ARL-TR-10236 ● Dec 2025

From Bench to Breakthrough: A Findable, Accessible, Interoperable, Reusable (FAIR)-Compliant Toolkit for Synthetic Biology Automation

by Jeanet V. Mante, Randi M. Pullen, and Randall A. Hughes

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-10236 ● DEC 2025

From Bench to Breakthrough: A Findable, Accessible, Interoperable, Reusable (FAIR)-Compliant Toolkit for Synthetic Biology Automation

Jeanet V. Mante, Randi M. Pullen, and Randall A. Hughes
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
December 2025	Technical Report	1 September 2023	30 September 2025

4. TITLE AND SUBTITLE
From Bench to Breakthrough: A Findable, Accessible, Interoperable, Reusable (FAIR)-Compliant Toolkit for Synthetic Biology Automation

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Jeanet V. Mante, Randi M. Pullen, and Randall A. Hughes

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLA-HD
Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-10236

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID: Jeanet V. Mante, 0000-0002-1450-5638

14. ABSTRACT
Automation is transforming synthetic biology, but without interoperable tools and standardized metadata, its benefits are undermined by fragmented workflows and irreproducible data. This work introduces a comprehensive suite of modular software tools that embed Findable, Accessible, Interoperable, Reusable (FAIR) principles into everyday laboratory operations. The toolkit supports sequence domestication and annotation, generates validated combinatorial part assembly instructions for acoustic liquid handlers like the Echo 525, automates liquid handling tasks across systems such as the Tecan Fluent and Formulatrix Mantis, and integrates high-throughput colony picking with the Singer Instruments PIXL. It also includes design-of-experiments modules for both binary and continuous variables, analytical workflows for expression factor importance, clustering algorithms for promoter classification, and gene identification tools for ribonucleic acid expression analysis. Together, these tools convert conceptual designs into executable robotic protocols while capturing machine-readable metadata at every step. Laboratories that fail to adopt such systems risk becoming bottlenecks in the scientific process, where data is trapped in proprietary formats, collaboration is stifled, and critical insights are lost in translation. In contrast, this framework enables rapid iteration, reproducibility, and cross-platform compatibility, making it not just a technical upgrade but a safeguard against stagnation in modern biological research.

15. SUBJECT TERMS
standardization, high-throughput, design of experiments, bioinformatics, statical analysis, data, siloes, artificial intelligence, AI, Biological and Biotechnology Sciences

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	66

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Jeanet V. Mante	(520) 691-5393

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

Modern defense biotechnology increasingly depends on rapid, design–build–test–learn (DBTL) cycles to address missions ranging from supply chain resilience to environmental remediation and emerging threat response. However, the ability to execute these cycles efficiently is constrained by the scarcity of Findable, Accessible, Interoperable, Reusable (FAIR) biological data. While the DoD benefits from an extensive research network, much of its experimental output remains siloed, locked into laboratory-specific formats, or insufficiently annotated for reuse. This slows the integration of discoveries into deployable capabilities and hinders the use of AI models that could accelerate everything from material development to biosurveillance.

A central challenge is that FAIR compliance is often perceived as an extra burden, with limited short-term incentive for the originating laboratory. The path forward is to embed FAIR data capture directly into the tools and workflows researchers already depend on, especially at the robot-to-robot integration level where interoperability failures currently fragment automated pipelines. Here, we present a suite of modular software tools designed to bridge instrumentation gaps, harmonize outputs across diverse platforms, and automatically generate machine-readable, standards-compliant datasets without disrupting ongoing operations.

To support scalable, interoperable, and high-throughput biotechnology, our tool suite delivers targeted enhancements across automation, data capture, and experimental design (Figure 1). Sequence domestication and assembly planning modules accelerate the build phase by streamlining DNA synthesis and cloning, while integration with robotic platforms (including Beckman Coulter Echo 525 [2025], Formulatrix Mantis [2025], Tecan Fluent 1080, and Singer Instruments PIXL) enables automated liquid handling and colony picking, reducing manual errors and increasing throughput. FAIR-compliant data capture is embedded directly into these workflows, ensuring that metadata is standardized and machine-readable from the outset. Automated format conversions and structured outputs allow seamless aggregation across laboratories and agencies, unlocking the potential for AI-driven analysis and predictive modeling. Finally, design of experiments (DOE) tools and analysis modules support smarter testing, from optimizing strain and media combinations to identifying key regulatory elements, dramatically accelerating the learn phase and enabling rapid iteration across DBTL cycles.

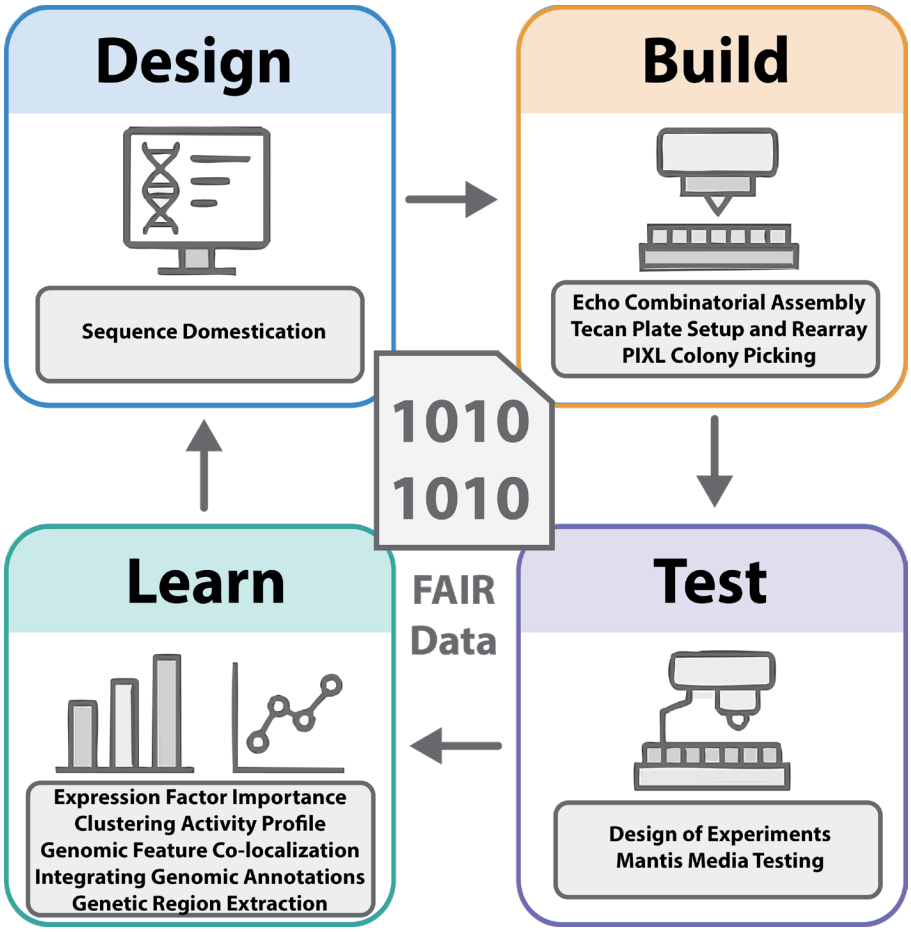

Figure 1. Overview of tools integrating in the DBTL pipeline. Modular tools bridge instrumentation gaps, facilitate high-throughput and cross-platform workflows, and automatically capture standardized data to accelerate iterative experimental cycles. The tools enable going from genetic parts found in literature, to DNA synthesis, combinatorial assemblies, transformation of organisms, testing constructs, and then analyzing the results.

2. Army Impact

Brig. Gen. Chris Hackler, U.S. Army Combat Capabilities Development Command (DEVCOM) Deputy Commanding General stated, "the Department of Defense must move at the speed of technology to maintain a competitive edge in the rapidly evolving global security landscape" (Hackler 2025).

Synthetic biology is rapidly becoming a cornerstone of Army modernization, enabling breakthroughs in organism domestication, biosensor development, and biomanufacturing. These capabilities are essential not only for operational readiness and threat response, but also for ensuring environmental resilience and securing critical supply chains. To meet these demands, the Army must accelerate its DBTL cycles while producing data that is immediately reusable and interoperable. Our toolkit addresses this need by embedding automation and FAIR

data standards directly into core workflows, allowing biological discoveries to be translated into deployable solutions with speed and precision. By harmonizing outputs across diverse instruments and platforms, the system supports scalable biomanufacturing pipelines, reduces resource waste, and ensures continuity of operations—even in austere or contested environments.

For the DoD, this approach delivers strategic advantages that extend well beyond the lab. It enables seamless collaboration across agencies and allied partners by standardizing data at the point of generation, not just during aggregation. This unlocks the ability to re-task research assets in real time, respond to emerging threats with agility, and build large, diverse datasets that fuel AI-driven models for predictive biosurveillance, advanced manufacturing, and mission-specific strain design. The tools also support modularity and technology transition, aligning with broader DoD priorities in scalable innovation and supply chain resilience. By reducing manual labor, minimizing experimental redundancy, and improving data quality, the system enhances both scientific productivity and operational impact—ensuring that synthetic biology investments deliver tangible returns across defense missions.

3. Methods

Our toolkit spans multiple DBTL stages. Sequence domestication and assembly planning modules split and annotate large sequences for synthesis while storing contextual metadata. Beckman Coulter's Echo 525 (acoustic liquid handler) combinatorial assembly instructions are generated for golden gate and other cloning procedures. This tool validates assemblies and tracks assembly and sequence metadata. Formulatrix Mantis (tipless nanolitre liquid handler), and Tecan Fluent 1080 (liquid handler) integration tools convert between vendor-specific formats and automated liquid handling tasks while logging execution details. The Singer Instruments PIXL colony picker integration enables high-throughput spot plating of transformations and picking of biological replicates. DOE tools support both binary factor screening for strain development and continuous-variable optimization, producing balanced experimental layouts for media optimization. Analysis modules quantify expression factor importance for both binary and continuous datasets, identifying key determinants of system performance. Clustering workflows use min–max scaling and weighted Euclidean distances to classify response surfaces of promoters across conditions. Gene ID and upstream region selection workflows make ribonucleic acid (RNA) expression analysis and predicted promoter verification simpler.

3.1 App Setup

All tools have been approved for public release via Form 1.[*] Each tool is set up as an independent tool that can be used in several ways:

- By calling the tool from the command line
- By using the front-end Hypertext Markup Language (HTML) form by calling the tool via a Flask application that serves as the Application Programming Interface (API)

3.2 Sequence Domestication Algorithm

Assumptions

- Sequences contain only adenine, thymine, guanine, and cytosine (A, T, G, and C), no ambiguous nucleotides.
- No further sequence optimization is required other than removal of enzyme sites.

Inputs

- Sequence table: Name, Sequence, Part Type
- Codon change table: original → synonymous codons with constraints
- Cloning template table with flanking regions
- Ligation fidelity table for overhang pairs

Procedure

1) Load input data.
   a) Sequence table (Name, Sequence, Part Type)
   b) Codon change table (original → new codons with constraints)
   c) Cloning template table (with flanking sequences)
   d) Overhang ligation fidelity table (e.g., from NEB PLOS One dataset[†])
2) For each sequence:
   a) Clean the sequence.
      i) Remove white space.
      ii) Convert to uppercase.
      iii) Validate that it only contains A, T, G, C and is greater than or equal to 6 base pairs (bp).
   b) Detect internal restriction sites using a predefined enzyme list.
   c) If internal sites are found:
      i) And if silent codon changes are allowed:

[*] https://helpdesk.arl.army.mil/tdm?id=tdm_home
[†] https://doi.org/10.1371/journal.pone.0238592

(1) Use codon table to propose alternative synonymous codons.
(2) Prioritize changes with minimal disruption (use "Maybe"/"Yes"/"No" logic).
(3) Avoid introducing other undesired sites (e.g., BsaI ↔ BsmBI).

ii) Otherwise:

(1) Directly mutate single bases to their complements (A↔T, G↔C).

iii) Perform up to three passes to remove all sites.

d) Add standard flanking regions using cloning templates.
   i) Locate placeholder region (e.g., "RRRR…") in template.
   ii) Replace placeholder with domesticated sequence.
e) If enabled, design colony polymerase chain reaction (PCR) primers upstream of the insert.
   i) Identify a region with valid length, nucleotide G or C flanks, and melting temperature.
   ii) Reverse-complement primer and record metadata.
f) Decide on synthesis strategy based on length and user settings.
   i) If below threshold: treat as single-stranded nucleotide
   ii) If short: optionally extend to minimum synthesis length
   iii) If above max length: proceed to split into blocks
b) Splitting into blocks
   i) Desired block size is specified (defaults are based on the Integrated DNA Technologies [IDT] ranges for gblocks [125–3000], eblocks [300–1500], and oligonucleotides [10–90 bp])
   ii) Identify split points that preserve minimum block size constraints.
   iii) At each candidate split site:
      (1) Extract 4 bp "sticky end" (overlap region).
      (2) Check if overhang has already been used (template or previous block).
      (3) Check New England Biolabs (NEB) ligation table to ensure low off-target ligation likelihood.
   iv) If valid overhang found:
      (1) Append enzyme site and overhang to left block.
      (2) Prepend overhang and enzyme site to right block.
      (3) Recurse on right block using same logic.
   v) Raise an error if no valid overhang is found that meets all constraints.
c) Design Primers (if extension PCR is required)
   i) Identify central overlap meeting melting temperature and GC constraints.
   ii) Split sequence into forward/reverse strands and locate primers on each.
   iii) Store melting temps and reverse complements.

d) Record processing steps:
    i) Which cut sites were removed
    ii) Whether template flanks were added
    iii) Block/primer sequences
    iv) Synthesis method assigned (e.g., "Split to GBlock")
e) If Synthetic Biology Open Language (SBOL) output is requested:
    i) Create modular SBOL components for each sequence (left flank, insert, right flank).
    ii) Use Sequence Ontology (SO) terms for part types (e.g., promoter, coding DNA sequence [CDS]).

