DELT-Hit: An end-to-end computational framework for DNA-encoded chemical library analysis

Authors

Adriano Martinelli+*, Alice Lessing+, Gary Hoppeler, Andreas Gloger, Louise Plais, Jörg Scheuermann*

Abstract

DNA-encoded chemical libraries (DELs) have emerged as a transformative technology in drug discovery, enabling the simultaneous screening of millions to billions of individually DNA-tagged small molecules and their identification after PCR amplification and high-throughput DNA sequencing. However, the computational analysis of DEL screening data is a challenge, requiring sophisticated integration of genomics, cheminformatics, and statistical analysis workflows for which currently no general, open-source software solution is available. This protocol presents DELT-Hit (DNA-Encoded Library Technology Hit Identification), a comprehensive open-source command line tool which allows computational and experimental researchers to analyze DEL data through an intuitive command-line interface. DELT-Hit offers a pipeline that converts raw FASTQ reads into machine learning-ready chemical information by five interconnected modules: (1) adaptive sequence demultiplexing using optimized algorithms with error correction and flexible barcode handling, (2) automated chemical structure reconstruction from building block libraries using reaction SMARTS templates with support for both single and dual display DEL architectures, (3) comprehensive molecular property calculation and descriptor generation using established cheminformatics libraries, (4) statistical analysis and hit ranking with multiple normalization strategies adapted from proven RNA-seq methodologies, and (5) integrated quality control and visualization tools specifically designed for DEL data interpretation. DELT-Hit addresses a critical computational gap identified in the DEL field and provides the standardization necessary for reproducible analysis across research groups. The protocol is accompanied by comprehensive documentation and a tutorial dataset, allowing for a consistent implementation across the rapidly growing DEL community.

Key Points

- **Scalability and performance**: DELT-Hit is designed to handle the computational demands of modern pharmaceutical DEL campaigns, efficiently processing libraries containing millions of compounds while maintaining a user-friendly interface

- **Comprehensive DEL architecture support**: The framework provides native support for both single and dual display DEL architectures, addressing the full spectrum of current library designs used in industry and academia

- **Validated algorithms**: Integrates proven bioinformatics tools (Cutadapt for sequence processing, edgeR for statistical analysis) with specialized DEL-specific optimizations, error handling, and quality control metrics
- **Flexible and robust design**: Modular architecture accommodates diverse library formats, custom reaction templates, and building block definitions
- **Machine learning ecosystem integration**: Generates standardized, analysis-ready datasets compatible with downstream machine learning workflows for advanced hit prediction, structure-activity relationship analysis, and virtual screening applications

Technical Overview

DELT-Hit is implemented as a Python package organized into five core modules:

- **init**: Project initialization and configuration management with Excel template support
- **demultiplex**: Sequence processing and demultiplexing with adaptive error correction
    - `qc`: Quality control plot generation and statistical summaries
    - `report`: Comprehensive reporting with sequence mapping statistics
- **library**: Chemical structure reconstruction and molecular property calculation
    - `enumerate`: SMILES construction from reaction steps and building blocks
    - `properties`: Molecular descriptor computation and distribution visualization
    - `represent`: Chemical representation generation for downstream machine learning
- **dashboard**: Interactive data exploration and real-time visualization interface
- **analyse**: Statistical analysis and hit ranking with multiple enrichment methods

Introduction

DNA-encoded chemical libraries (DELs) have revolutionized modern small-molecule drug discovery by enabling the synthesis and screening of chemical spaces that would be impractical to explore using traditional high-throughput screening approaches. In analogy to biological display technologies, DEL features a direct link between a library member's phenotype (i.e., the small molecule) and genotype (i.e, the encoding DNA oligonucleotide) which allows for the simultaneous and inexpensive screening of libraries of very large sizes (millions to billions) against chosen biological targets, thus greatly facilitating hit discovery. These affinity-based selections on targets can be automatized and have been reported in a previous Nature Protocol. At the last step of the affinity-based selection process, the DNA tags of the remaining/binding compounds need to be PCR amplified and deep-sequenced

by high-throughput DNA sequencing, to reveal the frequency (code counts) of each selected library member.

As outlined in comprehensive review articles on DEL technology (e.g., Nature Reviews Methods Primers (Satz et al., 2022), DELs are now employed by essentially all large pharmaceutical companies, and also by a multitude of academic laboratories worldwide.

Various types of DEL architectures have been developed, which can be classified as single-display and dual-display libraries, depending on the pharmacophore being displayed by just one linked DNA fragment, or jointly displayed by two strands of stably or dynamically pairing oligonucleotides. While standard DELs are usually encoded by a double stranded "DNA headpiece", other setups feature single-stranded DNA or Y-shaped encoding systems. However, regardless of the encoding system used, after selection and PCR amplification, all displays eventually lead to double-stranded amplicons which are DNA sequenced and analyzed.

Considering the large volume of data generated, the computational analysis of DEL screening data is a critical factor, requiring sophisticated integration of genomics, cheminformatics, and statistical analysis workflows that are currently not publicly available. DEL screening data presents unique challenges that require specialized approaches distinct from conventional genomics workflows: accurate demultiplexing of complex DNA barcode combinations from sequencing reads, reconstruction of chemical structures from building block combinations defined by reaction schemes, statistical analysis of enrichment patterns across multiple selection conditions, and integration with cheminformatics workflows for hit optimization and structure-activity relationship analysis.

While individual steps of a typical DEL analysis workflow can be completed by adapting tools originally developed for RNA-seq, these tools lack the flexibility to accommodate diverse library architectures and experimental designs. Most existing approaches do not integrate well with standard bioinformatics pipelines or provide adequate quality control mechanisms for systematic troubleshooting and validation.

Development of the protocol

As a research lab that is practicing DEL technology in all aspects (i.e., design, synthesis, target-based screening), we designed DELT-Hit to address the scale and complexity of modern DEL campaigns, supporting libraries with millions of compounds while maintaining computational efficiency and user friendliness. Its modular architecture enables workflow customization while preserving reproducibility through configuration files and standardized outputs. The framework combines established bioinformatics methods with DEL-specific optimization, provides flexible configuration for diverse library designs and experimental protocols, incorporates comprehensive quality controls at each analysis stage, and remains accessible to users with varying computational backgrounds through intuitive command-line interfaces and extensive documentation.

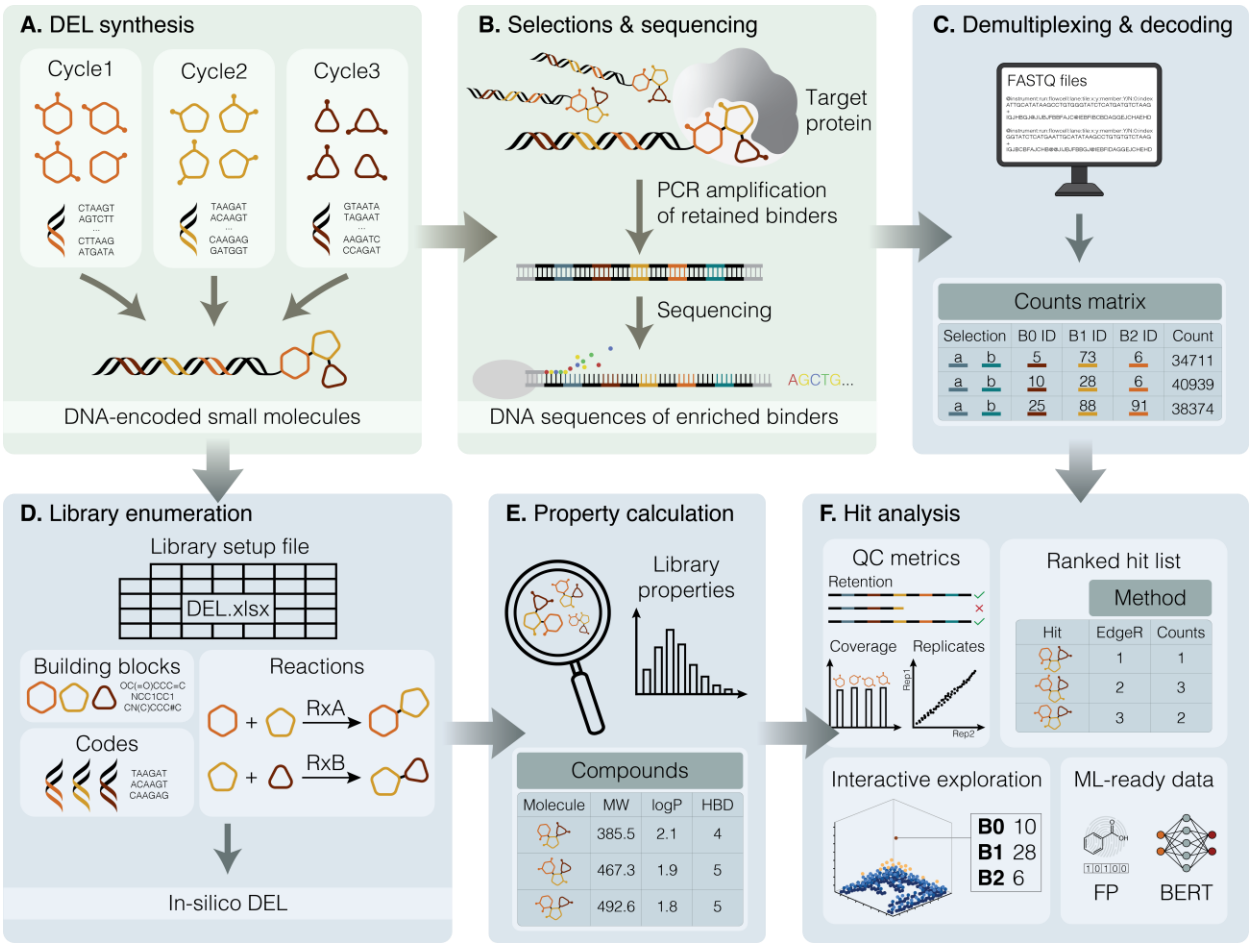


Figure 1 DELT-Hit is organized in several modules (blue rectangles) which integrate seamlessly with the wet-lab DEL workflow (green rectangles). The synthesis of the DEL **(A.)** from selected building blocks is done over iterative cycles of building block addition and encoding. DELs are then interrogated in affinity-based selection experiments **(B.)** against relevant biological targets. Whilst non-binding library members are washed away, DNA-encoded small molecules bound to the target can be selectively eluted. The coding information of selection eluates is then amplified by PCR and sequenced through next-generation sequencing. To reveal the identities of the selected compounds, DELT-Hit uses cutadapt to demultiplex the obtained sequencing data **(C.)**, allowing for user-specified error rates. The obtained counts matrix tabulates sequence counts for all combinations of codes and enables resolution between different selection experiments through distinct selection codes. In parallel, in silico enumeration of library compounds **(D.)** can be performed using SMILES to represent building block structures and customizable reaction SMARTS to define the coupling steps. Properties and molecular descriptors for the compounds comprised within the in silico libraries can be computed **(E.)** to assess drug-likeness and overall library properties. Importantly, DELT-Hit can already be used to guide library planning and building block selection through its library enumeration **(D.)** and property calculation **(E.)** modules by generating prospective libraries in silico and analysing their chemical properties. Hit analysis by DELT-Hit **(F.)** integrates the sequencing data within the counts matrix and the molecular information generated through library enumeration and property calculation. DELT-Hit also enables an assessment of data quality, through quality control metrics giving insight into retention of reads during demultiplexing, barcode coverage and reproducibility between replicates. To evaluate hits, library members can be ranked according to their enrichment, using several ranking methods such as edgeR and counts and the dashboard provides an intuitive visualisation of selection fingerprints and enables their interactive

