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

Authors

Adriano Martinelli, Alice Lessing, Garry Hoppeler, Andreas Glogger, 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 small molecules through DNA-tag identification via high-throughput sequencing. As outlined in the comprehensive Nature Reviews Methods Primers on this technology (Satz et al., 2022), DELs are now employed by numerous pharmaceutical companies and academic laboratories worldwide. However, the computational analysis of DEL screening data remains a critical bottleneck, requiring sophisticated integration of genomics, cheminformatics, and statistical analysis workflows that are currently accessible only through proprietary or highly specialized software solutions.

This protocol presents DELT-Hit (DNA-Encoded Library Technology Hit identification), a comprehensive open-source computational framework that makes DEL data analysis accessible through an intuitive command-line interface to both computational and experimental researchers. DELT-Hit is specifically designed to handle the scale and complexity of modern industrial DEL campaigns, supporting libraries containing hundreds of millions of compounds while maintaining computational efficiency and user accessibility.

DELT-Hit offers a complete pipeline that converts raw FASTQ reads into machine learning-ready chemical information through five interconnected modules: (1) adaptive sequence demultiplexing using optimized RNA-seq algorithms with DEL-specific 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 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.

The modular architecture allows researchers to customize workflows while maintaining reproducibility through configuration files and standardized output formats. We demonstrate the protocol's effectiveness using representative single and dual display DEL screening datasets, showcasing the complete analysis pipeline from raw sequencing reads to ranked lists of chemical hits with computed chemical properties and representations for downstream machine learning tasks. The entire analysis, including quality control and visualization, can be completed within 2-6 hours on standard computational hardware for typical datasets, making it accessible to laboratories without specialized computing infrastructure.

DELT-Hit addresses the 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 tutorial datasets with both single and dual display examples, ensuring broad adoption and consistent implementation across the growing DEL community.

Key Points

- **Industrial-scale capabilities**: DELT-Hit is designed to handle the computational demands of modern pharmaceutical DEL campaigns, efficiently processing libraries containing hundreds of millions of compounds while maintaining user-friendly operation
- **Comprehensive dual 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 developed through extensive validation studies
- **Flexible and robust design**: Modular architecture accommodates diverse library formats, custom reaction templates, and building block definitions while maintaining rigorous quality control standards
- **Research-grade quality assurance**: Built-in quality control metrics, automated validation checks, and standardized reporting ensure reliable results and facilitate systematic troubleshooting across different experimental conditions
- **Machine learning ecosystem integration**: Generates standardized, analysis-ready datasets fully 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:

Core Analysis 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 drug discovery by enabling the synthesis and screening of chemical spaces that would be impractical to explore using traditional high-throughput screening approaches. In DEL technology, each chemical compound is covalently linked to a unique DNA barcode, allowing millions to billions of compounds to be screened simultaneously against biological targets through DNA sequencing-based identification of enriched library members.

The computational analysis of 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 seamless integration with cheminformatics workflows for hit optimization and structure-activity relationship analysis.

Current computational tools for DEL analysis often focus on individual workflow components rather than providing comprehensive end-to-end solutions, require significant programming expertise for implementation, or lack the flexibility needed 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 result validation.

Development of the protocol

DELT-Hit addresses these limitations through a unified, modular framework built around several key design principles: leveraging established bioinformatics tools where appropriate while incorporating DEL-specific optimizations, providing flexible configuration systems for diverse library designs and experimental protocols, implementing comprehensive quality control at each analysis stage, and maintaining accessibility for users with varying computational backgrounds through intuitive command-line interfaces and extensive documentation.

The framework integrates multiple specialized modules: sequence demultiplexing using adapted Cutadapt workflows with DEL-optimized parameters, chemical structure reconstruction using RDKit with reaction SMARTS validation, statistical analysis using edgeR with DEL-appropriate normalization strategies, comprehensive molecular property calculation 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, 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.

Comparison with other methods

Several academic and commercial solutions address components of the DEL informatics workflow, but few provide comprehensive, openly available end-to-end pipelines. **Table 1** compares DELT-Hit with representative existing methods across key criteria including availability, scope, and performance characteristics.

Table 1 | Comparison of DELT-Hit with existing DEL analysis methods

Feature	DELT-Hit	DELi (UNC)	Commercial Platform A[1]	Academic Tool B[2]
Open source	Yes	Yes	No	Partial
Complete pipeline	Yes	Limited	Yes	No
Dual display support	Yes	No	Yes	No
Statistical analysis	edgeR integration	Basic counts	Proprietary	Custom
ML-ready	Yes	No	Yes	No
Scalability	>500M compounds	<50M compounds	>1B compounds	<10M compounds
Documentation	Comprehensive	Basic	Commercial	Limited
Cost	Free	Free	License required	Free

[1]Commercial platforms vary in capabilities and are not directly comparable
[2]Representative of specialized academic tools focusing on specific workflow components

DELT-Hit provides unique advantages in combining industrial-scale performance with open-source accessibility, comprehensive dual architecture support, and seamless integration with machine learning workflows. Unlike commercial solutions, DELT-Hit enables full methodological transparency and customization, while providing more complete functionality than existing academic tools.