Outputs

- Create logging file with all processing steps.
- Export IDT-ready Excel files for nucleotides, eBlocks, and gBlocks.
- Write SBOL file (if enabled).
- Write README explaining contents.

3.3 Echo (Combinatorial Assembly) Algorithm

Assumptions

- For DNA assemblies, only one part from each input DNA is required.

Inputs

- Source plate file with reagent metadata (plate, well, name, volume, molarity, sequence, and type)
- Composition file specifying desired DNA assemblies using parts, reagents, vectors, and enzymes

Procedure

1) Load and validate the source plate and composition tables.
    a) Ensure required columns are present.
    b) Check for consistent plate types and molarities.
2) Parse combinatorial assemblies.
    a) Identify all "Part" and "Reagent" columns.
    b) For each row in the composition file, expand all combinations of parts and reagents.
    c) Generate a flat list of all assembly reactions with metadata.
3) Validate source availability.
    a) Confirm all required reagents and parts exist in the source plate file.
4) Simulate restriction-ligation assemblies.

a) For each reaction with parts:
    i) Perform in silico digestion of vector and parts using the specified enzyme.
    ii) Identify valid assemblies by testing all permutations for matching overhangs.
    iii) Annotate reactions with sequence, assembly validity, and part order.
5) Optionally, generate SBOL representations:
    a) For each valid assembly, construct an SBOL2 or SBOL3 document.
    b) Annotate with part boundaries and roles based on SO ontology.
6) Compute liquid handling instructions.
    a) For each reaction:
        i) Determine required volume for each part and reagent based on molarity or defaults.
        ii) Simulate volume withdrawal from available wells.
        iii) Add water to reach the specified final volume.
        iv) Generate a list of Echo-compatible transfer instructions.
7) Output results:
    a) If volume estimation is requested, write required volume per reagent.
    b) Otherwise, write:
        i) Updated plate file with remaining volumes
        ii) Sequence outputs and Echo instruction files
        iii) Log of reaction contents and validity

Outputs

- Echo input file(s) with transfer instructions
- SBOL document (if enabled)
- Reaction logs, annotated sequences, and optional volume requirement summary

3.4 Tecan Algorithm (Plate Setup and Rearray)

Inputs

- Source plate file (well positions, reagent names, and volumes)
- Transfer plan file (reagent names, target plate, optional target wells, and required volumes)
- Optional: Destination plate file (current contents of target wells)
- Flexible Channel Arm (FCA) tip configuration (size, min–max volume constraints)

Procedure

1) Data Cleaning
    a) Remove commented rows/columns.
    b) Infer whether destination wells are prespecified.
2) Initialize internal data structures.
    a) Create well-mapped dictionaries for all output plates.
    b) Load and index source wells by reagent name.
    c) Initialize plate-specific movement lists from the transfer plan.
3) For each requested transfer:
    a) Determine destination well
        i) If specific well is provided: use it
        ii) ELSE if top-up mode is enabled:
            (1) Locate any existing well for the reagent on that plate.
            (2) If found, compute how much additional volume is needed.
            (3) If not found, select the first empty well.
        iii) ELSE (no top-up): select the next empty well
    b) Calculate how to fulfill the volume request from available source wells.
        i) Sort available source wells for that reagent.
        ii) Select the smallest FCA tip that can accommodate the volume.
        iii) Split across multiple wells if needed.
        iv) Track actual volumes withdrawn and update source plate.
    c) Record each transfer step with:
        i) Source plate/well, destination plate/well
        ii) Volume transferred
        iii) Tip size used
    d) Update destination well content to reflect volume delivered.
        i) If top-up mode and well was previously filled, accumulate volume.
        ii) ELSE, create a new entry for the reagent in that well.
4) Generate Outputs
    a) Write updated source plate table.
    b) Write updated destination plate table.
    c) Write human-readable log file (if top-up logic was used).
    d) Write robotic transfer instructions.
        i) One file per tip size (if splitting enabled)
        ii) Or a single combined file
        iii) Format instructions according to Tecan Fluent worklist syntax.
5) Append a README file explaining output structure and usage.

Outputs

- Updated Excel files for source and destination plates

- One or more robotic instruction files (.txt, .xlsx)
- Processing log (optional)
- README file

3.5 PIXL (Transformation and Colony Picking) Algorithm

Assumptions

- Spot plating was carried out in a column-by-column manner with the first four columns to a dilution.

Inputs

- One or more plate files with colony detection data (Tab Separated Values [TSV] format)
- Plate layout metadata
  - Number of sectors per plate (columns × rows)
  - Number of replicate target plates
  - Target plate type and format (e.g., 96, 384, or 1536 wells)
  - Optional sort column (e.g., Colony_Brightness)

Procedure

1) For each source plate:
   a) Read colony data.
   b) Remove commented rows and columns (those starting with "#").
   c) Assign source plate name from filename.
2) For each plate:
   a) If sorting is enabled:
      i) Sort colonies by specified column (e.g., brightness).
   b) For each sector (defined by row × column indices):
      i) Identify colonies from that sector.
      ii) Extract X/Y coordinates of colonies.
   c) For each replicate (up to the number of replicates specified):
      i) If a colony exists for this replicate:
         (1) Determine target well index based on placement mode.
            (a) Column-wise (default): fill by column
            (b) Row-wise (optional): fill by row
         (2) Convert index to alphanumeric well (e.g., A01, B12).
         (3) Record source plate, coordinates, target plate, and well.
3) Create header rows describing:
   a) Each target plate (with plate type and number of wells)

b) Each source plate (with type Society for Biomolecular Screening ["SBS"] and "NONE" for well count)

4) Combine header and rearray instructions into a single output table.

Outputs

- Tab-separated rearray instruction file suitable for input to Singer Instruments PIXL
  - Includes both source and target plate metadata
  - Each row encodes one transfer: source plate, colony position, target plate, and target well

3.6 DOE Binary Algorithm (Strain Optimization)

Assumptions

- All factors are binary (present/absent, high/low, mutated/wild type) or can be approximated as such.
- 3rd-order interactions are minimal.

Inputs

- List of binary input factors (e.g., gene names)
- Maximum number of experiments to be run
- Optional output directory

Procedure

1) Determine experiment design strategy.
   a) If the number of factors is less than or equal to 3, or if the full factorial design fits within the desired experiment count:
      i) Generate full factorial matrix using all factors (2^n combinations).
   b) Otherwise:
      i) Estimate minimum number of experiments needed to avoid aliasing of main effects and 2-factor interactions.
         (1) Solve inequality based on number of higher-order combinations (derivation of the inequality below).
      ii) Choose the largest number of experiments less than or equal to the maximum requested, that is, a power-of-two multiple of the minimum required.
2) Build design matrix:
   a) Assign a subset of the input factors as independent factors.
   b) Generate full-factorial matrix over independent factors (entries: –1 and 1).
   c) Identify all combinations of 3+ independent factors (used for aliasing).

d) For each dependent factor:
    i) Assign it as the product of a unique higher-order interaction (alias)
    ii) Add this calculated column to the matrix
3) Annotate and Label
    a) Convert each value in the matrix from {–1, 1} to {0, 1}.
    b) Construct column headers using gene names and aliasing expressions.
4) Output
    a) Save resulting design matrix to Comma Separated Values (CSV) file named "Discrete Output.csv" in the specified output directory.

Outputs

- A CSV of binary experiments to try

3.7 DOE Continuous Algorithm (Media Optimization)

Assumptions

- All factors are continuous quantitative values or close enough to be approximated as such.
- A single output must be minimized/maximized.

Inputs

- factors: DataFrame with factor names and their min, max, and step values
- max_runs: Integer, number of new experiments to generate
- Optional: Previous experimental data with an output column and target value

Procedure

1) Data Cleaning
    a) Remove commented rows and columns from factors and previous data (if provided).
    b) Determine the total number of discrete points in the design space based on Min, Max, and Step values.
2) Case A: Previous data is provided.
    2.1. If regression-based optimization is selected:
    a) Standardize prior factor values (X) and response (y).
    b) Fit a linear regression model.
    c) Generate Latin Hypercube Sampling (LHS).
    d) Rescale LHS points to factor bounds.
    e) Predict outputs for LHS points and select top N closest to target_value, where N is based on optimisation_split.

f) Inverse-transform selected points to original scale.

2.2. If local search around optima is selected:

a) Identify up to five previous points closest to target_value.
b) Around each selected point, generate a specified number of new points spaced by min_dist and within max_dist.
c) Use rejection sampling to ensure spacing between generated points.

2.3. Diversity supplement

a) If total selected points < max_runs, fill the remainder using rejection sampling across design space (ensuring min_dist separation from all other points).

2) Case B: No previous data is provided.
   a) If design space is small (below threshold):
      i) Use LHS to generate max_runs samples.
      ii) Rescale to factor limits.
   b) If design space is large:
      i) Use rejection sampling to generate max_runs well-spaced random points.
3) Postprocessing
   a) Round each factor in the generated design matrix to the nearest valid value defined by its step size.
   b) Format results as a DataFrame.
   c) If previous_data was provided, append new designs with blank output column to the bottom.
   d) Otherwise, add an empty "Output" column.
4) Output
   a) Save result as "Continuous Output.csv" to specified directory or current working directory.

Output

- CSV of experiments to try

3.8 Mantis Algorithm (Media Variation Testing)

Inputs

- Df: Dataframe where each row corresponds to a well and each column to a reagent volume
- plate_type: Number of wells (96, 384, or 1536)
- column_wise: Boolean; if true, fills plate column-by-column, else, row-by-row
- quadrant: Boolean; if true and plate_type = 384, fill quadrants sequentially

- mantis: Boolean; if true, output format is for Mantis, else, Tempest

Procedure

1) Data Cleaning
   a) If a column named "#" exists:
      i) Remove rows where # column contains "#."
      ii) Remove any columns whose names begin with "#."
2) Plate Formatting
   a) Select target reagent column values.
   b) Set plate dimensions based on plate_type.
   c) Pad data with zeros to fill all wells.
   d) If 384-well with quadrant filling enabled:
      i) Define quadrants as alternating rows and columns.
      ii) For each quadrant, fill wells in column-major order with corresponding subset of data.
   e) ELSE:
      i) Fill plate in column-major (column_wise = True) or row-major (False) order.
3) Output Matrix Generation
   a) Convert the filled plate array into tab-separated strings.
   b) For Tempest format: prepend reagent name, apply specific formatting.
4) File Assembly
   a) For Mantis format:
      i) Add header with version, plate type, and reagent count.
      ii) For each reagent column:
         (1) Add reagent metadata line (name, class, and constant pressure setting).
         (2) Append formatted plate matrix.
   b) For Tempest format:
      i) Add each reagent column: append formatted matrix with reagent name.
   c) Join all lines into a single string for saving to .dl.txt.

Output

- A txt file with the reagents to add to each well for Mantis/Tempest

3.9 Expression Algorithm Factor Importance and SHapley Additive exPlanations (SHAP) Analysis Pipeline

Inputs

- conditions_df: DataFrame of continuous predictor variables (factors)
- output_df: Single or multi-output DataFrame of measured responses
- multi_output: Boolean; determines whether output_df contains multiple outputs
- tree_depth: Decision tree maximum depth (controls complexity and overfitting risk)
- importance_cutoff: Minimum SHAP importance to trigger detailed analysis
- min_theory/max_theory: Theoretical or observed minimum and maximum values for each factor
- cluster_shap: Boolean; if true, perform feature clustering in SHAP plots
- cluster_cutoff: Dendrogram cutoff distance for clustering
- Output control arguments for directories, file naming, and graphics generation.

Procedure

1) Data Cleaning
   a) Remove any rows where the "#" column contains "#."
   b) Remove any columns with names starting with "#."
2) Min–Max Value Handling
   a) If theoretical_min_max = True:
      i) Read from min_max_df
   b) ELSE:
      i) Compute min and max from conditions_df.
3) Model Training and SHAP Analysis (condition_for_output)
   a) Split data into training (80%) and testing (20%) sets with a fixed random seed.
   b) Fit DecisionTreeRegressor with specified tree_depth.
   c) Use shap.Explainer to compute SHAP values for training data.
   d) Calculate mean absolute SHAP values for each feature and store in importance_df.
4) Feature Importance Visualization (if output_graphics = True)
   a) Sort features by importance into sorted_importance_df.
   b) Select features with importance above importance_cutoff.
   c) If cluster_shap = True, cluster SHAP values and plot a clustered bar chart.
   d) Save importance plots to Portable Network Graphics (png) and HTML (via mpld3).