exploration. Finally, DELT-Hit provides the obtained enrichment data and representations of library compounds as Morgan fingerprints and BERT embeddings in a format ready for downstream machine learning applications.

The framework integrates multiple specialized modules, including sequence demultiplexing via adapted Cutadapt workflows; chemical compound representation as SMILES and chemical reaction step definitions as reaction SMARTS implemented in RDKit; statistical analysis with edgeR; comprehensive molecular property calculations for drug-likeness assessment; and interactive visualization dashboards for real-time data exploration and hit interpretation.

This modular architecture enables users to execute complete workflows for routine analysis or utilize individual components for specialized applications. Each stage of the pipeline produces an executable script, allowing users to easily adapt the workflow to more complex experimental designs and custom settings, while maintaining reproducibility through standardized configuration files and output formats. The protocol has been successfully validated across diverse DEL architectures including multi-cycle libraries, hybridized libraries combining independent synthetic routes, and large-scale pharmaceutical screens with millions of compounds.

Overview of the procedure

The DELT-Hit protocol is organized into five sequential stages executed through a command-line interface designed to support both computational chemists experienced with bioinformatics tools and experimental scientists new to DEL data analysis:

(i) Project setup and library specification (Steps 1-3): Configuration file creation from Excel templates, library architecture definition, and experimental metadata specification

(ii) Chemical structure enumeration and property calculation (Steps 4-6): Automated SMILES generation from reaction schemes, comprehensive molecular descriptor calculation, and chemical space visualization

(iii) Sequence demultiplexing and quality assessment (Steps 7-9): High-throughput sequence processing, barcode identification with error correction, and quality control metric generation

(iv) Statistical analysis and hit detection (Steps 10-12): Enrichment analysis using established RNA-seq methods, hit ranking with multiple statistical approaches, and result validation

(v) Data visualization and interpretation (Step 13): Interactive dashboard exploration

Applications of the method

The DELT-Hit framework accommodates various library architectures from simple two-cycle libraries to complex multi-branch synthetic schemes and has been successfully applied across our diverse DEL screening campaigns:

A single-display DEL of two diversity elements comprising 366,600 glutamic acid derivatives (GB-DEL) was encoded in single-stranded DNA format and was tested in affinity-based selections, eventually allowing for the de novo identification of specific binders against several pharmaceutically relevant proteins. In addition, the GB-DEL was used for affinity maturation in dual display fashion by combining it with low affinity hits displayed on a pairing strand.

A further single-display DEL of two diversity elements 670,752 derivatives based on 2-azido-3-iodophenylpropionic acid scaffold (NF-DEL) which was also encoded in single-stranded DNA format and used for investigating the impact of stereo- and regiochemistry on ligand discovery. The NF-DEL was tested for a set of protein targets of pharmaceutical relevance, yielding selective enzyme inhibitors and binders to tumor-associated antigens, which enabled conditional chimeric antigen receptor T-cell activation and tumor targeting.

A single-display, double-stranded proof-of-principle DEL of three amino acid diversity elements of 1'100 was synthesized using a novel solid phase-based approach on magnetic beads featuring a final release of the library from the support. This release was enabled by both simple cleavage of the connecting linker to the solid support, resulting in a standard DEL (unpurified DEL), as well as by installing a terminal second linker to the bead, followed by cleavage of the first linker (thereby releasing incomplete synthetic structures) and then the second linker, releasing only the desired structures (self-purified DEL). Both versions of the DEL were then tested in affinity-based selections against carbonic anhydrase II and IX, revealing a markedly improved enrichment of the positive control for the self-purified DEL version. While the final synthetic yields of the individual library members of the self-purified DEL still vary depending on the individual building block, unlike standard DELs, the yields of chemical display of the self-purified DEL can be assessed by DNA sequencing of the unselected library because synthesis and encoding yields correlate. Furthermore, as synthesis occurs on solid support, this PureDEL technology allows the use of anhydrous reaction conditions, thereby enabling higher conversion yields and the implementation of - so far DEL-incompatible - water-free chemistry.

A dual display, DEL encoded Self-Assembling Chemical (ESAC) Libraries of 56 million members allowed for the two displayed peptides to be connected in an incremental fashion. The DP-DEL was synthesized and tested for modulating macrocycle flexibility using a two-step cyclization strategy. By performing affinity-based selections on three different targets, specific preferences for the various degrees of flexibility were observed, i.e., a 314 nM thrombin inhibitor from the conformationally restrained macrocyclic subset of the ESAC library, a more flexible streptavidin binder from the semi-closed conformation, as well as binders to human alkaline phosphatase (PLAP) from the open, conventional dual-display setup. The results obtained from this approach demonstrated the utility of dual-display DEL technology for facilitating the de novo discovery of peptide ligands through flexibility-tuning.

Limitations

While DELT-Hit addresses many challenges in DEL analysis, several limitations should be considered:

- Computational complexity: library size grows combinatorially with the number of diversity elements, which primarily impacts enumeration, property computation and representation. In contrast, demultiplexing complexity is limited by the number of sequencing reads, making this step largely independent of the library size.

- Library complexity: DELT-Hit supports arbitrary sequences of chemical reactions, provided that each synthetic route yields a single, well-defined final product, as is expected for high-quality DNA-encoded libraries. Libraries with complex architectures, such as non-standard reaction schemes, atypical building block representations, or unconventional precursor handling may require additional custom configuration and validation. In particular, the library enumeration step assumes accurate specification of reaction chemistry using reaction SMARTS. While DELT-Hit provides templates for commonly used DEL reactions, defining library-specific SMARTS may require advanced cheminformatics expertise to ensure correct product generation and structure reconstruction.

Comparison with other methods

Pharmaceutical companies that actively conduct DEL campaigns typically rely on internally developed or licensed software platforms that integrate sequencing, barcode decoding, hit identification, and downstream hit prioritization within proprietary ecosystems. While these systems support scalability, automation, and tight integration with experimental follow-up, they are closed-source, which restricts methodological transparency, limits user-driven customization, and hinders iterative improvement by the broader research community.

In parallel, contract research organizations (CROs) provide DEL screening as a service against user-supplied targets. These services generally return processed results without disclosing the underlying computational workflows or intermediate data products. Consequently, users have limited ability to adapt analytical procedures, interrogate intermediate outputs, or incorporate DEL informatics into their own computational pipelines.

To our knowledge, the only other open-source DEL framework introduced to date is DELi, that addresses several core aspects of DEL informatics, including barcode decoding, enrichment analysis, and library enumeration. DELi and DELT-Hit share key design principles, such as modular architecture, transparent configuration, and integration with established bioinformatics and cheminformatics tools (e.g., Cutadapt, RDKit). However, our framework differentiates itself through several distinct capabilities:

- Support for arbitrary reaction graphs for enumeration, enabling faithful reconstruction of complex synthetic schemes.

- Integration of statistical models developed for RNA-seq analysis for hit calling that account for replicate structure and dispersion.
- Support for arbitrary library designs, including single- and dual-display formats
- Standardized output of SMILES representations for library members, facilitating downstream machine-learning workflows.

DELT-Hit explicitly and deliberately exposes intermediate data products and executable scripts, facilitating reproducibility, benchmarking, and integration, which allows users to inspect, adapt, and extend each analysis step. This positions DELT-Hit as a complementary alternative for academic and industrial users seeking industrial-scale performance without the constraints of proprietary software.

Experimental design

Input requirements

The DELT-Hit framework processes three primary input categories, each with specific formatting requirements that are described in the following sections in detail:

(1) Library definition files (.xlsx): Defines building block structures in SMILES format, reaction SMARTS templates defining reaction steps, DNA constant sequences, and barcode-to-building block mapping tables. These are typically provided in Excel format with standardized sheet names for automated parsing.

(2) Raw sequencing data (fastq.gz): FASTQ files generated by Illumina or compatible sequencing platforms are supported in both compressed and uncompressed formats.

(3) Experimental metadata (.yaml): Selection condition specifications including target proteins, control experiments, and replicate groupings. This information defines the statistical comparisons and quality control parameters for downstream analysis.

Library architecture considerations

DELT-Hit supports diverse library architectures with flexible configuration options:

Single display libraries: Linear synthetic schemes where DNA tags are appended after each reaction cycle, suitable for most academic and industrial applications and focused screening campaigns.

Dual display libraries: Architectures where compounds are displayed on both complementary DNA strands, enabling higher diversity and more specialized DEL applications.

Multi-cycle libraries: Complex reaction schemes with arbitrary numbers of synthetic steps, branching reactions, and multiple building block incorporation sites.

The framework automatically validates library architecture consistency and provides warnings for potential issues such as incomplete reaction definitions or missing building blocks.