Applications of the method

DELT-Hit has been successfully applied across diverse DEL screening campaigns, including target classes such as . The framework accommodates various library architectures from simple two-cycle libraries to complex multi-branch synthetic schemes. Representative applications include:

- Publication 1:
- Publication 2:

Limitations

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

- **Computational requirements**: Memory usage and processing time scale significantly with library size and sequencing depth, requiring high-memory systems for very large datasets (>1 billion compounds)
- **Library complexity**: Complex architectures with non-standard reaction schemes or unusual building block formats may require custom configuration and validation
- **Error model assumptions**: Demultiplexing algorithms assume independence of sequencing errors across barcode positions, which may not hold for all sequencing platforms
- **Chemical structure dependency**: Structure reconstruction accuracy depends on precise reaction SMARTS definitions and building block structure quality

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

The workflow supports both automated execution for routine screening analysis and step-by-step processing for method development and troubleshooting.

Experimental design

Input requirements

The DELT-Hit framework processes three primary input categories, each with specific formatting requirements:

(1) Library definition files: Building block structures in SMILES format, reaction SMARTS templates defining synthetic transformations, 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) Experimental metadata: Selection condition specifications including target proteins, control experiments, multiplexing barcodes, and replicate groupings. This information defines the statistical comparisons and quality control parameters for downstream analysis.

(3) Raw sequencing data: FASTQ files from Illumina or compatible sequencing platforms, with typical read lengths of 150-300 bp to accommodate full barcode sequences and quality scores for error correction algorithms.

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 applications and focused screening campaigns.

Dual display libraries: Architectures where compounds are displayed on both DNA strands, enabling higher diversity.

Multi-cycle libraries: Complex 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 performed during library construction are defined through standardized SMARTS notation in user-provided configuration files. DELT-Hit constructs a reaction graph representation that supports arbitrary reaction sequences, branching pathways, and multiple product formation routes.

Key considerations for reaction definition:

- **SMARTS validation**: Automatic checking of reaction template syntax and chemical feasibility
- **Building block compatibility**: Verification that building blocks contain required functional groups
- **Product prediction**: Enumeration validation to ensure expected chemical structures are generated
- **Error handling**: Systematic identification and reporting of problematic reactions or building blocks

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 methodologies.

Replicate consistency: Correlation analysis and batch effect detection across biological and technical replicates.

Chemical structure validation: Automated detection of problematic structures, stereochemistry issues, and drug-likeness assessment.

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, 500 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/.gz)	Raw sequencing reads	Illumina sequencer

Example datasets (available for download):

Table 3 | Tutorial datasets for protocol validation

File	URL	Description
NF-selection-campaign.fastq.gz	https://figshare.com/ndownloader/files/58345816	Representative dual display screening
NF-selection-campaign.xlsx	https://figshare.com/ndownloader/files/58345864	Complete library and experimental de

Software installation

The following installation procedure needs to be performed once before running the DELT-Hit protocol. All subsequent analyses will use this same 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