5) Dependence Plots (if output_graphics = True)
    a) For each selected feature:
        i) Generate SHAP dependence scatter plots showing feature versus model output effect.
        ii) Save plots as .png and HTML.
6) Residual Analysis
    a) Predict outputs using full dataset.
    b) Compute residuals (actual – predicted).
    c) If output_graphics = True, plot residual histogram, save .png and HTML.
7) Model Performance Metrics
    a) Calculate and store:
        i) R-squared (R^2) score
        ii) Mean squared error (MSE)
        iii) Root MSE (RMSE)
8) Single-Feature Variation Analysis (if output_graphics = True)
    a) For each feature:
        i) Hold all others at mean values.
        ii) Vary the selected feature across its min–max range (100 points).
        iii) Predict model outputs.
        iv) Plot predicted output versus feature value; save .png and HTML.
    b) Reorder feature plots based on importance ranking.
9) Multi-Output Handling (continuous_overarching)
    a) For each output in output_df:
        i) Run generate_single_report without graphics to collect importance and metrics.
        ii) Merge results into a master Excel file.
    b) Combine all factors and output names into a flattened dataset; save as .xlsx.
    c) Append output name index to min_theory and max_theory and run full report with graphics.
10) Report Generation (generate_single_report)
    a) Load HTML template.
    b) Replace placeholders with:
        i) Sorted importance table
        ii) SHAP importance HTML
        iii) Residual histogram HTML
        iv) Model performance metrics (formatted as HTML list)
        v) Mean factor values table
        vi) Ordered single-feature variation plots
    c) Save report as .html.

Outputs

- Importance tables (CSV/XLSX)
- Performance metrics (XLSX)
- SHAP plots and dependence plots (PNG + HTML)
- Residual histogram (PNG + HTML)
- Single-factor variation plots (PNG + HTML)
- Combined multi-output summary file (XLSX, if applicable)
- Final HTML report per output

3.10 Expression Algorithm Binary Factor Analysis of Variance (ANOVA) and Interaction Analysis

Inputs

- df: DataFrame of binary-coded input factors + one output column
- output_col: Name of the output column in df
- significance_threshold: p-value cutoff for identifying significant terms
- test_interactions: Boolean; True to include two-way factor interactions
- output_to_cwd: Boolean; write output files to working directory if True
- out_dir: Alternative output directory if not writing to cwd
- input_file_name: String for file name tagging
- formatted_time: String timestamp for file naming

Procedure

1) Data Cleaning
   a) Remove rows where "#" column contains "#."
   b) Remove columns starting with "#."
   c) Validate output directory path.
   d) Separate factor_cols ← all columns except output_col.
   e) Remove interaction annotations from factor column names (e.g., "= interaction [...]").
2) n_way_anova_type_iii
   a) IF test_interactions = True:
      i) Generate all two-factor interaction terms (facA:facB).
      ii) Calculate how many interactions can be tested given dataset size.
      iii) Loop through subsets of interactions:
         (1) Fit ordinary least squares (OLS) model with subset interactions.
         (2) Perform Type III ANOVA.
         (3) Keep top-ranked interactions by p-value.

iv) Build final ANOVA formula: output_col ~ all main factors + chosen interactions.

b) ELSE:

i) Formula: output_col ~ all main factors

c) Fit OLS model, perform Type III ANOVA, sort terms by p-value.

d) any_sig_terms ← whether any p-value < significance_threshold.

e) Generate Pareto chart of sum-of-squares (save PNG, encode to base64).

f) Plot histogram of residuals (save PNG, encode to HTML).

g) Return ANOVA table, any_sig_terms, pareto_html, residuals_html.

3) Interaction Plots (if applicable)

a) IF test_interactions AND any_sig_terms:

i) Identify significant interactions (":" in term name).

ii) For each:

(1) Plot interaction_plot(facA, facB, output_col).

(2) Save PNG and store HTML string.

b) ELSE:

i) inter_plots_html ← []

4) Assumption Checks

a) IF any_sig_terms:

i) For each significant main factor:

(1) Run Levene's test on output_col between factor = 0 and factor = 1.

(2) Store results text.

5) Output Generation

a) Save ANOVA results table to Excel.

b) Load HTML report template.

c) Replace placeholders with:

(1) ANOVA table (HTML)

(2) Pareto plot (HTML/base64)

(3) Residuals histogram (HTML)

(4) Levene's test results (if any)

(5) Interaction plots (if any)

d) Save completed HTML report to out_dir.

e) Copy binary analysis ReadMe.txt to out_dir.

6) Cleanup

a) Close all matplotlib figures to free memory.

Outputs

- Tables
  - ANOVA results (Excel, HTML in report)
  - Levene's test results (HTML list in report, if applicable)
- Plots

    - Pareto chart of factor/interactions contribution (PNG, HTML in report)
    - Residuals histogram (PNG, HTML in report)
    - Interaction plots for significant interactions (PNG, HTML in report)
- Documentation
    - HTML report with embedded results and plots
    - ReadMe.txt with analysis methodology

3.11 Clustering Algorithm (Activity Profiling Across Conditions)

Assumptions

- Similar behavior under similar circumstances weighs less heavily than similar behavior under different circumstances.
- The number of factors to be considered is less than 1000 (you have issues with Euclidean distance in high dimensional space).
- The factors examined are uniformly distributed with no outliers (otherwise min–max scaling distorts the data).

Inputs

- conditions_df: DataFrame of condition measurements for each promoter
- promoter_df: DataFrame of experimental results for each promoter
- condition_name: Prefix identifying relevant columns in conditions_df
- Threshold: Distance cutoff for hierarchical clustering
- make_dendrogram: Boolean; if true, generate and save dendrogram
- Output control arguments for file paths and naming

Procedure

1) Data Cleaning
    a) Remove any rows in conditions_df or promoter_df where the "#" column contains "#."
    b) Remove columns with names starting with "#."
    c) Reset index if it contains numeric values.
2) Condition Selection
    a) From promoter_df, select only columns starting with condition_name.
3) Condition Distance Calculation
    a) Scale condition columns to a 0–1 range (min–max scaling).
    b) Compute pairwise Euclidean distances between conditions.
    c) Normalize distances so their sum equals 1.
4) Promoter Distance Calculation
    a) Sum distances per condition to generate weighting factors.
    b) Compute weighted Euclidean distances between promoters.

c) Return promoter distances in square matrix form.

5) Hierarchical Clustering
   a) Perform agglomerative clustering on promoter distance matrix using linkage method.
   b) Determine cluster assignments at the specified threshold.
   c) Record for each promoter:
      i) First join height.
      ii) Total cluster height.
      iii) Cluster ID.
      iv) Cluster member count.
6) Dendrogram Generation (if make_dendrogram=True)
   a) Plot hierarchical clustering dendrogram with promoter labels.
   b) Save to PNG file in output directory.
7) Output Generation
   a) Create DataFrame of cluster assignments and metrics.
   b) Save to CSV in output directory.

Outputs

- Cluster Assignments Table (CSV), contains each promoter's:
  - Cluster ID
  - Cluster member count
  - First join height
  - Total cluster height
- Promoter Distance Matrix: weighted Euclidean distances between promoters (optional intermediate output for validation)
- Dendrogram Image (PNG): hierarchical clustering tree showing promoter relationships (if make_dendrogram = True)
- Cluster Metrics: quantitative measures of within-cluster similarity and between-cluster separation, stored alongside promoter IDs

3.12 Matching (Genomic Feature Co-localization Analysis) Algorithm

Inputs

- Promoter dataset: Table containing genomic positions, strand orientation, and identifiers for promoter sequences (group 1)
- RNA dataset: Table containing genomic positions, strand orientation, and identifiers for RNA sequences (group 2)
- Notes text file: Plain text file with metadata or usage instructions for the datasets

Procedure

1) Input Processing
   a) Read promoter and RNA datasets from input files.
   b) Load text file containing metadata or usage notes.
2) Matching Procedure
   a) For each RNA entry (group 2), search for corresponding promoter entries (group 1) based on genomic position and strand orientation.
   b) Assign a match type code (0–3) to describe the spatial relationship between RNA and promoter
      i) 0 = no overlap or special relationship
      ii) 1 = directly adjacent or matching start
      iii) 2 = partial overlap
      iv) 3 = complete overlap or identical position
   c) Determine whether the RNA is located "in between" promoter entries and flag accordingly.
   d) Calculate the distance between each RNA and its matched promoter.
3) Data Assembly
   a) Merge the RNA dataset with matched promoter details, prefixing promoter columns with the group 1 label to avoid naming conflicts.
   b) Append match type codes, "in between" flags, and distance metrics to the combined dataset.
4) Summary Statistics
   a) Create a pivot table counting the number of matches per RNA entry for each match type (0–3) and overall totals.
5) Output Writing
   a) Export the combined match data, pivot table, and original notes to a multi-sheet Excel file for downstream analysis.

Outputs

- Matched.xlsx (Excel workbook) containing:
  - ReadMe sheet: Text file contents describing the data format and usage
  - Matches sheet: One row per match between group 1 (e.g., promoters) and group 2 (e.g., RNAs), including:
    - All columns from the RNA input file
    - Group 1 match type (0–3)
    - Group 2 "in between" flag
    - All columns from the promoter input file, prefixed with the group 1 label
    - Distance between group 1 and group 2

- Pivot_Table_Match_Types sheet: Counts of match types (0, 1, 2, 3) and totals, indexed by group 2 ID

3.13 Gene ID Algorithm (Integration of Genomic Annotations with Experimental Datasets)

Inputs

- Genome annotation folder: contains one or more .gff and/or .fasta files with locus tags and associated annotation data
- CSV folder: contains one or more .csv files with locus IDs in the first column. Other columns are preserved and carried through to the output.
- Optional parameters
  - current_directory_output (bool): if True, saves results to the current working directory
  - output_path (str): destination path if not using the current directory
  - --not_current_directory (CLI flag): forces output to a specified path via -output_path

Procedure

1) Load genome annotations.
   a) For each .gff or .fasta file in the genome annotation folder:
      i) Parse locus tags, gene names, products, and coordinates.
      ii) Store strand as a Boolean "compliment" flag.
      iii) Map each locus ID to one or more annotation records, retaining file of origin.
2) Load CSV data.
   a) For each .csv in the CSV folder:
      i) Read into a DataFrame.
      ii) Remove rows with "#" in the first column.
      iii) Remove columns starting with "#."
3) Match locus IDs to annotations.
   a) For each locus ID in the CSV's first column:
      i) Find matching genome annotation(s).
      ii) For each match:
         (1) Append gene, product, start, end, compliment, and genome_file columns.
         (2) If multiple matches, add numerical suffixes to column names.
4) Finalize output table.
   a) Remove any appended annotation columns that are entirely empty.
   b) Preserve original CSV columns in their original order.

5) Save outputs.
    a) Save processed table as .xlsx with filename containing:
        i) Original CSV name
        ii) Basenames of matching genome files
        iii) Timestamp of run
    b) Copy ReadMe.txt to the output directory.

Outputs

- One .xlsx file per input CSV, with
    - All original CSV columns
    - Annotation columns from matched genome records
        - gene, product, start, end, compliment, genome_file (with numerical suffixes if multiple matches)
    - Only non-empty annotation columns are retained in the final output
- ReadMe.txt: copied from repository data folder to the output location, describing input format and usage

3.14 Promoter Region Algorithm (Automated Extraction of Upstream Genomic Regions)

Inputs

- Genome FAST-ALL (DNA sequence format) (FASTA) files: one or more unannotated genome sequences in .fasta format (headers ignored, sequences concatenated)
- Excel files: input .xlsx files containing gene location data with columns
    - start (1-indexed start position)
    - end (1-indexed end position)
    - compliment (Boolean indicating reverse strand)
    - Optional genome_file_name (to disambiguate when multiple genome files are provided)
- Optional parameters
    - include_gene (bool): include the gene sequence itself in output (default True)
    - upstream_region (int): number of bases upstream to include (default 1000)
    - current_dir / output_dir: output location control
    - file_addition: string to append to output filenames

Procedure

1) Genome Loading
    a) Read each .fasta file, strip the header line, remove whitespace, and concatenate sequence lines into a single continuous string.
    b) Store results in a dictionary mapping {filename: genome_sequence}.
2) Upstream Region Extraction
    a) For each gene record in each Excel file:
        i) Identify the target genome sequence using genome_file_name (or use the single available genome if unspecified).
        ii) Convert start/end to 0-indexed positions.
        iii) For forward strand: start extraction at (start – 1 – upstream_region); end at end if including gene, or (start – 1) otherwise.
        iv) For reverse strand: adjust start/end accordingly and extend end by upstream_region before clipping to genome length.
        v) Extract sequence substring for the region.
3) Batch Processing
    a) Process all Excel files in the specified input folder.
    b) Remove commented rows (with "#" in the "#" column) and columns starting with "#."
    c) Append extracted sequence as a new column named "{upstream_region}bp upstream seq."
4) Output Generation
    a) Save each processed file as .xlsx with the file_addition suffix in the chosen output directory.

Outputs

- Processed Excel files: one per input .xlsx, containing:
    - Original data (cleaned of commented rows/columns)
    - Additional column with the extracted upstream (and optionally gene) sequence
- Output filenames: original name plus _file_addition.xlsx suffix, saved to current or specified output directory

3.15 Software and Instrument Versions

These are the software and instrument versions that are used for the various tools. It does not mean that the tools will not work with another version, just that they have not been tested and shown to be working.

- Beckman Coulter Echo 525 (Software ver. 1.7.2)

- Tecan Fluent 1080 (Software ver. 3.4.10.62215)
- Singer Instruments PIXL (Software ver. 3.25.0814.1)
- Formulatrix Mantis (Software ver. 5.1.2.17)
- Integra AssistPlus (Software ver. 3.7.0)

4. Results

A set of tools are presented that can be used individually or as part of a pipeline. Each tool is designed to address a specific step of the automation pipeline in the DBTL cycle. In general, all inputs allow flagging of rows with a "#" to enable a row to be left in a dataset but not processed.