Library chemistry and reaction definition

Reaction cycles used during library construction are specified using standardized reaction SMARTS notation provided via structured Excel configuration files. DELT-Hit translates these definitions into an explicit reaction graph that supports arbitrary reaction sequences, branching synthetic pathways, and multi-step transformations.

DELT-Hit accommodates libraries of varying complexity, including architectures with multiple reaction cycles, provided that each synthetic route resolves to a single, well-defined final product, consistent with best practices for high-quality DNA-encoded libraries. Libraries employing non-standard reaction schemes, unconventional building block representations, or complex precursor handling may require additional customization and validation.

The library enumeration process critically depends on accurate specification of reaction schemes through SMARTS definition. Although DELT-Hit supplies validated templates for commonly used DEL reactions, library-specific adaptations often necessitate advanced cheminformatics expertise to ensure chemically correct product formation and reliable structure reconstruction.

Quality control parameters

DELT-Hit monitors comprehensive quality metrics throughout the analysis workflow:

- **Demultiplexing efficiency**: Percentage of sequencing reads successfully assigned to valid barcode combinations, typically >70% for high-quality datasets.
- **Barcode recovery rates**: Coverage of each barcode to identify abnormalities in the demultiplexing.
- **Statistical significance assessment**: Multiple testing correction and false discovery rate control using established RNA-seq analysis methodologies.
- **Replicate consistency**: Correlation analysis across biological and technical replicates.
- **Chemical reaction validation**: Strict requirements on reaction products.

Materials

Equipment

Computing hardware:

- Minimum configuration: 8 GB RAM, 8 CPU cores, 50 GB available storage
- Recommended configuration: 32 GB RAM, 16 CPU cores, 100 GB available storage
- High-performance setup: 64 GB RAM, 32 CPU cores, 250 GB SSD storage

Operating systems:

- Linux (Ubuntu 20.04+)
- macOS (12.0+)
- Windows 10/11 (with Windows Subsystem for Linux recommended)

Software dependencies

Core requirements:

- Python 3.12 or higher with pip package manager
- Conda package manager (Miniconda or Anaconda)
- R statistical computing environment (version 4.1+)
- Git version control system (version 2.0+)

Python packages (automatically installed):

- cutadapt (4.9+): Sequence adapter trimming and demultiplexing
- rdkit (2024.3+): Chemical structure processing and property calculation
- pandas (2.2+): Data manipulation and statistical analysis
- numpy (1.24+): Numerical computing and array operations
- matplotlib/seaborn: Data visualization and publication-quality plotting
- plotly: Interactive visualization and dashboard components
- scipy (1.10+): Statistical functions and optimization algorithms

R packages (manual installation required):

- edgeR: Differential expression analysis adapted for count data
- limma: Linear modeling and empirical Bayes statistics
- tidyverse: Data manipulation and visualization tools
- GGally: Extension to ggplot2 for correlation matrices
- BiocManager: Bioconductor package management

Input file preparation

Required input files:

Table 2 | Input file specifications and sources

File Type	Format	Description	Example Source
Campaign definition	Excel (.xlsx)	Selection conditions, building blocks, reactions, constants	Laboratory records
Sequencing data	FASTQ (.fastq / .fasta)	Raw sequencing reads	Illumina sequencer
analysis.yaml	Yaml (.yaml)	Replicates used for hit identification with edgeR	Laboratory records

Example datasets (available for download):

The provided example dataset is based on a toy library based on the architecture of a published library.(GB library reference)

Table 3 | Tutorial datasets for protocol validation

File	URL	Description
campaign.fastq.gz	https://figshare.com/ndownloader/files/61487743	Representative screening data
template.xlsx	https://figshare.com/ndownloader/files/61487731	Complete library and experimental definitions
analysis.yaml	https://figshare.com/ndownloader/files/61487728	
Supporting Material	https://doi.org/10.6084/m9.figshare.31198468	All output files generated by running the described protocol on the tutorial dataset

Software installation

The following installation procedure needs to be performed once before running the DELT-Hit protocol and is executed entirely from the command line. All subsequent analyses are likewise performed via terminal-based commands within the same software environment.

Step 1: Install Git version control system

Git is required to download the DELT-Hit package from the GitHub repository.

For macOS:

```
# Install using Homebrew (recommended)
brew install git
```

```
# Or download from: https://git-scm.com/download/mac
```

For Linux (Ubuntu/Debian):

```
sudo apt-get update
sudo apt-get install git
```

For Windows:

```
# Download and install from: https://git-scm.com/download/win
# Then use Git Bash or Windows Subsystem for Linux (WSL)
```

Verify Git installation:

```
git --version
# Expected output: git version 2.x.x or higher
```

Step 2: Environment setup with Conda

We recommend using Miniconda package manager to create an isolated environment ensuring proper dependency management. This step only needs to be performed once.

```
# Download and install Miniconda for your operating system
# Follow instructions at: https://docs.anaconda.com/miniconda

# Create dedicated environment (only needed once)
conda create -n delt-hit python=3.12 -y

# Activate environment (required for each new terminal session)
conda activate delt-hit
conda install pygraphviz -y

# Install DELT-Hit package
pip install git+https://git@github.com/DELTechnology/delt-hit.git
```

Note: You will need to activate the `delt-hit` environment (`conda activate delt-hit`) each time you open a new terminal session before running DELT-Hit commands.

Step 3: R environment configuration

Statistical analysis components require R with specific Bioconductor packages. This configuration only needs to be performed once.

Open R or RStudio and execute:

```
# Install required CRAN packages
install.packages(c("tidyverse", "GGally"))

# Install BiocManager for Bioconductor packages
```

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

# Install Bioconductor packages
BiocManager::install(c("edgeR", "limma"))
```

Step 4: Installation verification

Verify that all components are correctly installed:

```
# Activate environment
conda activate delt-hit

# Test core functionality
delt-hit init –help


# Expected output: Help message describing the init command
```

Procedure

Phase 1: Project initialization and configuration • TIMING 30-60 min

The configuration file defines library structure, experimental design, and analysis parameters. DELT-Hit supports initialization from standardized Excel templates for user convenience.

In this phase, we will create a YAML configuration file from an Excel spreadsheet that contains all the information about your DEL library and experimental design.

Step 1 | Prepare Excel configuration file

Create an Excel file with multiple sheets defining your experiment. Each sheet contains specific information required for the analysis. The following sections describe the required sheets and their format. For convenience, users can start from the template provided in Table 3.

1.1 Create the `experiment` sheet

This sheet contains basic experiment parameters including experiment name, file paths, and computational resources.

Table 4 | Experiment configuration parameters

variable	value
name	template
fastq_path	campaign.fastq.gz

save_dir	experiments
num_cores	11

CRITICAL STEP: Use absolute paths or paths relative to your working directory for `fastq_path` and `save_dir`.

1.2 Create the `selections` sheet

This sheet defines the experimental design including selection conditions, multiplexing barcodes, and replicate grouping. Each row describes an experimental setting (for example a different protein target or experimental condition like controls) and is identified by a unique combination of S0 and S1 primers to demultiplex the sequencing data into the individual experiments.

Table 5 | Selection experimental design

name	operator	date	group	target	S0	S1
AG24_4	A. Smith	26-Sep-24	no_protein		ACACAC	TCGATA
AG24_5	A. Smith	26-Sep-24	no_protein		ACAGCA	TCGATA
AG24_6	A. Smith	26-Sep-24	no_protein		ACATGT	TCGATA
AG24_13	A. Smith	26-Sep-24	protein	A	ACGACG	TCGATA
AG24_14	A. Smith	26-Sep-24	protein	A	ACGCGA	TCGATA
AG24_15	A. Smith	26-Sep-24	protein	A	ACTAGC	TCGATA

Required columns for demultiplexing:

- `name`: Unique identifier for each selection
- `S0`, `S1`: Multiplexing barcodes for selection identification

Required columns for statistical analysis:

- `group`: Statistical grouping (e.g., "protein", "no_protein", "naive")

Optional columns:

- `operator`: Person who performed the selection
- `date`: Date of selection experiment
- other columns can be added up to the users discretion

1.3 Create the `structure` sheet

This sheet defines the DNA construct architecture and error tolerance for demultiplexing.

There are three types of DNA regions:

- selection: This region is used for the selection primer as defined in the `selection` sheet and need to be named S0 and S1.
- constant: These regions are used as constant regions across all DNA constructs and need to be named sequentially C0, C1, etc.
- building_block: This region identifies a chemical building block used and needs to be named sequentially B0, B1, etc.

Each region in the DNA construct is identified by a unique *name* and a *max_error_rate* and whether *indels* are allowed during multiplexing can be configured for each region.

Table 6 | DNA sequence structure definition

name	type	max_error_rate	indels
S0	selection	0	FALSE
C0	constant	1.01	FALSE
B0	building_block	0	FALSE
C1	constant	1.01	FALSE
B1	building_block	0	FALSE
C2	constant	1.01	FALSE
S1	selection	0	FALSE

Column descriptions:

- `name`: Unique identifier for each DNA region
- `type`: Region type (`selection`, `constant`, or `building_block`)
- `max_error_rate`: Maximum allowed error rate for adapter trimming (0 = perfect match, 1.1 = ~10% errors)
- `indels`: Whether to allow insertions/deletions (TRUE/FALSE)

CRITICAL STEP: Selection barcodes should require perfect matches (`max_error_rate = 0`, `indels = FALSE`) to ensure accurate multiplexing, while constant regions can tolerate higher error rates depending on their length.

1.4 Create the `constant` sheet

This sheet contains the DNA sequences for all constant regions defined in the `structure` sheet.

Table 7 | Constant DNA region sequences

name	codon
C0	GGAGCTTCTGAATTCTGTGTGCTG
C1	CGAGCGTCAGGCAGCCTGTGTGCTG
C2	CGAGTCCCATGGCGC

Note: The `name` column must match the constant region `names` defined in the `structure` sheet.

Step 2 | Define chemical building blocks and reactions

DELT-Hit supports libraries with an arbitrary number of building blocks. For each building block region defined in the structure sheet, the user must define the used DNA codons in a sheet named according to the `structure` sheet (i.e. B0, B1, ..., BN). To enable library enumeration and property calculation, users must define the chemical structure of the building blocks and reactions used in the library synthesis as well.

2.1 Create building block sheets (B0, B1, etc.)

For each building block position defined in your `structure` sheet, create a separate sheet containing all building blocks for that position.

Table 8 | Building block definition sheet (example for B0)

codon	smiles	educt	reaction	product
ATCTAT	NCC1=CC=CS1	scaffold_1	ABF1	product_1
AGAATA	COC1=CC=C(OC)C(CN)=C1	scaffold_1	ABF1	product_1
TCATTA	NCCCSCC1=CC=C(F)C=C1	scaffold_1	ABF1	product_1
TTGACT	NCC1(CC1)C1=CC=CC=C1	scaffold_1	ABF1	product_1

Required columns for demultiplexing:

- `codon`: DNA barcode sequence identifying this building block

Required columns for library enumeration:

- `smiles`: Chemical structure in SMILES notation
- `reaction`: Name of the reaction using this building block
- `educt`: Name of the compound this building block reacts with if any
- `product`: Name of the product formed

2.2 Create the `reactions` sheet

Only required for library enumeration. Define all chemical reactions used during library synthesis using SMARTS notation. SMARTS must be adapted to the building blocks used to yield a single product for each transformation. If a reaction is used several times throughout library synthesis, these must be defined as individual reactions. Table 9 demonstrates this with two amide-bond forming reactions (ABF1 and ABF2)

Table 9 | Reaction SMARTS templates

name	smirks

ABF 1	[CX3:1](=[O:2])[OX2;H1].[N;$([N;H1,H2]),$([N;H3]);!$([N;H1,H2][S]);!$([NX3][CX3](=[OX1])[#6]);!$([NX3][CX3](=[OX1])[OX2]):4]>>[CX3:1](=[O:2])[N:4]
ABF 2	[CX3:1](=[O:2])[OX2;H1].[N;$([N;H1,H2]),$([N;H3]);!$([N;H1,H2][S]);!$([NX3][CX3](=[OX1])[#6]);!$([NX3][CX3](=[OX1])[OX2]):4]>>[CX3:1](=[O:2])[N:4]
SR	[#6:1][$([NX2-][NX2+]#[NX1]),$([NX2]=[NX2+]=[NX1-])]>>[#6:1][N;H2]
CuAAC	[CX2:1]#[CX2;H1:2].[N:3]=[N+:4]=[N-:5]>>[C:1]1=[C:2][N-0:3][N-0:4]=[N-0:5]1

Note: The `name` column must match the reaction names used in the building block sheets.

2.3 Create the `compounds` sheet

Only required for library enumeration. Define any additional chemical compounds (scaffolds, reagents) used in the reactions.

Table 10 | Compound definitions

name	smiles
scaffold_1	O=C(N)CCC(N=[N+]=[N-])C(O)=O

2.4 Create the additional reaction steps

Only required for advanced library enumeration. If the synthesis of the library requires additional reaction steps, for example for conversion of precursor molecules or deprotection steps before reactions, those can be described in the sheet `reaction_graph`. A reaction is defined by at most two educts that react through a reaction to a new product. For each additional reaction step you want to add, include a new row with the corresponding compound names and reaction names from the reactions and compounds sheet.

Table 11: Additional reaction step definitions

educt_1	educt_2	reaction	product
product_1		SR	product_1B

It might be necessary to define multiple reaction paths for precursors even though they share the same reaction node in theory. Table 11 demonstrates this with two scaffolds that are activated with a Staudinger reduction (SR) reaction before reacting with the building block. The definition in Table 12 adds two independent reaction paths to the reaction graph and the reaction pathway can be resolved. In contrast, using the same reaction name SR for both would result in an invalid reaction graph that defines:

scaffold_1 + scaffold_2 -> SR -> scaffold_1A + scaffold_1B

Table 12 | Additional reaction step definitions (special case)

educt_1	educt_2	reaction	product

scaffold_1		SR_precursor_1	scaffold_1B
scaffold_2		SR_precursor_2	scaffold_2B

Step 3 | Generate configuration file

Once your Excel file is complete with all required sheets, generate the YAML configuration file:

```
# Activate the delt-hit environment
conda activate delt-hit

# Generate configuration from Excel template
delt-hit init --excel_path=/path/to/your/experiment.xlsx
```

This command creates a YAML configuration file at `save_dir/name/config.yaml` as specified in your Excel file in sheet `experiment`.

Expected output:

```
Configuration created at /path/to/results/experiments/experiment-
1/config.yaml
```

CRITICAL STEP: Review the generated YAML file to verify all information was parsed correctly. The YAML file will be used for all subsequent analysis steps.

TROUBLESHOOTING: While the tool supports some basic input validation it is helpful to check the following if the command fails:

- All building blocks are present in the configuration file
- Column names match exactly as specified in the tables above
- Excel file was saved with the latest changes before running the command

---

Phase 2: Chemical library enumeration and analysis • TIMING 10 min - 2 h

This phase generates all possible chemical structures from your building block combinations and calculates molecular properties for downstream analysis. Library enumeration requires advanced knowledge of the SMARTS language. While we provide common reaction patterns observed in DNA encoded libraries, library specific tweaks are often necessary.

Step 4 | Enumerate library compounds from building blocks

Generate all possible chemical structures by combining building blocks according to the defined reaction schemes:

```
delt-hit library enumerate --config_path=/path/to/config.yaml --
graph_only=True  # produce reaction graph only for inspection

delt-hit library enumerate --config_path=/path/to/config.yaml
```

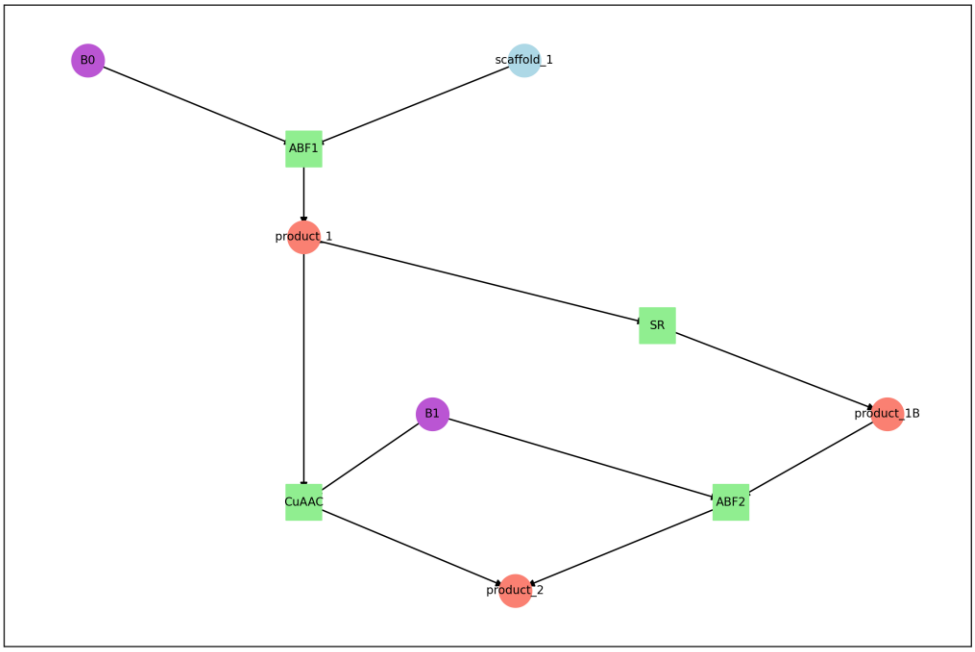
Those commands:

1. Construct a reaction graph from your building blocks and reactions
2. Enumerates all valid compound combinations
3. Validates chemical structures
4. Saves the complete library catalog

Expected outputs:

- `library.parquet`: Complete compound catalog with SMILES and barcode mappings
- `reaction_graph.png`: Visual representation of synthetic schema
- `additional_reactions_graph.png`: Visualizes the additional reactions from the `reaction_graph` sheet
- `building_block_reactions_graph.png`: Visualizes the reactions derived from the building block sheets `B0, B1`, etc.

Figure: Complete reaction graph for a DEL with two building blocks (violet), reactions (green), products (red) and starting compounds (blue).



Performance: Enumeration speed is approximately 1'000-10'000 compounds/second depending on reaction complexity. For a 1 million compound library, expect 2-10 minutes of processing time depending on the machine.

TROUBLESHOOTING:

- Check that all reaction SMARTS are valid and specific enough to produce a single product
- Ensure reactant and product names match between building blocks and compounds sheets
- Re-run enumeration in debugging mode to visualize the reaction graphs that fail: `delt-hit library enumerate --config_path=/path/to/config.yaml --debug=invalid`

Step 5 | Calculate molecular properties and descriptors

Compute comprehensive molecular descriptors for drug-likeness assessment and chemical space characterization:

