gene Symbols identifiers. These files can be individually downloaded from MSigDB Human Collections or Mouse Collections, or complete collection from GSEA Downloads.

Note 2: Since the default value of collapse is ‘Collapse’ it’s **advised** to set the parameter value yourself. If not set, GSEA falls to the default value which requires the chip annotation file, leading to an error if not provided.

Note 3: Chip annotations files list each identifier on a platform and its matching HGNC Gene Symbol. Optional for Gene Set Enrichment Analysis. CHIP format can be downloaded from the GSEA Downloads (Human or Mouse Chip Annotations files).

Note 4: Besides the above mentioned recommended parameters, the module supports most command-line options available on GSEA, including: metric, scoring_scheme, rpt_label, sort, set_max, set_min, norm, rnd_seed, zip_report, create_svgs, etc. A full list of parameters and their descriptions can be found in the GSEA CLI User Guide[10].

In either approach, the raw results from GSEA, such as comprehensive logs, plots, and other detailed data, are saved in a dedicated results directory. These files are not always necessary for interpretation and storing them separately ensures reproducibility and traceability while allowing users to focus on simplified, cleaner outputs for reporting and downstream analysis.

Key results are processed and formatted into two main CSV files: short_results.csv, which provides a concise summary of core enrichment metrics for quick interpretation,

[10] https://docs.gsea-msigdb.org/#GSEA/GSEA_User_Guide/

visualization, or integration into reports; and long_results.csv, which includes the full set of GSEA output fields, supporting transparency and enabling advanced or customized analyses.

Additionally, the gene set file (GMT) used during the analysis, defining the gene sets, is parsed to extract term-to-gene associations. This enables users to easily trace which input genes contributed to the enrichment of each term, facilitating hypothesis generation and biological validation. These final annotations are saved in terms_annotations_results.csv. To further organize the output, the module creates a structured directory where each significantly enriched term has its own subdirectory containing input genes associated with it, delivered one per line in either Gene IDs or Gene Symbols depending on initially used nomenclature, and a list file listing all genes, one per line, in the original term.

3.3.7. Module 7: Process EA results

This module is designed to enhance the biological relevance and interpretability of enrichment analysis results by filtering out genes and terms that are overly common or biologically nonspecific. By applying user-defined thresholds based on gene frequency across enriched terms, total gene annotations per term, and a minimum ratio score, it ensures that only the most specific and meaningful enriched terms are retained for downstream analysis. This targeted filtering reduces noise and helps focus on gene-term associations that are more likely to provide insightful biological interpretations.

The filtering process automatically takes as input a directory containing enrichment results generated by Modules 3, 4, and 6, along with filtering parameters that must be defined in the pipeline configuration file, named config0. The output is a refined terms annotations results CSV file, named “terms_annotatios_filtered.csv” located in the same input directory, that excludes nonspecific genes and broadly annotated terms, which often correspond to generic biological processes or housekeeping functions. By removing these potentially confounding elements, the module improves the clarity and precision of the enrichment results, supporting more accurate downstream analyses. Users can customize filtering using three main parameters.

Example:

max_occur=5

min_ratio=10

max_annot=500

Parameters description (set up in config0):

- max_occur Defines the maximum value of occurrences a gene, from the input gene list, can have in different enriched terms. Genes with a total number of occurrences higher than the set threshold are discarded from the "genes_in_list" field in the term's annotations results file. In the example: "5" means that any gene occurring in more than five enriched terms will be filtered out.
- min_ratio Defines the minimum ratio value [0,100] to filter out the terms results file. The ratio is equivalent to the total number of genes in the list associated with a given term divided by the number of genes annotations of that term, multiplied by 100. In the example: "10" means all enriched terms with a ratio lower than 10% will be excluded.
- max_annot Defines the maximum value for the total number of genes annotated in a term. Terms with a higher number of annotated genes than the set value will be excluded from the terms annotations results. In the example: a value of 500 means all enriched terms with over 500 annotations will be excluded.

Note 1: When max_occur is specified, it takes priority and is applied first, followed by any additional filtering based on min_ratio and max_annot.

The module allows flexible application of these filters. One, two, or all three parameters can be set independently or in combination. To help users select appropriate filter values, all enrichment analysis module produces, alongside other results, a cumulative distribution TSV file (named "cumulative_distribution.tsv"). This file supports downstream filtering strategies by allowing users to empirically define thresholds for gene inclusion based on frequency. For example, genes appearing in a high number of terms (e.g., the top 10% most frequent) may be flagged as overly common, reflecting broad or non-specific biological functions (e.g., housekeeping genes) that could dilute the interpretability of the enrichment results. By selecting a threshold informed by the cumulative distribution, users can filter out such genes and focus on more selectively enriched, potentially more biologically relevant gene-term associations.

Ultimately, this filtering module plays a vital role in improving the specificity of enrichment results by removing generic genes and broadly annotated terms that often appear across multiple enrichment tests. This increases the interpretability and biological significance of the final results, allowing researchers to focus on the most informative gene-term associations. Users are encouraged to review unfiltered results first, and then tailor filtering parameters to their specific biological questions and dataset characteristics to maximize the utility of this module.

3.4. The pipeline configuration file and behaviour

The pipeline configuration file contains the parameters used to define the modules to be executed, the target species name and the post-processing enrichment analysis filtering parameters. Figure 2 shows an example of the pipeline configuration file, config0 (where the values for Module 7 are also defined; see above), set up to make use of all the available modules by taking as input the expression matrix file through pre-processing, enrichment analysis and post-processing steps. This pipeline configuration file must be named config0 and be placed in the assigned data directory (/data).

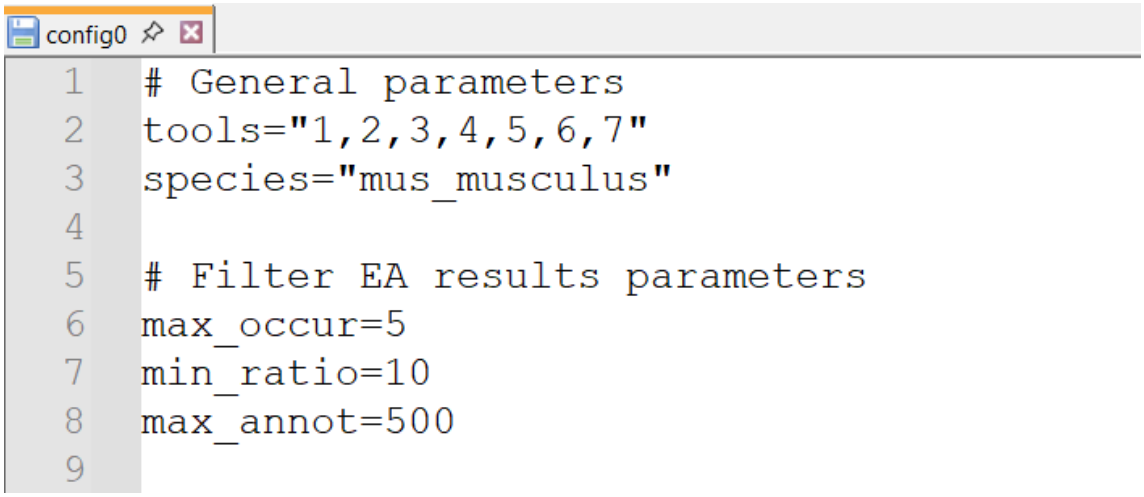

Figure 2. Example of a config0 file. The lines starting with "#" are comments. Line 2 defines the selected modules to run on the pipeline, line 3 is the target species name and lines 6, 7 and 8 are the filtering parameters to filter enrichment analysis terms annotations results by Module 7.

In addition to the main pipeline configuration file (config0), modules that require their own configuration, specifically Module 1, Module 5, and Module 6, must have their respective configuration files placed in the assigned data directory (/data). To run Module 1, the file config1 must be present; to run Module 5, the file config5 is required; and to run Module 6 for GSEA (either Classic or Preranked), the parameters file (gsea_parameters) must be available. Module 1 and Module 6 also depend on the indicated input file, defined in "input" variable, (e.g. input="expression_matrix.tsv") and any additional input files, such as the GMX file to run GSEA, must reside in the same assigned data directory (/data). Figure 3 illustrates the structure of a correctly populated data directory for this example pipeline.

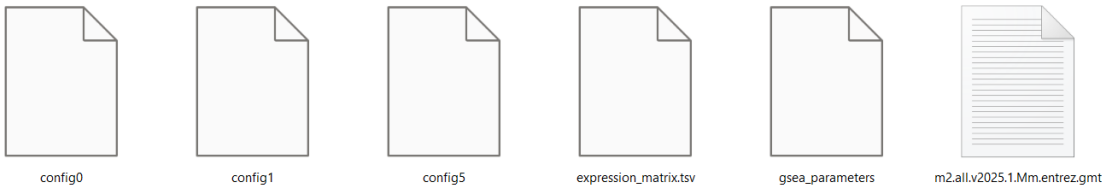

Figure 3. Look of the data directory, assigned as /data in the execution of the docker pipeline command, showing the data and configuration files needed to run the different modules: the pipeline configuration file (named “config0”), the Module 1 configuration file (named “config1”), the Module 5 configuration file (named “config5”), the input expression matrix TSV file (named “expression_matrix.tsv”), the GSEA parameters file (named “gsea_parameters”) and the gene set file GMX required to run GSEA (“m2.all.v2024.Mm.symbols.gmt” from MSigDB Mouse Collections).

Modules are run sequentially, as defined by the “tools” variable string in config0 file, allowing flexible customization of the analysis flow. Conditional logic enables certain modules, such as ids_mapping_info, gProfiler_plus, and PANTHER_plus, to adapt their behavior based on whether Module 1: prepare_gene_lists has been executed. This interdependence ensures consistent input formatting and compatibility between steps. The pipeline leverages file detection based on standardized naming conventions, automatically identifying and processing outputs from previous modules without requiring manual input. Internally, state flags that help coordinate logic are used, ensuring that each module runs under the appropriate conditions. Sanity checks are embedded throughout to verify the presence of the necessary files and correct configurations, reducing the likelihood of runtime errors, and actively reporting the cause of the error.

Upon execution of Module 1 to prepare gene lists from a gene expression matrix file followed with Module 2 to map gene information, every gene list generated by Module 1 is automatically processed by Module 2 for gene identifier mapping. This seamless integration ensures that all prepared lists are consistently annotated and ready for downstream analysis, eliminating the need for manual intervention between the two steps. The pipeline detects the output of Module 1 and triggers the mapping process for each list, maintaining alignment between gene identifiers and the target species defined in the configuration.

It’s important to note that Modules 3 and 4, which perform ORA using the g:Profiler g:GOSt tool and the PANTHER overrepresentation test, respectively, require as input the mapped gene information file generated by Module 2. Therefore, it is strongly recommended to run Module 2 before executing either of these ORA modules to ensure they have the necessary input data. Additionally, the specification of the target species

is critical, not only for correctly mapping gene identifiers but also for determining the appropriate term-gene annotation sets used in these analyses. As such, when running Module 3 or Module 4, the target species name must be explicitly defined in the config0 file; failure to do so will result in an error or incomplete analysis due to missing context for gene mappings and functional annotations.

Similarly, if opting to use Module 5 to prepare inputs for GSEA and Module 6 to execute the GSEA itself, it is essential to ensure that the required configuration and input files are correctly set up. When Module 5 is run prior to Module 6, it generates the necessary input files and automatically updates the set GSEA parameters file, gsea_parameters, by appending the relevant tab-separated key-value pairs of each prepared input. This automation ensures a smooth transition to Module 6, whether running GSEA Classic or GSEA Preranked, with the pipeline managing parameter updates transparently. In cases where multiple pre-ranked gene lists are prepared for GSEA Preranked, the pipeline iteratively runs GSEA for each list, using the gene set file (GMX) specified in the parameters file. This enables streamlined execution of batch GSEA analyses without requiring manual intervention for each list.

3.5. Pipeline and Modules Outputs Directories

3.5.1. Pipeline Outputs

Regardless of the pipeline built and the user configurations, the final results are always saved in the assigned data directory (/data). Figure 4 illustrates the organization of this main working directory after running the entire pipeline with all modules selected (tools="1,2,3,4,5,6,7"). Within this directory, users will find the original input and configuration files, ensuring all parameters and raw data are preserved for reproducibility. The directory also contains distinct output subdirectories corresponding to each executed module: “/prepared_gene_lists” holds the gene lists prepared by Module 1, “/mapped_gene_lists” stores the mapped gene lists information generated by Module 2, “/gprofiler” and “/panther” contain the over-representation analysis results from Modules 3 and 4 respectively, while “/gsea” includes outputs from Modules 5 and 6 related to GSEA input preparation and analysis. Additionally, the “/annotations” directory consolidates mapping and term annotation files produced by Modules 2, 3, and 4, facilitating easy cross-referencing of enriched terms and genes. If isoform filtering by Modules 2 and 5 is enabled, filtered isoform expression matrices generated are also saved within this directory structure.

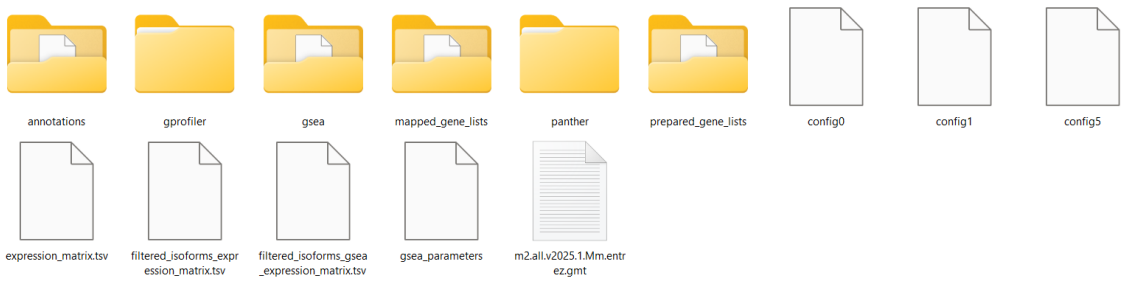

Figure 4. Structure of the data directory assigned as /data after executing the full pipeline using all modules. This directory includes the pipeline and modules configuration files (config0, config1, config5, and gsea_parameters), the input gene expression matrix TSV file (named "expression_data.tsv"), the isoform free filtered versions of this last file (generated by Module 2: "filtered_isoform_expression_data.tsv"; generated by Module 5: "filtered_isoform_gsea_expression_data.tsv") and the gene set file used in GSEA run (named "m2.all.v2025.1.Mm.entrez.gmt"). Modules results are organized into clearly defined subdirectories: /prepared_gene_lists contains the outputs of Module 1; /mapped_gene_lists holds the outputs of Module 2; /gprofiler and /panther store the over-representation analysis results from Modules 3 and 4 for each processed gene list, respectively; /gsea includes both the prepared inputs and the enrichment results generated by Modules 5 and 6; /annotations includes the mapping and terms annotations.