4.1 Sequence Domestication

Sequence domestication is a tedious and time-consuming task for synthetic biologists, who often must manually edit DNA sequences to make them compatible with standardized cloning systems. A major challenge is removing internal recognition sites for restriction enzymes like BsaI or BsmBI, which are essential for modular assembly but can cause destructive cuts if present within the sequence itself. These sites must be carefully identified and eliminated through silent mutations that preserve the protein's function while avoiding disruption during cloning. Researchers also need to add specific ends to the sequence that match the syntax of their genetic toolkit, ensuring seamless integration with other parts. Finally, long sequences must be split into synthesis-ready blocks that avoid problematic features such as high GC content or repetitive regions. Despite being a routine part of the design process, this work is repetitive, error-prone, and demands a high level of attention to detail, making it a significant bottleneck in the rapid development of engineered biological systems.

Currently, sequence domestication is handled through a patchwork of tools and manual interventions that fall short of a fully integrated solution. Some commercial platforms offer highly specific functionality, such as NEB's HiFi Assembly tool (New England Biolabs 2025), which assists with primer design and fragment joining for a particular cloning method. However, these are limited to narrow use cases and do not address upstream sequence editing or downstream synthesis constraints. Others, like TWIST Bioscience's codon optimization tool (TWIST Biosciences 2025), provide partial capabilities by adjusting codon usage and removing restriction sites, yet they often lack the flexibility to accommodate custom assembly standards or to split sequences into synthesis-ready fragments. In many labs, the bulk of domestication work is still performed manually using general-purpose software like Geneious, where researchers must identify restriction

sites, design silent mutations, and format sequences by hand. This manual process is not only slow but also prone to errors, especially when scaling up to dozens or hundreds of constructs. The lack of automation and standardization at this stage creates a bottleneck that delays the build phase and undermines the efficiency of the broader DBTL cycle.

The sequence domestication tool (Figure 2) was developed to automate the entire domestication workflow (from cut site removal to adding the appropriate modular cloning [MoClo] [Bird et al. 2022] ends), synthesis planning, and documentation of genetic sequences for cloning workflows. The system accepts multiple structured input tables, including a sequence table with part metadata, a codon-change table defining allowable synonymous substitutions, cloning template definitions with flanking regions, and an empirical overhang-ligation fidelity dataset. For each sequence, the tool performs rigorous cleaning and validation, removing whitespace, enforcing uppercase format, and rejecting sequences with invalid bases or lengths under six nucleotides. It then scans for internal restriction enzyme recognition sites from a predefined list. When sites are detected, the program applies either codon-based synonymous replacements that are prioritized to minimize disruption and avoid introducing other undesired sites, or direct base complementation if silent changes are not permitted. Up to three iterative passes are used to eliminate all targeted sites.

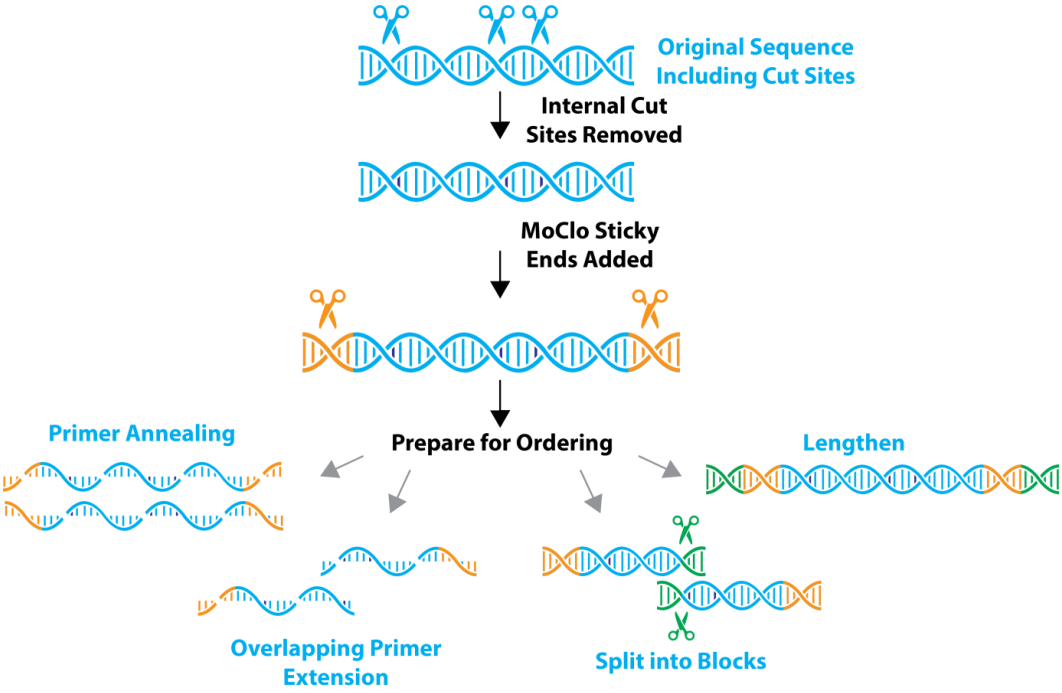

Figure 2. The sequence domestication tool works by removing internal cut sites, adding MoClo sticky ends for a toolkit or standard specified by the user, and then splitting sequences to order them as cheaply as possible from synthesis companies (as defined by the user).

Following domestication, standard flanking sequences are incorporated by substituting placeholders in the provided cloning templates (e.g., MoClo, Loop, CIDAR, or a custom syntax). Optional colony PCR primers are automatically designed upstream of the insert, using sequence context to ensure acceptable GC content, length, and melting temperature. The tool then selects an optimal synthesis strategy based on part length and user thresholds (these generally are based on synthesis length thresholds of companies like IDT), with logic to treat short inserts as single-stranded oligos, extend sequences to minimum synthesis lengths, or split large constructs into synthesis blocks. For block fragmentation, candidate split points are chosen to satisfy block size constraints, preserve unique and high-fidelity 4-bp overlaps, and avoid reusing overhangs already present in the design. NEB ligation fidelity data are used to reject overlaps with high off-target ligation likelihood, and recursive splitting is applied until all segments meet constraints. Where extension PCR is required, primers are designed to target central overlaps with suitable melting temperature and GC composition.

All processing steps are logged, including removed cut sites, applied template flanks, designed blocks and primers, and the synthesis pathway assigned. If SBOL output is requested, the program generates modular SBOL components for each construct segment (e.g., flanks, and insert), applies sequence assembly constraints, and annotates part types with SO terms. This integrated pipeline enables automated, constraint-aware sequence domestication, synthesis block optimization, primer design, and standards-compliant documentation for diverse synthetic biology assembly schemes.

4.2 Echo (Combinatorial Assembly)

Combinatorial assembly of genomic toolkit parts is a powerful strategy in synthetic biology that enables researchers to rapidly generate and test large libraries of genetic constructs. MoClo systems like Golden Gate allow scientists to mix and match promoters, coding sequences, terminators, and other regulatory elements in a single reaction, producing dozens or even hundreds of unique combinations in parallel. These subtle variations can dramatically influence gene expression, protein function, and cellular behavior, making combinatorial design essential for identifying optimal configurations for a desired phenotype or function.

Despite its power, setting up these assemblies remains labor-intensive and error-prone. Each reaction requires precise pipetting and careful tracking of part combinations, and the reagent cost can quickly escalate across large libraries. The Beckman Coulter Echo 525 acoustic liquid handler addresses part of this challenge by miniaturizing reaction volumes down to the nanoliter scale, reducing reagent use

by up to 40- to 200-fold (Bailey et al. 2025). However, generating accurate instructions for each unique assembly remains a bottleneck. To solve this, we developed a flexible tool that automatically generates build instructions for combinatorial assemblies, regardless of the modular cloning standard being used (e.g., MoClo, Loop, CIDAR, or a custom syntax). By abstracting the assembly logic from the specific overhangs or part types, the tool supports diverse workflows and ensures compatibility across platforms. This dramatically reduces setup time, minimizes human error, and enables high-throughput exploration of genetic design space with minimal resource consumption. The Echo tool was developed to automate the planning, validation, and output generation for combinatorial DNA assembly workflows (Figure 3). The program accepts a source plate file containing reagent metadata (plate, well, name, volume, molarity, sequence, and type) and a composition file defining desired assemblies using parts, reagents, vectors, and enzymes. It begins by loading and validating these inputs, ensuring required columns are present and that plate types and molarities are consistent. The composition table is parsed to identify "Part" and "Reagent" columns, and each row is expanded into all combinatorial permutations, producing a flattened list of planned reactions with full metadata.

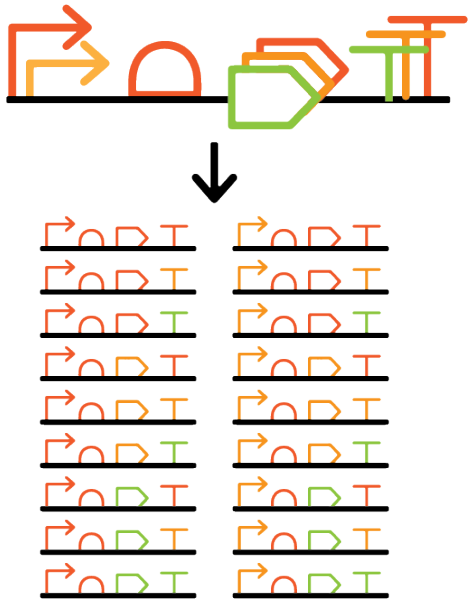

Figure 3. A single combinatorial description of an assembly can be turned into many individual assemblies for the Echo robot to carry out.

The system validates reagent availability by cross-checking that every part or reagent in the composition file exists in the source plate file. For assemblies involving restriction-ligation, it performs in silico digests of each vector and part using the designated enzyme, then tests all permutations to confirm matching overhangs and correct assembly order. Each reaction is annotated with assembly

validity, part arrangement, and resulting sequence. Optional SBOL generation is supported, with assemblies encoded in SBOL2 (Madsen et al. 2019) or SBOL3 (Buecherl et al. 2023), annotated for part boundaries and SO roles (Eilbeck et al. 2005).

The tool also computes liquid-handling instructions. For each reaction, it determines the required volume of each reagent based on molarity or default settings, simulates withdrawal from available wells, adds water to reach the target final volume, and generates an Echo-compatible transfer instruction file. Outputs can be tailored for either volume estimation, summarizing total reagent needs, or full execution, in which case the software produces updated plate files with adjusted remaining volumes, sequence outputs, transfer instruction files, and reaction logs summarizing contents and validity.

Final outputs include Echo input files for automated liquid handling, SBOL documents when enabled, and reaction logs with annotated sequences. Together, these capabilities allow automated expansion of combinatorial design tables into physically executable assembly instructions, with integrated validation, simulation, and standards-compliant documentation.

While the tool was designed with combinatorial DNA assembly in mind, the tool can be used for setting up any kind of combinatorial reaction, for example, for testing various enzyme cocktails for diagnostic methods.

4.3 Tecan (Plate Setup and Rearray)

Setting up plates for synthetic biology experiments is a repetitive and time-consuming task that often slows down high-throughput workflows. Researchers must manually pipette media, reagents, or cultures into dozens or hundreds of wells, carefully topping up volumes and rearranging samples into specific formats for downstream analysis. This process requires precision to avoid inconsistencies. Rearranging layouts to match experimental designs or to prepare for automated readouts adds another layer of complexity, often involving tedious tracking and reformatting. These steps are prone to human error and demand significant attention to detail, making them a bottleneck in synthetic biology workflows.

To streamline this process, we developed a tool that generates precise worklists for automated plate setup, beginning with support for the Tecan Fluent liquid handler. This high-throughput platform enables consistent, large-scale execution of complex experimental layouts. While the tool currently targets Fluent workflows, its architecture is designed to be extensible to a wide range of other instruments, including budget-friendly systems like Opentrons OT-2, mid-range platforms such

as Integra Assist Plus and Formulatrix Mantis, and high-end systems like Hamilton Microlab STAR and Beckman Echo 525 (overview of the instruments is given in Table 1). By abstracting experimental design from hardware-specific syntax, the tool lays the foundation for broader automation across the synthetic biology community, enabling consistent execution, reduced setup time, and improved reproducibility.

Table 1. Comparison of common liquid handling robots.

Instrument	Plate spaces	Dispense volume (microliter)	Modules	Learning curve	Worklist option	Scripting options	Community	Labware definitions	Special labware	Cost (USD)	Source
Opentrons OT-2	11	1–1000	Magnetic, temp, thermocycler, heater-shaker, and HEPA filter modules available	Low (GUI or Python API)	Yes	Python protocol script	Large open community	JSON files	Tips	$15K	(Opentrons 2025)
Integra Assist Plus	4	0.5–1250	Magnetic, temperature modules	Very low (GUI)	Yes	Worklists	Limited	GUI or JSON files	Tips	$35K	(INTEGRA 2025)
Formulatrix Mantis	1 (plus 6–12 reagent tubes)	0.1–1000	None	Low (GUI-based)	Yes	Worklists	Limited	GUI or JSON files	Dispensing diaphragms	$65K	(Formulatrix 2005)
Hamilton STAR	55+	1–5000	Temperature modules, shakers, centrifugation, plate reader, vacuum manifold	High (VENUS software very complicated low-level programming)	Yes	Worklist (proprietary API firmware commands)	Limited	GUI	Tips	$175K–$275K	(HAMILTON 2025)
Tecan Fluent 1080	72	0.25–1	Temperature modules, hotels, spiral plater, shakers, plate reader, vacuum manifold	High (GUI but very complicated liquid classes, labware, etc.)	Yes	Worklists (proprietary API commands)	Limited	GUI	Tips	$350K	(TECAN 2025)
Beckman Echo 525	1 source and 1 destination	0.025–5	None	Moderate to high (GUI but less intuitive)	Yes	Worklists	Limited	Limited	Acoustic dispensing plates	$500K	(Beckman Coulter Inc. 2025)

Note that all can handle input as a worklist (a set of instructions uploaded as a CSV, TSV, or Excel document, though each uses a different format).