```
delt-hit library properties --config_path=/path/to/config.yaml
```

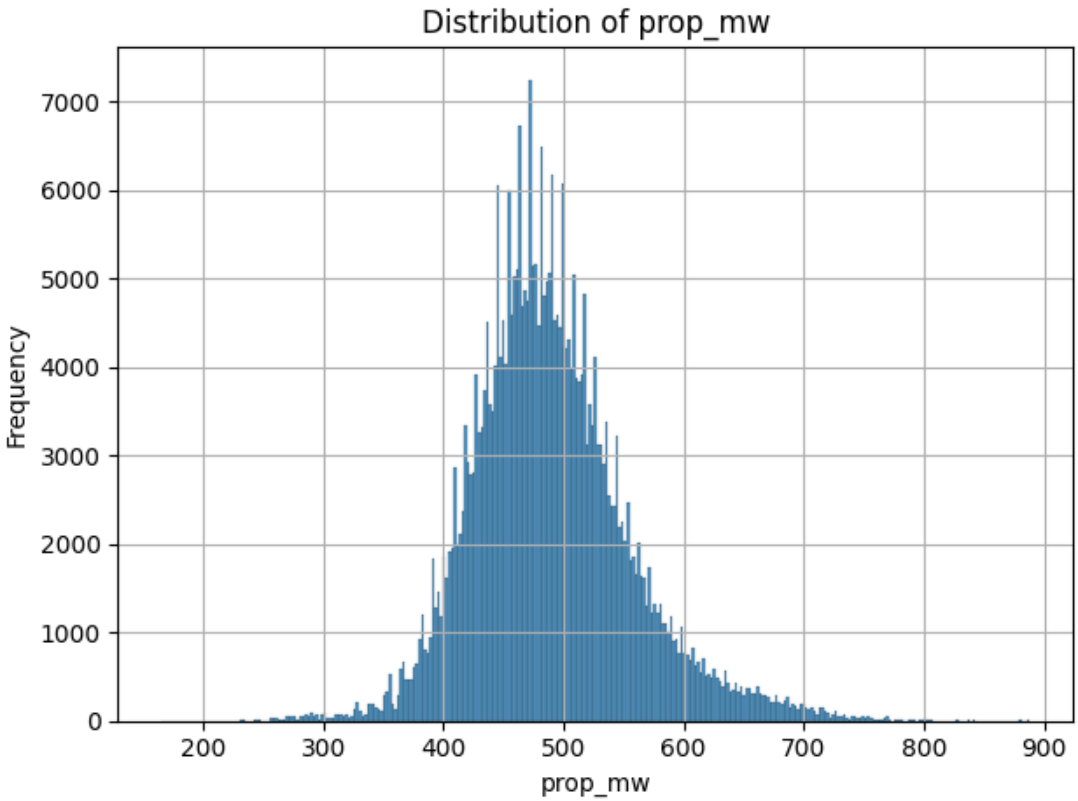
Computed properties include:

- **Lipinski descriptors**: Molecular weight (MW), partition coefficient (LogP), hydrogen bond donors (HBD), hydrogen bond acceptors (HBA)
- **Topological indices**: Topological polar surface area (TPSA), rotatable bonds
- **Structural complexity metrics**: Number of aromatic rings, heavy atom count, formal charge
- **Drug-likeness scores**: Quantitative estimate of drug-likeness (QED), fraction of sp^3 carbons

Expected outputs:

- `properties/properties.parquet`: Library catalog augmented with molecular descriptors
- `properties/prop_*.png`: Distribution plots for each calculated property

Figure: Exemplary plot for the molecular weight distribution of library members.



Performance note: Property calculation processes approximately 600-1000 compounds/second. For a 1 million compound library, expect 15-30 minutes processing time.

Step 6 | Generate molecular representations for machine learning

Create standardized chemical representations compatible with machine learning frameworks:

```
# Morgan fingerprints for similarity analysis
# ~600-1000 compounds/s
delt-hit library represent --method=morgan --config_path=/path/to/config.yaml

# BERT embeddings for deep learning applications
# ~600-1000 compounds/s
delt-hit library represent --method=bert --config_path=/path/to/config.yaml
```

Available representation methods:

- `morgan`: Morgan circular fingerprints (2048-bit vectors) for similarity searching and clustering
- `bert`: Transformer-based molecular embeddings (768-dimensional vectors) for deep learning

Expected outputs:

- `representations/morgan.npz`: Sparse matrix of Morgan fingerprints
- `representations/bert.npz`: Dense matrix of BERT embeddings

Note: These representations are compatible with scikit-learn, PyTorch, and TensorFlow for downstream machine learning applications.

Phase 3: Sequence processing and demultiplexing • TIMING 30 min - 4 h

This phase processes raw sequencing reads to identify and count each library member in your selection experiments.

Step 7 | Configure demultiplexing parameters

The demultiplexing parameters defined in your Excel file control how sequence reads are processed by *cutadapt*. Key parameters include:

- `max_error_rate`: Tolerance for sequencing errors (0 = perfect match, 1.1 = ~10% errors)
- `indels`: Whether to allow insertions/deletions during barcode matching

As mentioned earlier, we recommend requiring exact matches for selection barcodes (max_error_rate = 0, indels = FALSE) to ensure accurate multiplexing, while allowing higher error rates for constant regions depending on their length.

Step 8 | Prepare and execute sequence demultiplexing

Generate the demultiplexing scripts with its dependencies and execute the analysis:

```
# Generate Cutadapt input files and demultiplexing script
delt-hit demultiplex prepare --config_path=/path/to/config.yaml

# Execute demultiplexing
/path/to/experiment/demultiplex/cutadapt_input_files/demultiplex.sh
```

The demultiplexing process performs:

1. **Adapter detection and trimming**: Removal of sequencing adapters
2. **Barcode extraction**: Sequential identification of selection and building block barcodes
3. **Quality filtering**: Rejection of reads with errors exceeding defined thresholds

The produced demultiplexing executable (.sh) uses cutadapt and to process the FASTQ file and maps the sequences to the selection and codon IDs. It generates the `reads_with_adapters.gz` file that contains all sequences that have been mapped.

Performance note: Processing speed is approximately 50,000-200,000 reads/second. For a typical dataset with 50-100 million reads, expect 30 minutes to 2 hours processing time.

Expected outputs:

- `demultiplex/cutadapt_output_files/*.cutadapt.json`: Detailed processing statistics for each demultiplexing step
- `demultiplex/cutadapt_output_files/reads_with_adapters.gz`: Intermediate files with extracted barcodes

Step 9 | Process demultiplexing results and generate count tables

After demultiplexing completes, aggregate the results into count tables for each selection:

```
# Process demultiplexing output and generate count tables
delt-hit demultiplex process --config_path=/path/to/config.yaml
```

This command:

1. Parses the demultiplexed reads
2. Maps barcode combinations to library members
3. Generates count tables for each selection
4. Saves results in standardized format

Expected outputs:

- `selections/[selection_name]/counts.txt`: Tab-delimited count table with columns:
    - `code_1`, `code_2`, `...`: Building block indices
    - `name`: Selection identifier
    - `count`: Number of reads for this compound in this selection

NOTE: alternatively, counts can be saved as `{selection_name}_counts.txt` by using the flag `--as_files=True`. To save the counts sorted by codon ids instead of observed counts, set the flag .

Step 10 | Generate quality control reports

Assess demultiplexing quality with comprehensive statistics and visualizations:

```
# Generate text-based quality control report
delt-hit demultiplex report --config_path=/path/to/config.yaml