# Install DELT-Hit package
pip install git+ssh://git@github.com/DELTechnology/delt-core.git@paper

# Verify installation
delt-hit --help
```

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 15-45 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.

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	Description
name	experiment-1	Unique identifier for this analysis
fastq_path	data/sequencing/NF-selection.fastq.gz	Path to raw sequencing data
save_dir	results/experiments	Directory where results will be saved
num_cores	16	Number of CPU cores to use for parallel tasks

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.

Table 5 | Selection experimental design

name	operator	date	target	group	S0	S1
CA_N1	A.Smith	15-Oct-24	-	no_protein	ACACAC	CGCTCGATA
CA_N2	A.Smith	15-Oct-24	-	no_protein	ACAGCA	CGCTCGATA
CA_P1	A.Smith	15-Oct-24	hCAII	protein	ACGACG	CGCTCGATA
CA_P2	A.Smith	15-Oct-24	hCAII	protein	ACGCGA	CGCTCGATA

name	operator	date	target	group	S0	S1
CA_A1	A.Smith	15-Oct-24	-	naive	ACGCGT	CGCTCGATA
CA_A2	A.Smith	15-Oct-24	-	naive	ACGCGG	CGCTCGATA

Required columns:

- `name`: Unique identifier for each selection
- `target`: Target protein or ”-” for controls
- `group`: Statistical grouping (e.g., ”protein”, ”no_protein”, ”naive”)
- `S0`, `S1`: Multiplexing barcodes for selection identification

Optional columns:

- `operator`: Person who performed the selection
- `date`: Date of selection experiment

CRITICAL STEP: The `group` column defines which selections will be compared during statistical analysis. Typically, ”protein” selections are compared against ”no_protein” controls.

1.3 Create the `structure` sheet This sheet defines the DNA construct architecture and error tolerance for demultiplexing.

Table 6 | DNA sequence structure definition

name	type	max_error_rate	indels
S0	selection	0	FALSE
C0	constant	1.1	TRUE
B0	building_block	0	FALSE
C1	constant	1.1	TRUE
B1	building_block	0	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.

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	CGAGTCCCATGGCGCCGGATCGACG
C2	GCGTCAGGCAGC

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

Step 2 | Define chemical building blocks and reactions To enable library enumeration and property calculation, you must define the building blocks and reactions used in your library synthesis.

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)

smiles	codon	reaction	reactant	product
OC(=O)C1=CC(=CN=C1)C#C	GCCTCG	CuAAC	scaffold_1	product_1
BrC1=NC=C(OCC#C)C=C1	TCCGAC	CuAAC	scaffold_1	product_1
CNC1=CC=C(OCC#C)C=C1	CAAGTG	CuAAC	scaffold_1	product_1
...	...	...	...	...

Required columns:

- `codon`: DNA barcode sequence identifying this building block
- `smiles`: Chemical structure in SMILES notation (required for enumeration)
- `reaction`: Name of the reaction using this building block (required for enumeration)
- `reactant`: Name of the compound this building block reacts with (required for enumeration)
- `product`: Name of the product formed (required for enumeration)

Note: If you only want to perform demultiplexing without library enumeration, only the `codon` column is required.

2.2 Create the `reactions` sheet Define all chemical reactions used during library synthesis using SMARTS notation.

Table 9 | Reaction SMARTS templates

name	smirks
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
Suzuki	[cX3:1][I].[#6:2][BX3]»[cX3:1][#6:2]

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

2.3 Create the `compounds` sheet Define any additional chemical compounds (scaffolds, reagents) used in the reactions.

Table 10 | Compound definitions

name	smiles
scaffold_1	Ic1ccc(CC(N=[N+]=[N-])C(O)=O)cc1
scaffold_2	[N-]=[N+]=NC(C(O)=O)Cc1cc(I)ccc1
scaffold_3	[N-]=[N+]=NC(C(O)=O)Cc1c(I)cccc1
scaffold_4	NC(Cc1ccc(I)cc1)C(=O)O
scaffold_5	NC(Cc1cccc(I)c1)C(=O)O
scaffold_6	NC(Cc1ccccc1I)C(=O)O

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.

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: If the command fails, verify that:

- All required sheets are present in your Excel file
- Column names match exactly as specified in the tables above
- DNA sequences contain only valid nucleotide characters (A, C, G, T)
- SMILES strings are valid chemical structures

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.

Step 4 | Enumerate library compounds from building blocks Generate all possible chemical structures by combining building blocks according to the defined reaction schemes:

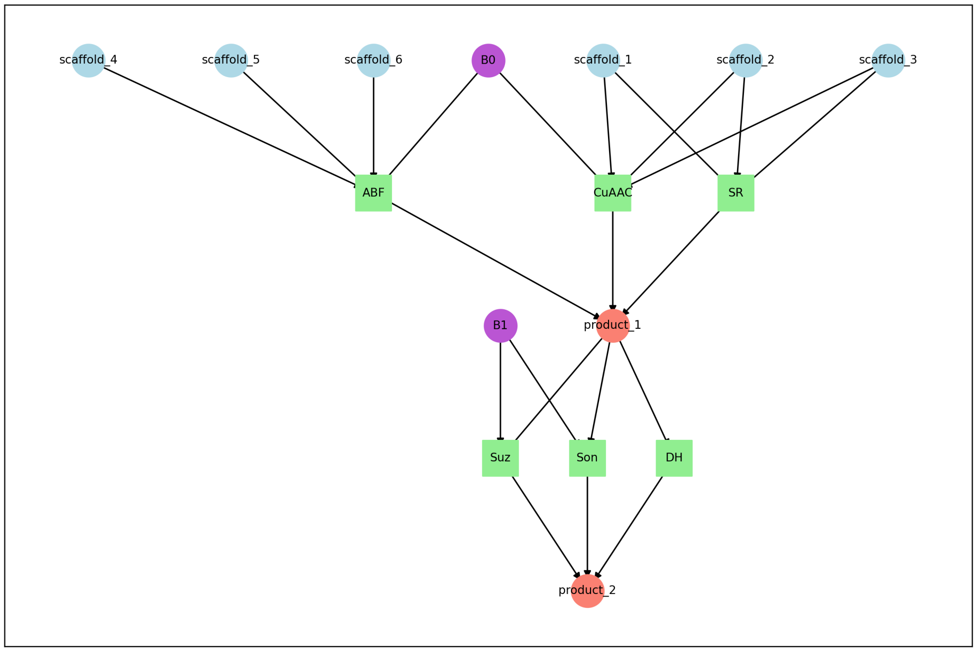
```
delt-hit library enumerate --config_path=/path/to/config.yaml
```

This process:

1. Constructs 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 scheme



Performance note: Enumeration speed is approximately 1000-10000 compounds/second depending on reaction complexity. For a 1 million compound library, expect 2-10 minutes processing time.

TROUBLESHOOTING: If enumeration fails:

- Check that all reaction SMARTS are valid
- Verify that building block SMILES are correctly formatted
- Ensure reactant and product names match between building blocks and compounds sheets

Step 5 | Calculate molecular properties and descriptors Compute comprehensive molecular descriptors for drug-likeness assessment and chemical space characterization:

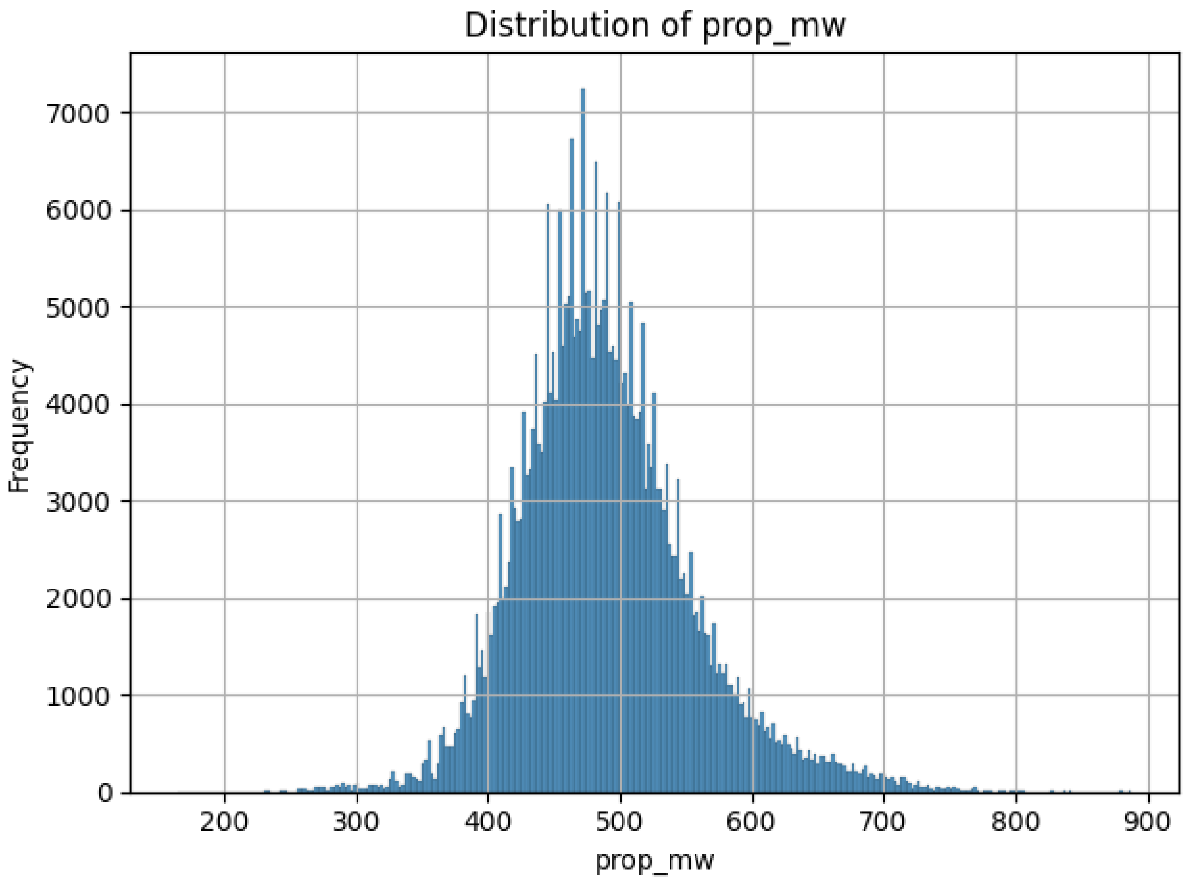
```
delt-hit library properties --config_path=/path/to/config.yaml
```

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



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 tasks.

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. 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

Recommended settings:

```yaml
structure:
  S0:
    type: selection
    max_error_rate: 0.0    # Perfect match required for multiplexing barcodes
    indels: false
  C0:
    type: constant
    max_error_rate: 1.01   # Allow moderate errors in constant regions
    indels: true
  B0:
    type: building_block
    max_error_rate: 0.0    # Perfect match required for building block identification
    indels: false
```

CRITICAL STEP: Selection and building block barcodes should require perfect matches to ensure accurate identification. Constant regions can tolerate higher error rates since they serve primarily as spacers.

Step 8 | Prepare and execute sequence demultiplexing Generate the demultiplexing scripts and execute the analysis:

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

# Make the script executable
chmod u+x /path/to/experiment/demultiplex/cutadapt_input_files/demultiplex.sh

# Execute demultiplexing (this step takes the longest)
/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
4. **Count aggregation**: Tabulation of reads per compound per selection

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
- `selections/*/counts.txt`: Raw count tables for each selection (generated in Step 9)

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

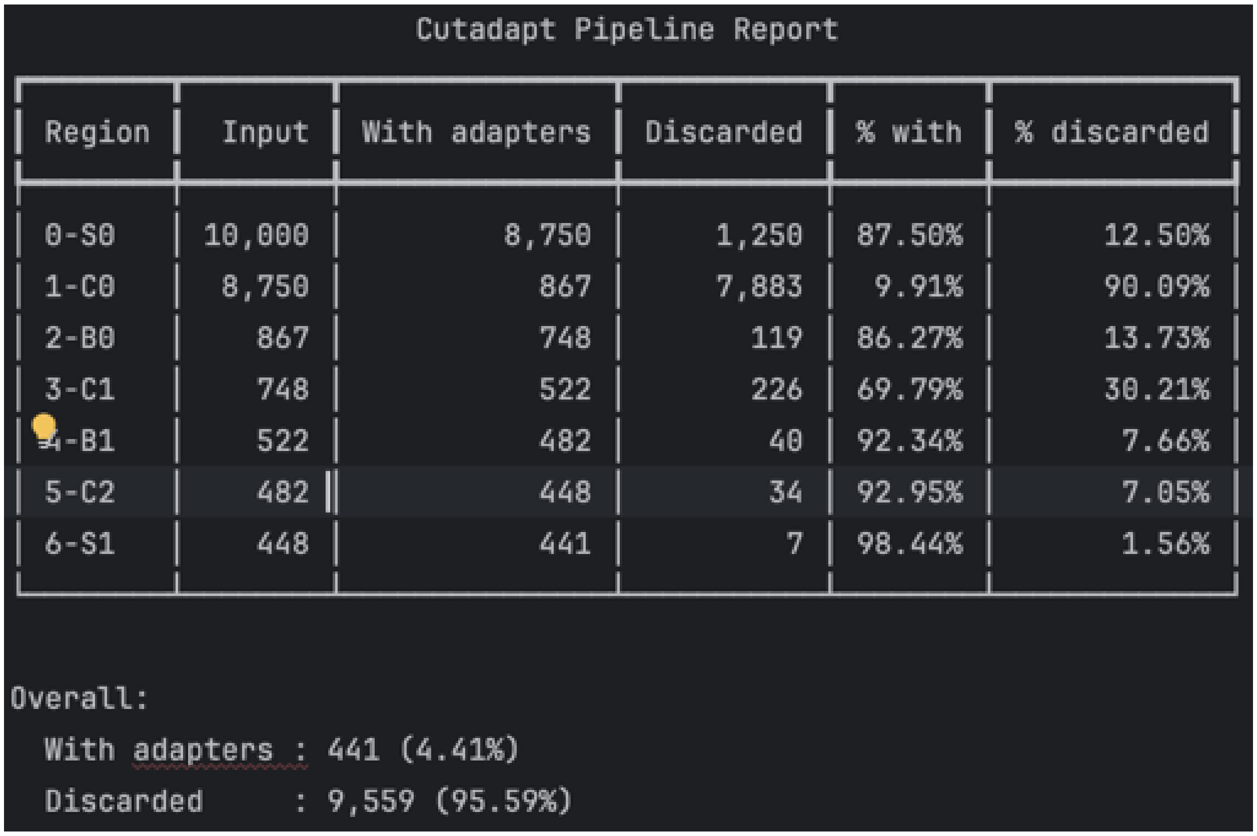
Step 10 | Generate quality control reports Assess demultiplexing quality with comprehensive statistics and visualizations:

```
# 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



```
                    Cutadapt Pipeline Report

┏━━━━━━━━┳━━━━━━━━┳━━━━━━━━━━━━━━━┳━━━━━━━━━━━┳━━━━━━━━┳━━━━━━━━━━━━━┓
┃ Region ┃  Input ┃ With adapters ┃ Discarded ┃ % with ┃ % discarded ┃
┡━━━━━━━━╇━━━━━━━━╇━━━━━━━━━━━━━━━╇━━━━━━━━━━━╇━━━━━━━━╇━━━━━━━━━━━━━┩
│ 0-S0   │ 10,000 │         8,750 │     1,250 │ 87.50% │      12.50% │
│ 1-C0   │  8,750 │           867 │     7,883 │  9.91% │      90.09% │
│ 2-B0   │    867 │           748 │       119 │ 86.27% │      13.73% │
│ 3-C1   │    748 │           522 │       226 │ 69.79% │      30.21% │
│ 4-B1   │    522 │           482 │        40 │ 92.34% │       7.66% │
│ 5-C2   │    482 │           448 │        34 │ 92.95% │       7.05% │
│ 6-S1   │    448 │           441 │         7 │ 98.44% │       1.56% │
└────────┴────────┴───────────────┴───────────┴────────┴─────────────┘


Overall:
  With adapters : 441 (4.41%)
  Discarded     : 9,559 (95.59%)
```