The software tool for the Tecan Fluent 1080 was developed to automate and optimize liquid transfer planning between plates for high-throughput workflows. The tool accepts a source plate file containing well positions, reagent names, and volumes; a transfer plan file specifying reagent names, destination plates, optional well assignments, and required volumes; and optionally, a destination plate file with existing contents. FCA tip configuration files defining size and min–max volume constraints are also used to enforce hardware limits.

The workflow begins by loading and sanitizing all input tables, removing commented rows or columns and detecting whether destination wells are prespecified. Internal data structures are then initialized, including dictionaries mapping wells to reagents, indexed source wells by reagent name, and movement lists for each destination plate based on the transfer plan.

For each requested transfer, the software determines the correct destination well. If a specific well is supplied, it is used directly; if "top-up" mode is enabled, the system identifies an existing well containing that reagent on the target plate, calculates the additional volume needed, and fills it, or selects the first empty well if no match is found. In standard mode without top-up, the next available empty well is chosen. The required volume is then sourced from available wells, prioritizing the smallest FCA tip capable of the transfer and splitting across wells if necessary. Each withdrawal updates the source plate's available volume and records the exact details of the transfer, including source/destination plate and well, volume, and tip size. Destination well records are updated to reflect accumulated or newly added reagent content.

Once all transfers are planned, the tool generates a complete output package. This includes updated source and destination plate tables, one or more robotic worklist files formatted for Tecan Fluent (either split by tip size or combined), and a human-readable processing log when top-up logic is used. A README file is also generated to explain the structure and use of the outputs.

4.4 PIXL (Transformation and Colony Picking)

Transformation is the step in the DBTL cycle where engineered DNA constructs are introduced into host cells to create living systems that express the desired genetic functions. Once transformed, cells are plated at different dilution factors to ensure well-isolated colonies, often resulting in multiple plates for each condition. Each colony represents a unique clone that may carry the desired construct, and researchers typically pick several colonies per condition to serve as biological replicates. The challenge lies in selecting colonies from across these plates and reformatting them into new plates that match the original experimental layout, with

each well representing a distinct experiment and each plate a replicate. Manually tracking colony origin and placement is slow and error-prone, especially at large numbers of plates. Using a colony picker to automate colony picking and plate formatting would eliminate this bottleneck, ensuring consistency, preserving experimental structure, and accelerating the transition to downstream testing.

A workflow was developed that uses the Integra Assist Plus to spot plate transformation results at several dilutions and the Singer Instruments PIXL to pick and rearray colonies into plates of biological replicates (Figure 4). This workflow includes a computational tool to streamline the conversion of plate-based colony detection data into rearray instructions compatible with the PIXL colony-picking robots, specifically to support high-throughput transformation workflows. The PIXL software accepts one or more TSV-formatted plate files containing colony coordinates and intensity measurements, along with metadata describing the number of sectors per plate (rows × columns), the number of replicate target plates, the target plate format (e.g., 96-, 384-, or 1536-well), and an optional sorting column such as Colony_Brightness.

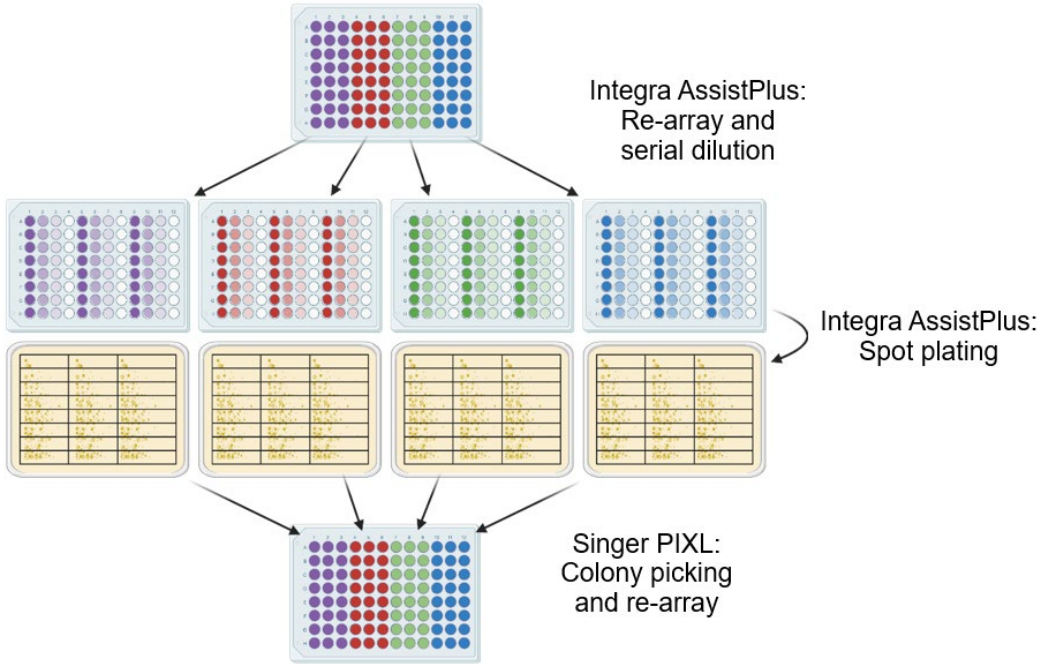

Figure 4. Dilution, spot plating, and colony picking scheme for high throughput transformations. From a single transformation recovery plate several dilution plates are created each of which are spotted onto flat SBS plates. The program helps pick a single colony per region per replicate and ensures that even if a transformation produced no colonies, or fewer than the number of biological replicates desired, the final plate layout mirrors the original.

To meet this need for automation, a range of colony picking systems has emerged, each offering distinct capabilities and levels of flexibility. QPix from Danaher Life Sciences is one of the most established platforms, known for its reliability but limited by a closed architecture that restricts user customization. In contrast, PIXL from Singer Instruments has gained popularity in academic settings for its openness and collaborative development model, allowing researchers to tailor functionality to specific workflows. Hudson Robotics offers the RapidPick series, which supports high-speed, morphology-based selection and integration with their other robotic systems. As academic labs increasingly adopt high-throughput approaches, the colony picking landscape is shifting toward platforms that prioritize modularity, user control, and adaptability, which make it possible to scale transformation workflows without compromising experimental design or data integrity.

The workflow begins by loading each source plate file, removing commented rows and columns, and assigning a source plate name from the filename. Colonies can be ranked by the chosen sort metric, and the program iterates through each grid-defined plate region, or sector, to extract colony coordinates. The system is designed to enable spot plating at four dilutions across four plates (e.g., A1 1× in A1 of plate 1, A1 2× in A2 of plate 1, and so on). For each region (e.g., the A1 sector in all its dilutions), the program automatically selects a user-defined number of colonies to pick.

Replicates are fully supported, with the tool preserving plate layout across biological replicates. This ordering is maintained even if some transformations produce few or no colonies. For each replicate, colonies are assigned to target wells using column-wise (default) or row-wise placement. The program converts well indices to alphanumeric codes (e.g., A01, B12) and records the complete mapping of source plate, colony coordinates, target plate, and well.

The output combines metadata headers for all source and target plates with detailed picking instructions, producing a single Singer PIXL-compatible tab-separated file. This ensures that picking is carried out according to the spot plating scheme and user-defined number of biological replicates.

4.5 DOE: Binary (Strain Optimization)

A DOE tool tailored for binary factors is useful for designing knockout strains in synthetic biology, where each gene is either present or deleted. Traditional approaches often rely on one-factor-at-a-time testing, which quickly becomes inefficient and blind to interactions when dealing with multiple gene knockouts. In contrast, a DOE framework allows researchers to systematically explore combinations of gene deletions, uncovering not only individual effects but also

synergistic or antagonistic interactions between genes. This is especially valuable when engineering strains for complex traits like metabolic optimization, stress tolerance, or synthetic circuit performance, where the phenotype may depend on subtle multi-gene dynamics. By using a binary-factor DOE tool, researchers can efficiently prioritize which combinations to build and test, reduce the number of experiments needed, and gain deeper insight into the genetic architecture of their system. It transforms knockout design from a trial-and-error process into a structured, data-driven strategy.

This tool generates optimized binary experimental designs that avoid aliasing of main effects and two-factor interactions by carefully choosing the number of independent factors (f) and assigning the remaining factors as dependent aliases of unique three-factor interactions. Given a total maximum number of experiments X and total factors T, the number of independent factors is first estimated as f = log2(X), since a full factorial design over f binary factors produces 2^f experiments. The number of dependent factors is $d = T - f$. To ensure each dependent factor is assigned to a unique three-factor interaction, the number of distinct three-way combinations from the independent factors must exceed d, giving the inequality

$$\binom{f}{3} > d$$

Substituting $d = T - f$ yields

$$\binom{f}{3} > T - f$$

Expanding the binomial coefficient,

$$\frac{f!}{3!\,(f-3)!} > T - f$$

which simplifies to

$$\frac{f(f-1)(f-2)}{6} > T - f$$

Adding f to both sides and combining terms gives

$$\frac{f(f-1)(f-2)}{6} + f > T$$

Factoring the numerator:

$$\frac{f(f^2 - 3f + 2)}{6} + f > T$$

Writing f as $6f/6$ allows combination:

$$\frac{f(f^2 - 3f + 2) + 6f}{6} > T$$

which simplifies to the final form:

$$\frac{f(f^2 - 3f + 8)}{6} > T$$

The minimum integer f satisfying this inequality for a given T is determined numerically, and the minimum required number of experiments is then $2\text{^}f$. If $2\text{^}f$ is below the user-specified X, additional independent factors may be added (up to X experiments) to increase resolution. The final design matrix is constructed with independent columns set to all binary combinations of 0/1 and dependent columns defined as the product (logical AND in binary form) of their assigned three-factor interaction. The output is saved as a CSV file (Discrete Output.csv), where each row is a proposed experiment and each column represents the state of a factor.

While this tool was designed with strain optimization in mind, it can also be used for other binary DOE problems such as microbial consortia design where each strain is either present or absent.

4.6 DOE: Continuous (Media Optimization)

Continuous DOE is a powerful strategy for optimizing media conditions when testing genetic constructs in synthetic biology. By varying concentrations of media components such as carbon sources, nitrogen, salts, and cofactors across defined ranges, researchers can systematically explore how these factors influence gene expression, protein production, or metabolic performance. Rather than testing one variable at a time, continuous DOE enables the simultaneous evaluation of multiple inputs and their interactions, dramatically reducing the number of experiments needed to uncover meaningful insights. This approach is especially valuable when biological responses are nonlinear or influenced by subtle combinations of nutrients, which traditional methods often miss. With a well-structured DOE, researchers can tease apart complex dependencies, identify optimal media formulations, and build predictive models that guide future designs while conserving time, resources, and experimental effort.

Continuous DOE is already being actively applied across a range of synthetic biology workflows, particularly in areas like protein expression optimization, metabolic engineering, and strain development. For example, companies such as Culture Biosciences use DOE in conjunction with cloud-connected bioreactors to fine-tune fermentation conditions at scale, enabling rapid iteration and data-driven decision-making in strain optimization (Culture Biosciences 2025). In academic settings, DOE is increasingly integrated with machine learning approaches to enhance the LEARN phase of the DBTL cycle (Thompson et al. 2023). We have built a tool to make DOE easier to use for the average researcher and to apply to many different contexts.

The continuous DOE tool generates optimized continuous experimental designs for multivariate factor spaces, with optional integration of prior experimental results to guide selection. Inputs include a factor table specifying the minimum, maximum, and step values for each variable, a user-defined limit on the number of new experiments to generate, and optionally, a dataset of previous experiments containing both factor values and their measured output.

If prior data is provided, two strategies are available. Regression-based optimization standardizes the historical factor values and outputs, fits a linear regression model, and uses LHS (Loh 1996) to propose new candidate experiments. LHS is a space-filling design method that ensures each factor's range is evenly sampled while still generating combinations that explore the multidimensional space efficiently. This helps avoid clusters of similar points that can occur with purely random sampling. Candidate designs are scaled to factor limits, evaluated by the regression model, and filtered to retain those whose predicted outputs are closest to a user-specified target. Local search optimization instead identifies a small set of prior experiments that performed closest to the target output and generates new points around them, constrained by a minimum and maximum search radius. In both strategies, rejection sampling is used to ensure that all selected points maintain a minimum separation distance, avoiding redundancy.

If the total number of selected points falls below the desired run count, additional designs are filled in using rejection sampling across the entire factor space. When no prior data is available, the algorithm defaults to space-filling designs: small design spaces (i.e., those where the total number of possible combinations is not much larger than the desired number of experiments) are explored using LHS for balanced coverage, while very large spaces instead rely on random sampling with distance constraints to avoid oversampling in any region. This cutoff is important because LHS becomes less efficient in extremely large or sparse spaces; it can overemphasize uniform coverage at the expense of exploring potentially more

informative regions, so random sampling with distance filtering is preferred in those cases.

All generated designs are snapped to the nearest valid step size for each factor. If previous data was provided, the tool appends new designs (with empty output fields) to the original dataset, maintaining consistent formatting for downstream analysis. The final design is saved as Continuous Output.csv, with each row representing an experiment and each column corresponding to factor levels or the output placeholder. This workflow allows researchers to efficiently explore large experimental spaces, focus on promising regions when prior results are available, and maintain reproducible, well-structured experiment plans. It can be applied to the synthetic biology DBTL pipeline in several ways including media formulations, growth conditions (temperature, CO_2, etc.), and inducer schedules.