# Generate quality control visualizations
delt-hit demultiplex qc --config_path=/path/to/config.yaml
```

Expected outputs:

- `qc/report.txt`: Comprehensive text report with processing statistics
- `qc/*.png`: Quality control plots showing barcode recovery and coverage distributions

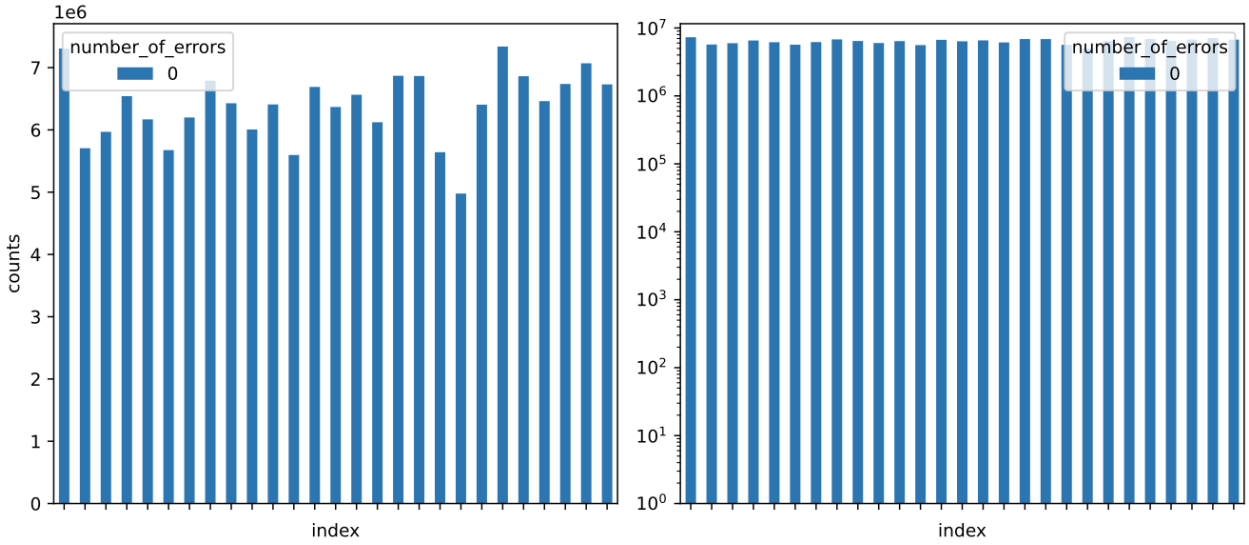
Figure: Example report for a poor selection with only 37% read retention. Report highlights the problematic region where more than 10% of the reads are lost (not mapped to codons).

Cutadapt Pipeline Report

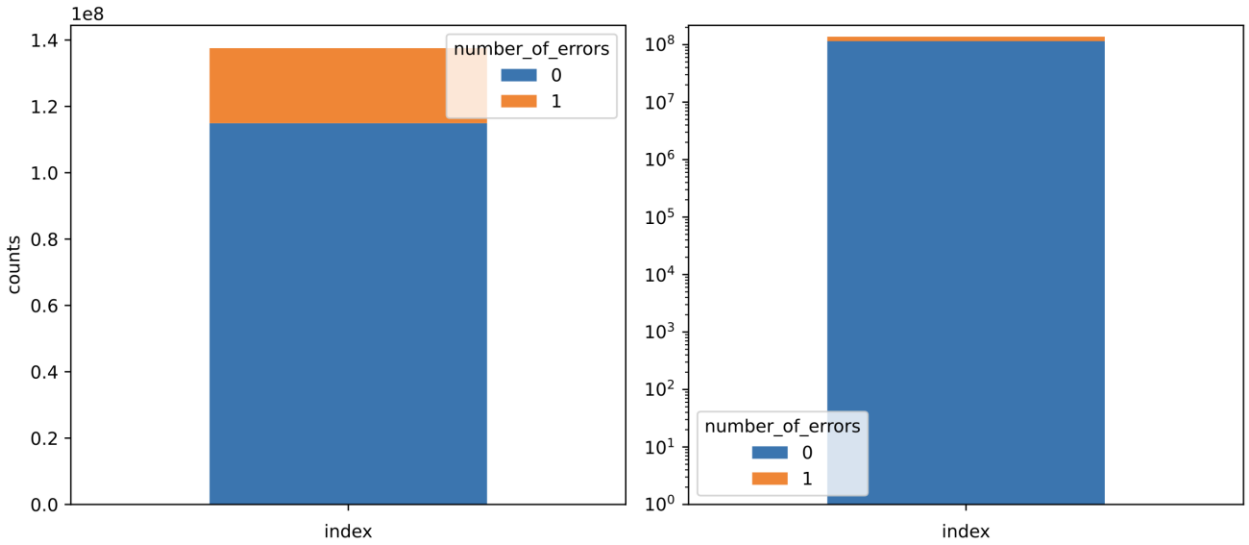
Region	Input	With adapters	Discarded	% with	% discarded
0-S0	190,703,154	172,500,482	18,202,672	90.45%	9.55%
1-C0	172,500,482	137,549,183	34,951,299	79.74%	20.26%
2-B0	137,549,183	93,215,748	44,333,435	67.77%	32.23%
3-C1	93,215,748	80,853,918	12,361,830	86.74%	13.26%
4-B1	80,853,918	73,998,527	6,855,391	91.52%	8.48%
5-C2	73,998,527	72,061,120	1,937,407	97.38%	2.62%
6-S1	72,061,120	71,472,279	588,841	99.18%	0.82%

Overall:
With adapters : 71,472,279 (37.48%)
Discarded : 119,230,875 (62.52%)

Figure: Stacked bar plots showing the number of reads per index position for selection or barcode codons as defined in the Excel file. Absolute read counts are shown on the right, and log-transformed counts on the left. For each index position, reads are displayed as stacked segments according to the number of sequencing errors. (TOP) Stacked barplot for the assigned counts to the different selection codons of S0. (BOTTOM) Stacked barplot for read assignment to constant region C0, in orange are the number of reads that were mapped to this region with 1 error, in blue the number of reads that were mapped with 0 errors.



1-C0, 1.375E+08 / 1.725E+08 (79.74%)
e0: 1.149E+08 (83.55%) e1: 2.262E+07 (16.45%)



Quality assessment criteria:

- **Read retention**: >70% of input reads should be successfully assigned
- **Barcode coverage**: Most building blocks should have >0 reads

TROUBLESHOOTING: Low read retention (<50%) may indicate:

- Incorrect barcode sequences in configuration
- Poor sequencing quality
- Poor library quality

Phase 4: Data visualization and interpretation • TIMING 15-60 min

The interactive dashboard provides real-time exploration of raw count data and experimental metadata.

Step 13 | Launch interactive analysis dashboard

Visualize demultiplexing results and explore count distributions:

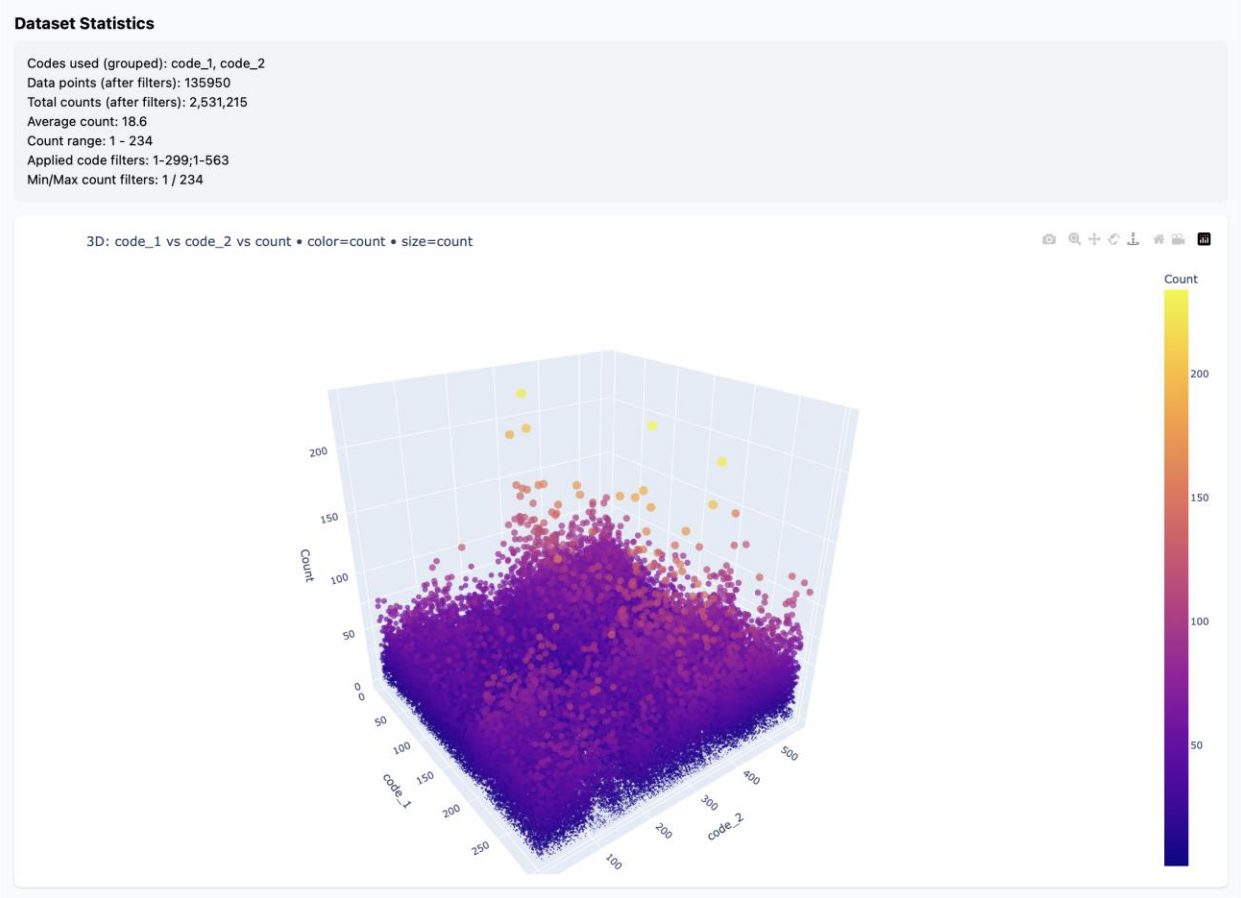
```
delt-hit dashboard \

--config_path=/path/to/config.yaml \

--counts_path=/path/to/experiment/selections/CA_P1/counts.txt



# Dash is running on http://127.0.0.1:8050/  # copy paste into browser

# * Serving Flask app 'delt_hit.cli.dashboard.api'

# * Debug mode: on
```

This command starts a server interface that can be opened from a web browser (typically at http://localhost:8050). Stop the server with ctrl+c or by closing the terminal.

Dashboard features:

- **Experimental metadata display**: Selection conditions, target information, and experimental parameters
- **Count distribution visualization**: Scatter/Histograms and summary statistics for the selected condition
- **Interactive filtering**: Explore specific regions of the count distribution

Figure 4 | Interactive dashboard interface. The dashboard provides interactive visualization of count data with filtering capabilities (not shown). The top panel shows experimental metadata (not shown), while the main panel displays count distributions and allows exploration of specific compounds.



Using the dashboard:

1. Select different count files to visualize different selection conditions
2. Visualize aggregated counts across codon dimensions:
3. Apply filters to focus on specific count ranges

Note: The dashboard displays raw counts from the demultiplexing step. For statistical comparisons and hit identification, use the enrichment analysis results from Phase 4. However, since the dashboard command accepts the counts with users can compute custom counts normalizations and visualize those normalized counts with the same command.

CRITICAL: The default folder hierarchy as produced by delt-hit process needs to be used or if a custom counts.txt is provided the user needs to pass the selection name with as specified in the selection sheet as well.

Phase 4: Statistical analysis and hit identification • TIMING 10-30 min

This phase identifies enriched library members by comparing selection conditions using established statistical methods.

Step 11 | Define statistical comparisons

From the information in the `selections` sheet, create a new dedicated configuration yaml file to specify which selections to compare. Add an `experiments` section to define a set of experiments that should be analysed. Each experiment in the list defines the parameters for statistical testing and requires a unique experiment name (protein_vs_no_protein). An experiment consists of multiple selections, typically technical/biological replicas which are used for the statistical testing and the outputs are saved the specified save_dir. Each selection is defined by its name and the path to the corresponding table with the counts (counts_path, as produced by the demultiplexing step) as well as an assignment to either the `protein` or `no_protein` group.

A new configuration file is used to support comparisons across different selection runs, i.e. runs that might come from multiple individual sequencing files.

```
# new analysis.yaml
experiments:
- name: protein_vs_no_protein
  save_dir: /Volumes/experiment-GB/analysis
  selections:
    - name: AG24_4
      counts_path: /Volumes/experiment-GB/selections/AG24_4/counts.txt
      group: no_protein
    - name: AG24_5
      counts_path: /Volumes/experiment-GB/selections/AG24_5/counts.txt
      group: no_protein
    - name: AG24_6
      counts_path: /Volumes/experiment-GB/selections/AG24_6/counts.txt
      group: no_protein
    - name: AG24_13
      counts_path: /Volumes/experiment-GB/selections/AG24_13/counts.txt
      group: protein
    - name: AG24_14
      counts_path: /Volumes/experiment-GB/selections/AG24_14/counts.txt
      group: protein
    - name: AG24_15
      counts_path: /Volumes/experiment-GB/selections/AG24_15/counts.txt
      group: protein
```

Step 12 | Perform enrichment analysis with statistical methods

DELT-Hit supports two statistical approaches for identifying enriched compounds. Both methods generate an Rscript that can be executed and/or further modified by the user.

Method 1: Simple counts method (suitable for single replicates)

```
# Generate R script for counts-based analysis
delt-hit analyse enrichment \
  --config_path=/path/to/analysis-config.yaml \
```

```
  --name=protein_vs_no_protein \
  --method=counts

# Execute the generated R script
Rscript --vanilla /path/to/experiment/analyses/analysis-
1/counts/enrichment_counts.R
```

The counts method:

- Averages counts across replicates within each group
- Computes simple differences between groups
- Does not provide statistical significance testing
- Suitable for exploratory analysis or when replicates are not available

Method 2: edgeR method (recommended when replicates are available)