3.5.2. Mapping and Terms Annotations directory (/annotations)

This directory is generated in the first run of the pipeline. It serves the purpose of keeping the main data directory organized and keeping all species-specific mapping files and terms annotations from the different data sources in the same place. If these essential files are not available in this directory, annotations and a mapping file will be created by the appropriate Modules and saved in this directory. By always having these files in the annotations directory, performance and consistency will be improved by avoiding redundant processing, removing the need to query the terms annotations and gene mapping information every time the modules are used and ensuring accurate gene and term annotations across the different data sources used by the pipeline modules.

Inside the /annotations directory, created after the first run of Modules 2, 3, and 4, the following species-specific files will be saved (with the prefix of the target species short format name).

These files are illustrated in Figure 5 and are the following:

- <species_prefix>_ids_map (e.g., hsapiens_ids_map) that saves the mapped Gene IDs information. This file is generated by Module 2 and updated with any new, unmapped Gene IDs in every new run.

- <species_prefix>_gprofiler_annotations.gmt (e.g., hsapiens_gprofiler_annotations.gmt) is utilized and downloaded from g:Profiler g:GOSt data sources by Module 3. It helps ensure consistent term annotations of the g:Profiler enriched results.
- <species_prefix>_REAC_annotations (e.g., hsapiens_REAC_annotations) provides REACTOME pathway annotations. This file is extracted from a larger g:Profiler dataset and filtered to include only REACTOME terms. It helps ensure consistent term annotations of the PANTHER Reactome enriched results.
- <species_prefix>_PTHR19.0_annotations (e.g., hsapiens_PTHR19.0_annotations) uted by PANTHER_plus, this file contains PANTHER-specific annotations parsed from the official PANTHER FTP resource, retaining only essential fields such as UniProtKB IDs and Term IDs.

Recommendation: Keep these files stored in the /annotations directory inside the data directory and **do not rename** these files, as Modules rely on their specific naming conventions to function properly. This way they can be reused in future analyses to ensure consistency and improve performance. During the run of the pipeline these files are pulled out of the /annotations directory into the working directory (/data) by the necessary modules and put back in when the module is over.

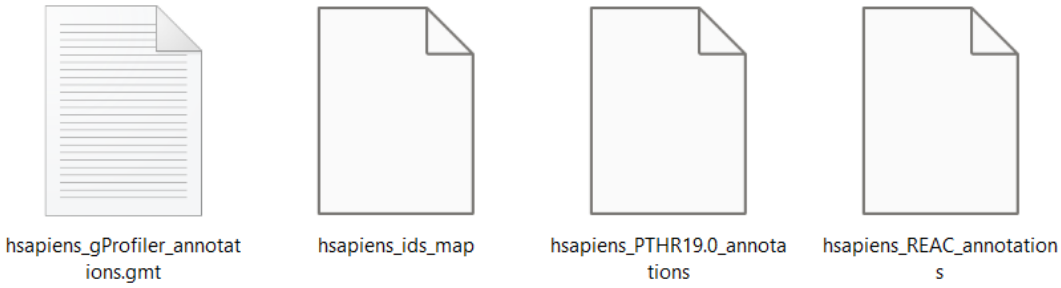

Figure 5. Contents of the /annotations directory present in the data directory, as generated after the initial execution of Modules 2 (ids_mapping_info), 3 (gProfiler_plus), and 4 (PANTHER_plus).

3.5.3. Module 1 Output Directory (/prepared_gene_lists)

This directory, located in the assigned data directory, stores the output results from Module 1, which processes the input gene expression matrix to produce curated gene lists based on user-defined conditions. These outputs are structured for clarity and reproducibility and include several key files necessary for downstream enrichment analyses. The outputs include a reference file (named column_header_mapping.txt) that maps columns names from the input expression matrix file to the average, condition and threshold parameters defined by the user in the configuration file, config1. It details each

parameter name, its assigned values or index positions, and the corresponding column headers from the original expression matrix, providing transparency in how the sample groups were interpreted during pre-processing.

The central output file is a TSV comprehensive table (named “evaluation_all_conditions.tsv”) that compiles gene-level evaluations across all defined conditions. Each row corresponds to a gene and includes: (i) the gene identifier column, as specified in the configuration; (ii) calculated average expression values (avgN) for the sample groups defined under the avgN variable in the configuration; (iii) condition comparison values (condN) resulting from pairwise comparisons between groups defined in the condN variable in the configuration; and (iv) binary evaluation results (evaluation_condN) indicating whether each gene’s expression meets (1) or doesn’t meet (0) the specified thresholds (expression_min/expression_max), letting users quickly spot genes of interest under each condition.

Additionally, for each defined condition, a separate TSV file is generated listing only the genes that meet the threshold criteria. This file includes the same three core fields, Gene ID, average expression values, and condition comparisons, but are filtered to retain only those genes that satisfy the set thresholds. The files created for each output follow a consistent naming convention using the prefixes overexpressed, underexpressed, or in_range, combined with the suffix condN for each condition. The file that contains the filtered genes and data, uses is base pattern (e.g., overexpressed_cond1). The genes list file, adds _genes before the condition suffix (e.g., overexpressed_genes_cond1).

All the output files generated by this module are illustrated in Figure 6, which represents the structure of the results directory created during execution of Module 1.

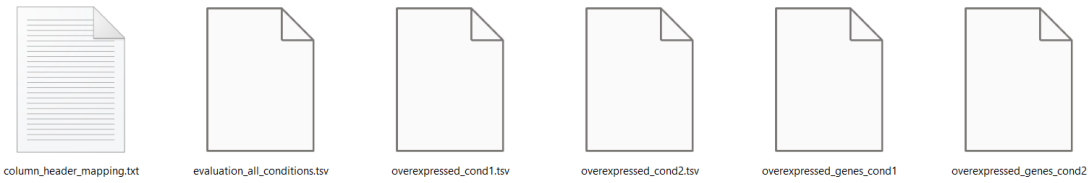

Figure 6. Contents of the /prepared_gene_lists directory in the assigned data directory, generated after the initial execution of Module 1, with the defined configuration files parameter expression_min=2 and two distinct conditions between groups. This directory contains: the reference file (named “column_header_mapping.txt”) file, the central results TSV file (named “evaluation_all_conditions.tsv”), the outputs specific to each defined condition for the set threshold (the “overexpressed_cond1.tsv” and “overexpressed_genes_cond1” for the condition 1; the “overexpressed_cond2.tsv” and “overexpressed_genes_cond2” for the condition 2).

This modular and condition-aware output structure enables a clear, scalable, and interpretable workflow that supports high-throughput gene filtering and prepares consistent inputs for subsequent mapping and enrichment analysis modules. This targeted reporting helps users focus on relevant subsets of genes for specific biological hypotheses or experimental conditions.

3.5.4. Module 2 Output Directory (/mapped_gene_lists)

This directory is created in the assigned data directory upon execution of Module 2 and contains the mapped gene lists. For each gene list from Module 1 (or found in /prepared_gene_lists), a corresponding TSV file is generated with four columns (five for *Mus musculus*, including MGI ID): Gene ID, UniProtKB ID, Gene Symbol, and Full Name. Files are named after the original lists with the _map suffix (see Figure 7).

Figure 7. Contents of the /mapped_gene_list directory in the assigned data directory, as generated after the initial execution of Module 2, ids mapping info, with the prepared gene list by Module 1. This directory contains: the mapped gene list files, namely, the “overexpressed_genes_cond1_map” and “overexpressed_genes_cond2_map” for the mapped gene information of the overexpressed genes in condition 1 and condition 2, respectively.

3.5.5. Modules 3 and 4 Outputs Directories (/gprofiler and /panther)

The enrichment analysis results produced by Modules 3 and 4, that implement ORA with g:Profiler and PANTHER, are processed into a unified and normalized output format to ensure interoperability between tools. Despite differences in their underlying statistical models and annotation databases, outputs are restructured to follow the same logical layout for each analysed gene list, enabling seamless downstream integration and comparative analysis.

Each run is stored under its respective root directory, /gprofiler or /panther, within a subdirectory named after the mapped gene list (/data/<tool>/<gene_list>/results/). Inside this results directory, standardized CSV files are generated, including short_results.csv

(containing key enrichment fields for summary), long_results.csv (providing a more comprehensive view of the full results), and terms_annotations_results.csv (summarizing the input genes contributing to each enriched term). Additionally, a /terms_annotations/ directory is included, containing one subdirectory per enriched term with lists of all genes annotated to that term and the subset from the user's input.

For each gene list analysed, the results directory includes the short_results.csv and long_results.csv files, a terms_annotations_results.csv summary, a /terms_annotations/ directory with term-specific gene mappings, and a TSV file (named "cumulative_distribution.tsv") to support data-driven gene filtering based on recurrence across enriched terms. As shown in Figure 8, the directory for PANTHER outputs run under /panther/<gene_list>/results/ exemplifies this structure, which is mirrored for the other tools, regardless of whether the underlying method is based on over-representation statistics (PANTHER and g:Profiler) or gene ranking and enrichment scores (GSEA).

Figure 8. Contents of /panther directory generated by Module 4, saved under /panther/<gene_list>/results/ in the assigned data directory. This directory contains: a subdirectory containing individual long-format result files for each enriched dataset (the /gene_sets_results directory), the terms annotations directory with a subdirectory for every enriched term (the /terms_annotations directory), the cumulative distribution TSV file of gene recurrence across enriched terms (named "cumulative_distribution.tsv"), and the generated results CSV files (namely the "short_results.csv", "long_results.csv" and "terms_annotations_results.csv").

Although the file structure is standardized, the content of long_results.csv varies according to the enrichment engine. For example, g:Profiler includes fields such as term ID, description, *P*-value, precision, recall, effective domain size, and hierarchical parent, reflecting its ontology-sensitive enrichment model and multiple testing corrections. PANTHER long_results.csv reports fold enrichment, FDR, plus/minus, and term-specific gene counts (e.g., number in list and reference), consistent with its traditional overrepresentation test model applied across diverse gene set libraries. This standardization facilitates consistent parsing, display, and filtering regardless of the tool used.

Each tool also produces a terms_annotations_results.csv, which offers a structured summary for each enriched term. This summary includes the term name and ID, source, the number of input genes associated with the term (ListCount), gene IDs and symbols from the input list that map to the term, total genes annotated to the term (TermCount), all genes annotated to the term, and a computed ratio representing the proportion of input genes relative to total genes annotated. This file serves as a biologically meaningful link between enrichment results and the input gene list. Complementing this, the /terms_annotations/ directory contains per-term subdirectories with two annotation files: genes_in_term, listing all genes annotated to the term from the reference dataset, and genes_in_list, listing the subset from the user input, typically including UniProtKB IDs and gene symbols.

All modules also generate a cumulative_distribution.tsv, which quantifies how frequently each gene in the input gene list appears across enriched terms. This file provides a quantitative overview of how frequently each gene from the input list is associated with enriched terms across the analysed gene sets. This distribution-based analysis enables users to assess the redundancy or specificity of gene-term associations, helping distinguish biologically informative signals from potentially non-specific or overly generic genes. The file contains four fields: N_Occurrences (the number of distinct enrichment terms in which a gene appears), N_Genes (the count of unique genes occurring exactly N times), Proportions (percentage of all input genes occurring exactly N times, calculated as (N_Genes / total number of unique genes) × 100), and Cumulative_Distribution (a running total of the Proportions column, indicating the percentage of genes appearing in up to N enriched terms).

In addition to these core standardized files, the PANTHER module uniquely provides a /gene_sets_results/ subdirectory that stores individual result files for each annotation dataset tested. These files (e.g., results_GO_BP.csv or results_REACTOME_PATHWAY.csv) contain statistically significant long-format outputs filtered at FDR *P*-value < 0.05, offering additional detailed insights into enrichment across specific gene set categories.

3.5.6. Modules 5 and 6 GSEA Outputs Directory (/gsea)

The GSEA workflow, implemented through Modules 5 and 6, is fully integrated into the enrichment pipeline and follows the same results output logic as ORA-based tools. Figure 9 illustrates the organization of inputs and outputs in the GSEA workflow. Module 5 prepares essential inputs, including gene expression data, phenotype class file, gene

set files (GMT format), and parameter configuration files, and generates the ranked gene lists for preranked analyses, stored under the /gsea/preranked_lists/ directory. Module 6 then consumes these inputs to perform GSEA runs in Classic or Preranked modes, producing the corresponding results directories.

Figure 9. Contents of the /gsea directory after running two GSEA Preranked analyses and one GSEA Classic analysis, by Modules 5 and 6. The directory contains the Module 5 prepared input files, including gene expression matrices (named “expression_dataset.gct”), phenotype class file (named “phenotype_labels.cls”), gene set file used in GSEA runs (named “m2.all.v2025.1.Mm.entrez.gmt”), and the generated parameter files by Module 5 for each individual run (namely the ”gsea_parameters_*”) and stored ranked gene lists in the /gsea/preranked_lists/ directory for preranked analyses. Module 6 execution saves results in the directory /results.

All results are stored in the /gsea/results/ directory, where each GSEA run is saved in its own subdirectory. For GSEA Classic runs, the directory name includes the phenotype classes followed by the GseaClassic suffix, while for GSEA Preranked runs the directory name reflects the ranked gene list used and ends with GseaPreranked. The name of the gene set used for this analysis is also included as prefix to the directory name, as exemplified in Figure 10.

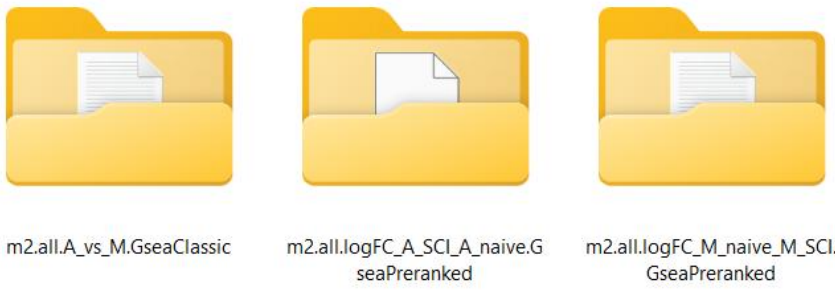

Figure 10. Contents of a results directory of Module 6 showing two GSEA Preranked runs and one GSEA Classic run. The directory includes: the results directory for a GSEA Classic run comparing phenotypes A and M using the m2.all gene set (named “m2.all.A_vs_M.GseaClassic”); and results directories for two GSEA Preranked runs using the same m2.all gene set with preranked lists, the /m2.all.logFC_A_SCI_A_naive.GseaPreranked and /m2.all.logFC_M_SCI_M_naive.GseaPreranked directories.

Each result directory produced by GSEA adheres to the same structural layout as those from g:Profiler and PANTHER. Inside each GSEA output directory are the short_results.csv and long_results.csv files summarizing and detailing enriched gene sets, respectively. These results files contain specific fields such as enrichment score (ES), normalized enrichment score (NES), nominal *P*-value, FDR *Q*-value, rank at max, and leading-edge subset, metrics central to GSEA enrichment strategy based on the distribution of gene ranks across sets. Also present are the terms_annotations_results.csv file and the /terms_annotations directory containing per-gene set mappings of all annotated genes and the subset found in the input ranked list. The cumulative_distribution.tsv file, computed from the GSEA results, enables the same type of frequency-based gene filtering as in ORA-based tools, allowing users to remove genes that appear across too many gene sets and may thus contribute noise to the interpretation. Additionally, the gene set file used for the analysis is stored alongside the results to maintain traceability of the gene sets applied.