4.7 Mantis (Media Variation Testing)

Implementing media conditions designed by continuous DOE using hand pipetting is tedious and time consuming due to the variability of components across wells. Each media component must be dispensed at different concentrations, which means constantly adjusting pipette volumes, recalculating mixtures, and tracking combinations across dozens or even hundreds of wells. This not only slows down the workflow but also introduces a high risk of human error. The complexity compounds quickly as the number of variables increases, making manual execution impractical for large-scale experiments. Liquid handlers offer a robust solution by automating precise dispensing across complex experimental matrices, enabling researchers to execute continuous DOE designs with speed, accuracy, and reproducibility. This transforms what would be a tedious and error-prone process into a scalable and reliable platform for uncovering nuanced biological insights.

The Formulatrix Mantis (tipless microliter liquid handler) can be used to accurately set up many plates with varying media components with minimal plasticware waste. A tool was developed to convert reagent allocation tables (as created by the continuous DOE tool) into formatted dispensing instructions for either Mantis or Tempest (larger version of the Mantis) liquid handling systems, accommodating multiple plate sizes and filling patterns. The input is a DataFrame in which each row corresponds to a well and each column (except metadata) specifies the volume of a given reagent to be dispensed. The user specifies the plate type (96, 384, or 1536 wells), whether filling should proceed column-wise or row-wise, whether 384-well plates should be filled quadrant-by-quadrant, and which instrument format (Mantis or Tempest) should be produced.

The process begins with data cleaning, removing rows flagged with a "#" in the dedicated metadata column and dropping any reagent columns whose names begin with "#." After cleaning, the tool sets plate dimensions according to the specified format and selects the target reagent's volume values. If the provided data contains fewer wells than the plate capacity, it pads with zeros until all wells are represented. For 384-well plates with quadrant filling enabled, wells are assigned in quadrant order, defined by alternating rows and columns, and filled in column-major sequence within each quadrant. Otherwise, wells are filled strictly column-by-column or row-by-row according to the column_wise setting.

Once the plate array is filled, the function generates a formatted output matrix. For Tempest, the reagent name is prepended to the matrix and formatting is adjusted to the Tempest specification. For Mantis, the output begins with a header containing software version, plate type, and the total reagent count, followed by metadata lines for each reagent (name, classification, and constant pressure setting) and the corresponding matrix of well volumes. In both cases, all reagent matrices are concatenated into a single .dl.txt output string ready for saving.

4.8 Expression Factor Importance and SHAP Analysis Pipeline

As researchers increasingly adopt DOE approaches, the need for robust analytical tools becomes paramount. DOE enables the simultaneous exploration of multiple input variables and their interactions, often resulting in complex, nonlinear datasets that are difficult to interpret using traditional statistical methods. Despite its power, many scientists shy away from DOE because it demands sophisticated analysis to extract meaningful insights, especially when dealing with high-dimensional data and subtle biological effects. This is where tools like those that are decision-tree based featuring importance analysis with SHAP (Mosca et al. 2022) become essential. SHAP provides a transparent and intuitive framework for quantifying how each continuous predictor variable contributes to one or more measured outputs. By decomposing model predictions into additive contributions, SHAP not only identifies which factors are influential but also reveals the direction and magnitude of their impact. This level of interpretability is critical for aligning computational results with experimental intuition, making it easier for researchers to validate findings, refine hypotheses, and guide future experimental designs. This tool is designed with accessibility in mind, enabling researchers with limited statistical training and no coding experience to easily apply these advanced analysis methods and gain clear, interpretable insights from complex experimental data.

The process begins with data cleaning, removing rows flagged with "#" and dropping any factor or output columns that are marked as commented-out.

Minimum and maximum factor values are then determined either from theoretical ranges provided by the user or from the observed dataset. Data are split into training (80%) and testing (20%) sets, and a DecisionTreeRegressor with user-specified maximum depth is trained to balance capturing relationships against overfitting risk. SHAP values are computed for the training data, producing a per-feature attribution for each prediction. The mean absolute SHAP value per factor is calculated to create an importance ranking, highlighting the predictors that most strongly drive model behavior.

For factors with importance above a user-defined cutoff, the tool can generate dependence plots that show how variation in a single factor correlates with changes in predicted output, optionally clustering features by similarity in SHAP patterns to reveal groups of interrelated variables. This clustering is valuable in biological contexts where multiple factors may act through related pathways or share mechanistic effects. Residuals between model predictions and actual measurements are computed to assess model fit, with histograms provided to detect systematic bias. Standard performance metrics (R^2, MSE, and RMSE) quantify predictive accuracy.

An optional single-feature variation analysis explores each factor individually by holding all other variables at their mean and sweeping the selected factor across its full range, plotting the predicted response curve. This can reveal nonlinear thresholds or saturating effects that may guide experimental design. In multi-output mode, the process repeats for each output, compiling all results into combined importance and metrics tables.

4.9 Expression Binary Factor ANOVA and Interaction Analysis

When testing the effects of knocking out different genes across multiple conditions, researchers often turn to ANOVA to determine which factors significantly influence an outcome. But interpreting ANOVA results is not always straightforward, especially for those without a strong statistical background. Understanding model assumptions like normality and equal variance, knowing how to test them, and making sense of the results can be tricky and time-consuming. This tool simplifies the process by running a Type III ANOVA on binary-coded factors, optionally including two-way interactions, and automatically checking assumptions. It presents results with clear visual summaries and explains how to interpret each plot, making it accessible to researchers who are not familiar with R or advanced statistics.

The analysis begins with preprocessing, where rows or columns flagged with "#" are removed, the output directory path is checked, and factor columns are identified

by excluding the output column. Interaction annotations in column names are stripped to standardize factor names.

If test_interactions is enabled, the script generates all possible two-factor interaction terms (e.g., FactorA:FactorB) and determines how many can be tested given the dataset size (since including too many terms with limited data can lead to unstable estimates). These interactions are evaluated in subsets, each fitted to an OLS model. The top-ranked interactions by p-value are retained, and the final ANOVA model includes both main factors and these selected interactions. If test_interactions is disabled, the model includes only main factors.

The Type III ANOVA tests each factor's effect on the output after accounting for all other terms in the model. This is particularly relevant in designs where factors are not orthogonal, as it isolates each term's unique contribution. P-values are compared to the significance_threshold; factors or interactions below this cutoff are considered significant. The output also includes a Pareto chart of sum-of-squares (showing relative contribution of each term) and a histogram of residuals for checking normality assumptions.

When interactions are significant, the tool generates interaction plots that show how the effect of one factor depends on the level of another, which is key in biological contexts where factor combinations may produce synergistic or antagonistic effects.

Assumption checks are performed for each significant main factor using Levene's test to evaluate homogeneity of variance between groups (factor = 0 vs. factor = 1). Levene's test assumes independence of observations and is robust to moderate departures from normality; a significant p-value here indicates unequal variances, which can affect ANOVA validity. The residual histogram provides a visual check of the normality assumption.

4.10 Clustering (Activity Profiling Across Conditions)

Clustering tools are widely used in biological research to group genes, species, or other entities based on shared characteristics. For example, hierarchical clustering (Figure 5) and k-means are commonly applied to gene expression data to identify co-regulated genes (Ren et al. 2020). In microbial ecology, tools such as QIIME 2 (Bolyen et al. 2019) and MEGAN (Huson and Weber 2013) cluster species based on sequence similarity and abundance profiles, helping researchers interpret complex community structures. However, most of these tools operate under a single condition or dataset, limiting their ability to capture context-dependent variation.

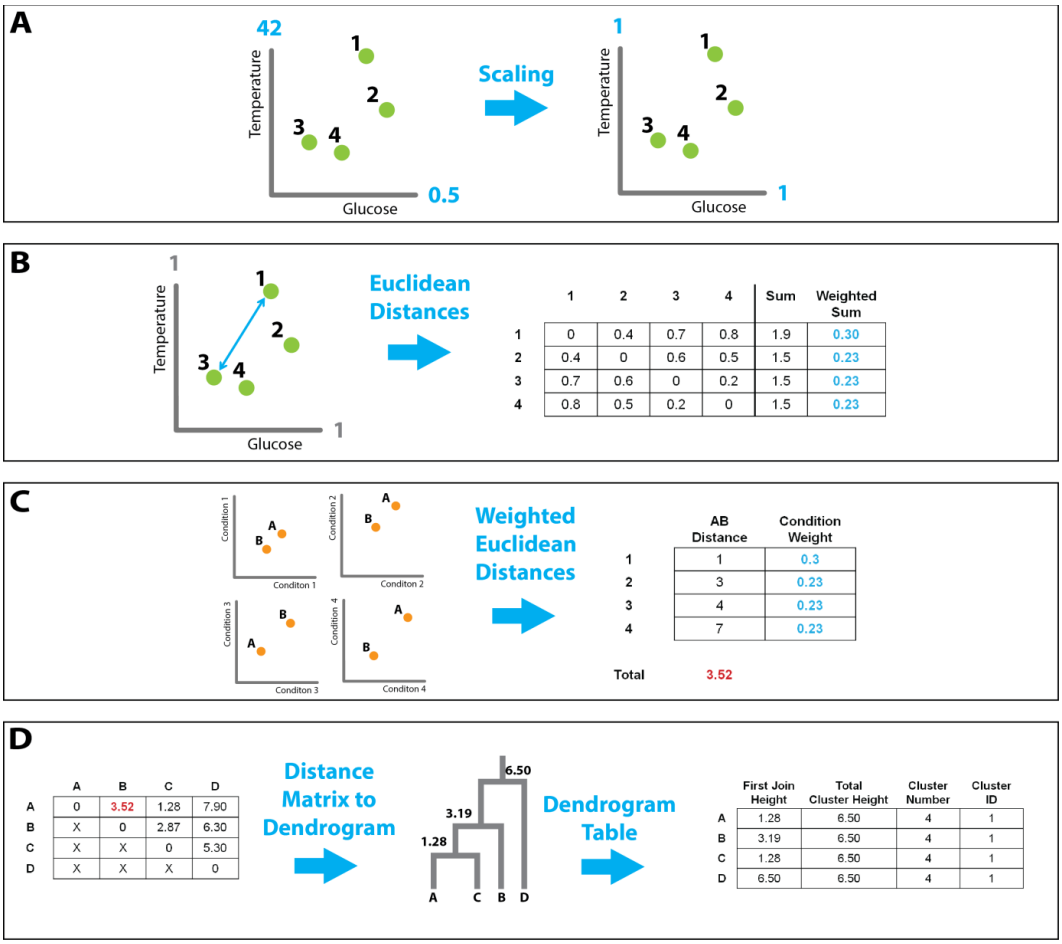

Figure 5. Workflow for weighted hierarchical clustering of promoter activity profiles. A) Min–max scaling of condition measurements to a standardized 0–1 range, ensuring comparability across conditions with differing dynamic ranges. B) Conversion of scaled condition profiles into a condition similarity matrix, followed by derivation of condition-specific weights based on inter-condition distances. C) Calculation of weighted Euclidean distances between promoters, where each condition's contribution is scaled according to its weight. D) Transformation of the weighted promoter–promoter distance matrix into a hierarchical clustering dendrogram, with corresponding cluster assignments summarized in a dendrogram table.

Thus, such tools cannot be applied to identify functionally similar biological across diverse experimental conditions. Promoters, for example, often exhibit complex and context-dependent activity profiles that are difficult to interpret or compare using simple metrics like average strength. As the scale of experiments grows, so does the need for systematic approaches to uncover patterns in regulatory behavior and group elements with shared functional characteristics. Without computational tools, researchers must manually sift through noisy data, which is both time-consuming and difficult to interpret. To address this, we developed a clustering framework that organizes promoters based on their activity patterns across conditions, enabling the discovery of regulatory modules and context-specific

responses. This approach not only streamlines analysis but is broadly applicable to any dataset where biological entities are profiled across multiple conditions, including genes, strains, metabolites, and synthetic constructs.

The tool uses a clustering framework to group promoters based on their activity patterns across experimental conditions, enabling identification of promoters with similar regulatory responses (see Figure 5). To ensure comparability between measurements taken in different contexts, we applied min–max scaling, which rescales each feature to a standard range of 0–1 according to:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

This transformation preserves the shape of the original distribution while ensuring that no single condition dominates the distance calculation simply due to having larger absolute values. This is particularly important in promoter activity datasets, where measurement ranges can vary substantially between conditions (e.g., due to differences in assay sensitivity or reporter dynamic range).

We then computed pairwise Euclidean distances between promoters, applying condition-specific weights derived from the similarity structure of the conditions themselves. The weighted Euclidean distance between two promoters can be calculated using the equation

$$d(A,B) = \sqrt{\sum_{i=1}^{n} w_i (activity_{A,i} - activity_{B,i})^2},$$

where w_i is the weight for condition i, $activity_{A,i}$ and $activity_{B,i}$ are the scaled activities of promoters A and B in condition i, and n is the total number of conditions. This weighting allows conditions with greater discriminatory power to contribute more to the overall similarity metric.

Euclidean distance is well-suited for continuous, scaled data with approximately linear relationships between conditions; however, it can become less informative if the data are highly nonlinear, sparse, or contain strong outliers, in which case alternative metrics (e.g., cosine similarity, correlation distance, or kernel-based measures) may be more appropriate.