```
# Generate R script for edgeR statistical analysis
delt-hit analyse enrichment \
  --config_path=/path/to/analysis-config.yaml \
  --name=protein_vs_no_protein \
  --method=edgeR

# Execute the generated R script
Rscript --vanilla /path/to/experiment/analyses/analysis-
1/edgeR/enrichment_edgeR.R
```

The edgeR method:

- Implements sophisticated RNA-seq-derived statistical models
- Uses empirical Bayes shrinkage for improved power
- Calculates false discovery rate (FDR) corrected p-values
- Requires replicates (minimum 2 per group)
- Provides normalized count tables

For both methods, the tool produces the Rscripts that implement the corresponding analysis. This enables users to easily further customize their analysis pipelines.

Expected outputs (both methods):

- `stats.csv`: Complete statistics for all library members with columns:
  - `code_1`, `code_2`, `etc.`: Building block indices
  - `LogFC`: Log2 fold-change (edgeR) or difference in mean counts (counts method)
  - `PValue`: Raw p-value (edgeR only)
  - `FDR`: False discovery rate corrected p-value (edgeR only)
- `enrichment_hits.csv`: Top enriched compounds (FDR < 0.05 for edgeR)

- `correlation_*.png`: Replicate correlation plots for quality control
- Normalized count files for each selection or group

Interpreting results:

- **LogFC > 0**: Compound enriched in protein selections vs controls
- **LogFC < 0**: Compound depleted in protein selections vs controls
- **FDR < 0.05**: Statistically significant enrichment (edgeR only)
- High replicate correlation ($R^2 > 0.7$) indicates good experimental consistency

CRITICAL STEP: Always examine replicate correlation plots to verify data quality before interpreting hit lists. Poor correlation may indicate technical problems or batch effects.

TROUBLESHOOTING: If no hits are identified:

- Check that experimental groups are correctly assigned
- Verify sufficient sequencing depth (>100 reads per compound for reliable statistics)
- Consider adjusting FDR threshold for exploratory analysis
- Examine correlation plots for replicate consistency issues

Figure: Replicate correlation plots for quality control produced by the analysis step.

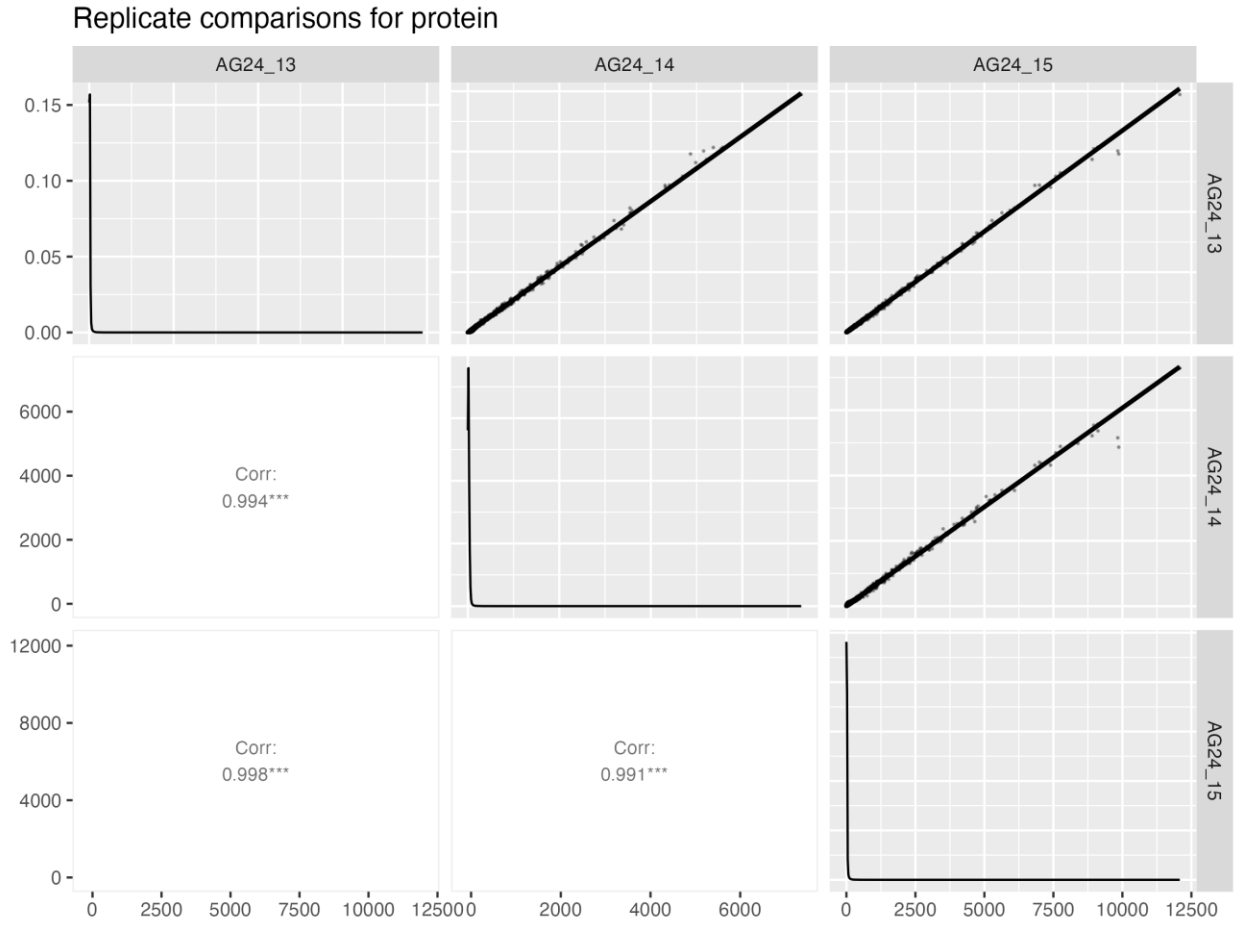


Table: Hit identification by edgeR. Compounds are for FDR<0.05 and sorted by logFC to identify the most relevant compounds.

code_1	code_2	logFC	logCPM	LR	PValue	FDR
94	472	9.76658103	4.65399379	140.973181	1.63E-32	2.99E-29
94	337	9.52452422	4.41931582	128.115071	1.06E-29	9.38E-27
250	250	9.50138143	4.39920396	136.048263	1.95E-31	2.62E-28
250	523	9.46551754	4.36448577	134.035088	5.37E-31	6.34E-28
94	364	9.29500646	4.19806216	117.172728	2.63E-27	1.35E-24
250	208	9.2845154	5.99659857	173.194896	1.48E-39	2.05E-35
250	337	9.06687397	3.98375428	115.539372	6.00E-27	2.80E-24
94	341	8.99590166	3.91723403	113.847311	1.41E-26	6.00E-24
250	414	8.99240694	3.90981012	105.694583	8.60E-25	2.69E-22
250	96	8.92430743	8.02773969	203.793994	3.10E-46	2.36E-41

Troubleshooting

This section provides solutions to common issues encountered during DELT-Hit analysis.

Table 11 | Common issues and solutions

Problem	Possible Cause	Solution
Command fails with unrecognized arguments	Trailing spaces after \ when using multiline commands; Mistyped the command	Ensure that multiline commands do not have trailing spaces and ensure that the delt-hit commands are correctly typed. You can find more information for each command by typing delt-hit <cmd> --help
Demultiplexing fails No such file or directory: '.../input.fastq.gz'	Trailing whitespaces in the fail name in as defined in the excel sheet experiment	Remove and whitespaces and spaces in the file name and regenerate the config file with the delt-hit init command
Low demultiplexing efficiency (<50%)	Incorrect barcode sequences, inverted barcode sequences	Increase `max_error_rate` parameters; validate barcode sequences against sequencing data; check for adapter contamination
R script failures	Missing R packages, R not available in environment	Reinstall R into conda environment; Reinstall R packages using BiocManager; verify R installation with `which R`; check R version (4.1+ required)
Empty hit lists	Stringent statistical thresholds	Adjust FDR cutoffs (try 0.1 or 0.2 for exploratory analysis); check replicate consistency
Chemical structure errors	Invalid SMILES, incorrect reaction SMARTS	Validate building block structures using RDKit; check reaction definitions; test reactions with small building block subset; run enumeration in debug mode
"Command not found" errors	Conda environment not activated	Run `conda activate delt-hit` before executing DELT-Hit commands
Permission denied for shell scripts	Executable permission not set	Run `chmod u+x <script_name.sh>` before executing shell scripts
Poor replicate correlation	Batch effects, technical variability	Review selection protocol for consistency; check for contamination

Performance optimization guidelines

Memory requirements by library size:

- **Small libraries** (<1M compounds): 16 GB RAM sufficient
- **Medium libraries** (1-50M compounds): 32GB RAM recommended
- **Large libraries** (>50M compounds): 32GB RAM required

Processing speed optimization:

- Use SSD storage for faster I/O operations
- Increase `num_cores` parameter for parallel processing
- Use compressed FASTQ files (.gz) to reduce disk space

Quality control thresholds

Demultiplexing quality indicators:

- **Excellent**: >90% initial read retention
- **Good**: 70-90% initial retention
- **Acceptable**: 50-70% initial retention
- **Poor**: <50% retention (requires troubleshooting)

Statistical analysis quality indicators:

- **Replicate correlation**: R^2 >0.7 for biological replicates (R^2 >0.9 for technical replicates)
- Positive / negative controls: That positive and negative controls give expected outputs

Timing

Protocol execution times depend on dataset characteristics and computational resources. This table provides estimates for different dataset scales using the recommended hardware configuration (32 GB RAM, 16 CPU cores).

Table 12 | Timing estimates for different dataset sizes

Analysis Phase	Small Dataset[1]	Medium Dataset[2]	Large Dataset[3]
Software installation (one-time)	15-30 min	15-30 min	15-30 min
Project initialization	15-30 min	15-60 min	60-120 min
Library enumeration	2-5 min	15-45 min	1-2 h
Property calculation	1-3 min	10-30 min	30 min-1 h
Molecular representation	2-5 min	15-45 min	45 min-1.5 h
Sequence demultiplexing	10-30 min	1-2 h	2-6 h

Analysis Phase	Small Dataset[1]	Medium Dataset[2]	Large Dataset[3]
Statistical analysis	2-5 min	5-15 min	15-45 min
Visualization and interpretation	5-15 min	10-30 min	15-45 min
Total workflow time	**45 min-1.5 h**	**2-4 h**	**4-10 h**

[1]Small dataset: <100K compounds
[2]Medium dataset: 100K-10M compounds
[3]Large dataset: >10M compounds

Notes:

- Software installation is required only once before first use
- Times assume recommended hardware configuration (32 GB RAM, 16 CPU cores)
- Parallel processing (`num_cores` parameter) significantly affects timing

---

Anticipated results

Output file structure

DELT-Hit generates a comprehensive, standardized output hierarchy:

```
project_root/
├── analysis.yaml                        # analysis specification
├── campaign.fastq.gz                    # Raw sequencing reads
├── template.xlsx                        # Input excel sheet
│
└── experiments/                         # experiments (one folder per campaign)
    └── template/                        # Example experiment instance
        ├── config.yaml                  # Experiment-level config
        │
        ├── demultiplex/                 # Cutadapt-based demultiplexing)
        │   ├── cutadapt_input_files/    # Files passed to Cutadapt (generated)
        │   │   ├── demultiplex.sh       # Generated shell script running Cutadapt steps
        │   │   └── {0-S0,1-C0,2-B0,...}.fastq   # Cutadapt adapter lists
        │   └── cutadapt_output_files/   # Cutadapt outputs per step
```

```
│       ├── *.cutadapt.json      # Per-step JSON stats from Cutadapt
│       ├── *.cutadapt.log       # Per-step logs (stdout/stderr capture)
│       ├── input.fastq.gz       # Cutadapt input for the demultiplexing
│       ├── out.fastq.gz         # Cutadapt output for the demultiplexing
│       └── reads_with_adapters.gz  # Reads retained with adapters
│
├── library/                     # Library reconstruction & enumeration outputs
│   ├── library.parquet          # Enumerated library table
│   ├── reaction_graph.png       # Main reaction graph visualization
│   ├── building_block_reactions_graph.png   # Graph focused on building blocks
│   ├── additional_reactions_graph.png       # Extra/optional reactions graph
│   │
│   └── properties/              # Calculated molecular descriptors for the library
│       ├── properties.parquet   # Descriptor table aligned with library.parquet
│       └── prop_*.png           # Descriptor distributions (e.g.,mw/logP/TPSA/QED/…)
│
├── representations/             # ML-ready representations derived from SMILES
│   ├── morgan.npz               # Morgan fingerprints (sparse array)
│   └── bert.npz                 # Embeddings (e.g., SMILES BERT)
│
├── selections/                  # Per-selection count tables (from demultiplexing)
│   ├── AG24_4/                  # One selection / condition / replicate
│   │   └── counts.txt           # Count table for this selection
│   ├── AG24_5/
│   │   └── counts.txt
│   ├── AG24_6/
```

```
        │   │   └── counts.txt
        │   ├── AG24_13/
        │   │   └── counts.txt
        │   └── ...                      # Many more selections
        │   # Note: AG24_*_counts.txt files at this level are flattened copies of the per-
folder counts
        │
        ├── qc/                          # Quality control outputs
        │   ├── report.txt               # Human-readable QC summary
        │   ├── hits_{0-S0,1-C0,2-B0,...}.parquet  # Per-step “hit” tables
        │   └── hits_{0-S0,1-C0,2-B0,...}.pdf      # Per-step QC report plots
        │
        └── analysis/                    # Downstream statistical analyses
            └── protein_vs_no_protein/   # One analysis definition (experiment name)
                ├── samples.csv          # metadata for statistical analysis
                ├── data.csv             # data for statistical analysis
                │
                ├── counts/              # Simple counts-based enrichment
                │   ├── enrichment_counts.R          # Script to compute enrichment
                │   ├── stats.csv                    # Full results table
                │   ├── hits.csv                     # Filtered top hits
                │   ├── protein.csv                  # Group-normalized counts
                │   ├── no_protein.csv               # Group-normalized counts
                │   └── correlation_{protein,no_protein}.png  # Replicate correlation plots
                │
                └── edgeR/               # edgeR-based differential enrichment
                    ├── enrichment_edgeR.R           # edgeR workflow script
```

```
├── enrichment_stats.csv         # Full results with p-values/FDR
├── enrichment_hits.csv          # Significant hits (e.g., FDR < 0.05)
├── {AG24_4,AG24_5,AG24_6,...}.csv  # Normalized counts per selection
└── correlation_{protein,no_protein}.png  # Replicate correlation plots
```

Chemical library characterization

Library enumeration generates comprehensive structural and chemical space information for all possible compounds in your DEL.

Library enumeration:

The reaction graph (**Figure**) visualizes the synthetic scheme, showing how building blocks combine through defined reactions to generate final products. This representation helps validate:

- Correct reaction connectivity
- Building block incorporation at each step
- Expected product formation

Library size:

Library enumeration should produce as many compounds as expected for the provided library setup.

Molecular property distributions:

Property calculation provides comprehensive descriptor analysis for drug-likeness assessment. **Figure 2** shows a typical molecular weight distribution with most compounds falling in the 300-600 Da range.

Demultiplexing performance

Successful demultiplexing indicates high-quality library and sequencing data.

Quality control visualizations (**Figure 3**) show:

- **Read retention through processing steps**: Tracks the percentage of reads retained after each adapter trimming step
- **Barcode coverage distributions**: Displays how evenly reads are distributed across building blocks
- **Error rate distributions**: Shows the frequency of sequencing errors detected at each position

Performance interpretation:

- **Excellent/Good performance**: Library is well constructed with high-quality sequencing
- **Acceptable performance**: Usable for analysis but may have reduced sensitivity
- **Poor performance**: Indicates technical issues requiring troubleshooting (see Troubleshooting section)

Common patterns:

- **High initial retention (>90%)**: Correct adapter sequences, good sequencing quality
- **Even barcode distribution**: Well-balanced library without synthesis bias
- **Low error rates (<2%)**: High-quality sequencing suitable for stringent matching

Statistical analysis outputs

Hit identification provides ranked lists of enriched compounds with statistical significance assessment.

Expected enrichment patterns:

For a typical protein target screen against negative controls:

- **Dynamic range**: 2-1000 fold enrichment over negative controls
- **Replicate consistency**: Pearson correlation R > 0.7 between biological replicates
- **Statistical power**: Reliable detection of >2-fold enrichment with >100 reads per compound
- Reproducible selection experiments
- Minimal batch effects
- Reliable statistical inference

Integration capabilities and downstream applications

DELT-Hit outputs integrate seamlessly with established cheminformatics and machine learning workflows.

Machine learning workflows:

The generated molecular representations enable:

- **Scikit-learn compatibility**: Morgan fingerprints for classification, regression, and clustering
- **Deep learning frameworks**: BERT embeddings compatible with TensorFlow and PyTorch
- **Chemical similarity analysis**: Tanimoto similarity calculations using Morgan fingerprints
- **Dimensionality reduction**: PCA, t-SNE, or UMAP visualization of chemical space

Cheminformatics pipelines:

Integration with RDKit and other tools for:

- **Structure-activity relationship (SAR) analysis**: Identification of structural motifs associated with activity
- **Virtual screening**: Use enriched compounds to search larger chemical databases

Statistical analysis:

Direct export to R for specialized modeling:

- **Dose-response modeling**: Analyze enrichment across multiple selection stringencies
- **Multi-target analysis**: Compare enrichment patterns across different protein targets

Data and code availability

Software access and documentation

DELT-Hit software is freely available under MIT License:

- **Primary repository**: https://github.com/DELTechnology/delt-hit
- **Installation**: `pip install git+https://git@github.com/DELTechnology/delt-hit.git@paper`
- **Documentation**: Comprehensive user guides, API reference, and tutorials at the repository
- **Issue tracking**: Community support and bug reporting via GitHub Issues

---

Author contributions

A.M. conceived the project, designed the software architecture, and implemented core algorithms. A.L. created and maintained the configuration files and testing frameworks. G.H implemented core algorithms. A.G. performed selection experiments and helped with result interpretation. L.P. performed software validation and contributed to code fine-tuning. J.S. provided scientific oversight and experimental validation. All authors contributed to manuscript preparation and revision.

Competing interests

The authors declare no competing financial interests.

Acknowledgments

We thank the DEL research community for valuable feedback during software development and beta testing. We acknowledge the developers of Cutadapt, RDKit, edgeR, and other

open-source tools that enable this framework. Special thanks to early adopters who provided critical feedback for improving usability and robustness.

References

1. Satz, A. L. *et al*. DNA-encoded chemical libraries. *Nat. Rev. Methods Primers* **2**, 2 (2022).

2. Goodnow, R. A., Dumelin, C. E. & Keefe, A. D. DNA-encoded chemistry: enabling the deeper sampling of chemical space. *Nat. Rev. Drug Discov.* **16**, 131–147 (2017).

3. Brenner, S. & Lerner, R. A. Encoded combinatorial chemistry. *Proc. Natl. Acad. Sci. USA* **89**, 5381–5383 (1992).

4. Franzini, R. M. & Randolph, C. Chemical space of DNA-encoded libraries. *J. Med. Chem*. **59**, 6629–6644 (2016).

5. Robinson, M. D., McCarthy, D. J. & Smyth, G. K. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* **26**, 139–140 (2010).

6. Martin, M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet J.* **17**, 10–12 (2011).

7. Landrum, G. *et al*. RDKit: Open-source cheminformatics. https://www.rdkit.org (2023).

8. McInnes, C. DNA-encoded library strategies for drug discovery. *Curr. Opin. Chem. Biol*. **26**, 77–83 (2015).

9. Machutta, C. A. *et al*. Prioritizing multiple therapeutic targets in parallel using automated DNA-encoded library screening. *Nat. Commun*. **13**, 6742 (2022).

10. Bigatti, M. *et al*. Impact of library design on the quality of hits identified from DNA-encoded chemical libraries. *ACS Comb. Sci.* **23**, 309–321 (2021).