In addition to this standardized structure, GSEA also generates a /raw_GSEA_output/ subdirectory within each results directory. This subdirectory contains the original GSEA output files, including all HTML visual reports, enrichment plots, ranked list files, and enrichment statistics as provided by the native GSEA software. These results directory structure is illustrated in Figure 11.

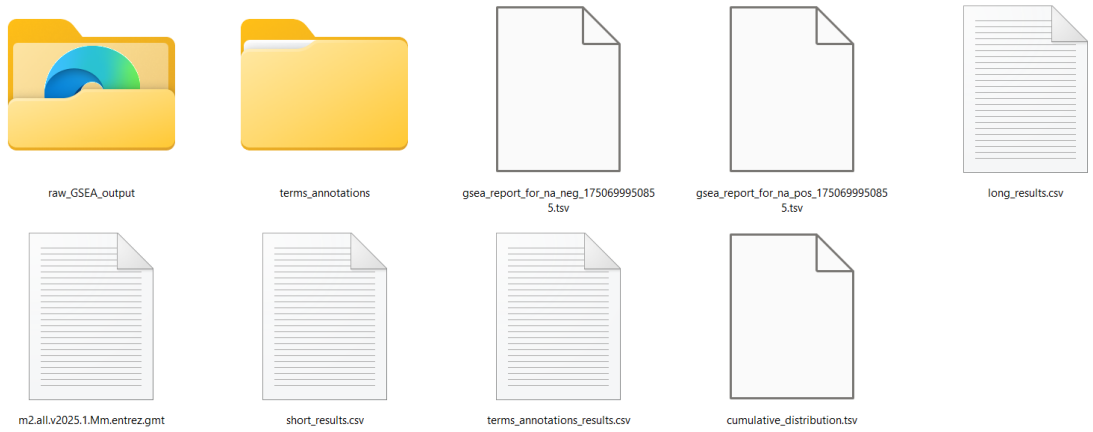

Figure 11. Contents of a results directory for a GSEA analysis generated by Module 6, located under the /gsea/results/<run_name> directory in the assigned data directory. This directory contains: the /raw_GSEA_output/ subdirectory with original GSEA output files, the terms annotations directory with a subdirectory for every enriched term (the /terms_annotations directory), the GSEA raw reports (namely gsea_report_*), the generated results CSV files (namely the “short_results.csv”, “long_results.csv” and “terms_annotations_results.csv”), the cumulative distribution TSV file of gene recurrence across enriched terms (named “cumulative_distribution.tsv”) and the gene set file used during GSEA run (named “m2.all.v2025.1.Mm.entrez.gmt”).

The consistent normalization and formatting applied to these results ensure that, despite GSEA methodological differences from ORA tools, its outputs can be interpreted, filtered, and integrated using the same frameworks and tools established in earlier modules.

3.6. Customization of the analysis

The pipeline's modular design allows users to run modules individually, providing their own pre-prepared inputs as needed. This flexibility makes it possible to skip steps, such as bypassing Module 1 to run multiple pre-prepared gene lists for over-representation analysis or skipping Module 5 when using ready-made inputs for Gene Set Enrichment Analysis. In this section, it is explained how to prepare proper input formats for each module and where to place these files for seamless integration.

3.6.1. Module 2: IDs mapping info

This module can take as input pre-prepared gene lists to map the gene information. These lists must be placed inside a directory named /prepared_gene_lists, located in the assigned data directory.

3.6.2. Modules 3 and 4: gProfiler_plus and PANTHER_plus

The Over-Representation Analysis method modules can take as input pre-mapped gene lists, as prepared by module 2. These mapped gene lists must be placed inside a directory named /mapped_gene_lists located in the assigned data directory.

3.6.3. Module 6: GSEA_plus

The Gene Set Enrichment Analysis method module can take as input pre-prepared input files for either GSEA Classic or GSEA Preranked. These files must be located in the assigned data directory and correctly configured in the gsea_parameters file, also located in the data directory.

3.6.4. Module 7: Process EA results

This module can be run as a standalone to process and filter enrichment results. To perform this, simply define the filter parameters in the config0 pipeline configuration file and it takes as input all the enrichment modules output directories present in the data directory. For this the directories must have the default name produced by the modules.

3.7. Usage cases

This section demonstrates the practical application of the developed pipeline by showcasing how it can be utilized with a real case study example. It details the setup process, including data preparation and tool configuration, to illustrate the pipeline workflow in a typical bioinformatics enrichment analysis scenario. Furthermore, the results obtained through the pipeline are compared against those from established methods to evaluate performance, accuracy, and usability. This comparison highlights the pipeline effectiveness and potential advantages in streamlining multi-tool enrichment analyses.

3.7.1. Usage case 1

In this usage case, an enrichment analysis is performed using data from a published article [91], and the results obtained using the pipeline compared with those reported in this article using PANTHER overrepresentation test. The goal is the identification of the key functional terms associated with spinal cord injury (SCI) in the spiny mouse (*Acomys cahirinus*) compared to the common laboratory mice (*Mus musculus*). The article highlights that *Acomys cahirinus* spontaneously regains motor and urinary functions after complete spinal cord transection, forming a scarless, pro-regenerative tissue at the injury site, unlike other adult mammals. The authors demonstrate that this remarkable recovery is linked to rewired glycosylation activity, specifically an increase in the enzyme *B3gnt7*, which promotes axon growth. The research identifies *Acomys cahirinus* as a unique model for understanding mammalian central nervous system regeneration and suggests *B3gnt7* as a potential therapeutic target for the treatment of human SCI.

In the study, 213 genes were identified as having a significant expression difference in at least one of four conditions, namely: uninjured and injured animals for both *Mus musculus* and *Acomys cahirinus*. The differentially expressed CDS (coding sequence) isoforms were partitioned into 11 clusters, by cutting the hierarchically clustered genes tree at 10% the height of the tree. Only clusters in which over 75% of members showed a statistically significant difference between *Mus musculus* and *Acomys cahirinus* were chosen for functional category analysis using PANTHER. Gene terms were found to be overrepresented in Clusters 4, 6, and 22. In the analysis here performed, only genes from these three distinct clusters were used to perform enrichment analysis using the developed pipeline.

In the published study, the terms identified by PANTHER, based on the analysis of average fragments per kilobase of transcript per million mapped reads (FPKM) values obtained through transcriptomic analysis, showed statistically significant enrichment in

inflammation-related pathways and acetylglucosaminyltransferase activity. These enrichments were specifically associated with clusters 4, 6, and 22 (Table 2).

Table 2. Summary of genes grouped by cluster and their involvement in significantly enriched pathways identified in the enrichment analysis.

Associated pathways	Cluster	Related genes
Toll Receptor signaling pathway	4	*Tlr4, Ly96, Irak4, Nfkb2, Ikbke*
Inflammation mediated by chemokine and cytokine signaling pathway	4	*Tlr4, Ly96, Irak4, Nfkb2, Ikbke, Plcb2, C5ar1, Myh9, Ifngr1*
	6	*Rgs17, Dclk3*
Interleukin signaling pathway	4	*Elk3, Shc1, Il4ra1, Il6ra, Il13ra1, Il10ra*
Acetylglucosaminyltransferase activity	4	*Pgm5, Mfng*
	6	*Ndst3, Ndst3, B3galnt1*
	22	*B3gnt7*

This application used five modules, beginning with Module 1 (prepare_lists) to extract member genes from each cluster into separate gene lists. Next, Module 2 (ids_mapping_info) was used to map gene information. Each mapped gene list was then analysed using Module 3 (gProfiler_plus) with g:Profiler g:GOSt tool, and Module 4 (PANTHER_plus) for the PANTHER statistical overrepresentation test. The main pipeline configuration file, config0, used to perform over-representation enrichment analyses using genes from a given cluster, is shown in Figure 12.

```
1  # General parameters
2  tools="1,2,3,4,7"
3  species="mus_musculus"
4
5  # Filter terms annotations results parameters
6  max_annot=500
7  min_ratio=
8  max_occur=
9
```

Figure 12. The config0 file configured to pre-process data (Modules 1 and 2), perform Over-Representation Analyses (Modules 3 and 4), and filter enrichment results (Module 7). Lines beginning with "#" are comments. Line 2 specifies the modules to run, line 3 sets the target

species name, and lines 6–8 define parameters for filtering enriched term annotation results (in this case only max_annot was defined).

To extract genes belonging to a specific cluster (with all clusters stored in the same file), Module 1 was used to filter out genes marked with a value of 1, generating an individual list of genes for that cluster (Figure 13).

```
1  # Index starts at 1!
2  input="gene_expression_test_data.txt"
3  gene=2
4
5  # Column index with pre-evaluated genes (either 1 or 0)
6  select=8
7
```

Figure 13. Module 1 configuration file, config1, setup to extract pre-evaluated genes. The lines starting with "#" are comments. Line 2 is the input dataset file. Line 3 is the indicated column index with the gene identification (Gene IDs). Line 6 is the indicated column index with the marked genes (value 1) from a given cluster.

Next, the extracted gene list, saved in the /prepared_gene_lists directory, is processed by Module 2 (ids_mapping_info), which maps gene information. The mapped gene list is saved in the /mapped_gene_lists directory and subsequently used for Over-Representation enrichment analysis with g:Profiler g:GOSt tool (Module 3) and the PANTHER overrepresentation test (Module 4). The analysis outputs are automatically filtered to retain only statistically significant results (corrected FDR *P*-value < 0.05), organized in a standard format, and saved in individual gene list subdirectories inside each module's output directory.

Example gProfiler results directory: **/data/gprofiler/cluster4/results/…**

Example PANTHER results directory: **/data/panther/cluster4/results/…**

Finally, Module 7 (process_ea_results) was used to filter out terms with limited relevance, or overly general, to the biological processes under investigation, enabling a more focused analysis aimed at identifying the most meaningful functional terms. The cutoff was set for the maximum total number of annotations a category could have, to improve interpretability and highlight biologically meaningful patterns. A threshold of 500 annotations was chosen based on our analysis, to filter out broad, high-level terms and retain more specific functional terms relevant to spinal cord injury recovery. This threshold is context-dependent and may need to be adjusted based on the enrichment results obtained for other analyses.

The enrichment analysis of Cluster 4 genes, performed using Module 4 for the PANTHER statistical overrepresentation test, revealed statistically significant enrichment in several immune-related pathways, many of which overlap with those reported in the case study (highlighted in yellow in Figure 14). Acetylglucosaminyltransferase activity, which was identified in the original case study, was not detected in this analysis. Enrichment was observed for terms such as *Neutrophil Degranulation* (R-MMU-6798695), *interleukins* and *cytokines* related pathways from the Reactome Pathways dataset. These results were obtained by reviewing the short results file (/data/panther/cluster4/results/short_results.csv) in Excel and filtering by dataset (see Figure 14).

TermID	Name	P-Value(FDR)	Source
P00031	Inflammation_mediated_by_chemokine_and_cytokine_signaling_pathway	0.004249716216058751	PANTHER_PATHWAY
P00036	Interleukin_signaling_pathway	0.00021646094554418277	PANTHER_PATHWAY
P00054	Toll_receptor_signaling_pathway	0.0001781700284287344	PANTHER_PATHWAY
null	UNCLASSIFIED	0.00005238213353175026	PANTHER_PATHWAY
R-MMU-936964	Activation_of_IRF3__IRF7_mediated_by_TBK1__IKBKE	0.013265714800884001	REACTOME_PATHWAY
R-MMU-1280215	Cytokine_Signaling_in_Immune_system	0.009498959088445195	REACTOME_PATHWAY
R-MMU-168256	Immune_System	3.741244322741035E-7	REACTOME_PATHWAY
R-MMU-168249	Innate_Immune_System	8.245386840734537E-7	REACTOME_PATHWAY
R-MMU-166166	MyD88_independent_TLR4_cascade	0.014026178869177843	REACTOME_PATHWAY
R-MMU-6798695	Neutrophil_degranulation	0.0012104461758806265	REACTOME_PATHWAY
R-MMU-449147	Signaling_by_Interleukins	0.0012626964149479464	REACTOME_PATHWAY
R-MMU-166016	Toll_Like_Receptor_4__TLR4__Cascade	0.025294721411736613	REACTOME_PATHWAY
R-MMU-937061	TRIF_TICAM1__mediated_TLR4_signaling	0.012272906510530613	REACTOME_PATHWAY

Figure 14. Excerpt from the short results file generated by Module 4 (PANTHER_plus), showing enriched pathways from Reactome and PANTHER Pathways datasets. Columns include TermID (pathway identifier), Name (pathway name), *P*-Value [FDR] (adjusted significance), and Source (datasets of origin). Pathways matching the case study are highlighted in yellow.

Filtering the annotation results using Module 7, excluding overly broad terms with more than 500 annotations, reduced the list of enriched terms from 209 to 105. This refined set highlights the genes from Cluster 4 that contribute meaningfully to the enriched terms across the analysed datasets.

Among these, three PANTHER pathways reported in the case study were found to contain Cluster 4 genes, confirming consistency between analyses. These matching pathways are highlighted in yellow in Figure 15. In addition, several other significantly enriched terms from the Reactome Pathways dataset, such as *cytokine regulation*, *interleukins* and *toll receptor* related activation were reported.

TermID	Name	Source	Ratio	ListCount	Genes_in_list	TermCount
P00054	Toll_receptor_signaling_pathway	PANTHER_PATHWAY	9.09	5	'Ly96' 'Irak4' 'Ikbke' 'Tlr4' 'Nfkb2'	55
P00036	Interleukin_signaling_pathway	PANTHER_PATHWAY	6.38	6	'Elk3' 'Il13ra1' 'Il6ra' 'Il10ra' 'Shc1' 'Il4ra'	94
P00031	Inflammation_mediated_by_chemokine_and_cytokine_signaling_pathway	PANTHER_PATHWAY	2.70	7	'C5ar1' 'Ifngr1' 'Shc1' 'Ikbke' 'Nfkb2' 'Plcb2' 'Myh9'	259
R-MMU-936964	Activation_of_IRF3__IRF7_mediated_by_TBK1__IKBKE	REACTOME_PATHWAY	16.67	3	'Ly96' 'Ikbke' 'Tlr4'	18
R-MMU-166166	MyD88_independent_TLR4_cascade	REACTOME_PATHWAY	5.21	5	'Ly96' 'Dusp6' 'Ikbke' 'Tlr4' 'Nfkb2'	96
R-MMU-937061	TRIF_TICAM1__mediated_TLR4_signaling	REACTOME_PATHWAY	5.21	5	'Ly96' 'Dusp6' 'Ikbke' 'Tlr4' 'Nfkb2'	96
R-MMU-166016	Toll_Like_Receptor_4__TLR4__Cascade	REACTOME_PATHWAY	4.67	5	'Ly96' 'Dusp6' 'Ikbke' 'Tlr4' 'Nfkb2'	107
R-MMU-449147	Signaling_by_Interleukins	REACTOME_PATHWAY	3.47	9	'Dusp6' 'Il13ra1' 'Il6ra' 'Irak4' 'Il10ra' 'Shc1' 'Syk' 'Il4ra' 'Nfkb2'	259
R-MMU-1280215	Cytokine_Signaling_in_Immune_system	REACTOME_PATHWAY	2.33	10	'Dusp6' 'Il13ra1' 'Il6ra' 'Irak4' 'Ifngr1' 'Il10ra' 'Shc1' 'Syk' 'Il4ra' 'Nfkb2'	430

Figure 15. Excerpt from the term annotations results file generated by Module 4 (PANTHER_plus), showing the set of enriched terms from Reactome and PANTHER Pathways datasets (highlighted in yellow) of Cluster 4 genes. Columns include TermID (term identifier), Name (category or pathway name), Source (annotation dataset), Ratio [%] (percentage of Cluster 4 genes in the term), ListCount (number of Cluster 4 genes in the term), Genes_in_list (their symbols), and TermCount (total genes in the term).

The annotation results for the pathways reported in the case study identified in this analysis are summarized in Table 3.

Table 3. Summary of genes of Cluster 4 and their involvement in significantly enriched pathways identified using Module 4.

Associated pathways	**Related genes**
Toll Receptor signaling pathway	*Tlr4, Ly96, Irak4, Nfkb2, Ikbke*
Inflammation mediated by chemokine and cytokine signaling pathway	*C5ar1, Ifngr1, Shc1, Ikbke, Nfkb2, Plcb2, Myh9*
Interleukin signaling pathway	*Elk3, Shc1, Il4ra1, Il6ra, Il13ra1, Il10ra*

The enrichment analysis method applied in this usage case successfully replicated the key findings reported in the original case study. By comparing the genes from Cluster 4 involved in enriched pathways (as shown in Table 1 and Table 2), the same inflammation-related and interleukin-related pathways were identified, and the gene overlap was found to be high. The only notorious difference was the absence of the genes *Tlr4*, *Ly96* and *Irak4* in the *Inflammation mediated by chemokine and cytokine signaling pathway* in this analysis, reported in the case study, as well as *acetylglucosaminyltransferase activity*, as mentioned above.

In contrast, the enrichment results of Cluster 6 are markedly distinct from those reported in the case study. Referring to the initial functional enrichment reported in Table 1 of the case study, neither the *Inflammation mediated by chemokine and cytokine signaling pathway* nor *acetylglucosaminyltransferase activity* appeared as significantly enriched in Cluster 6. Instead, the results obtained for Cluster 6 identified 25 enriched terms, dominated by terms associated with synaptic structure and function, especially components of the synaptic vesicle cycle and postsynaptic membrane architecture (e.g.,

synaptic vesicle membrane, *presynaptic membrane*, *postsynaptic density membrane*), Figure 16). This strong neuronal signature suggests that the Cluster 6 genes may be involved in neurotransmission, synaptic plasticity, or neuronal signaling pathways.

TermID	Name	P-Value(FDR)	Source
GO:0098588	bounding_membrane_of_organelle	0.0010469167867083824	GO_CC
GO:0036477	somatodendritic_compartment	0.014244287223106264	GO_CC
GO:0097060	synaptic_membrane	0.010593578956296527	GO_CC
GO:0098793	presynapse	0.01471871720044463	GO_CC
GO:0045202	synapse	0.01310431494894986	GO_CC
GO:0030133	transport_vesicle	0.017194330560886855	GO_CC
GO:0030658	transport_vesicle_membrane	0.015571281469569797	GO_CC
GO:0099501	exocytic_vesicle_membrane	0.014705611459743962	GO_CC
GO:0030672	synaptic_vesicle_membrane	0.013071654630883521	GO_CC
GO:0043025	neuronal_cell_body	0.014126721245972232	GO_CC
GO:0042734	presynaptic_membrane	0.01643044310382669	GO_CC
GO:0030659	cytoplasmic_vesicle_membrane	0.018897560527685178	GO_CC
GO:0030425	dendrite	0.017499176874503662	GO_CC
GO:0097447	dendritic_tree	0.016609526404719934	GO_CC
GO:0098978	glutamatergic_synapse	0.01550222464440527	GO_CC
GO:0008021	synaptic_vesicle	0.015111853658188991	GO_CC
GO:0012506	vesicle_membrane	0.015953173155824275	GO_CC
GO:0031090	organelle_membrane	0.015912963454938738	GO_CC
GO:0043005	neuron_projection	0.017795587834419483	GO_CC
GO:0070382	exocytic_vesicle	0.017296606399118293	GO_CC
GO:0045211	postsynaptic_membrane	0.01745029328318868	GO_CC
GO:0044297	cell_body	0.01773044397418997	GO_CC
GO:0048786	presynaptic_active_zone	0.018351851199386017	GO_CC
GO:0098839	postsynaptic_density_membrane	0.03178329018151424	GO_CC
R-MMU-112315	Transmission_across_Chemical_Synapses	0.03338339940806513	REACTOME_PATHWAY

Figure 16. Excerpt from the short results, generated by Module 4, of the Cluster 6 analysis with the PANTHER overrepresentation test, showing enriched pathways across the Gene Ontology Cellular Component (GO_CC) dataset and Reactome Pathways. Columns include pathway ID, name, *P*-value (FDR corrected), and source dataset.

Filtering the annotation results using Module 7, excluding overly broad terms with more than 500 annotations, reduced the list of enriched terms from 25 to 11. This refined set highlights the genes from Cluster 6 that contribute meaningfully to the enriched terms across the analysed datasets. A consistent group of genes (*Lin7a*, *Rab3c*, *Slc2a3*, *Sv2a*, *Dlg2*, *Gria1*, *Gabbr1*, and *Dgki*) appear across multiple terms, indicating that they are likely hub genes with broad involvement in synaptic-related cellular components. These findings indicate that Cluster 6 likely plays a role in synaptic activity, as the genes in this cluster appear consistently across terms, demonstrating strong functional coherence in neuronal processes (Figure 17).

TermID	Name	Source	Ratio(%)	ListCount	Genes_in_list	TermCount
GO:0030133	transport_vesicle	GO_CC	1.99	9	'Nts' 'Rab3c' 'Lin7a' 'Slc2a3' 'Sv2a' 'Dlg2' 'Gria1' 'Gabbr1' 'Dgki'	452
GO:0030658	transport_vesicle_membrane	GO_CC	2.70	7	'Rab3c' 'Lin7a' 'Slc2a3' 'Sv2a' 'Dlg2' 'Gria1' 'Dgki'	259
GO:0099501	exocytic_vesicle_membrane	GO_CC	3.31	6	'Rab3c' 'Slc2a3' 'Sv2a' 'Dlg2' 'Gria1' 'Dgki'	181
GO:0030672	synaptic_vesicle_membrane	GO_CC	3.31	6	'Rab3c' 'Slc2a3' 'Sv2a' 'Dlg2' 'Gria1' 'Dgki'	181
GO:0042734	presynaptic_membrane	GO_CC	2.52	7	'Rims3' 'Nptn' 'Kcnj3' 'Slc1a6' 'Gria1' 'Gabbr1' 'Dgki'	278
GO:0008021	synaptic_vesicle	GO_CC	2.35	7	'Rab3c' 'Slc2a3' 'Sv2a' 'Dlg2' 'Gria1' 'Gabbr1' 'Dgki'	298
GO:0070382	exocytic_vesicle	GO_CC	2.22	7	'Rab3c' 'Slc2a3' 'Sv2a' 'Dlg2' 'Gria1' 'Gabbr1' 'Dgki'	315
GO:0045211	postsynaptic_membrane	GO_CC	1.91	8	'Lin7a' 'Lhfpl4' 'Nptn' 'Slc1a6' 'Dlg2' 'Gria1' 'Gabbr1' 'Dgki'	419
GO:0048786	presynaptic_active_zone	GO_CC	3.60	5	'Rims3' 'Nptn' 'Sv2a' 'Gria1' 'Dgki'	139
GO:0098839	postsynaptic_density_membrane	GO_CC	3.05	5	'Lin7a' 'Nptn' 'Dlg2' 'Gria1' 'Dgki'	164
R-MMU-112315	Transmission_across_Chemical_Synapses	REACTOME_PATHWAY	3.65	7	'Lin7a' 'Gnal' 'Kcnj3' 'Slc1a6' 'Dlg2' 'Gria1' 'Gabbr1'	192

Figure 17. Excerpt of enriched term annotations for Cluster 6 genes, identified using the PANTHER Overrepresentation Test (Module 4) and filtered by Module 7. The figure displays enriched pathways from the Gene Ontology Cellular Component (GO_CC) and Reactome Pathways datasets. Columns include term ID, term name, source dataset, enrichment ratio (%), number of genes from Cluster 6 associated with the term (ListCount), the specific genes from the cluster involved (Genes_in_list), and the total number of genes annotated to the term in the reference database (TermCount).

Conversely, the enrichment analysis of Cluster 22 did not identify any statistically significant terms. This result is likely attributable to the small cluster size (13 genes) and limited annotation density, as only one gene (*B3gnt7*) was previously linked to an enriched term in the case study. The PANTHER Overrepresentation Test is statistically conservative with small gene lists, and enrichment detection requires a minimum level of term co-occurrence that was not met.

The pipeline and modules implemented (notably Module 4 for the PANTHER overrepresentation test and Module 7 for post-processing) have successfully replicated most key functional enrichments from the original case study, particularly for Cluster 4. Inflammatory pathways such as Toll receptor signaling, Cytokine signaling and Interleukin-related pathways were confirmed with the same core gene contributors in most cases. This replication strengthens the validity of the auto-Enrich pipeline, showing that it can reproduce curated case study findings under controlled conditions.

The divergence in enrichment results for Cluster 6 reflects a biological rather than technical difference, since the same exact genes were used. This discrepancy may be due to differences in the underlying databases used for enrichment analyses, as these resources are continuously updated and refined over time. Such changes in database content and annotation structures can lead to variations in detected terms, complementing and refining earlier findings. It is also possible that the case study included more liberal filtering or emphasized results with weaker statistical support.

Cluster 22, on the other hand, did not yield statistically significant enrichment results. This is expected given the small size of the cluster (13 genes) and the low annotation density, only one gene (*B3gnt7*), was associated with a relevant term in the case study. The PANTHER Overrepresentation Test, which uses Fisher's Exact Test followed by

multiple testing correction (FDR), is less sensitive for small input lists. Statistical power in enrichment analysis depends on detecting robust, non-random co-occurrences of gene annotations, which is difficult to achieve with sparse data. Consequently, isolated hits are strongly penalized, and unless multiple genes contribute to the same term, significance is unlikely.

The enrichment analysis of Cluster 4 using the g:Profiler g:GOSt tool (Module 3) identified 862 enriched terms across eight annotation datasets. Of these, 610 terms were from the Gene Ontology Biological Process (GO:BP) dataset, 84 from Molecular Function (GO:MF), and 51 from Cellular Component (GO:CC), totaling 745 GO-related terms. The remaining 117 enriched terms were distributed across TransFac (46), CORUM (25), KEGG Pathways (22), Reactome Pathways (21), and WikiPathways (3).

A direct comparison between the g:Profiler g:GOSt tool and the PANTHER overrepresentation test, using the same dataset, Reactome Pathways (*REAC*; Figure 18), shows a high degree of consistency in the biological themes identified. Both methods revealed enrichment for immune-related pathways, including *cytokine* and *interleukin signaling* and *Toll-like receptor activation*, in Cluster 4 genes of Module 3 (g:GOSt) and Module 4 (PANTHER), as highlighted in yellow in Figure 18. This overlap suggests minimal discrepancy between the two enrichment approaches. However, g:Profiler reported a larger number of enriched Reactome pathways (21) compared to PANTHER (9), although many g:Profiler results represent variations of the same core pathway (e.g., four separate *Toll receptor*-related terms).

Beyond the shared Reactome pathways, g:Profiler also identified enriched terms from WikiPathways (*WP*, Figure 18), such as *Fibrin Complement Receptor 3 Signaling* and the *Tyrobp Causal Network in Microglia* (highlighted in orange in Figure 18). These additional findings offer a microglia-centric perspective and suggest alternative immune mechanisms potentially underlying the *Acomys cahirinus* capacity for immune-mediated regeneration. In particular, the enrichment of the *Tyrobp network* points to a coordinated, possibly anti-inflammatory microglial phenotype supportive of tissue repair and scarless healing. This expands the biological insight beyond Reactome pathways, capturing fine-scale immune and phagocytic processes relevant to spontaneous spinal cord repair.

TermID	Name	P-Value	Source
REAC:R-MMU-936964	Activation_of_IRF3__IRF7_mediated_by_TBK1__IKK__IKBKE_	0.013121490102043179	REAC
REAC:R-MMU-1280215	Cytokine_Signaling_in_Immune_system	0.013121490102043179	REAC
WP:WP5128	Fibrin_complement_receptor_3_signaling_pathway	0.019776108809568357	WP
REAC:R-MMU-9707616	Heme_signaling	0.033601960084959255	REAC
REAC:R-MMU-168256	Immune_System	0.00013129249651539776	REAC
REAC:R-MMU-168249	Innate_Immune_System	0.00023757235780290568	REAC
REAC:R-MMU-9020558	Interleukin_2_signaling	0.033601960084959255	REAC
REAC:R-MMU-6785807	Interleukin_4_and_Interleukin_13_signaling	0.033601960084959255	REAC
REAC:R-MMU-8851805	MET_activates_RAS_signaling	0.033601960084959255	REAC
REAC:R-MMU-975155	MyD88_dependent_cascade_initiated_on_endosome	0.033601960084959255	REAC
REAC:R-MMU-166166	MyD88_independent_TLR4_cascade_	0.013121490102043179	REAC
REAC:R-MMU-6798695	Neutrophil_degranulation	0.013121490102043179	REAC
REAC:R-MMU-416700	Other_semaphorin_interactions	0.04276268407810407	REAC
REAC:R-MMU-449147	Signaling_by_Interleukins	0.0033551652148213714	REAC
REAC:R-MMU-4755510	SUMOylation_of_immune_response_proteins	0.033601960084959255	REAC
REAC:R-MMU-166016	Toll_Like_Receptor_4__TLR4__Cascade	0.019234397934680576	REAC
REAC:R-MMU-168181	Toll_Like_Receptor_7_8__TLR7_8__Cascade	0.033601960084959255	REAC
REAC:R-MMU-168138	Toll_Like_Receptor_9__TLR9__Cascade	0.03523400383070198	REAC
REAC:R-MMU-168898	Toll_like_Receptor_Cascades	0.033601960084959255	REAC
WP:WP88	Toll_like_receptor_signaling	0.00966412405216	WP
REAC:R-MMU-975138	TRAF6_mediated_induction_of_NFkB_and_MAP_kinases_upon_TLR7_8_or_9_activation	0.033601960084959255	REAC
REAC:R-MMU-937061	TRIF__TICAM1__mediated_TLR4_signaling_	0.013121490102043179	REAC
REAC:R-MMU-2562578	TRIF_mediated_programmed_cell_death	0.033601960084959255	REAC
WP:WP3625	Tyrobp_causal_network_in_microglia	0.00966412405216	WP

Figure 18. Excerpt from the short results, generated by Module 3, of Cluster 4 genes analysis with the g:Profiler g:GOSt tool, showing enriched pathways across Reactome Pathways (REAC) and WikiPathways (WP) datasets. Related pathways matching the case study are highlighted in yellow. Other possibly relevant enriched terms from WikiPathways dataset are highlighted in orange. Columns include pathway ID (TermID), name, *P*-value (FDR), and source dataset.

The enrichment analysis for Cluster 4 genes yielded 862 initial enriched terms, which was refined to 538 by excluding terms associated with more than 500 genes to remove broad, non-specific annotations. This reduction sharpened the focus on biologically relevant pathways, enhancing interpretability and highlighting the specific functional contributions of Cluster 4 genes across datasets.

The g:Profiler g:GOSt analysis not only recapitulated the core immune-related pathways identified by PANTHER (highlighted in yellow in Figure 19) but also introduced novel gene-pathway associations through its inclusion of WikiPathways terms. Notably, the *Tyrobp causal network in microglia* (*WP:WP3625*) identified genes such as *Il13ra1*, *Il10ra*, *Gal3st4*, *Hlx*, and *Ppp1r18*, which were absent in the PANTHER results (highlighted in orange in Figure 19). These genes suggest a nuanced microglial response in *Acomys cahirinus*, possibly involving immunomodulation, ECM remodelling, and chromatin regulation, all crucial for promoting a regenerative tissue environment. Furthermore, the *Fibrin complement receptor 3 signaling* (*WP:WP5128*) pathway introduces a dimension of immune clearance and receptor-mediated signaling involving the genes *Syk* and *Tlr4*, underscoring the complexity of the immune landscape post-injury.

TermID	Name	Source	Ratio(	ListCount	Genes_in_list	TermCount
REAC:R-MMU-936964	Activation_of_IRF3__IRF7_mediated_by_TBK1__IKK__IKBKE_	REAC	16.67	3	'Ly96 'Ikbke 'Tlr4	18
REAC:R-MMU-1280215	Cytokine_Signaling_in_Immune_system	REAC	2.33	10	'Dusp6 'Il13ra1 'Il6ra 'Irak4 'Ifngr1 'Il10ra 'Shc1 'Syk 'Il4ra 'Nfkb2	430
WP:WP5128	Fibrin_complement_receptor_3_signaling_pathway	WP	9.09	4	'Ly96 'Irak4 'Syk 'Tlr4	44
REAC:R-MMU-9707616	Heme_signaling	REAC	18.18	2	'Ly96 'Tlr4	11
REAC:R-MMU-9020558	Interleukin_2_signaling	REAC	22.22	2	'Shc1 'Syk	9
REAC:R-MMU-6785807	Interleukin_4_and_Interleukin_13_signaling	REAC	18.18	2	'Il13ra1 'Il4ra	11
REAC:R-MMU-8851805	MET_activates_RAS_signaling	REAC	22.22	2	'Shc1 'Hgf	9
REAC:R-MMU-975155	MyD88_dependent_cascade_initiated_on_endosome	REAC	4.82	4	'Ly96 'Dusp6 'Tlr4 'Nfkb2	83
REAC:R-MMU-166166	MyD88_independent_TLR4_cascade_	REAC	5.21	5	'Ly96 'Dusp6 'Ikbke 'Tlr4 'Nfkb2	96
REAC:R-MMU-416700	Other_semaphorin_interactions	REAC	15.38	2	'Plxnc1 'Plxnd1	13
REAC:R-MMU-449147	Signaling_by_Interleukins	REAC	3.47	9	'Dusp6 'Il13ra1 'Il6ra 'Irak4 'Il10ra 'Shc1 'Syk 'Il4ra 'Nfkb2	259
REAC:R-MMU-4755510	SUMOylation_of_immune_response_proteins	REAC	18.18	2	'Ikbke 'Nfkb2	11
REAC:R-MMU-166016	Toll_Like_Receptor_4__TLR4__Cascade	REAC	4.67	5	'Ly96 'Dusp6 'Ikbke 'Tlr4 'Nfkb2	107
REAC:R-MMU-168181	Toll_Like_Receptor_7_8__TLR7_8__Cascade	REAC	4.76	4	'Ly96 'Dusp6 'Tlr4 'Nfkb2	84
REAC:R-MMU-168138	Toll_Like_Receptor_9__TLR9__Cascade	REAC	4.55	4	'Ly96 'Dusp6 'Tlr4 'Nfkb2	88
REAC:R-MMU-168898	Toll_like_Receptor_Cascades	REAC	3.76	5	'Ly96 'Dusp6 'Ikbke 'Tlr4 'Nfkb2	133
WP:WP88	Toll_like_receptor_signaling	WP	12.12	4	'Irak4 'Ikbke 'Tlr4 'Nfkb2	33
REAC:R-MMU-975138	TRAF6_mediated_induction_of_NFkB_and_MAP_kinases_upon_TLR7_8_or_9_activation	REAC	4.88	4	'Ly96 'Dusp6 'Tlr4 'Nfkb2	82
REAC:R-MMU-937061	TRIF__TICAM1__mediated_TLR4_signaling_	REAC	5.21	5	'Ly96 'Dusp6 'Ikbke 'Tlr4 'Nfkb2	96
REAC:R-MMU-2562578	TRIF_mediated_programmed_cell_death	REAC	22.22	2	'Ly96 'Tlr4	9
WP:WP3625	Tyrobp_causal_network_in_microglia	WP	8.62	5	'Il13ra1 'Gal3st4 'Hlx 'Il10ra 'Ppp1r18	58

Figure 19. Excerpt from the terms annotations results, generated by Module 3, of Cluster 4 genes analysis with g:Profiler g:GOSt tool, filtered by Module 7, showing enriched pathways across Reactome Pathways (REAC) and WikiPathways (WP) datasets. Pathways corresponding to the case study are highlighted in yellow, while additional potentially relevant WikiPathways terms are highlighted in orange.Columns include term ID, name, source dataset, ratio (%), number of genes from the cluster (ListCount), genes from the cluster (Genes_in_list) and the total number of annotations from the enriched term (TermCount).

The analysis of Cluster 6 genes using the g:Profiler g:GOSt tool identified 179 enriched terms across all functional annotation datasets, a considerably higher number than the 25 terms reported by the PANTHER overrepresentation test using the same gene set. This trend aligns with the pattern observed for Cluster 4, where g:Profiler consistently reported a broader spectrum of enriched terms compared to PANTHER.

Despite this difference in term number, the Reactome Pathways results from both methods converge on a common biological theme: neuronal function and synaptic activity. Specifically, enriched pathways such as *Activation of GABAB receptors*, *Transmission across Chemical Synapses*, and *Neurotransmitter receptors and postsynaptic signal transmission* indicate a functional role for Cluster 6 genes in inhibitory synaptic signaling and neuronal system regulation (Figure 20). This is further supported by the enrichment of the WikiPathways term *Dravet syndrome SCN1A A1783V point mutation model* (*WP:WP5298*), which implicates genes potentially associated with epileptogenic or neurodevelopmental mechanisms.

TermID	Name	P-Value	Source
WP:WP5298	Dravet_syndrome_Scn1a_A1783V_point_mutation_model	0.004600714143416062	WP
REAC:R-MMU-112315	Transmission_across_Chemical_Synapses	0.005990437358359157	REAC
REAC:R-MMU-991365	Activation_of_GABAB_receptors	0.03424390029150863	REAC
REAC:R-MMU-977444	GABA_B_receptor_activation	0.03424390029150863	REAC
REAC:R-MMU-112314	Neurotransmitter_receptors_and_postsynaptic_signal_transmission	0.03424390029150863	REAC
REAC:R-MMU-112316	Neuronal_System	0.03424390029150863	REAC

Figure 20. Excerpt from the short results, generated by Module 3, of Cluster 6 genes analysis with the g:Profiler g:GOSt tool, showing enriched pathways across Reactome Pathways (REAC) and WikiPathways (WP) datasets. Columns include pathway ID (TermID), name, *P*-value (FDR), and source dataset.

The overlap between PANTHER and g:Profiler enrichment results highlight a functionally cohesive cluster centered on synaptic structure and neurotransmission, with recurrent involvement of core genes such as *Gabbr1*, *Gria1*, *Dlg2*, and *Lin7a* (presented in Figure 21). The consistency across independent annotation tools, along with additional specificity from g:Profiler, suggests that Cluster 6 may represent a neuro-functional module engaged in synaptic signaling homeostasis, potentially linked to recovery processes following spinal cord injury.

TermID	Name	Source	Ratio(%)	ListCount	Genes_in_list	TermCount
WP:WP5298	Dravet_syndrome_Scn1a_A1783V_point_mutation_model	WP	5.48	4	'Slc2a3 'Kcnj3 'Gria1 'Gabbr1	73
REAC:R-MMU-112315	Transmission_across_Chemical_Synapses	REAC	3.65	7	'Lin7a 'Gnal 'Kcnj3 'Slc1a6 'Dlg2 'Gria1 'Gabbr1	192
REAC:R-MMU-991365	Activation_of_GABAB_receptors	REAC	7.69	3	'Gnal 'Kcnj3 'Gabbr1	39
REAC:R-MMU-977444	GABA_B_receptor_activation	REAC	7.69	3	'Gnal 'Kcnj3 'Gabbr1	39
REAC:R-MMU-112314	Neurotransmitter_receptors_and_postsynaptic_signal_transmission	REAC	3.79	5	'Gnal 'Kcnj3 'Dlg2 'Gria1 'Gabbr1	132
REAC:R-MMU-112316	Neuronal_System	REAC	2.28	7	'Lin7a 'Gnal 'Kcnj3 'Slc1a6 'Dlg2 'Gria1 'Gabbr1	307

Figure 21. Excerpt from the terms annotations results, generated by Module 3, of cluster 6 genes analysis with g:Profiler g:GOSt tool, showing enriched pathways across Reactome Pathways (REAC) and WikiPathways (WP) datasets. Columns include term ID, name, source dataset, ratio (%), number of genes from the cluster (ListCount), genes from the cluster (Genes_in_list) and the total number of annotations from the enriched term (TermCount).

Although Cluster 22 genes did not yield statistically significant enrichment through PANTHER, the g:Profiler analysis identified several terms associated with sugar metabolism and transport. Specifically, genes such as *Slc2a5* (a fructose transporter), *Npl* (involved in sialic acid catabolism), and *Sult1c2* (a sulfotransferase) were associated with terms linked to fructose handling, glycan modification, and sugar metabolism (Figure 22). These findings, while based on individual gene associations, complement the case study's emphasis on rewired glycosylation and the role of *B3gnt7* in regenerative outcomes.

TermID	Name	Source	Ratio(%)	ListCount	Genes_in_list	TermCount
REAC:R-MMU-8963676	Intestinal_absorption	REAC	50.00	1	'Slc2a5	2
REAC:R-MMU-8981373	Intestinal_hexose_absorption	REAC	50.00	1	'Slc2a5	2
GO:0004062	aryl_sulfotransferase_activity	GO:MF	20.00	1	'Sult1c2	5
GO:0005353	fructose_transmembrane_transporter_activity	GO:MF	16.67	1	'Slc2a5	6
GO:0008747	N_acetylneuraminate_lyase_activity	GO:MF	50.00	1	'Npl	2
GO:0016833	oxo_acid_lyase_activity	GO:MF	16.67	1	'Npl	6
GO:0070061	fructose_binding	GO:MF	16.67	1	'Slc2a5	6
GO:0019828	aspartic_type_endopeptidase_inhibitor_activity	GO:MF	10.00	1	'Wfdc2	10

Figure 22. Excerpt from the terms annotations results, generated by Module 3, of Cluster 22 genes analysis with g:Profiler g:GOSt tool, showing enriched pathways across Reactome Pathways (REAC) and Gene Ontology Molecular Function (GO:MF) datasets. Columns include term ID, name, source dataset, ratio (%), number of genes from the cluster (ListCount), genes from the cluster (Genes_in_list) and the total number of annotations from the enriched term (TermCount).

Collectively, the results suggest that Cluster 22 may reflect supportive sugar-processing functions relevant to the glycosylation shifts observed during tissue regeneration.

The markedly higher number of enriched terms returned by g:Profiler g:GOSt compared to PANTHER reflects fundamental methodological differences between the tools in dataset scope, statistical filtering, and term handling. g:Profiler integrates a broader range of annotation sources and favors sensitivity, capturing a wider functional landscape, especially valuable for detecting weak or nuanced biological signals in smaller or functionally sparse gene clusters. In contrast, PANTHER provides more conservative, high-confidence pathway-level associations, emphasizing robust and well-supported annotations. This divergence is illustrated in the comparative enrichment results from Clusters 4, 6, and 22: Cluster 6 showed consistent neuronal pathway signals across both tools, validating functional coherence, while only g:Profiler identified sugar metabolism-related processes in Cluster 22, aligning with the biological premise of altered glycosylation during regeneration. Together, these tools offer complementary perspectives, g:Profiler enabling broad exploratory analysis, and PANTHER reinforcing focused, stringent pathway-level inference, both essential for comprehensive enrichment analysis in systems biology.

3.7.2. Usage case 2

This usage case demonstrates the analytical utility of Module 1 by applying it to a real-world case study focused on spontaneous spinal cord regeneration in mammals. The analysis uses a gene expression matrix comprising 1,145 unique genes measured across 8 samples. These include control (naive) conditions for two species: *Acomys cahirinus* (spiny mouse) and *Mus musculus*. For *Acomys cahirinus* there are four naive (A1, A4, A5, A7) samples; for *Mus musculus* there are four naive (M2, M3, M4, M9) samples.

The data comes from a published study [91] that investigated the molecular basis for the scarless and regenerative spinal cord response in *Acomys cahirinus*, which contrasts with the fibrotic and non-regenerative response seen in *Mus musculus*. The study highlights enhanced glycosylation activity, particularly via the enzyme *B3gnt7*, as a major contributor to axon growth and regeneration in *Acomys cahirinus*.

In this use case the aim is to identify genes that may predispose *Acomys cahirinus* to a pro-regenerative state. These baseline expression differences could help explain

the molecular foundations of spontaneous Central Nervous System (CNS) repair in this species and offer insight into potential therapeutic targets for enhancing regeneration in non-regenerative species.

Following gene list extraction, this application continues with downstream enrichment analysis to identify biological pathways and functional terms associated with the species-specific genes. This includes over-representation analysis (ORA) using Modules 3 and 4 (g:Profiler and PANTHER, respectively), and Gene Set Enrichment Analysis using Modules 5 and 6. Together, these analyses serve the broader goal of the pipeline: to uncover biologically meaningful patterns and pathways that may underlie regenerative capabilities, enabling robust functional interpretation of transcriptomic data.

To achieve this, the pipeline began with Module 1 that was used to identify species-specific gene expression differences under baseline (control) conditions by averaging expression in naive samples and selecting genes where expression in *Acomys cahirinus* exceeds that seen in *Mus musculus*. Next, Module 2 was used to map gene information. The mapped gene list was then analysed using Module 3 with g:Profiler g:GOSt tool, and Module 4 with the PANTHER statistical overrepresentation test.

The main pipeline configuration file, config0, used to run these analyses is shown in Figure 23.

config0

```
1  # General parameters
2  tools="1,2,3,4"
3  species="mus_musculus"
4
```

Figure 23. config0 file setup used to pre-process (modules 1 and 2) and perform Over-Representation analyses (modules 3 and 4). The lines starting with "#" are comments. Line 2 declares the modules to run. In Line 3 the target species name is listed.

In this context, Module 1 is used to extract species-specific gene expression patterns under baseline conditions. This is achieved by computing the average gene expression across naive samples for both species and applying a fold-change filter, to select genes where *Acomys cahirinus* control expression significantly exceeds that of *Mus musculus* (e.g., fold change > 0.95). The configuration file that is used for this purpose, config1, is shown in Figure 24.

config1

```
1  input="genes_expression_samples.txt"
2  gene=1
3
4  avg1="2,3,4,5"      # Acomys control (naive) group samples
5  avg2="10,11,12,13"  # Mus musculus control (naive) group samples
6
7  # Proportion of Acomys naive expression vs. total naive expression (Acomys + Mus)
8  cond1="avg1/(avg1 + avg2)"
9
10 # Retain genes where Acomys contributes >95% of the combined expression
11 expression_min=0.95
12
```

Figure 24. Configuration file, config1, for Module 1, used to extract species-specific genes that show much higher expression in *Acomys cahirinus* than in *Mus musculus*, under baseline conditions. The input file (line 1) contains raw expression values across samples, with gene identifiers indicated in column index 1 (declared in line 2). The calculation of the average expression values for *Acomys cahirinus* (avg1) and *Mus musculus* (avg2) naive samples are defined on lines 4 and 5, where the column indexes to be considered are indicated. Line 8 sets the fold-change condition (cond1) as the proportion of *Acomys cahirinus* expression over the total naive expression (*Acomys cahirinus* + *Mus musculus*). Line 11 sets a minimum expression threshold (expression_min=0.95) to retain genes where *Acomys cahirinus* contributes over 95% of the combined expression, aiming to isolate baseline expression signatures potentially related to the species' regenerative capacity. Comments (lines starting with #) provide context for each parameter.

The custom filter implemented in Module 1 successfully extracted 76 out of 1,145 genes that showed much higher expression in *Acomys cahirinus* control samples than in *Mus musculus* control samples. This selection highlights potential species-specific baseline expression patterns relevant to the regenerative phenotype. The entire process, including average calculations, application of the fold-change condition, and gene filtering, was completed in approximately 40 seconds. This demonstrates not only the module's efficiency and scalability but also how it enables rapid, automated identification of biologically meaningful candidates, a task that would be highly time-consuming and error-prone if done manually, especially across large datasets with multiple samples per condition.

Using Module 2's mapping capabilities, the names and descriptions of these genes gives more biological insight on reasons why *Acomys cahirinus* show a different behavior than in *Mus musculus*. Among the 12 most highly expressed genes (in comparison with *Mus musculus*) in *Acomys cahirinus* under uninjured conditions (naive), several suggest critical roles, present in Figure 25 with their Gene IDs, in maintaining the species' unique regenerative environment, *Cd59b* (Gene ID: 333883), encoding the *CD59b* antigen that protects cells from complement-mediated lysis, suggests a role in immune modulation. *Cct6b* (Gene ID: 12467), a subunit of the chaperonin-containing TCP1 complex involved in protein folding and cytoskeletal organization, also ranks prominently, indicating cellular

maintenance and structural readiness. Noteworthy is the presence of genes like *Gstm2* (Gene ID: 14863), a glutathione S-transferase important in detoxification and oxidative stress defence, and *Rtp4* (Gene ID: 67775), involved in protein trafficking and possibly antiviral responses, which, although less characterized in regeneration, exhibit robust expression, hinting at novel factors potentially involved in tissue maintenance or repair. Immune-related genes such as *Cd68* (Gene ID: 11808), a macrophage marker linked to phagocytosis, and *Lgals3* (Gene ID: 11737), a modulator of inflammation and tissue remodelling, highlight innate immune surveillance and extracellular matrix interactions. The antioxidant transcription factor *Nfe2l2* (Gene ID: 18952), critical for oxidative stress responses, suggests metabolic readiness for injury prevention and repair. Genes such as *Map2* (Gene ID: 17756), essential for microtubule stabilization in neurons, and *Tcl1b5* (Gene ID: 27382), linked to cell survival and proliferation signaling, suggest an active structural plasticity even in steady state. Many genes within this top set, including several with currently unknown or predicted functions (e.g., Gene IDs: 102640710, 97243), provide a valuable resource for uncovering novel molecular pathways that may underpin *Acomys cahirinus* intrinsic regenerative capacity. Overall, this expression profile supports a molecular landscape primed for immune surveillance, structural readiness, and potential regenerative signaling.

	geneID	avg1	avg2	cond1
2	333883	519,3725	8,56	0,9838
3	102640710	477,8375	0	1
4	14863	448,6375	13,31	0,9712
5	67775	246,315	11,065	0,957
6	12467	222,8075	0,4175	0,9981
7	11629	159,2225	0	1
8	97243	145,245	0	1
9	17756	128,915	6,1075	0,9548
10	11808	112,415	0	1
11	18952	107,9375	0	1
12	11737	98,935	0	1
13	27382	96,8675	0	1

Figure 25. Excerpt from the “overexpressed_cond1” file showing the 12 most highly expressed genes (in comparison with *Mus musculus*) in *Acomys cahirinus* under uninjured (naive) conditions, listed in descending order of expression in *Acomys cahirinus*. Row 1 contains the column headers: geneID (column 1), avg1 (average expression across naive *Acomys cahirinus* samples), avg2 (average expression across naive *Mus musculus* samples), and cond1 (fold change between *Acomys cahirinus* and *Mus musculus* expression).

These findings provide early clues, even before formal over-representation analysis, about the extracted gene list that should capture biologically coherent signals relevant to

regeneration and already hints at species-specific programs that may underlie *Acomys cahirinus* unique regenerative traits.

The ORA conducted with PANTHER did not return any statistically significant enriched terms across the available datasets at the applied significance threshold (FDR-corrected *P*-value < 0.05).

In contrast, g:Profiler identified a significant enrichment in several categories within the CORUM database and Gene Ontology Cellular Component (GO:CC) category, showed in Figure 26. Notably, enriched CORUM complexes include multiple sarcoglycan-sarcospan-syntrophin-dystrobrevin complexes (e.g., *CORUM:349*, *CORUM:131*, *CORUM:317*), primarily involving the genes *Sgca* and *Dtna*. The CCT complex (*CORUM:3072*), including *Cct6b*, was also enriched. Additionally, the transcription factor *Creb3l3* appeared in an enriched CORUM complex (*CORUM:7230*). Within GO:CC, the term *intercellular bridge* (*GO:0045171*) was significantly enriched, with contributions from genes such as *Gstm2* and *Phldb1*.

TermID	Name	Source	Ratio(%)	ListCount	GeneIDs_in_list	Genes_in_list	TermCount
CORUM:349	Sarcoglycan_sarcospan_syntrophin_dystrobrevin_complex	CORUM	28.57	2	'13527 '20391	'Dtna 'Sgca	7
CORUM:131	Skeletal_muscle_sarcoglycan_complex_SGC__alpha_beta_gamma_delta	CORUM	25.00	1	'20391	'Sgca	4
CORUM:317	Dystrobrevin_syntrophin_complex__brain_derived	CORUM	25.00	1	'13527	'Dtna	4
CORUM:318	Muscle_derived_dystrobrevin_syntrophin_complex	CORUM	50.00	1	'13527	'Dtna	2
CORUM:329	Skeletal_muscle_sarcoglycan_complex_SGC__alpha_beta_gamma_delta	CORUM	25.00	1	'20391	'Sgca	4
CORUM:332	Skeletal_muscle_sarcoglycan_complex_SGC__alpha_beta_epsilon_gamma	CORUM	25.00	1	'20391	'Sgca	4
CORUM:333	Skeletal_muscle_sarcoglycan_complex_SGC__alpha_beta_gamma_delta	CORUM	25.00	1	'20391	'Sgca	4
CORUM:347	Sarcoglycan_sarcospan_complex_SG_SPN	CORUM	25.00	1	'20391	'Sgca	4
CORUM:7230	Creb3l3_Ppara_Sirt1_mouse	CORUM	33.33	1	'208677	'Creb3l3	3
CORUM:346	Sarcoglycan_sarcospan_dystroglycan_complex	CORUM	20.00	1	'20391	'Sgca	5
CORUM:337	Dystrophin_sarcoglycan_syntrophin_complex__skeletal_muscle	CORUM	16.67	1	'20391	'Sgca	6
CORUM:3072	CCT_complex__chaperonin_containing_TCP1_complex___testis_specific	CORUM	12.50	1	'12467	'Cct6b	8
GO:0045171	intercellular_bridge	GO:CC	3.92	4	'102693 '14863 '329207 '74107	'Phldb1 'Gstm2 'Rbm44 'Cep55	102

Figure 26. Excerpt from the terms annotations results, generated by Module 4, of highly expressed genes (in comparison with *Mus musculus*) in *Acomys cahirinus* control samples, after the analysis with g:Profiler g:GOSt tool. The results show enriched pathways across CORUM and Gene Ontology Cellular Component (GO:CC) datasets. Columns include term ID, name, source dataset, ratio (%), number of genes from the list (ListCount), gene from the list (GeneIDs_in_list and Genes_in_list) and the total number of annotations from the enriched term (TermCount).

While PANTHER, known for its conservative and high-confidence pathway associations, did not identify significant enrichments in this gene set, g:Profiler revealed multiple enriched protein complexes and cellular components. This difference likely reflects g:Profiler greater sensitivity, especially when analysing smaller or functionally less characterized gene clusters. The enrichment of sarcoglycan-sarcospan-syntrophin-dystrobrevin complexes, key to muscle membrane stability and signaling, points to molecular features that may support muscle integrity and regenerative capacity in *Acomys cahirinus*, although its relevance in the context of spinal cord injury remains elusive. Similarly, the identification of the CCT chaperonin complex emphasizes the importance of protein folding and cytoskeletal organization in tissue maintenance and

repair. The significant GO term for intercellular bridges suggests active cellular communication and cytoskeletal remodelling processes. Collectively, these results indicate that *Acomys cahirinus* expresses gene programs geared toward structural stability and dynamic cellular interactions in steady state, potentially underlying its enhanced regenerative abilities compared to *Mus musculus*.

3.7.3. Usage case 3

To investigate pathway-level differences between naive *Acomys cahirinus* and *Mus musculus* samples, GSEA Classic was performed using a full expression matrix, comprising 1,145 unique genes measured across 8 samples. These include control (naive) conditions for the two species. For *Acomys cahirinus* there are four naive (A1, A4, A5, A7) samples; for *Mus musculus* there are four naive (M2, M3, M4, M9) samples. In this application, the naive samples from *Acomys cahirinus* and *Mus musculus* were compared by association to phenotypes, A and M respectively.

The goal is to show how incorporating natural inter-sample variability, rather than using pooled or averaged values or a limited set of selected genes, affects enrichment analysis results. By using the complete expression matrix and defining phenotypes based on the control condition samples, GSEA Classic highlights pathway-level differences that reflect the true biological variability present in the data.

In this application, three modules were used, including Modules 5 and 6 configured in the pipeline configuration file, config0 (Figure 27). Module 5 was used to prepare the GSEA Classic required inputs, namely the expression dataset (GCT) and the phenotype labels file (CLS). Module 6 performed the Gene Set Enrichment Analysis using the Classic approach but also formats and organizes the outputs.

config0

```
1  # General parameters
2  tools="5,6"
3
```

Figure 27. config0 file setup to pre-process (module 5) and perform Gene Set Enrichment Analysis using the Classic approach (module 6). The lines starting with "#" are comments. Line 2 declares the modules to run.

To run Module 5 the respective configuration file, config5 (Figure 28), was set to generate GSEA Classic method inputs, by transforming the full expression matrix (TSV)

into an expression dataset (GCT) and assigning the phenotype label to the naive conditions samples of *Acomys cahirinus* (A) and *Mus musculus* (M).

config5

```
1  method="classic"
2  input="genes_expression_samples.tsv"
3  gene="Gene ID"
4
5  # Total number of samples
6  number_samples=8
7  # Column indexes of samples in the naive conditions
8  samples="2,3,4,5,10,11,12,13"
9
10 # Number of distinct groups (Acomys naive and Mus naive)
11 number_groups=2
12 # Assign phenotype labels to groups and number of samples
13 group_order="A 4 M 4"
14
```

Figure 28. Configuration file, config5, for Module 5 to prepare GSEA Classic inputs using the naive conditions samples of *Acomys cahirinus* and *Mus musculus*. Line 1 defines the GSEA method to be used. Line 2 declares the name of the full expression matrix file. Line 3 sets the name of the column with the Gene Identifiers (Gene IDs in this case). Line 6 defines the total number of samples (8). Line 8 is the column indexes of the samples. Line 11 number the distinct groups/conditions. Line 13 assigns phenotypes to the different groups and respective number of samples of each. Comments (lines starting with #) provide context for the parameter.

The parameters used for running GSEA using Module 6 were specified in the configuration file, gsea_parameters (Figure 29). The **"**M2: curated gene sets**"** from the Molecular Signature Database (MSigDB) is a collection of manually curated gene sets representing biological processes, pathways, and functional categories derived from expert knowledge and published studies. These sets are based on canonical pathway databases (e.g., KEGG, Reactome), literature curation, or experimentally validated signatures. In this analysis, the gene set file used was composed of NCBI (Entrez) Gene IDs specific to *Mus musculus*, ensuring compatibility with the expression dataset and enabling direct mapping without additional annotation files, therefore collapsing was disabled. This choice allows the analysis to focus on established, well-defined pathways and biological processes relevant to mouse models, facilitating interpretation and comparison with prior studies. The permutation selected was gene set (recommended when there are fewer than seven samples per condition), with 1000 permutations, as generally advised.

gsea_parameters

```
1  gmx m2.all.v2025.1.Mm.entrez.gmt
2  permute gene_set
3  nperm   1000
4  collapse    No_Collapse
5
```

Figure 29. Parameters configuration file (gsea_parameters) containing tab-separated key-value pairs used by Module 6 to run GSEA Classic. Line 1 specifies the gene set file used (m2.all.v2025.1.Mm.entrez.gmt). Line 2 sets the permutation type to gene_set, which is recommended for conditions with fewer than seven samples. Line 3 defines the number of permutations (1000). Line 4 disables collapse (No_Collapse), as the expression dataset and gene set file use compatible gene identifiers, eliminating the need for a chip annotation file.

The gsea_parameters file is automatically updated by Module 5, which appends the expression dataset and phenotype labels files to the parameter configuration file using the appropriate key-value combinations. In addition, the default output directory parameter value ("results") is also added. Figure 30 shows the contents of the gsea_parameters_classic file (named according to the GSEA method used), which is a tab-separated key-value file prepared by Module 5 from the initial base configuration.

gsea_parameters_classic

```
1  gmx m2.all.v2025.1.Mm.entrez.gmt
2  nperm   1000
3  permute gene_set
4  collapse    No_Collapse
5  res expression_dataset.gct
6  cls phenotype_labels.cls
7  out results
8
```

Figure 30. Parameters configuration file (gsea_parameters_classic) containing tab-separated key-value pairs used by Module 6 to run GSEA Classic. This file is prepared by Module 5 based on the initial parameter file provided by the user, with the addition of the addition of the expression dataset (res), the phenotype labels file (cls), and the output directory (out), along with initial set parameters.

The GSEA Classic analysis comparing naive *Acomys cahirinus* to naive *Mus musculus* samples identified 41 significantly enriched terms (pathways), all associated with the *Acomys cahirinus* phenotype. These enriched pathways reveal a broad transcriptional landscape characterized by enhanced immune system activity, including both adaptive and innate immune responses, such as Reactome immnune system related pathways (*REACTOME_IMMUNE_SYSTEM*,

REACTOME_ADAPTIVE_IMMUNE_SYSTEM, *REACTOME_INNATE_IMMUNE_SYSTEM* and *REACTOME_NEUTROPHIL_DEGRANULATION*), indicating a heightened baseline immune state in *Acomys cahirinus*. Additionally, several metabolic pathways (*REACTOME_METABOLISM_OF_CARBOHYDRATES_AND_CARBOHYDRATE_DERIVATIVES*, and *REACTOME_METABOLISM_OF_RNA*) are upregulated, reflecting fundamental metabolic differences between the species. Enrichment of pathways linked to tissue remodeling and cell signaling (*REACTOME_PLATELET_ACTIVATION_SIGNALING_AND_AGGREGATION*) and the microglia-related pathway (*WP_TYROBP_CAUSAL_NETWORK_IN_MICROGLIA*), further suggest complex regulation of immune and regenerative processes. Moreover, gene sets associated with cancer-related pathways (*SWEET_LUNG_CANCER_KRAS_UP*) and tumor invasiveness highlight potential differences in cellular proliferation and regeneration. The enrichment of immune checkpoint blockade predictive gene sets (e.g., *ZENG_GU_ICB_CONTROL_METAGENE_25_PRECICTIVE_ICB_RESPONSE*) also points to distinct immunoregulatory networks. These key enriched pathways are shown in Figure 31.

Name	NES	FDR_q-val	EnrichDirection
LIAN_LIPA_TARGETS_6M	2.7251658	0.0	A
LIU_OVARIAN_CANCER_TUMORS_AND_XENOGRAFTS_XDGS_DN	2.2138562	0.0012796277	A
MARKEY_RB1_ACUTE_LOF_DN	2.3441343	5.0507794E-4	A
MARKEY_RB1_CHRONIC_LOF_DN	2.1211276	0.0024379948	A
MARTINEZ_RB1_TARGETS_DN	1.7657393	0.04180515	A
MIKKELSEN_NPC_ICP_WITH_H3K4ME3	1.8466175	0.02615561	A
NABA_ECM_REGULATORS	1.8035657	0.032367513	A
NABA_MATRISOME_ASSOCIATED	1.8807704	0.020395027	A
NADLER_OBESITY_UP	2.289902	6.9279695E-4	A
QI_PLASMACYTOMA_UP	1.831838	0.028472057	A
REACTOME_ADAPTIVE_IMMUNE_SYSTEM	2.0746398	0.0039002239	A
REACTOME_GLYCOSAMINOGLYCAN_METABOLISM	1.7524422	0.04389442	A
REACTOME_HEMOSTASIS	1.9851447	0.0104353065	A
REACTOME_IMMUNE_SYSTEM	2.1282945	0.0023531222	A
REACTOME_INNATE_IMMUNE_SYSTEM	2.1667306	0.0013533052	A
REACTOME_METABOLISM_OF_CARBOHYDRATES_AND_CARBOHYDRATE_DERIVATIVES	1.7372493	0.048350085	A
REACTOME_METABOLISM_OF_RNA	1.9585456	0.011849385	A
REACTOME_NEUTROPHIL_DEGRANULATION	2.1915133	0.0012788507	A
REACTOME_PLATELET_ACTIVATION_SIGNALING_AND_AGGREGATION	1.810926	0.031846922	A
SANSOM_APC_TARGETS	1.7575743	0.04384498	A
SCHAEFFER_PROSTATE_DEVELOPMENT_48HR_DN	2.0019362	0.009161303	A
SWEET_LUNG_CANCER_KRAS_UP	2.5462992	0.0	A
WANG_TUMOR_INVASIVENESS_UP	1.8837146	0.0206469	A
WP_TYROBP_CAUSAL_NETWORK_IN_MICROGLIA	2.1726837	0.0013575888	A
ZENG_GU_ICB_CONTROL_METAGENE_25_PRECICTIVE_ICB_RESPONSE	2.4559326	1.5645161E-4	A
ZENG_GU_ICB_CONTROL_METAGENE_8_PRECICTIVE_ICB_RESPONSE	1.9597672	0.012172409	A
ZHONG_SECRETOME_OF_LUNG_CANCER_AND_FIBROBLAST	2.2747195	9.530394E-4	A

Figure 31. Excerpt from the short results file generated by Module 6, showing some of the enriched pathways. Columns include Name (pathways data source identifier and name), NES (normalized enrichment score), FDR q-val (adjusted significance), and EnrichDirection (associative phenotype).

The examination of the genes underlying the enriched pathways reveals consistent involvement of key regulators of myeloid lineage differentiation, immune activation, and microglial function, as demonstrated in Figure 32. Several genes occur across multiple pathways, including *Cd14* (Gene ID:12260), a co-receptor for LPS detection and monocyte activation; *Cd68* (Gene ID:12262), a macrophage marker linked to phagocytosis; and *Csf1r* (Gene ID:12514), a receptor essential for macrophage development and survival. *Itgam* (Gene ID:13040) and *Itgb2* (Gene ID:13030) encode integrin subunits critical for leukocyte adhesion and migration. *Ly86* (Gene ID:14130) modulates TLR4 signaling, amplifying innate immune responses. The microglial receptor complex genes *Trem2* (Gene ID:17476) and *Tyrobp* (Gene ID:22177) are vital for phagocytic activity and anti-inflammatory signaling. *Fn1* (Gene ID:12759) and *Mmp9* (Gene ID:14824) play important roles in extracellular matrix remodeling during wound healing and inflammation, while *Lgals3* (Gene ID:64138) contributes to macrophage activation and fibrosis. These genes are enriched in pathways such as myeloid cell development, graft-versus-host disease, platelet activation, and microglial signaling, reflecting an integrated axis of innate immune readiness, cellular trafficking, and tissue remodeling.

Name	EnrichDirection	Ratio(%)	ListCount	Genes_in_list	TermCount
LIAN_LIPA_TARGETS_6M	A	22.7273	20	11810'12259'12260'12262'12267'12514'12523'13030'12140'14130'14131'15368	88
LIU_OVARIAN_CANCER_TUMORS_AND_XENOGRAFTS_XDGS_DN	A	16.1441	242	232345'217262'18670'13349'11517'11568'11629'11689'11690'16790'69538'117	1499
MARKEY_RB1_ACUTE_LOF_DN	A	17.7419	44	13733'11746'11810'11815'12122'12260'12262'12267'12273'12363'20305'14049	248
MARKEY_RB1_CHRONIC_LOF_DN	A	26.4000	33	11816'12259'12260'12262'12267'12475'140497'12508'13040'13800'12140'1081	125
MARTINEZ_RB1_TARGETS_DN	A	8.1395	49	13349'52123'26358'11690'11816'75329'68010'12759'223453'13386'112407'141	602
MIKKELSEN_NPC_ICP_WITH_H3K4ME3	A	9.8291	46	70419'67392'232345'217262'11600'11761'80287'26362'12505'12580'229445'32	468
NABA_ECM_REGULATORS	A	8.5809	26	232345'12153'235505'13030'13032'13033'13039'229445'13040'64138'112407'2	303
NABA_MATRISOME_ASSOCIATED	A	6.7470	56	232345'11600'12306'11745'11746'12153'12259'12260'12262'23829'81799'1240	830
NADLER_OBESITY_UP	A	38.3333	23	11690'12111'12260'12508'12514'13030'13033'13039'13040'64138'14131'18301	60
QI_PLASMACYTOMA_UP	A	12.4514	32	11690'11857'26362'12363'12483'12508'12759'12798'14190'14676'70110'15979	257
REACTOME_ADAPTIVE_IMMUNE_SYSTEM	A	7.0013	53	142688'17060'12229'12483'217303'140497'52685'12524'12757'13030'13032'13	757
REACTOME_GLYCOSAMINOGLYCAN_METABOLISM	A	13.8211	17	227327'14595'12111'12505'72136'212898'14264'50917'74577'75612'110006'15	123
REACTOME_HEMOSTASIS	A	10.7203	64	232345'227290'11600'12306'54519'53313'11941'12389'235505'12505'12512'12	597
REACTOME_IMMUNE_SYSTEM	A	9.3129	164	227290'11593'11689'16790'12306'11821'142688'11980'14595'11796'17060'122	1761
REACTOME_INNATE_IMMUNE_SYSTEM	A	11.3636	115	227290'11593'11689'16790'12306'11821'11980'14595'11796'12229'12259'1226	1012
REACTOME_METABOLISM_OF_CARBOHYDRATES_AND_CARBOHYDRATE_DERIVATIVES	A	9.9237	26	11668'227327'14595'12111'12505'72136'232449'212898'14264'50917'74577'75	262
REACTOME_METABOLISM_OF_RNA	A	3.7099	22	11737'11810'80287'108062'14109'14853'15507'230082'73826'17749'238831'18	593
REACTOME_NEUTROPHIL_DEGRANULATION	A	14.4195	77	11593'11689'16790'12306'11821'11980'14595'12267'12273'12475'1404971'250	534
REACTOME_PLATELET_ACTIVATION_SIGNALING_AND_AGGREGATION	A	14.5098	37	232345'227290'54519'235505'12512'67072'12759'320127'104418'13601'13645'	255
SANSOM_APC_TARGETS	A	7.6923	17	77087'11829'12006'12444'12505'68977'16010'218121'192654'71592'71729'677	221
SCHAEFFER_PROSTATE_DEVELOPMENT_48HR_DN	A	10.1415	43	74211'319387'140577'16790'69538'232201'17060'12259'12262'12291'14049'7	424
SWEET_LUNG_CANCER_KRAS_UP	A	13.8277	69	11746'11847'26362'12260'12262'12319'20305'12475'12505'12512'12514'12606	499
WANG_TUMOR_INVASIVENESS_UP	A	7.1795	28	12306'12260'12325'12512'12757'12798'64138'13132'22350'14109'27055'54393	390
WP_TYROBP_CAUSAL_NETWORK_IN_MICROGLIA	A	31.0345	18	54519'12262'12523'330217'238875'15284'16154'16164'16409'218454'17095'67	58
ZENG_GU_ICB_CONTROL_METAGENE_25_PRECICTIVE_ICB_RESPONSE	A	21.5000	43	11629'11690'16790'12259'12260'12262'12363'217303'140497'12524'13030'723	200
ZENG_GU_ICB_CONTROL_METAGENE_8_PRECICTIVE_ICB_RESPONSE	A	10.5000	21	76117'246746'70086'108670'14190'171285'212032'16154'16408'16768'140795'	200
ZHONG_SECRETOME_OF_LUNG_CANCER_AND_FIBROBLAST	A	13.4328	18	11746'12111'12759'13030'13033'64138'227929'13601'18606'14860'14870'1903	134

Figure 32. Excerpt of enriched term annotations, generated by Module 6, of identified expression matrix genes. Columns include Name (pathways data source identifier and name), EnrichDirection (associative phenotype), Ratio (%), ListCount (number of genes from the matrix), Genes_in_list (genes from the matrix) and TermCount (the total number of annotations from the enriched term).

In the context of the case study comparing *Acomys cahirinus* naive samples to *Mus musculus* naive samples, these enrichment patterns suggest a fundamentally different immunological baseline between species. In the context of the case study comparing *Acomys cahirinus* naive samples to *Mus musculus* naive samples, these enrichment patterns suggest a fundamentally different immunological baseline between species.

The prominence of immune-related pathways in *Acomys cahirinus* implies a more activated or primed innate immune and inflammatory state under basal conditions. In contrast, *Mus musculus* appears to maintain a more quiescent immune profile, with reduced representation of these genes and pathways.

From an analytical perspective, GSEA offers a unique approach by leveraging the entire ranked gene expression matrix across all samples. This methodology captures coordinated changes in gene expression, accounting for the variability within the dataset and enabling the detection of subtle but biologically meaningful shifts in pathway activity. Therefore, GSEA provides valuable insight into the global transcriptional landscape and pathway-level differences that underlie distinct biological states or conditions across various studies.

4. Conclusion

The auto-Enrich pipeline represents a comprehensive, standardized, and reproducible framework addressing critical challenges in pathway enrichment analysis. By integrating multiple well-established tools (g:Profiler, PANTHER, and GSEA) with complementary strengths, the pipeline ensures robust statistical methods, broad species coverage, and consistent updates. The modular architecture, harmonized outputs, Docker-based deployment, and uniform application of statistical corrections further improve reproducibility, usability, and interpretability. These advances collectively enable researchers to extract more reliable and meaningful biological insights from high-throughput data with reduced manual intervention and methodological inconsistencies.

The auto-Enrich pipeline stands out for its modular design, built around seven core modules covering pre-processing, enrichment analysis, and post-processing. This architecture lets researchers customize workflows easily, from generating differentially expressed gene lists to preparing GSEA input files (Classic and Preranked), with simple configuration changes. It delivers results quickly, supports multiple queries and comparisons simultaneously, and takes advantage of the different databases and gene-term annotation resources. Results filtering and standardized, clear outputs further boost usability, helping researchers quickly identify genes driving specific enrichments and obtain focused, biologically relevant insights. With Docker-based deployment and full traceability of raw outputs, auto-Enrich is a powerful, reproducible, and efficient solution for pathway enrichment analysis.

By reducing barriers to performing robust enrichment analyses, auto-Enrich accelerates discovery, promote reproducibility, and foster broader adoption of rigorous bioinformatics practices across the scientific community, ultimately advancing our understanding of complex biological systems.

The auto-Enrich pipeline has yet to reach its full potential. Several future enhancements could significantly expand its capabilities, including the integration of additional enrichment analysis tools, advanced visualization modules like interactive Venn diagrams, UpSet plots, and pathway network overlays, as well as standardized benchmarking metrics and statistical concordance analyses. These additions would help users better interpret overlaps and detect tool-specific biases. Additionally, developing a graphical user interface (GUI) would streamline data input and analysis execution, making the pipeline more accessible to researchers without programming expertise. It

would also enhance reproducibility by guiding parameter selection and offering intuitive dashboards for exploring results.

5. References

1. Chicco, D. & Jurman, G. A brief survey of tools for genomic regions enrichment analysis. *Front. Bioinform.* **2**, 968327 (2022).
2. Huang, D. W., Sherman, B. T. & Lempicki, R. A. Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Research* **37**, 1–13 (2009).
3. Marco-Ramell, A. *et al.* Evaluation and comparison of bioinformatic tools for the enrichment analysis of metabolomics data. *BMC Bioinformatics* **19**, 1 (2018).
4. Subramanian, A. *et al.* Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences* **102**, 15545–15550 (2005).
5. Zhao, K. & Rhee, S. Y. Interpreting omics data with pathway enrichment analysis. *Trends in Genetics* **39**, 308–319 (2023).
6. Merico, D., Isserlin, R., Stueker, O., Emili, A. & Bader, G. D. Enrichment Map: A Network-Based Method for Gene-Set Enrichment Visualization and Interpretation. *PLoS ONE* **5**, e13984 (2010).
7. Nam, D. & Kim, S.-Y. Gene-set approach for expression pattern analysis. *Briefings in Bioinformatics* **9**, 189–197 (2008).
8. García-Campos, M. A., Espinal-Enríquez, J. & Hernández-Lemus, E. Pathway Analysis: State of the Art. *Front. Physiol.* **6**, (2015).
9. Wijesooriya, K., Jadaan, S. A., Perera, K. L., Kaur, T. & Ziemann, M. Urgent need for consistent standards in functional enrichment analysis. *PLoS Comput Biol* **18**, e1009935 (2022).
10. Consortium, T. G. O. Creating the Gene Ontology Resource: Design and Implementation. *Genome Res.* **11**, 1425–1433 (2001).
11. Ashburner, M. *et al.* Gene Ontology: tool for the unification of biology. *Nat Genet* **25**, 25–29 (2000).

12. Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research* **32**, 258D – 261 (2004).
13. The Gene Ontology Consortium. The Gene Ontology in 2010: extensions and refinements. *Nucleic Acids Research* **38**, D331–D335 (2010).
14. The Gene Ontology Consortium. Gene Ontology Annotations and Resources. *Nucleic Acids Research* **41**, D530–D535 (2012).
15. The Gene Ontology Consortium. The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Research* **47**, D330–D338 (2019).
16. The Gene Ontology Consortium *et al.* The Gene Ontology knowledgebase in 2023. *GENETICS* **224**, iyad031 (2023).
17. Li, L., Cai, S., Liu, S., Feng, H. & Zhang, J. Bioinformatics analysis to screen the key prognostic genes in ovarian cancer. *J Ovarian Res* **10**, 27 (2017).
18. Karp, P. D. The EcoCyc and MetaCyc databases. *Nucleic Acids Research* **28**, 56–59 (2000).
19. Wittig, U. Analysis and comparison of metabolic pathway databases. *Briefings in Bioinformatics* **2**, 126–142 (2001).
20. Kanehisa, M. The KEGG databases at GenomeNet. *Nucleic Acids Research* **30**, 42–46 (2002).
21. Kanehisa, M. & Goto, S. KEGG: Kyoto Encyclopedia of Genes and Genomes.
22. Joshi-Tope, G. Reactome: a knowledgebase of biological pathways. *Nucleic Acids Research* **33**, D428–D432 (2004).
23. Mubeen, S., Tom Kodamullil, A., Hofmann-Apitius, M. & Domingo-Fernández, D. On the influence of several factors on pathway enrichment analysis. *Briefings in Bioinformatics* **23**, bbac143 (2022).
24. Mi, H. & Thomas, P. PANTHER Pathway: An Ontology-Based Pathway Database Coupled with Data Analysis Tools. in *Protein Networks and Pathway Analysis* (eds. Nikolsky, Y. & Bryant, J.) vol. 563 123–140 (Humana Press, Totowa, NJ, 2009).

25. Slenter, D. N. *et al.* WikiPathways: a multifaceted pathway database bridging metabolomics to other omics research. *Nucleic Acids Research* **46**, D661–D667 (2018).
26. Draghici, S. Onto-Tools, the toolkit of the modern biologist: Onto-Express, Onto-Compare, Onto-Design and Onto-Translate. *Nucleic Acids Research* **31**, 3775–3781 (2003).
27. Huang, D. W. *et al.* The DAVID Gene Functional Classification Tool: a novel biological module-centric algorithm to functionally analyze large gene lists. *Genome Biol* **8**, R183 (2007).
28. Identifying biological themes within lists of genes with EASE.
29. Castillo-Davis, C. I. & Hartl, D. L. GeneMerge—post-genomic analysis, datamining, and hypothesis testing. *Bioinformatics* **19**, 891–892 (2003).
30. Mootha, V. K. *et al.* PGC-1α-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. *Nat Genet* **34**, 267–273 (2003).
31. Abatangelo, L. *et al.* Comparative study of gene set enrichment methods. *BMC Bioinformatics* **10**, 275 (2009).
32. Newton, M. A., Quintana, F. A., Den Boon, J. A., Sengupta, S. & Ahlquist, P. Random-set methods identify distinct aspects of the enrichment signal in gene-set analysis. *Ann. Appl. Stat.* **1**, (2007).
33. Tarca, A. L. *et al.* A novel signaling pathway impact analysis. *Bioinformatics* **25**, 75–82 (2009).
34. Mitrea, C. *et al.* Methods and approaches in the topology-based analysis of biological pathways. *Front. Physiol.* **4**, (2013).
35. Nguyen, T.-M., Shafi, A., Nguyen, T. & Draghici, S. Identifying significantly impacted pathways: a comprehensive review and assessment. *Genome Biol* **20**, 203 (2019).
36. Ma, J., Shojaie, A. & Michailidis, G. A comparative study of topology-based pathway enrichment analysis methods. *BMC Bioinformatics* **20**, 546 (2019).

37. Khatri, P., Sirota, M. & Butte, A. J. Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges. *PLoS Comput Biol* **8**, e1002375 (2012).
38. Gao, X., Starmer, J. & Martin, E. R. A multiple testing correction method for genetic association studies using correlated single nucleotide polymorphisms. *Genetic Epidemiology* **32**, 361–369 (2008).
39. Geistlinger, L. *et al.* Toward a gold standard for benchmarking gene set enrichment analysis. *Briefings in Bioinformatics* **22**, 545–556 (2021).
40. Machado, C. M., Freitas, A. T. & Couto, F. M. Enrichment analysis applied to disease prognosis. *J Biomed Sem* **4**, 21 (2013).
41. Wijesooriya, K., Jadaan, S. A., Perera, K. L., Kaur, T. & Ziemann, M. Guidelines for reliable and reproducible functional enrichment analysis. Preprint at https://doi.org/10.1101/2021.09.06.459114 (2021).
42. Reimand, J. *et al.* Pathway enrichment analysis and visualization of omics data using g:Profiler, GSEA, Cytoscape and EnrichmentMap. *Nat Protoc* **14**, 482–517 (2019).
43. Karp, P. D., Midford, P. E., Caspi, R. & Khodursky, A. Pathway size matters: the influence of pathway granularity on over-representation (enrichment analysis) statistics. *BMC Genomics* **22**, 191 (2021).
44. Buzzao, D., Castresana-Aguirre, M., Guala, D. & Sonnhammer, E. L. L. Benchmarking enrichment analysis methods with the disease pathway network.
45. Wieder, C. *et al.* Pathway analysis in metabolomics: Recommendations for the use of over-representation analysis. *PLoS Comput Biol* **17**, e1009105 (2021).
46. Irizarry, R. A., Chi Wang, Yun Zhou & Speed, T. P. Gene set enrichment analysis made simple. *Stat Methods Med Res* **18**, 565–575 (2009).
47. Kanehisa, M., Furumichi, M., Sato, Y., Matsuura, Y. & Ishiguro-Watanabe, M. KEGG: biological systems database as a model of the real world. *Nucleic Acids Research* **53**, D672–D677 (2025).

48. Milacic, M. *et al.* The Reactome Pathway Knowledgebase 2024. *Nucleic Acids Research* **52**, D672–D678 (2024).
49. Karp, P. D. *et al.* The BioCyc collection of microbial genomes and metabolic pathways. *Briefings in Bioinformatics* **20**, 1085–1093 (2019).
50. Piñero, J. *et al.* The DisGeNET knowledge platform for disease genomics: 2019 update. *Nucleic Acids Research* gkz1021 (2019) doi:10.1093/nar/gkz1021.
51. Agrawal, A. *et al.* WikiPathways 2024: next generation pathway database. *Nucleic Acids Research* **52**, D679–D689 (2024).
52. Tryputsen, V., Cabrera, J., De Bondt, A. & Amaratunga, D. Using Fisher's Method to Identify Enriched Gene Sets. *Statistics in Biopharmaceutical Research* **6**, 154–162 (2014).
53. Liberzon, A. *et al.* Molecular Signatures Database (MSigDB) v2025.1.Hs. *Broad Institute* https://www.gsea-msigdb.org/gsea/msigdb (2025).
54. Andrade, C. Multiple Testing and Protection Against a Type 1 (False Positive) Error Using the Bonferroni and Hochberg Corrections. *Indian Journal of Psychological Medicine* **41**, 99–100 (2019).
55. García-Pérez, M. A. Use and misuse of corrections for multiple testing. *Methods in Psychology* **8**, 100120 (2023).
56. Hollestein, L. M. *et al.* MULTIPLE ways to correct for MULTIPLE comparisons in MULTIPLE types of studies. *Br J Dermatol* **185**, 1081–1083 (2021).
57. Liu, G. & Shiraito, Y. Multiple Hypothesis Testing in Conjoint Analysis. *Polit. Anal.* **31**, 380–395 (2023).
58. Korthauer, K. *et al.* A practical guide to methods controlling false discoveries in computational biology. *Genome Biol* **20**, 118 (2019).
59. Ziemann, M., Schroeter, B. & Bora, A. Two subtle problems with overrepresentation analysis. *Bioinformatics Advances* **4**, vbae159 (2024).

60. Xie, C., Jauhari, S. & Mora, A. Popularity and performance of bioinformatics software: the case of gene set analysis. *BMC Bioinformatics* **22**, 191 (2021).
61. Mathur, R., Rotroff, D., Ma, J., Shojaie, A. & Motsinger-Reif, A. Gene set analysis methods: a systematic comparison. *BioData Mining* **11**, 8 (2018).
62. Hahne, F., Huber, W., Gentleman, R. & Falcon, S. *Bioconductor Case Studies*. (Springer New York, New York, NY, 2008). doi:10.1007/978-0-387-77240-0.
63. Chicco, D. & Agapito, G. Nine quick tips for pathway enrichment analysis. *PLoS Comput Biol* **18**, e1010348 (2022).
64. Tamayo, P., Steinhardt, G., Liberzon, A. & Mesirov, J. P. The limitations of simple gene set enrichment analysis assuming gene independence. *Stat Methods Med Res* **25**, 472–487 (2016).
65. Roder, J., Linstid, B. & Oliveira, C. Improving the power of gene set enrichment analyses. *BMC Bioinformatics* **20**, 257 (2019).
66. Korotkevich, G. *et al.* Fast gene set enrichment analysis. Preprint at https://doi.org/10.1101/060012 (2016).
67. Wang, Y. *et al.* Comparing Bayesian-Based Reconstruction Strategies in Topology-Based Pathway Enrichment Analysis. *Biomolecules* **12**, 906 (2022).
68. Yang, Q. *et al.* Pathway enrichment analysis approach based on topological structure and updated annotation of pathway. *Briefings in Bioinformatics* **20**, 168–177 (2019).
69. Sigmund, O. & Maute, K. Topology optimization approaches: A comparative review. *Struct Multidisc Optim* **48**, 1031–1055 (2013).
70. Bayerlová, M. *et al.* Comparative study on gene set and pathway topology-based enrichment methods. *BMC Bioinformatics* **16**, 334 (2015).
71. Tipney, H. & Hunter, L. An introduction to effective use of enrichment analysis software. *Hum Genomics* **4**, 202 (2010).

72. Maleki, F., Ovens, K., Hogan, D. J. & Kusalik, A. J. Gene Set Analysis: Challenges, Opportunities, and Future Research. *Front. Genet.* **11**, 654 (2020).
73. Djaffardjy, M. *et al.* Developing and reusing bioinformatics data analysis pipelines using scientific workflow systems. *Computational and Structural Biotechnology Journal* **21**, 2075–2085 (2023).
74. Cohen-Boulakia, S. *et al.* Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. *Future Generation Computer Systems* **75**, 284–298 (2017).
75. Boettiger, C. An introduction to Docker for reproducible research, with examples from the R environment. *SIGOPS Oper. Syst. Rev.* **49**, 71–79 (2015).
76. Ahmadi, M. An Introduction to Docker and Analysis of its Performance. (2017).
77. Laitos, T. ADVANTAGES OF DOCKER.
78. Kwon, C., Kim, J. & Ahn, J. DockerBIO: web application for efficient use of bioinformatics Docker images. *PeerJ* **6**, e5954 (2018).
79. Di Tommaso, P. *et al.* Nextflow enables reproducible computational workflows. *Nature Biotechnology* **35**, 316–319 (2017).
80. Mölder, F. *et al.* Sustainable data analysis with Snakemake. *F1000Res* **10**, 33 (2021).
81. Landau, W. The targets R package: a dynamic Make-like function-oriented pipeline toolkit for reproducibility and high-performance computing. *JOSS* **6**, 2959 (2021).
82. Wu, T. *et al.* clusterProfiler 4.0: A universal enrichment tool for interpreting omics data. *The Innovation* **2**, 100141 (2021).
83. Zhou, Y. *et al.* Metascape provides a biologist-oriented resource for the analysis of systems-level datasets. *Nat Commun* **10**, 1523 (2019).
84. Kuleshov, M. V. *et al.* Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. *Nucleic Acids Res* **44**, W90–W97 (2016).

85. Thomas, P. D. *et al.* PANTHER: Making genome-scale phylogenetics accessible to all. *Protein Science* **31**, 8–22 (2022).
86. Young, M. D., Wakefield, M. J., Smyth, G. K. & Oshlack, A. Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome Biol* **11**, R14 (2010).
87. Liao, Y., Wang, J., Jaehnig, E. J., Shi, Z. & Zhang, B. WebGestalt 2019: gene set analysis toolkit with revamped UIs and APIs. *Nucleic Acids Research* **47**, W199–W205 (2019).
88. Jr, G. D. *et al.* DAVID: Database for Annotation, Visualization, and Integrated Discovery.
89. Kolberg, L. *et al.* g:Profiler—interoperable web service for functional enrichment analysis and gene identifier mapping (2023 update). *Nucleic Acids Research* **51**, W207–W212 (2023).
90. Bu, D. *et al.* KOBAS-i: intelligent prioritization and exploratory visualization of biological functions for gene enrichment analysis. *Nucleic Acids Research* **49**, W317–W325 (2021).
91. Nogueira-Rodrigues, J. *et al.* Rewired glycosylation activity promotes scarless regeneration and functional recovery in spiny mice after complete spinal cord transection. *Developmental Cell* **57**, 440-450.e7 (2022).