Agglomerative hierarchical clustering was applied to the weighted distance matrix, using a user-defined threshold to control the granularity of the resulting clusters. The threshold parameter defines the granularity of clustering: lower thresholds produce more clusters with finer distinctions, while higher thresholds merge

clusters to highlight broader similarity patterns. For each promoter, we record its cluster ID, the number of promoters in the same cluster, the height at which it first merges with another cluster, and the total height at which its final cluster is formed (Figure 6). When enabled, the method generates a dendrogram image to visually represent the promoter similarity tree, allowing intuitive inspection of the clustering structure.

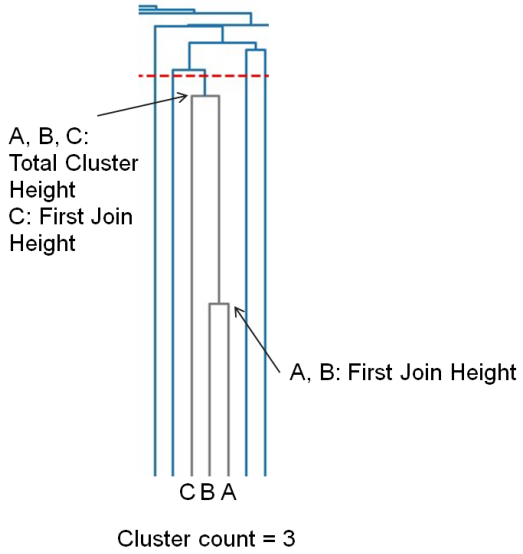

Figure 6. Clustering metrics for the dendrogram table. For the cluster CBA, the cluster count is 3, and the A and B first join height is the same. The total cluster height is the same as the C for join height.

Outputs include a tabular summary of cluster assignments with associated metrics, a dendrogram image, and, optionally, the promoter distance matrix itself for validation or downstream computational analyses. This combined visual and quantitative output enables researchers to both explore and formally analyze similarity relationships among promoters. The same workflow can be adapted to any high-dimensional biological dataset where features are measured under multiple conditions, including clustering of gene expression profiles, metabolite abundance patterns, or phenotypic assay results.

4.11 Matching (Genomic Feature Co-localization Analysis)

Comparing genomic feature sets is often complicated by the need to account for both spatial proximity and strand orientation, especially when datasets originate from different sources or represent distinct biological entities. Traditional overlap-based methods may miss meaningful relationships when features are close but not directly overlapping, or when strand-specific context matters. To address this, we developed a position-based matching approach that uses only the start, end, and

orientation of genomic features to assess spatial relationships. This abstraction allows flexible comparisons across a wide range of datasets, regardless of their biological content. While demonstrated here by comparing promoter sequences to RNA features, the method is broadly applicable to tasks such as aligning transcription factor binding sites with regulatory elements, mapping genetic variants to coding or noncoding regions, or identifying overlaps between Chromatin Immunoprecipitation Sequencing (ChIP-seq) peaks and annotated enhancers.

For each RNA feature, all promoter entries with compatible strand orientation and nearby genomic positions are identified. Matches are assigned to one of four categories describing their positional relationship: (0) no overlap or adjacency, (1) direct adjacency or matching start coordinate, (2) partial overlap, or (3) complete overlap or identical position. Additional flags are used to mark RNA features positioned between two promoters without overlapping either, and genomic distances between matched pairs are computed.

This classification spans a diverse set of spatial relationships, ranging from complete positional concordance between promoters and RNA features (Type 3), to partial overlaps suggesting alternative transcription start sites (Type 2), to adjacent features (Type 1) consistent with promoter-proximal initiation. Type 0 matches, no direct spatial relationship, highlight regions where annotated promoters are unlikely to be driving the observed RNA feature. Features flagged as "in-between" indicate genomic intervals that may represent unannotated regulatory elements or intergenic transcription events.

All matches, along with the corresponding promoter and RNA metadata, are assembled into a combined dataset and summarized in a pivot table showing the distribution of match types per RNA feature. This output structure provides a reproducible and extensible framework for quantifying positional relationships in genomic datasets and is adaptable for broader comparative analyses beyond the promoter–RNA context.

4.12 Gene ID (Integration of Genomic Annotations with Experimental Datasets)

Functional genomics experiments often produce large datasets containing locus identifiers but lack the biological context needed for meaningful interpretation. Researchers typically rely on genome annotation files—such as GFF and FASTA records—to retrieve metadata like gene names, products, and genomic coordinates. However, manually merging these annotations with experimental results is time-consuming, error-prone, and difficult to scale across multiple datasets or organisms.

To address this challenge, we developed an automated approach that links genome annotation data with experimental outputs, enabling rapid enrichment of raw results with biologically relevant metadata. This workflow streamlines downstream analysis and is broadly applicable to any dataset where locus-level resolution is available, including transcriptomics, mutational screens, and synthetic biology assays.

Annotation files are parsed to extract locus-level details, including start and end coordinates, strand orientation, and product descriptions, while retaining the source file identity for traceability. Each experimental dataset is scanned for locus IDs in the first column, and for each match, the corresponding annotation fields are appended. Multiple matches for a single locus ID are supported, with suffixes distinguishing the alternate records. Columns that remain entirely empty after annotation are automatically removed to produce concise outputs.

The resulting annotated datasets preserve the full structure of the original experimental CSV files while augmenting them with relevant genomic context—such as gene function, genomic location, and strand orientation—that is essential for biological interpretation. Output files are generated in Excel format, one per input dataset, with filenames containing the source dataset name, the contributing genome files, and a timestamp for version tracking. Each output is accompanied by a ReadMe file detailing input formats and usage guidelines. This approach streamlines the integration of experimental results with genome annotations, reducing manual curation time and minimizing transcription errors, and is adaptable to a wide range of genomic and functional data types.

4.13 Promoter Region (Automated Extraction of Upstream Genomic Regions)

Extracting upstream DNA sequences from annotated genes is a common prerequisite for many sequence-based analyses in synthetic biology and genomics, yet it remains a manual and error-prone task when done at scale. Researchers often need to retrieve promoter regions or regulatory elements from genome assemblies to study transcriptional control, design primers, or build datasets for computational modeling. This process typically involves parsing genome annotation files, calculating upstream coordinates, and extracting sequences from reference genomes. To streamline this workflow, we implemented an automated approach that retrieves upstream DNA sequences for sets of annotated genes, enabling rapid and reproducible access to promoter regions and other regulatory elements. While our primary use case focused on extracting the 1000 bp upstream of each gene, the method is fully configurable to accommodate different upstream lengths and

optionally include coding sequences, making it broadly applicable to tasks such as motif discovery, comparative promoter analysis, and preparing training datasets for machine learning models of transcriptional regulation.

The system accepts one or more unannotated genome assemblies in FASTA format and a corresponding set of gene location tables in Excel format. Genomes are read and concatenated into continuous strings, ensuring accurate coordinate-based retrieval. For each gene, the start and end coordinates, along with strand orientation, are used to calculate the correct extraction window. Reverse-strand genes are automatically reverse-complemented, and all sequences are clipped to genome boundaries when upstream regions overlap the sequence origin. Users may also disambiguate multiple genomes by specifying a genome file name in the input table.

All Excel files in a batch are processed identically. Any rows marked as comments (prefixed with "#") or columns beginning with "#" are removed to preserve only relevant data. The extracted upstream (and optionally coding) sequence is appended as a new column, labeled with the upstream length for traceability. Final outputs retain all original dataset columns, augmented with the extracted sequence, and are written to Excel files with modified filenames to indicate the extraction parameters.

By automating sequence retrieval in a batch-compatible format, this workflow eliminates manual coordinate lookups, minimizes errors, and greatly accelerates preparation of sequence datasets for downstream analyses. While designed for upstream regulatory region extraction, the method can also be adapted to obtain downstream sequences, intergenic regions, or any arbitrary genomic interval defined by start and end coordinates.

4.14 Increased Efficiency

The full efficiency improvements enabled by this toolkit are challenging to quantify directly. However, for certain steps, like sequence domestication, the time savings are both measurable and dramatic.

4.14.1 Sequence Domestication

In a trial using seven sequences of various lengths, it took 51 min and 33 s to manually remove internal cut sites, add correct MoClo ends, and break the parts down into synthesizable fragments. In contrast, the sequence domestication tool completed the same task in just 4.85 s, roughly 0.1% of the time.

4.14.2 Combinatorial Assembly Simplification

Similarly, the toolkit streamlines combinatorial Echo assembly. What can be described in just 8 lines of input for the tool expands to 128 lines when fully written out manually, making it significantly harder to interpret and gain an overview. Writing the full Echo format took 16 min and 51 s, whereas the tool completed it in only 5.06 s (0.5% of the time). Beyond time savings, the tool also prevented critical errors. In the manual setup, three issues were identified:

1) A reagent name typo
2) A volume typo requesting 1000 µL instead of 100 µL
3) A calculation error resulting in insufficient enzyme supply

Each of these would have caused the Echo to terminate mid-run, requiring a full restart, additional reagents, and updating the source plate file with the new volumes. Instead, the tool flagged all errors during setup, allowing the run to be completed successfully in a single attempt.

4.14.3 Easier Robotic Setup

As well as the Echo, the toolkit also integrates with robotic platforms like Tecan, PIXL, and Mantis to facilitate experimental setup. These tools reduce both the time required and the expertise needed, encouraging users to opt for robotic execution over manual handling. In medical sample processing pipelines, this shift has led to a greater than 95% (Holman et al. 2002) reduction in error rates. Moreover, most of the remaining errors in multi-robot workflows stem from manual data transfers between automated steps (Lam and Jacob 2012), a vulnerability the toolkit helps eliminate. As a result, error rates are reduced on multiple fronts.

4.14.4 Reduction in Reagent Use: Miniaturization and Design Strategies

By lowering the barriers to robotic use, the toolkit also enables experimental miniaturization, conserving valuable reagents. It further enhances experimental coverage by supporting advanced design strategies like DOE. For instance, testing all combinations of 10 binary factors (e.g., gene knockouts) would require 1024 experiments. Using the DOE tool and the Expression Binary Factor tool, this number is reduced to just 32 experiments, without compromising analytical power. These experiments can be executed using robots like the Mantis, eliminating the need for manual reformatting and reducing the risk of errors. Robots also outperform humans in dispensing liquids, especially when volumes vary widely (Kong et al. 2012; Chai et al. 2013; Guan et al. 2023). Additionally, the Mantis system minimizes dead volume and reduces plasticware usage compared to manual workflows, resulting in meaningful cost savings (Formulatrix 2025). While DOE

tools offer powerful optimization capabilities, many researchers avoid them due to the tedious pipetting and complex setup they often require, barriers that are effectively removed when workflows are directly integrated with the Mantis.

4.14.5 Lower Entry Barrier: Streamlined Analysis Tools

The toolkit's analysis suite (Binary and Continuous Expression Factor, Clustering, Matching, Gene ID, and Promoter Region) accelerates data processing while lowering the barrier to entry. Users no longer need extensive coding experience (Mangul et al. 2019) in R, Python, or Stata, nor deep statistical knowledge (Leek and Peng 2015; Weissgerber et al. 2016). Additionally, the tools minimize errors from copy–paste mistakes or typos (Sandve et al. 2013; Ivimey-Cook et al. 2023; Veseli et al. 2025), further improving reliability.

4.14.6 Capabilities Conclusion

Collectively, these capabilities result in broader accessibility, improved reliability, and more efficient execution of experimental workflows. The toolkit not only saves time but also empowers users to design, execute, and analyze experiments with greater confidence and precision.

5. Discussion

The combined suite of tools described here, covering sequence domestication, Echo and Tecan plate-format generation, PIXL colony picking, DOE frameworks for both binary and continuous factors, liquid handler configuration for Mantis, expression factor importance analysis, clustering, feature matching, genome ID annotation, and upstream region extraction, represents an end-to-end computational and automation ecosystem for experimental biology.

5.1 Standardization Across Platforms and Devices

By producing unified, device-ready outputs for different hardware systems (Echo, Tecan, PIXL, and Mantis), this toolkit reduces the friction in moving between experimental planning and execution. The direct generation of machine-readable formats minimizes transcription errors, improves reproducibility, and allows the same upstream data structures to be deployed across multiple platforms. This enables labs to shift from a fragmented, per-device scripting approach to a centralized, version-controlled workflow.

5.2 Linking Experimental Design to Execution

Both DOE modules link statistical design principles directly to automated execution. This integration ensures that plates generated in Echo or Tecan tools reflect optimal experimental layouts, while also providing statistical validation (effect sizes, variance components, and interaction terms) immediately after data collection. The feedback loop between DOE output and automated liquid handling is critical for adaptive experiments, where factor importance can be reassessed in near real time.

5.3 Multilayer Data Integration

The feature-matching, genome annotation, and upstream sequence extraction tools extend the platform beyond laboratory robotics into bioinformatics. This enables direct integration of experimental readouts (e.g., expression levels, and phenotypic scores) with genomic context, regulatory regions, and functional annotations. Such a linkage supports hypothesis-driven iteration; for example, allowing researchers to identify high-performing constructs via Mantis-generated plates, map them to promoter variants, and use the upstream extraction tool to identify motifs for the next design round.

5.4 Scalable and Modular Clustering for Biological Insights

The clustering module connects high-dimensional experimental datasets (e.g., multi-condition promoter assays) to interpretable groupings. By incorporating weighted Euclidean distances and min–max scaling, the method balances condition-specific discrimination with overall comparability. This approach can be applied not only to promoter activity datasets but also to metabolomics profiles, colony growth curves, or any continuous assay readouts, enabling consistent classification logic across domains.