Quality assessment criteria:

- **Read retention**: >70% of input reads should be successfully assigned
- **Barcode coverage**: Most building blocks should have >0 reads
- **Selection consistency**: Similar read counts across replicates

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

- Incorrect barcode sequences in configuration
- Poor sequencing quality
- Unexpected adapter sequences
- Library degradation during selection

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 Edit your configuration file to specify which selections to compare. Add an `analyses` section defining comparison groups:

```yaml
# Add to config.yaml
analyses:
  analysis-1:
    selections:
      - CA_P1
      - CA_P2
      - CA_N1
      - CA_N2
      - CA_A1
      - CA_A2

  analysis-2:
    selections:
      - CA_P1
      - CA_P2
      - CA_N1
      - CA_N2
```

Note: Each analysis can include different subsets of selections. The `group` assignments from your experimental design (Step 1.2) determine which selections are compared (e.g., ”protein” vs ”no_protein”).

Step 12 | Perform enrichment analysis with statistical methods DELT-Hit supports two statistical approaches for identifying enriched compounds:

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

```
# Generate R script for counts-based analysis
delt-hit analyse enrichment \
  --config_path=/path/to/config.yaml \
  --name=analysis-1 \
  --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 fold-change 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/config.yaml \
  --name=analysis-1 \
  --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 biological replicates (minimum 2 per group recommended)
- Provides normalized count tables (CPM - counts per million)

Expected outputs (both methods):

- `stats.csv`: Complete statistics for all library members with columns:
  - `code_1`, `code_2`: 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

Phase 5: 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
```

This command opens a web-based interface (typically at `http://localhost:8050`) providing:

Dashboard features:

- **Experimental metadata display**: Selection conditions, target information, and experimental parameters
- **Count distribution visualization**: Histograms and summary statistics for the selected condition
- **Interactive filtering**: Explore specific regions of the count distribution
- **Building block analysis**: Compare counts across different building block positions
- **Multi-selection comparison**: Load different selections to compare enrichment patterns

Figure 4 | Interactive dashboard interface. The dashboard provides real-time visualization of count data with interactive filtering capabilities. The top panel shows experimental metadata, 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. Use the histogram to identify outliers or unusual count distributions
3. Apply filters to focus on specific count ranges
4. Export filtered data for further analysis

Note: The dashboard displays raw counts from demultiplexing. For statistical comparisons and hit identification, use the enrichment analysis results from Phase 4.

Troubleshooting

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

Table 11 | Common issues and solutions

Problem	Possible Cause	Solution
Low demultiplexing efficiency (<50%)	High error rates, incorrect barcode sequences	Increase `max_error_rate` paramete
Memory errors during enumeration	Large library size, insufficient RAM	Reduce library size by filtering buil
R integration failures	Missing R packages, PATH issues	Reinstall R packages using BiocMa
Empty hit lists	Stringent statistical thresholds	Adjust FDR cutoffs (try 0.1 or 0.2
Chemical structure errors	Invalid SMILES, incorrect reaction SMARTS	Validate building block structures u
”Command not found” errors	Conda environment not activated	Run `conda activate delt-hit` befo
Permission denied for shell scripts	Executable permission not set	Run `chmod u+x script.sh` before ex
Poor replicate correlation	Batch effects, technical variability	Review selection protocol for consis

Performance optimization guidelines

Memory requirements by library size:

- **Small libraries** (<1M compounds): 8 GB RAM sufficient
- **Medium libraries** (1-50M compounds): 16-32 GB RAM recommended
- **Large libraries** (>50M compounds): 32-64 GB RAM required
- **Very large libraries** (>500M compounds): High-performance computing recommended

Processing speed optimization:

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

Quality control thresholds

Demultiplexing quality indicators:

- **Excellent**: >90% initial read retention, >85% barcode assignment
- **Good**: 80-90% initial retention, 75-85% assignment
- **Acceptable**: 70-80% initial retention, 60-75% assignment
- **Poor**: <70% retention or <60% assignment (requires troubleshooting)

Statistical analysis quality indicators:

- **Library coverage**: 10-90% of theoretical compounds detected (depends on selection stringency)
- **Replicate correlation**: $R^2 > 0.7$ for biological replicates ($R^2 > 0.9$ for technical replicates)
- **Statistical power**: >100 reads per compound for reliable enrichment detection
- **Hit rate**: 0.1-2% of library showing significant enrichment (typical for specific targets)

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	5-10 min	10-15 min	15-30 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
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, <10M reads
[2]Medium dataset: 100K-10M compounds, 10-100M reads
[3]Large dataset: >10M compounds, >100M reads

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
- SSD storage reduces I/O time by 2-3× compared to traditional hard drives
- Very large datasets (>500M compounds or >500M reads) may require proportionally longer times

---

Anticipated results

Output file structure

DELT-Hit generates a comprehensive, standardized output hierarchy organized by analysis phase:

```
project_name/
├── config.yaml                      # Master configuration file
├── library.parquet                  # Complete enumerated library
├── reaction_graph.png               # Synthetic scheme visualization
│
├── properties/                      # Molecular descriptors
│   ├── properties.parquet          # Library with calculated properties
│   └── prop_*.png                  # Property distribution plots
│
├── representations/                 # Machine learning representations
│   ├── morgan.npz                  # Morgan fingerprints
│   └── bert.npz                    # BERT embeddings
│
├── demultiplex/                    # Sequence processing results
│   ├── cutadapt_input_files/      # Input files for Cutadapt
│   │   ├── demultiplex.sh         # Demultiplexing script
│   │   └── *.fasta                # Adapter sequences
│   └── cutadapt_output_files/     # Cutadapt results
│       ├── *.cutadapt.json        # Processing statistics
│       └── reads_with_adapters.gz # Processed reads
│
```

```
│
├── selections/                       # Per-selection count tables
│   ├── CA_P1/
│   │   └── counts.txt               # Counts for selection CA_P1
│   ├── CA_P2/
│   │   └── counts.txt               # Counts for selection CA_P2
│   └── ...
│
├── qc/                               # Quality control outputs
│   ├── report.txt                   # Text-based QC report
│   └── *.png                        # QC visualization plots
│
└── analyses/                         # Statistical analysis results
    ├── analysis-1/
    │   ├── counts/                  # Counts-based analysis
    │   │   ├── stats.csv           # Complete statistics
    │   │   ├── hits.csv            # Top hits
    │   │   └── correlation_*.png   # Replicate correlation plots
    │   └── edgeR/                   # edgeR-based analysis
    │       ├── enrichment_stats.csv  # Complete statistics with p-values
    │       ├── enrichment_hits.csv   # Significant hits (FDR < 0.05)
    │       ├── *.csv               # Normalized counts per selection
    │       └── correlation_*.png   # Replicate correlation plots
    └── analysis-2/
        └── ...
```

Chemical library characterization

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

Expected library characteristics for a representative 2-cycle DEL with 1000 building blocks per position:

- **Library size**: ~1 million unique structures
- **Molecular weight range**: 200-800 Da (pharmaceutically relevant space)
- **Chemical diversity**: Tanimoto similarity <0.4 between randomly selected pairs
- **Drug-likeness**: 70-90% compounds passing Lipinski's Rule of Five
- **Structural complexity**: Mean heavy atom count 15-35 atoms

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, consistent with oral drug space.

Chemical space analysis:

The reaction graph (**Figure 1**) 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

Sequence processing performance

Successful demultiplexing indicates high-quality library and sequencing data.

Table 13 | Expected demultiplexing performance metrics

Metric	Excellent	Good	Acceptable	Poor
Initial read retention	>90%	80-90%	70-80%	<70%
Barcode assignment	>85%	75-85%	60-75%	<60%
Library coverage	>50%	30-50%	10-30%	<10%
Error rate per position	<2%	2-5%	5-10%	>10%
Mean reads per compound	>1000	100-1000	10-100	<10

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:

- **Hit rate**: 0.1-2% of library members show significant enrichment (FDR < 0.05)
- **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

Table 14 | Representative statistical analysis results

Analysis Type	Hits Identified[1]	Median LogFC[2]	FDR Threshold	Replicate Correlation
Enzyme target	156 (0.15%)	3.2	0.05	0.82
PPI disruption	89 (0.09%)	4.1	0.01	0.79
Membrane receptor	234 (0.23%)	2.8	0.05	0.85

[1]Number and percentage of library members with significant enrichment
[2]Log2 fold-change for significantly enriched compounds

Output file contents:

enrichment_stats.csv (edgeR method):

```
code_1,code_2,logFC,logCPM,PValue,FDR
24,427,4.23,8.56,1.2e-08,3.4e-05
104,205,3.87,7.92,3.5e-07,4.2e-04
...
```

Column descriptions:

- `code_1`, `code_2`: Building block indices identifying the compound
- `logFC`: Log2 fold-change (positive = enriched in protein selections)
- `logCPM`: Log2 counts per million (measure of abundance)
- `PValue`: Raw p-value from statistical test
- `FDR`: False discovery rate corrected p-value (multiple testing correction)

Interpreting hit lists:

1. **Rank by FDR**: Most statistically significant hits first
2. **Filter by logFC**: Consider only substantially enriched compounds (logFC > 1, i.e., $>$2-fold)
3. **Examine replicate correlation**: Verify consistent enrichment across replicates
4. **Cross-reference with properties**: Check drug-likeness of top hits

Replicate correlation plots show the consistency between biological replicates. High correlation ($R^2 > 0.7$) indicates:

- 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

Example applications:

```python
import numpy as np
from scipy import sparse
from sklearn.ensemble import RandomForestClassifier

# Load Morgan fingerprints
fps = sparse.load_npz('representations/morgan.npz')

# Load hit annotations (e.g., from enrichment analysis)
# hits = pd.read_csv('analyses/analysis-1/edgeR/enrichment_hits.csv')

# Train machine learning model
# model = RandomForestClassifier()
# model.fit(fps, labels)
```

Cheminformatics pipelines:

Integration with RDKit and other tools for:

- **Structure-activity relationship (SAR) analysis**: Automated identification of structural motifs associated with activity
- **Virtual screening**: Use enriched compounds to search larger chemical databases
- **Lead optimization**: Multi-parameter optimization considering potency, selectivity, and ADMET properties
- **Pharmacophore modeling**: Identify common 3D features among active compounds

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
- **Batch effect correction**: Advanced normalization methods for complex experimental designs

Quality assurance and validation

DELT-Hit implements comprehensive validation throughout the pipeline.

Built-in validation features:

1. **Chemical structure verification**
   - Automated detection of invalid SMILES notation
   - Stereochemistry consistency checks
   - Unusual valence or charge state warnings
   - Expected: >99.5% valid structures from well-defined building blocks
2. **Reaction template validation**
   - SMARTS syntax checking
   - Product structure prediction
   - Reactant-product consistency verification
   - Expected: All reactions produce valid molecular graphs
3. **Statistical model diagnostics**
   - Dispersion estimates for count data
   - Mean-variance relationship plots (edgeR)
   - Residual analysis
   - Expected: Overdispersion parameter estimates between 0.1-1.0
4. **Cross-platform reproducibility**
   - Consistent results across Linux, macOS, and Windows
   - Deterministic algorithms (same input $\rightarrow$ same output)
   - Version-controlled dependencies
   - Expected: <5% coefficient of variation between independent analyses

Validation best practices:

- Always inspect quality control plots before interpreting results
- Verify building block structures using chemical drawing software
- Test reaction SMARTS on small subsets before full enumeration
- Compare results between counts and edgeR methods for consistency
- Examine top hits manually for chemical plausibility

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+ssh://git@github.com/DELTechnology/delt-core.git@paper`
- **Documentation**: Comprehensive user guides, API reference, and tutorials at the repository
- **Issue tracking**: Community support and bug reporting via GitHub Issues
- **Continuous integration**: Automated testing across multiple platforms and Python versions

Tutorial datasets:

Example datasets for protocol validation are available on Figshare:

- **NF-selection-campaign.fastq.gz**: Representative dual display screening data
- **NF-selection-campaign.xlsx**: Complete library and experimental definitions
- URLs provided in Table 3

Community resources:

- **Discussion forum**: GitHub Discussions for user questions and best practices
- **Example workflows**: Complete analysis notebooks in the repository
- **Custom modifications**: Fork the repository for lab-specific customizations

---

Author contributions

A.M. conceived the project, designed the software architecture, and implemented core algorithms in collaboration with G.H. A.L. created and maintained the configuration files and testing frameworks. A.G. performed selection experiments and helped with results interpretation. 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. This work was supported by [funding information to be added].

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).