5.5 Enabling Closed-Loop Experimentation

Taken together, these software components make it possible to run closed-loop synthetic biology workflows.

- Design and Domestication: generate sequence variants free of problematic restriction sites.
- Automated Assembly and Screening: use Echo, Tecan, and Mantis outputs to assemble, grow, and assay constructs.
- Data Acquisition: capture colony positions with PIXL, and link experimental outputs to construct IDs.

- Statistical Analysis: apply DOE or clustering to identify key factors and promising variants.
- Bioinformatic Contextualization: map results back to genomic features, regulatory elements, and sequence motifs for rational redesign.

5.6 Broader Biological Applications

Although demonstrated here in synthetic biology and molecular cloning contexts, these tools generalize to many areas of biology. The same matching and annotation methods can link epigenomic marks to gene bodies, align metagenomic contigs to reference genomes, or associate proteomic features with pathway maps. Likewise, DOE frameworks and clustering approaches can optimize and interpret experiments ranging from fermentation conditions to cell culture media formulation.

6. Conclusion

The suite of tools and workflows described here demonstrates how targeted software development can close gaps between experimental design, laboratory automation, and downstream data analysis in biological research. We have established a unified framework for transforming heterogeneous data and operational formats into interoperable, automation-ready outputs by creating tools for sequence domestication, robotic liquid handling integration (Echo, Tecan, Mantis), colony picking (PIXL), experimental design optimization (binary and continuous variables), expression factor analysis, clustering, dataset matching, gene identification, and upstream sequence extraction.

A key strength of this approach lies in its modularity—each tool can operate independently to address a specific workflow bottleneck, yet all share a consistent philosophy: make data and protocols both human-readable and machine-ready. This enables researchers to rapidly transition from conceptual experimental designs to executable robotic protocols, while maintaining traceable metadata and reproducible computational steps.

Importantly, these tools directly address a persistent challenge in modern biology: siloed data. Experimental outputs, instrument logs, and analytical results are often stored in incompatible formats across different systems, preventing their reuse or integration. By ensuring outputs follow FAIR (Findable, Accessible, Interoperable, and Reusable) principles, we promote data longevity, cross-platform compatibility, and reproducibility. This is particularly critical in multi-institutional projects where consistent data structures allow teams to merge results without loss of fidelity or costly manual reformatting.

The tools and workflows described in this framework are not merely conveniences. They serve as essential safeguards against the inefficiencies and data degradation that continue to hinder progress in biological research. Without a unified system for converting experimental designs into automation-ready and interoperable outputs, laboratories risk becoming sources of delay rather than engines of innovation. Fragmented data, incompatible formats, and manual protocol translation do more than slow progress. They actively obstruct it. Critical insights are lost during transitions, reproducibility is compromised, and collaboration across institutions becomes increasingly difficult.

Failure to adopt these solutions leads to compounding consequences. Research teams will continue to devote excessive time to managing data rather than interpreting it. Instrument logs and experimental results will remain confined to proprietary formats, making long-term reuse nearly impossible. This not only limits innovation but also undermines confidence in published findings. In collaborative projects across institutions, the absence of standardized data structures results in costly delays, duplicated efforts, and outcomes that cannot be reliably merged or compared.

In addition, the lack of portable and automation-friendly tools reinforces dependence on specific vendors. This restricts laboratories to narrow ecosystems and discourages experimentation across platforms. Smaller laboratories may find themselves unable to participate in high-throughput methods, which increases the divide between well-funded institutions and those with limited resources. Over time, this fragmentation threatens to create a stratified research environment in which only a select group of laboratories can fully engage in advanced scientific exploration.

By contrast, the adoption of modular and interoperable tools enables a future in which experimental design, robotic execution, and computational analysis are seamlessly integrated. This approach supports not only efficiency but also resilience, scalability, and scientific integrity. The decision is not between convenience and rigor. It is a choice between stagnation and meaningful progress.

7. Future Outlook

The next phase of development for this framework will focus on scalability, accessibility, and seamless integration with a wide range of laboratory instruments. To overcome the current fragmentation caused by proprietary control formats and poorly documented APIs, the framework will introduce modular adapters and push for standardization at the robot-control level. This shift would enable direct communication between diverse platforms (such as liquid handlers, colony pickers,

and assay systems) without costly custom integrations. Inspired by standards like STEP-NC in Computer Numerical Control (CNC) manufacturing (Latif et al. 2021), a shared protocol for low-level instructions would allow workflows to be ported across hardware with minimal reconfiguration, making high-throughput automation more attainable for labs with varying resources.

In parallel, the framework will incorporate a centralized library of validated, FAIR-compliant protocols for common synthetic biology tasks, reducing duplication and fostering collaboration. AI and machine learning will further enhance the system by enabling predictive modeling, real-time optimization, and adaptive experimentation. These tools will analyze intermediate results and suggest protocol adjustments on the fly, improving reproducibility and accelerating discovery. Together, these advancements will transform the framework into a flexible, intelligent platform that empowers labs of all sizes to tackle complex biological challenges with speed and precision.

These enhancements align closely with DoD priorities by enabling rapid decision-making, resilient biomanufacturing, and flexible deployment of experimental workflows. Integrating AI into the framework allows researchers to analyze data in real time and adjust protocols dynamically, which is critical for developing countermeasures against emerging biological threats or optimizing biosynthetic production in unpredictable environments. The modular, hardware-agnostic design ensures that instruments can be swapped or reconfigured and mitigates vendor lock-in. Furthermore, it enables seamless collaboration across labs, agencies, and allied partners. By standardizing low-level control and enabling seamless interoperability, the framework allows labs to move at the speed of software. This means they can rapidly adapt to new challenges, deploy solutions across diverse platforms, and accelerate the pace of discovery in mission-critical contexts.

8. References

Bailey J, Lai J, Khan R, Lesnick J. Nanoliter scale DNA assembly utilizing the NEBuilder HiFi cloning kit with the Echo 525 liquid handler. Beckman Coulter, Inc.; c2025 [accessed 2025 Sep 9]. https://www.beckman.com/resources/reading-material/application-notes/nanoliter-scale-dna-assembly-with-the-echo-525-liquid-handler

Beckman Coulter, Inc. Echo 525 acoustic liquid handler. Beckman Coulter, Inc.; c2025 [accessed 2025 Sep 9]. https://www.beckman.com/liquid-handlers/echo-525

Bird JE, Wright JM, Giachino A. A user's guide to Golden Gate cloning methods and standards. ACS Synthetic Biology. 2022;11(11):3551–3563. https://doi.org/10.1021/acssynbio.2c00355

Bolyen E et al. Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. Nature Biotechnology. 2019;37:852–857. https://doi.org/10.1038/s41587-019-0209-9

Buecherl L et al. Synthetic biology open language (SBOL) version 3.1.0. Journal of Integrative Bioinformatics. 2023;20(1).

Chai SC, Goktug AN, Cui J, Low J, Chen T. Practical considerations of liquid handling devices in drug discovery. In: El-Shemy HA, editor. Drug Discovery. Rijeka: IntechOpen; 2013 Jan 23.

Culture Biosciences. Synthetic biology applications. Culture Biosciences; c2025 [accessed 2025 Sep 9]. https://www.culturebiosciences.com/applications/synthetic-biology

Eilbeck K et al. The sequence ontology: a tool for the unification of genome annotations. Genome Biology. 2005;6(5):R44. https://doi.org/10.1186/gb-2005-6-5-r44

Formulatrix. MANTIS liquid dispenser. Formulatrix; c2025 [accessed 2025 Sep 9]. https://formulatrix.com/liquid-handling-systems/mantis-liquid-dispenser/

Guan XL, Chang DPS, Mok ZX, Lee B. Assessing variations in manual pipetting: an under-investigated requirement of good laboratory practice. Journal of Mass Spectrometry and Advances in the Clinical Lab. 2023;30:25–29. https://doi.org/10.1016/j.jmsacl.2023.09.001

Hackler GCH. DEVCOM. Personal communication, 2025.

HAMILTON. Microlab STAR line. Hamilton Company; n.d. [accessed 2025 Sep 9]. https://www.hamiltoncompany.com/Microlab-STAR

Holman JW, Mifflin TE, Felder RA, Demers LM. Evaluation of an automated preanalytical robotic workstation at two academic health centers. Clinical Chemistry. 2002;48(3):540–548.

Huson DH, Weber N. Microbial community analysis using MEGAN. Methods in Enzymology, Ch. 21; vol 531. In: DeLong EF, editor. Academic Press; c2013. p. 465–485. https://doi.org/10.1016/b978-0-12-407863-5.00021-6

INTEGRA. Assist plus. INTEGRA; c2025 [accessed 2025 Sep 9]. https://www.integra-biosciences.com/united-states/en/pipetting-robots/assist-plus

Ivimey-Cook ER et al. Implementing code review in the scientific workflow: insights from ecology and evolutionary biology. Journal of Evolutionary Biology. 2023;36(10):1347–1356.

Kong F, Yuan L, Zheng YF, Chen W. Automatic liquid handling for life science: a critical review of the current state of the art. SLAS Technology. 2012;17(3):169–185.

Lam CW, Jacob E. Implementing a laboratory automation system: experience of a large clinical laboratory. Journal of Laboratory Automation. 2012;17(1).

Latif K, Adam A, Yusof Y, Kadir AZA. A review of G code, STEP, STEP-NC, and open architecture control technologies based embedded CNC systems. The International Journal of Advanced Manufacturing Technology. 2021;114:2549–2566.

Leek JT, Peng RD. Statistics: P values are just the tip of the iceberg. Nature. 2015;520(7549):612.

Loh W-L. On Latin hypercube sampling. Annals of Statistics. 1996;24(5):2058–2080.

Madsen C et al. Synthetic Biology Open Language (SBOL). Ver. 2.3. Journal of Integrative Bioinformatics. 2019;16(2).

Mangul S et al. Challenges and recommendations to improve the installability and archival stability of omics computational tools. PLOS Biology. 2019;17(6):e3000333.

Mosca E, Szigeti F, Tragianni S, Gallagher D, Groh G. SHAP-based explanation methods: a review for NLP interpretability. Proceedings of the 29th

International Committee on Computational Linguistics; 2022 Oct; Gyeongju, Republic of Korea.

New England Biolabs. NEBuilder HiFi DNA assembly. New England Biolabs; c2025 [accessed 2025 Sep 20]. https://www.neb.com/en-us/applications/cloning-and-synthetic-biology/dna-assembly-and-cloning/nebuilder-hifi-dna-assembly

Opentrons. Opentrons compare robots. Opentrons Flex. OT-2. Opentrons; c2025 [accessed 2025 Sep 9]. https://opentrons.com/ot-2/

Ren Q, Ma A, Ma Q, Zou Q. Clustering and classification methods for single-cell RNA-sequencing data. Briefings in Bioinformatics. 2020;21(4):1196–1208.

Sandve GK, Nekrutenko A, Taylor J, Hovig E. Ten simple rules for reproducible computational research. PLOS Computational Biology. 2013;9(10):e1003285.

TECAN. Fluent automation workstation. TECAN; n.d. [accessed 2025 Sep 9]. https://lifesciences.tecan.com/fluent-laboratory-automation-workstation

Thompson JC, Zavala VM, Venturelli OS. Integrating a tailored recurrent neural network with Bayesian experimental design to optimize microbial community functions. PLOS Computational Biology. 2023;19(9):e1011436.

TWIST Biosciences. Codon optimization tool. TWIST Biosciences; c2025. [accessed 2025 Sep 9]. https://www.twistbioscience.com/faq/using-your-twist-account/what-does-twist-codon-optimization-tool-do

Veseli I, Kananen K, Eren AM, Bradley PJH. Coding mistakes in bioinformatics, their consequences for research, and how to correct them. The anvi'o Project; n.d. [accessed 2025 Sep 9]. https://anvio.org/blog/bioinformatics-bugs/

Weissgerber TL et al. Reinventing biostatistics education for basic scientists. PLOS Biology. 2016;14(4):e1002430.

List of Symbols, Abbreviations, and Acronyms

AI	artificial intelligence
ANOVA	analysis of variance
API	Application Programming Interface
A, T, G, and C	adenine, thymine, guanine, and cytosine
bp	base pair
CDS	Coding DNA Sequence
ChIP-seq	Chromatin Immunoprecipitation Sequencing
CNC	Computer Numerical Control
CSV	Comma Separated Values
DBTL	design–build–test–learn
DEVCOM	U.S. Army Combat Capabilities Development Command
df	DataFrame
DNA	deoxyribonucleic acid
DoD	Department of Defense
DOE	design of experiments
FAIR	Findable, Accessible, Interoperable, Reusable
FASTA	FAST-ALL (DNA sequence format)
FCA	Flexible Channel Arm
GFF	general feature format (DNA sequence format)
GUI	graphical user interface
HEPA	high-efficiency particulate air
HTML	Hypertext Markup Language
ID	identification
IDT	Integrated DNA Technologies (DNA synthesis company)
JSON	JavaScript Object Notation
LHS	Latin Hypercube Sampling

min–max	minimum–maximum
MoClo	modular cloning
MSE	mean squared error
NEB	New England Biolabs
OLS	ordinary least squares
PCR	polymerase chain reaction
PNG	Portable Network Graphics
R^2	R-squared
RMSE	root MSE
RNA	ribonucleic acid
SBOL	Synthetic Biology Open Language
SBS	Society for Biomolecular Screening (plate sizing standard)
SHAP	SHapley Additive exPlanations
SO	Sequence Ontology
TSV	Tab Separated Values

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB