



AFRL-RI-RS-TR-2023-059

AN ADAPTIVE PIPELINE FROM SCIENTIFIC DATA TO MODELS

DUKE UNIVERSITY

MARCH 2022

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2023-059 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

STEVEN L. DRAGER
Work Unit Manager

/ S /

GREGORY J. HADYNSKI
Assistant Technical Advisor
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE		2. REPORT TYPE		3. DATES COVERED	
MARCH 2023		FINAL TECHNICAL REPORT		START DATE SEPTEMBER 2017	END DATE SEPTEMBER 2022
4. TITLE AND SUBTITLE AN ADAPTIVE PIPELINE FROM SCIENTIFIC DATA TO MODELS					
5a. CONTRACT NUMBER FA8750-17-C-0054		5b. GRANT NUMBER N/A		5c. PROGRAM ELEMENT NUMBER 61101E	
5d. PROJECT NUMBER		5e. TASK NUMBER		5f. WORK UNIT NUMBER R2DQ	
6. AUTHOR(S) Steven B. Haase, Ph.D.					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Duke University 2200 W. Main St STE 710 Durham NC 27708				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RITA 525 Brooks Road Rome NY 13441-4505			10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI & DARPA		11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-RI-RS-TR-2023-059
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Under the Defense Advanced Research Projects Agency's Synergistic Discovery and Design program, the Duke Team, composed of scientists from Duke, Rutgers, Montana State, and Florida Atlantic Universities, as well as Geometric Data Analytics, and Netrias, Inc., broadly researched and developed data driven techniques for scientific discovery and robust design, proving feasibility through program challenge problems with Yeast States, Novel Chassis, Protein Stability, and Perovskite. Their efforts developed an adaptable computational pipeline with approaches, methods, and tooling which learns the structure and function of regulatory networks for model construction from time-series data, providing an ability to utilize high-fidelity models and simulations to account for process perturbations and component interactions for robust scientific discovery and system design. Aside from their innovative Dynamic Signatures Generated by Regulatory Networks (DSGRN), tools were developed for automating data pre-processing, normalization, quality control, scientific extraction, data aggregation, and data analyses; accelerating the design, build, test, learn loop; and transitioned to the National Institute of Health's Accelerating COVID-19 Therapeutic Interventions and Vaccines Program. Additionally, when it became clear that the foundry-style, high-throughput laboratories charged with duties of experimentation and data collection did not have sufficient capabilities to produce some data types and perform developmental protocols, Duke Team members stepped in with their own wet-lab capabilities to fulfill the gap in data collection. This work facilitated new analyses aimed at integrating data from both high-throughput and benchtop laboratories, substantially extending the capabilities of the SD2 architecture with the innovative extension of Aquarium for the benchtop.					
15. SUBJECT TERMS Dynamic Signatures Generated by Regulatory Networks (DSGRN), Data Driven Techniques for Scientific Discovery, Robust Design, Automated Data Aggregation of the Design Build Test Learn Loop, Automating Scientific Extraction					
16. SECURITY CLASSIFICATION OF:				17. LIMITATION OF ABSTRACT	
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	SAR		18. NUMBER OF PAGES 86
19a. NAME OF RESPONSIBLE PERSON STEVEN L. DRAGER					19b. PHONE NUMBER (Include area code) N/A

Table of Contents

List of Figures	v
List of Tables	vii
1.0 Summary	1
1.1 Data Acquisition: Merging High-Throughput Automated Experimentation with Low-Throughput Benchtop Data Acquisition	1
1.2 Robustness in Synthetic Biology	1
1.3 Tools for Automated Data Pre-Processing, Normalization and Quality Control	2
1.4 Tools for automated data aggregation and acceleration of Design, Build, Test, Learn loop	2
1.5 Tools for Data Analyses and Automation.....	3
1.6 Automating Scientific Knowledge Extraction	5
1.7 Transitioning Synergistic Discovery and Design Tools and Ecosystem to Accelerating COVID-19 Therapeutic Interventions and Vaccines Program Performers	5
2.0 Introduction.....	6
3.0 Methods, Assumptions, and Procedures	7
3.1 Data Acquisition: Merging High-Throughput Automated Experimentation with Low-Throughput Benchtop Data Acquisition	7
3.1.1 Aquarium at Duke.....	7
3.2 Robustness in Synthetic Biology	7
3.2.1 Dynamic Signatures Generated by Regulatory Networks: Predicting the Robustness of Circuit Function Across Parameter Space/Growth Conditions	7
3.2.2 Diagnosing Circuit Failure.....	12
3.2.3 DSGRN Functionality outside of Gene Regulatory Networks	13
3.2.4 DSGRN: Designing Circuits that Function Robustly Across Growth Conditions ..	14

3.2.5	SD2 Tool Integration	20
3.2.6	Network Discovery	21
3.3	Tools for Automated Data Pre-Processing, Normalization and Quality Control	22
3.3.1	Transcriptomics Processing	22
3.3.2	Processing RNA-seq Data.....	23
3.3.3	gRNA Quantification for CRISPR circuits.....	23
3.3.4	Processing Microarray Data.....	25
3.3.5	RNA-seq & Microarray Data: Normalizing Transcriptomics across platforms	26
3.3.6	Quality Control Reporter for RNA-seq.....	28
3.3.7	Flow Cytometric Data Analyses: Removing Dead Cells from the Analyses (Autogater).....	28
3.3.8	Protein Design CP, Data Processing Pipelines	30
3.3.9	Versioned Dataframe Repository.....	31
3.3.10	Build Quality Control Unit	32
3.3.11	Signal Prediction.....	32
3.3.12	Replicate Checking	35
3.4	Tools for automated data aggregation and acceleration of Design, Build, Test, Learn loop	37
3.4.1	Build Request and Build Request Parser	37
3.4.2	Data Converge	37
3.4.3	Precomputed Data Table.....	38
3.5	Tools for Data Analyses and Automation.....	39
3.5.1	Performance Metrics.....	39
3.5.2	Data Diagnosis	41
3.5.3	Yeast States Analysis Pipeline.....	43

3.5.4	Circuit Scoring	45
3.5.5	Cell Growth & Cell States Transcriptomics Analysis	47
3.5.6	Cell State Markers, Max Separation	49
3.5.7	Enhanced Gene Regulation Inference Technologies	51
3.5.8	Immortality Challenge Problem Proposal.....	51
3.5.9	Perovskite Data Processing Pipelines: Accounting for Temperature Variation Across 96-Well Plates.....	52
3.5.10	Data Sharing Initiative: Improving the Reproducibility and Generalizability of Published Results	53
3.5.11	Topological Features of Protein Structure for Machine Learning Models	54
3.5.12	Topological Data Analysis: Protein Stability	55
3.5.13	Responding to Experimental Surprise: Data Driven Models for Identifying New Informative Experiments	57
3.5.14	Identification of DNA-binders from Primary Sequence	58
3.6	Automating Scientific Knowledge Extraaction	59
3.6.1	Support Data Acquisition For Mathematical Modelers	59
3.6.2	Data-Driven Rapid Adaptation of Models.....	60
3.6.3	Metrics and Evaluations On Model Performance	62
3.7	Transitioning Technology to New Stakeholders.....	64
3.7.1	National Primate Center: Tulane (NIH).....	64
3.7.2	Xilis, Inc.....	67
3.7.3	Duke External Quality Assurance Program Oversight Laboratory	67
3.7.4	Publications.....	68
4.0	Results and Discussion	69
4.1	Integrated Tools: Pipelines and Automation.....	69

4.2	Metrics for Success	69
4.3	Purpose built vs. Generalizability: Data Sharing and Transition.....	70
5.0	Conclusions.....	71
5.1	Lessons of COVID-19 Pandemic: SD2 Infrastructure for National Bio-preparedness .	71
6.0	References.....	73
	List of Symbols, Abbreviations, and Acronyms	74

List of Figures

Figure 1. Flow Diagram of Integrating Haase Lab Aquarium into the SD2 Infrastructure.....	7
Figure 2. Schematic of the DSGRN Paradigm: (a) Regulatory Network, (b)-(c) Combinatorialized Dynamic Behavior, and (d) Decomposition of Parameter Space.	8
Figure 3. Analysis of all 3 Node Networks for Robust Hysteresis.....	11
Figure 4. (Left) Original Circuit Design with Glitch, (Right) Alternative Design Produced by DSGRN.....	13
Figure 5. DSGRN Model of the Toxin-Antitoxin Two Population System.	14
Figure 6. Two Inducer FAST OFF Designs – The Repressors Dominate the Activators to Achieve the Desired Behavior in the Third and Fifth Networks, While Activators Dominate the First, Second and Fourth Networks.....	16
Figure 7. DSGRN Design Interface and Combinatorial Design Model Improvements to Process.	18
Figure 8. (Top) Original Simple Circuits, (Bottom) DSGRN+CDM Circuit Designs with Redundancy.....	19
Figure 9. Schematic of the Inherent Dynamics Pipeline.....	22
Figure 10. Small RNA-seq of yeast circuits. A) OR gate, B) Distribution of total reads, C) Percent of reads that passed QC, D) Heatmap for input state of 11.	24
Figure 11. Expected and Unexpected gRNAs for the OR Gate with an Input State of 11 from Various Resources, Where the Red “X” Indicates the gRNA is Missing from the Resource.	25
Figure 12. Set of 13 Invariant Genes Found Across RNA-seq and Microarray Experiments.....	27
Figure 13. Flow Cytometry Data from Yeast Cells Killed with Various Ethanol Concentrations; Rows are Different Ethanol Concentrations, Columns are Time Points, Purple Distributions are Cells not Stained with Sytox, and Yellow Distributions are Cells Stained with Sytox..	30
Figure 14. Initial Design of Build QC.	32
Figure 15. The Signal Prediction tool builds a controls model from cleaned flow cytometry data to improve predictions on experimental samples.....	34
Figure 16. Plot of QC Data for RNA-seq Replicates, Showing the Visualization of the Results for a Replicate and Comparing it to a Flag of Upstream QC Checks.....	36

Figure 17. (Top) Previous Method for getting from Data to Results, (Bottom) Integration of Data Converge into the Method.....	38
Figure 18. (Top) Method for getting from data to results with Data Converge integrated, (Bottom) Integration of Precomputed Data Table into the method.....	39
Figure 19. How the metrics are computed using percentiles (top) and standard deviations (bottom). The normal distribution is used purely for illustrative purposes - no assumptions are made about the distribution of the data.....	40
Figure 20. Perform Metrics plots of ON vs. OFF states for groups of data, sorted by their performance score.....	43
Figure 21. Data Diagnose showing the top variables that associate with differences in performance. The plots are ordered from most significant to least significant, top to bottom. Only the first few variables are shown in this plot.	44
Figure 22. Cut Scoring as a Weighted Complete Graph.....	46
Figure 23. (Left) A correctly performing OR circuit that is scored as OR under both the normalized cut and average cut scores. (Right) A correctly performing XNOR circuit that was misclassified as OR under the normalized cut score but was correctly identified under the average cut score.....	46
Figure 24. Heatmap of the Correlations between Differentially Expressed Genes.	49
Figure 25. One set of possible "sentinel" genes that uniquely identify different cell states to be used as fluorescent gene fusions.....	50
Figure 26. The DBTL Architecture for the Yeast States Challenge Problem.	69
Figure 27. SD2 Program Metrics.....	70
Figure 28. The Investigator-Initiated Model for Discovery.....	71
Figure 29. The SD2 Model for Discovery.	72

List of Tables

Table 1. Experiment Data Used in Analysis for Cell Growth and Cell States.	48
---	----

1.0 Summary

The Duke Team, composed of scientists from Duke University, Rutgers The State University of New Jersey (Rutgers), Montana State University, Florida Atlantic University, Geometric Data Analytics (GDA), and Netrias, Inc., has worked broadly within the Defense Advanced Research Projects Agency (DARPA) Synergistic Discovery and Design (SD2) program, contributing to efforts in the Yeast States, Novel Chassis, Protein Stability, and Perovskite challenge problems (CP).

1.1 Data Acquisition: Merging High-Throughput Automated Experimentation with Low-Throughput Benchtop Data Acquisition

The SD2 program was structured across five technical areas (TAs), TA1 – Data-Centric Scientific Discovery, TA2 – Design in the Context of Uncertainty, TA3 – Hypothesis and Design Evaluation, TA4 – Data and Analysis Hub, and TA5 – Challenge Problem Integrator, where the Duke Team supported TA1 and TA3 capabilities. Early in the program it was realized that some of the data needed to achieve goals of this effort could not be produced by the automated and semi-automated TA3 laboratories. To merge the data collected in the benchtop laboratory of the Duke Team with the automated labs, it was necessary to utilize approaches for protocol execution and data collection that were utilized by the TA3 labs. In collaboration with the University of Washington (UW) Biofab team, Aquarium for the benchtop was developed, enabling the Duke Team’s benchtop lab to execute protocols and collect data, that when delivered to the database, was indistinguishable from data collected at the automated TA3 labs.

1.2 Robustness in Synthetic Biology

Before SD2, circuit designs in synthetic biology were focused on proposed functions. Findings early in the program indicated that synthetic genetic circuits did not perform functions consistently across varying growth conditions that likely alter parameters in which the circuit functions. At that time there were no tools for designing circuits taking into account both function and robustness to growth conditions. The practical importance of robust designs lies in the ability of circuits to perform their functions in the face of varying conditions of a deployment, even within highly controlled conditions such as a fermenter.

Dynamic Signatures Generated by Regulatory Networks (DSGRN) was developed as a design tool that could computationally assess the robustness of a particular circuit topology across parameter conditions. As well, with circuit performance data, DSGRN could infer the mode of failure of circuits so that they might be “repaired”.

DSGRN has been wrapped with extensive tooling to improve the design process, especially for logic circuits with arbitrary biological parts. Multiple design problems were tackled in Yeast States and Novel Chassis including the comprehensive analysis of all three node networks for bistability, the redesign of 2-input logic circuits for enhanced robustness, the redesign of a 3-input logic circuit for glitch removal, and the analysis of experimental data from an external Department of Defense (DoD) partner. The design process was reduced from months to days with 50-100% qualitative matching to data where it was available.

Several SD2 achievements facilitated this work. An easy-to-use design tool was created that predicts the robustness of logic circuit designs given user-supplied experimental constraints and network functionality requirements. The concepts of robustness of performance incorporating design parameters, neighboring parameters, and continuation in Hill function models were improved. Concrete connections between DSGRN parameters were developed as well as build constraints that are easily communicated to experimentalists. The number of DSGRN-computable network topologies was increased, including more complex networks (more in-edges, multiple edges between nodes, self-repressors, no in-edges, no out-edges, non-monotone interactions).

With these achievements, computational predictions of the relative performance of a variety of complex network topologies under disparate experimental conditions subject to build constraints were provided.

Transition partners are able to evaluate the trade-off between robustness and complexity in a collection of networks exhibiting a desired functionality and explore the performance of previously unconsidered synthetic network models.

DSGRN specifies circuit designs at the level of circuit topology. A fully automated design tool will need to be integrated with tools with parts-level recommendations for realization of the robust topologies such as Combinatorial Design Model (CDM) from Netrias and Cello from Boston University (BU). Initial steps towards integration were made but not completed.

1.3 Tools for Automated Data Pre-Processing, Normalization and Quality Control

To facilitate standardized analyses, the Duke Team developed tools for automated data pre-processing, normalization, and Quality Control (QC). Tools were developed for multiple data types including Ribonucleic Acid Sequencing (RNA-seq), Microarray, flow cytometry and protein stability. These tools were integrated into the Design, Build, Test, Learn (DBTL) loop to improve the speed and accuracy of automated analyses.

1.4 Tools for automated data aggregation and acceleration of Design, Build, Test, Learn loop

As the development of the SD2 Database progressed, it became clear that moving from data to analysis to results was very slow and not reproducible. Complicated, evolving data was run through a variety of changing scripts, without standardization or provenance tracking. Analysis was manually run, custom built for each question, and used different data sets with different formats. There were different measures of performance, different aggregations of data, and a lack of automated tools for debugging experiment issues. When multiplied by the number of analysts, the time wasted writing customized scripts was substantial, and the confusion about different performance measures and aggregations created confusion. Without documented versioned data, it was unclear how to compare the outputs of analyses to accurately assess reproducibility.

Data Converge (DC) and the Precomputed Data Table (PDT) were created with the help of Netrias, Inc. These are a system of processing and analysis tools, in containers/apps, that are automated in a pipeline, use standardized/versioned data and analysis, and organize and store output and provenance. It enabled faster and more reproducible analyses, added new analyses faster,

and responded more rapidly to new experimental data. The metrics show reduced time from data generation to analysis results, from months to hours.

Data Converge is a tool designed as a data access layer (DAL) that gathers data from files and databases and formats it for analysts and other tools. DC provides a flexible bridge between changing, complicated data sources and multiple data consumers that require structured, consistent data with provenance. DC has saved analysts hundreds of development/maintenance hours for their analysis tools. It has also increased reproducibility, going from each tool/analyst separately using data to now all challenge problem – Yeast States (CP-YS) analysts/tools, and other CPs, using standardized data products.

The Precomputed Data Table automatically executes analyses and provides versioned, organized, analysis-ready, precomputed data for custom analyses. Before the PDT, analysts performed their own computations of values such as growth rate. These values are commonly used as features in more sophisticated machine learning (ML) analyses by multiple analysts, so having them computed automatically saves time. As well, the method for computing such values may vary from analyst to analyst, adding unnecessary variance in analyses. Thus, the PDT saves time and increases reproducibility across multiple analyses. In conjunction with Data Converge, analysis time was reduced from days to hours.

Transition partners see value in a well-integrated, automated, and extensible data processing and analysis system that improves speed and reproducibility of analysis of complex data sets.

What was significantly harder than expected was collecting, storing, and providing any information on analysis intent. It was also difficult to create standardized formats needed for a variety of analysts from the flexible data descriptions for different experiments. For increased scale, analysis intent and standardized formats for analysis are dependent on data and experiment type. Thus, a more generalized and automated method for defining and consuming these definitions would reduce human configuration and make analysis faster and more flexible.

A fully operational version of PDT should have a Quality Control module to indicate some measure of data quality.

1.5 Tools for Data Analyses and Automation

Several tools were developed to fill needs in the Design, Build, Test loop of CP-Yeast States. A method to assess the performance of circuits was developed to measure the distance between fluorescence distributions of control and experimental populations in flow cytometric analyses (Wasserstein Distance). Another method to assess circuit performance is a tool called Performance Metrics (PM) which consists of a suite of performance metrics that compare ON vs. OFF states at several thresholds and across different groupings of samples and is used to assess circuit performance in the DBT loop. Wasserstein Distance, Performance Metrics and an entropy-based method for circuit scoring were all used together to score circuit performance. Data Diagnosis is a trouble shooting tool that identifies variables associated with variation in performance and identifies dependent variables that may cause redundancies in analysis. This diagnosis tool sits upstream of the design module in the DBT so that new designs can avoid variables that lack robustness (e.g. poorly performing promoter sequence).

Before SD2, Rosetta was the modeling suite of choice for tools that allowed researchers to explore and predict biophysical properties of proteins (e.g. stability) from primary amino acid sequence. Over a few weeks of model conceptualization and implementation the team developed a data-driven ML method employing a novel feature-learning technology called Cover-tree differencing via entropy reduction (CDER) to learn stability-discriminating topological/structural features. The team found that a small number (6-21) of completely data-driven, shape-based features could be used on their own to achieve 85-90% of the performance on stability classification as state-of-the-art models trained on ~100 expert-curated biophysical features. Moreover, by providing a quantified, interpretable characterization of protein structure, these features revealed the intuitive biological insight that more stable designs have an increased number of smaller voids and fewer large voids than unstable designs.

While exploring the use of Topological Data Analysis (TDA) methods to predict protein stability, the team found that several published methods of transforming topological features into machine-learning-ready vectors required *ad hoc* choices of parameters, which cannot be made ahead of time in a principled way. Also, some methods proved inefficient to compute on the massive, designed protein datasets.

The SD2 innovations that made a real difference were two technology solutions which promise to enhance the application of TDA methods to ML problems across domains and beyond the SD2 program. The team developed software to automatically optimize the choices of parameters determining the representation of TDA features simultaneously with the choice of ML model and its hyperparameters. The TopOpt tool reduces uncertainty and makes model selection systematic and automatic, thereby massively reducing the human effort required for complex model development and exploration. The team also optimized components of the tool, reducing compute time on High Performance Computers (HPCs) by as much as 135-fold as compared to publicly available versions. The team has also hardened these tools, making them scalable, feature-rich, and user-friendly, and are transitioning them into a widely used open-source ecosystem to make these tools easily accessible to the broad community of researchers and modelers using TDA methods.

Part way into the SD2 program, this effort aimed to identify limitations in the Rosetta modelling framework and potentially discover previously unknown principles of protein stabilization. Towards these ends, the team developed a machine learning approach to identify unstable designs which were proximal to highly stable designs in biophysical feature space. It was hypothesized that these “unexpected unstable designs” would be highly informative since their instability may have been due to seemingly small, but highly destabilizing errors in the amino-acid sequences generated by Rosetta.

The team’s data-driven approach suggested 21 from ~16,000 of the original designs to re-assay for stability following exhaustive single-amino-acid mutations. Excitingly, for most designs, there were single amino-acid mutations that rescued the design, making it stable. Many of these stabilizing mutations were found to relieve energetically unfavorable steric clashes that Rosetta had designed in the protein core, suggesting a physical inaccuracy in Rosetta’s energy function. This observation led to re-tuning Rosetta’s energy function parameters, which resolved the bias and substantially improved Rosetta’s performance in a standard set of benchmarks used to quantify Rosetta’s physical accuracy.

Previous to SD2 yeast cell viability was assayed using a variety of fluorescence dyes that could be analyzed by flow cytometry. In close collaboration with Smart Information Flow Technologies (SIFT) and Netrias, this effort helped develop a new ML approach to assess live and dead populations of cells using data from all channels of the flow cytometer. The Duke group performed a variety of experiments where cells were killed by various methods in time series and then analyzed by flow cytometry as well as assessing viability at each time point by a gold standard method of determining the percentage of colony-forming units. The ML method developed by Netrias and SIFT was able to identify dead (and likely dying) cell populations with high accuracy as compared to colony forming units. This method is simpler, faster and cheaper than traditional methods using expensive dyes that require extra steps in the protocol. Transition partners appreciate the speed and ease of this method. The team currently has interest from Thermo Fisher, who make flow cytometers and package analysis software with the dedicated computer systems that accompany the cytometer. Fully realized versions will extend the results across cell types from organisms from bacteria, to plants, to humans.

1.6 Automating Scientific Knowledge Extraction

The team extended SD2 tools for the DARPA Automating Scientific Knowledge Extraction (ASKE) program aimed at increasing the speed and accuracy of mathematical modeling. Tools were developed to support data acquisition for mathematical modelers and for data driven approaches for rapid adaptation of models. Approaches were also developed for evaluating model performance.

1.7 Transitioning Synergistic Discovery and Design Tools and Ecosystem to Accelerating COVID-19 Therapeutic Interventions and Vaccines Program Performers

Duke University has been leading a transition effort that may eventually include the bulk of the SD2 ecosystem to an effort at Tulane University sponsored by the National Institutes of Health (NIH) Foundation's Accelerating Coronavirus Disease 2019 (COVID-19) Therapeutic Interventions and Vaccines (ACTIV) program. This effort is a consortium of 7 National Primate Centers doing harmonized research on COVID-19 therapeutics and vaccines in non-human primate systems. Tulane is the coordinating center and will be collecting data from all other data-providing centers in order to archive and organize data for analysis by ML/Artificial Intelligence (AI) approaches. The challenge here will be to port the SD2 socio-technical system and expand SD2 tooling to accommodate new data types and analyses. Initial efforts have focused on automating high-dimensional flow cytometric analyses. Additional transitions include Xilis, Inc. and the External Quality Assurance Program Oversight Laboratory (EQAPOL) Program at Duke.

2.0 Introduction

The Duke Team was comprised of scientists from Duke University, Rutgers The State University of New Jersey, Montana State University, Florida Atlantic University. The team also included scientists from Geometric Data Analytics, and was eventually joined by Netrias, Inc. The expertise of the team was primarily in mathematics and computational biology but performer Haase, principal investigator (PI), was trained in yeast genetics and genomics. Previous to SD2, members of the Duke Team have worked together on multiple quantitative biology projects sponsored by the NIH, National Science Foundation (NSF), and DARPA.

As a group, the team brought a deep understanding of gene regulatory networks (circuits) that evolved to control the timing of events during yeast cell growth and division. The Duke TA1 team utilized deep knowledge of evolved circuits to develop tools for the evaluation and design of synthetic genetic circuits in both the Yeast States and Novel Chassis challenge problems. PI Haase served as co-lead for the Yeast States effort and helped direct efforts within that challenge problem towards designing synthetic genetic circuits that would be robust across growth conditions that might be encountered in a field deployment (e.g. elevated temperature, dehydration, low nutrients). Duke Team performers also developed computational tools and pipelines for data normalization and pre-processing and data analyses that were also deployed on problems within the Protein Design and Perovskite challenge problems.

Part way into the program, it became clear that the foundry-style, high-throughput laboratories that were charged with TA3 duties of experimentation and data collection did not have sufficient capabilities to produce some data types as well as performing developmental protocols. Thus, the Duke Team stepped in with its own wet-lab capabilities to fulfill the gap in data collection. This work facilitated new analyses aimed at integrating data from both high-throughput and benchtop laboratories, substantially extending the capabilities of the SD2 architecture.

In the sections following, this report documents the efforts of the Duke Team to develop approaches, methods, and tooling to achieve goals broadly across all challenge problems.

3.0 Methods, Assumptions, and Procedures

3.1 Data Acquisition: Merging High-Throughput Automated Experimentation with Low-Throughput Benchtop Data Acquisition

3.1.1 Aquarium at Duke

Transitioning Aquarium to academic labs at Duke University was important for the SD2 program as it demonstrated transferability of Aquarium as well as enabled Duke University with low-to-mid throughput TA3 capabilities. Aquarium was initially set up in Steve Haase's lab, starting with several basic protocols ported from BioFab's Aquarium instance. Some of the protocols, such as making media, were updated at Duke with Haase Lab-specific steps. The Haase Lab times series RNA-seq protocol was the first protocol to be converted into an Aquarium protocol. BioFab representatives visited the Haase Lab and a dry-run demo of the time series RNA-seq protocol was performed for them. During Phase II of the SD2 program, new experimental protocols at Strateos were designed, e.g., Dose Response and Growth Curves. A Dose Response protocol in Aquarium was created for use in the Haase Lab and successfully tested using yeast circuits sent by BioFab. The next steps for fully integrating the Haase Lab into the SD2 infrastructure was enabling Experimental Requests to be submitted to Aquarium instances and for data generated and saved into Aquarium to be uploaded to Texas Advanced Computing Center's (TACC) Data Catalog. Figure 1 shows a flow diagram of the full integration using the Dose Response protocol and flow cytometry data, with red labeling indicating yet-to-be developed functions. This project was halted as the core Aquarium developers at BioFab moved on.

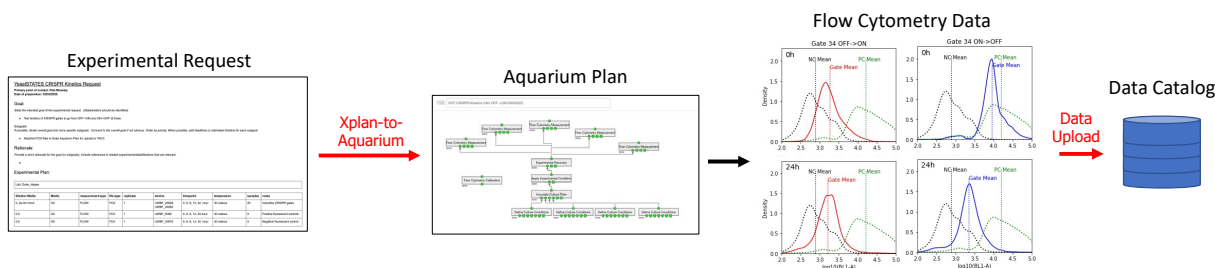


Figure 1. Flow Diagram of Integrating Haase Lab Aquarium into the SD2 Infrastructure.

3.2 Robustness in Synthetic Biology

3.2.1 Dynamic Signatures Generated by Regulatory Networks: Predicting the Robustness of Circuit Function Across Parameter Space/Growth Conditions

Dynamic Signatures Generated by Regulatory Networks is both a theoretical framework and software tool that comprehensively models the dynamic behavior of genetic networks. To apply DSGRN confidently and successfully to the analysis and design of circuits requires that at a minimum the following four conditions are satisfied: (i) a theoretical mathematical foundation, (ii) sufficient modeling capabilities, (iii) computability, and (iv) the ability to directly relate DSGRN output with experimental data. The Rutgers group focused on the first three conditions but pro-

vided specific code in support of the fourth condition. The DSGRN software can be found at <https://github.com/marciogameiro/DSGRN>.

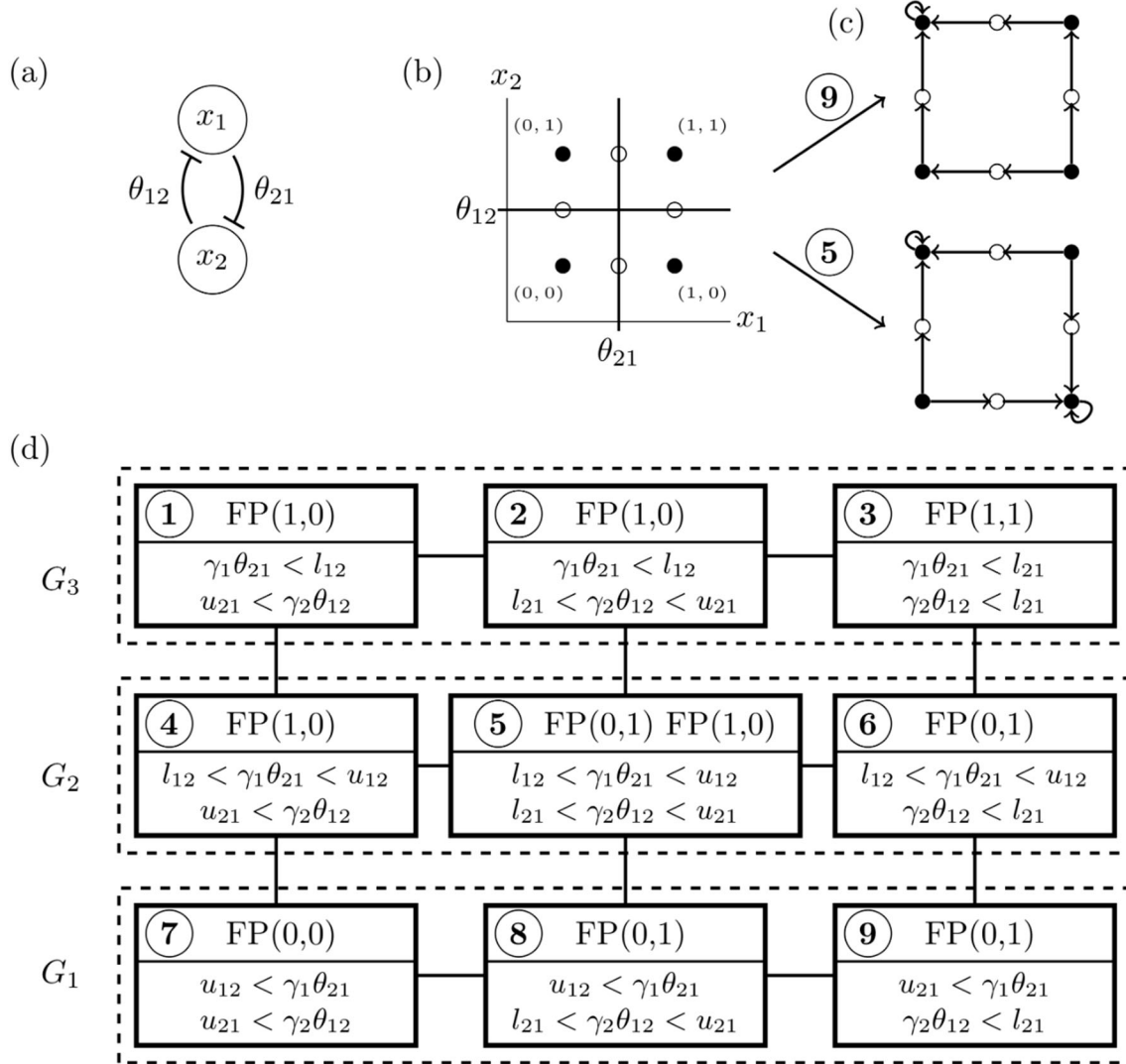


Figure 2. Schematic of the DSGRN Paradigm: (a) Regulatory Network, (b)-(c) Combinatorialized Dynamic Behavior, and (d) Decomposition of Parameter Space.

The discussion begins by discussing the theoretical foundations. DSGRN takes as input an annotated directed graph and interaction functions assigned to each node, as shown in Figure 2. The systems biology interpretation is that the directed graph represents a regulatory network. The nodes indicate biochemical species, e.g., proteins or Messenger Ribonucleic Acid (mRNA), the edges indicate whether one species up or down regulates the production of another species, and the interaction function indicates how multiple influences at a single node are processed. The output of DSGRN is a finite decomposition of an associated parameter space such that for every

region of this decomposition there is a valid combinatorial description of the global dynamics. Thus, DSGRN provides a purely combinatorial description of what is typically assumed to be a continuous process. The combinatorial structure allows for extremely efficient computations. However, ensuring the mathematical validity of these computations with respect to the underlying continuous process is a challenge.

The mathematical paradigm motivating the development of DSGRN is that global dynamics can be decomposed in a robust manner using a finite lattice of attracting blocks. Having identified such a lattice, the continuous dynamics can be recovered via algebraic topological invariants (Conley indices) that can be computed from the topology of the elements of the lattice. The Conley indices provide information about the dynamics of the Ordinary Differential Equations (ODEs), e.g. existence of fixed points, periodic orbits, heteroclinic connections, and even chaotic dynamics. Furthermore, the topological information in the lattice of attracting blocks can be codified via a chain complex where the chains are Conley indices of Morse sets and the boundary operator is called a connection matrix.

A major effort that was completed with partial support of SD2 was formalizing this mathematical paradigm. This was done using the language of order theory. Theorems were produced about the structures of lattices of attracting blocks and attractors, producing theorems about the corresponding partial order structures, Morse tiles and Morse decompositions, respectively. These structures and theorems were done both on the continuous and on the combinatorial level and theorems obtained relating the continuous with the combinatorial structures.

Specifically, the theory of connection matrices was redeveloped, recasting it in the language of combinatorial cell complexes. In this setting, using techniques from discrete Morse theory provided an algorithm that begins with a lattice of attracting blocks represented via a finite cell complex and produces a minimal chain complex whose chains are derived from homological computations involving the Morse tiles and whose boundary operator is a connection matrix in the continuous setting, referring to this reduced chain complex as a Conley complex.

The team believes that via this theory, it is able to prove that given an ordinary differential equation, then the DSGRN computational machinery developed during the SD2 project will produce a rigorous homological description of the global dynamics and thus via the Conley invariants and Conley complex computations mathematical rigorous information about the dynamics of the ordinary differential equation will be obtained. Preliminary results for a particular class of ordinary differential equations have been obtained.

Now, let's discuss sufficient modeling capabilities. A description of the capabilities of DSGRN existed at the beginning of SD2. DSGRN was originally designed to model gene regulatory networks and could handle regulatory networks in which each node had at least one and at most three incoming edges and at least one and at most three outgoing edges. Research under the SD2 effort expanded the set of admissible networks significantly. The current code is now capable of handling networks with nodes that have no in edges or no out edges. More significantly it can now handle nodes with up to six input edges and arbitrarily many out edges. This latter extension represents a nontrivial accomplishment. The challenge is that the interaction functions act on values associated with in-edges to a node. The resulting value of the interaction function impacts the global dynamics by increasing or decreasing production of the species. The value of the in-

teraction function is parameter dependent. The DSGRN decomposition of parameter space is based on identifying as a function of parameter values, values on in-edges, and the interaction function whether production of the species increases or decreases. The number of regions in the decomposition grows extremely fast as a function of the number of in-edges. While there is an abstract result that guarantees that this decomposition can be computed, the worst-case cost of computation for each potential region of parameter space is doubly exponential in the number of parameters. Therefore, an alternative method of computation needed to be developed and was achieved by developing new techniques based on computational algebraic geometry.

From systems biology it is known that regulatory networks depend on both transcriptional and post-transcriptional regulation. Thus, the DSGRN framework was extended to include post-transcriptional regulation. DSGRN can now handle post-transcriptional activity such as phosphorylation and ubiquitination. The code has been written and tested and a paper justifying the modeling framework published.

Finally, there are numerous examples in biological systems, e.g., neuroscience, where the interaction between species is not monotone. The team has identified how to obtain the appropriate parameter space decomposition in this setting and is in the process of implementing the associated code. Papers justifying and demonstrating these capabilities are in process.

Considerable effort for the project revolved around developing, extending, and exploiting the computational capabilities of DSGRN. Much of the development was directed towards making DSGRN user friendly. The extensions involved developing algorithms and code to implement the extended modeling effort, and it involved developing the computational homological algebra necessary to move to the next step where the combinatorial DSGRN output can be interpreted in the context of continuous processes.

During the first stages of the SD2 project considerable effort went into building interfaces to the C++ DSGRN code to allow for ease of use, portability, the ability to query DSGRN for particular information, and parallelization. This included the introduction of Jupyter notebooks that act as user manuals (this continues to be the way that new capabilities are explained).

It is useful to think of the output of DSGRN as being organized via a graph (called the *parameter graph*) where each node of the parameter graph represents a region in parameter space over which the combinatorial dynamics is constant and an edge in the parameter graph indicates that the two associated regions share a codimension-one face. Thus, understanding how dynamics does or does not change as a function of parameters can be determined by choosing a path in the parameter graph and observing changes or lack thereof in the associated Morse graph that codifies the global dynamics. Code was developed that allows for identification and construction of neighborhoods and paths through the parameter graph while querying the associated dynamics.

Considerable time was also spent developing code for analysis of the DSGRN output using algebraic topology. Particular projects included the following:

- Identifying an appropriate cell complex based on the decomposition of parameter space so that the homology of unions of regions of parameter space can be computed. The challenge of performing the computations is that DSGRN parameter space is typically high

dimensional. Computations were able to be performed in systems involving more than 20 parameters. The biological importance is that this provides a means of identifying if a particular phenotype exists over complicated or simple regions of parameter space.

- Computation of Conley complexes. The research obtained an extremely efficient algorithm that allows one to perform on laptops computations that involve trillions of cells. Thus, the team was able to derive Conley complexes for networks on which DSGRN was routinely applied. This is not only a challenging computation because of size, but also represents a rather sophisticated use of algebraic topology. Thus, the implementation and subsequent usability of these techniques required the development of a broad range of code. The code can be found at <https://github.com/marciogameiro/pyCHomP2>.

Via the parameter graph, DSGRN provides a description of global dynamics over all of parameter space. This is a feature that to the best of the team's knowledge is unique to DSGRN. The research is attempting to exploit this feature by using DSGRN to identify if a given network exhibits a particular phenotype over large ranges of parameter values. From the point of view of synthetic biology one might expect that this property would suggest that if DSGRN indicates that a phenotype persists over large ranges of parameter values, then the associated regulatory network can function over a wide range of growth conditions or within a wide range of organisms.

A particular example of this approach is the team's work on identifying the most robust 3-node network that exhibits hysteretic switching. Enoch Yeung's group at the University of California, Santa Barbara (UCSB) is currently trying to construct the network that the team identified as being most robust. In current work, this idea has been extended to searching for logic circuits in larger regulatory networks, shown in Figure 3.

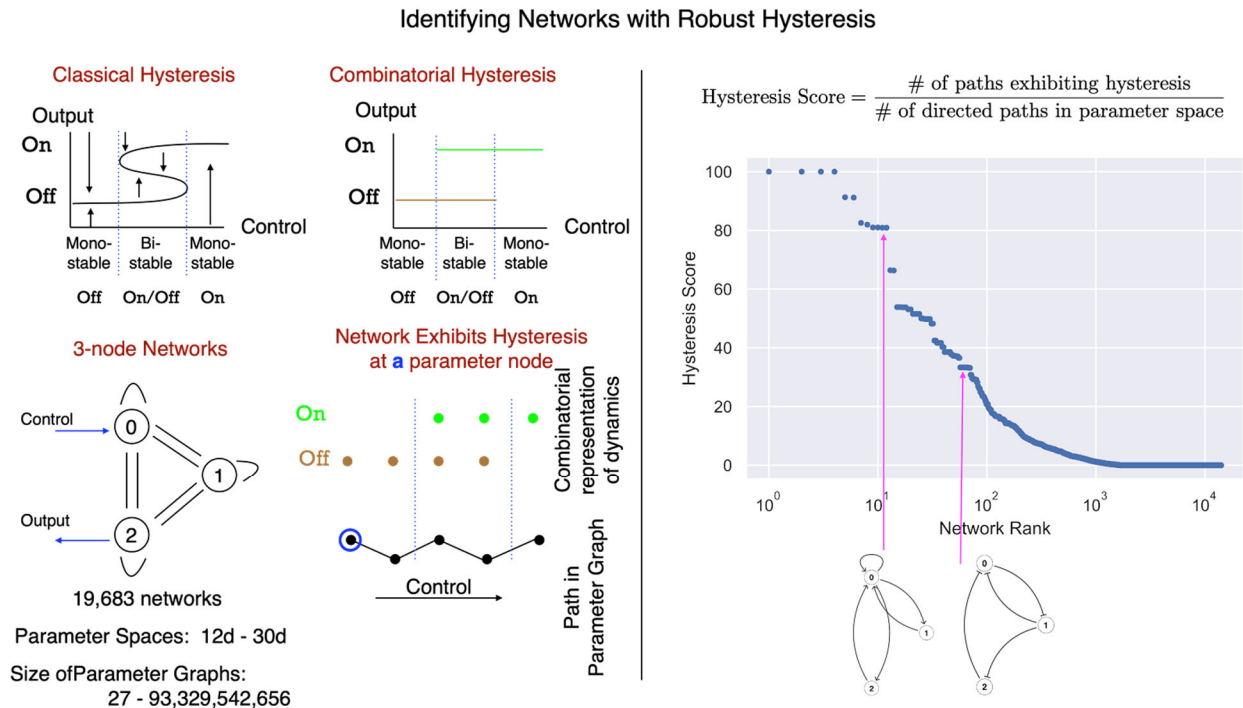


Figure 3. Analysis of all 3 Node Networks for Robust Hysteresis.

The Montana State University group also contributed to (i) a theoretical and mathematical foundation, deriving an important theoretical advance gaining a clear understanding of the relationship between DSGRN parameters and discrete models of gene network dynamics using Boolean functions. This popular class of models assigns binary values to each gene product of 0 (inactive or absent) and 1 (active or present). The states are updated based on the values of the input nodes via a fixed Boolean function, which can also be represented as a truth table.

The team realized that each DSGRN parameter node can be uniquely associated to a collection of Monotone Boolean functions (MBFs). This collection is composed of subcollections of MBFs, one for every node in the regulatory network, where each MBF in the subcollection is associated to a single out-edge from the regulatory node. Therefore, the DSGRN parameter graph can be viewed as a set of all possible collections of MBFs that are compatible with the structure of the network. This implies that the paths in the DSGRN parameter graph represent “bifurcation” paths connecting the corresponding MBFs. The insight of tight connection between DSGRN and Boolean models allows easier communication between these fields.

The discovery of this relationship enabled a contribution to (iii) computability. An ongoing issue with DSGRN since its inception is scalability. The exhaustive and combinatorial description of network dynamics precludes the DSGRN analysis of very large networks. Early on, the team discovered an informative subset of DSGRN parameter space called “essential parameters”. These parameters greatly reduce the size of DSGRN output, but also suffer from scaling difficulties. During the SD2 program, another subset of informative parameters was identified and called Boolean parameters. A DSGRN Boolean parameter is one in which for each node, the associated subcollection of MBFs is composed of identical maps. The DSGRN Boolean parameters are a very small set of DSGRN parameters. In one example network, a reduction of 9 orders of magnitude of DSGRN parameter space onto DSGRN Boolean parameter space was observed. This particular sampling method over DSGRN parameters greatly increases the size of networks that are amenable to analysis.

Beyond aims (i)-(iv), the Montana State University group focused on user interfaces for DSGRN and applications of DSGRN. Because of its capacity to comprehensively model network behavior across a finite decomposition of parameter space, DSGRN can be used to predict the robustness of a network architecture when the parameterization of the network or the cellular environment varies. This lends itself well to design and discovery problems. When a robust behavior is desired for a synthetic biology build or is suspected to exist in a natural genetic network, DSGRN can analyze up to hundreds of thousands of networks to proffer hypotheses for the ideal build or potential gene interaction network. Five applications are presented along with associated software advances.

3.2.2 Diagnosing Circuit Failure

The Voigt lab, at the Massachusetts Institute of Technology (MIT), had an interesting situation in which a 3-input logic circuit designated 0xA1 exhibited time series data with an unexpected drop in fluorescence that recovered over time, a phenomenon formally called a “glitch”. The Duke Team determined that this glitch was due to the presence of incoherent feed-forward loops in the network architecture. The input signal was forced to propagate downward with different timings, leading to unexpected loss of fluorescence before the second signaling pathway was activated.

The team was able to give advice on which branch of the network to shorten, as well as supply an alternative DSGRN design that would be feasible to build with a new modular part, as shown in Figure 4.

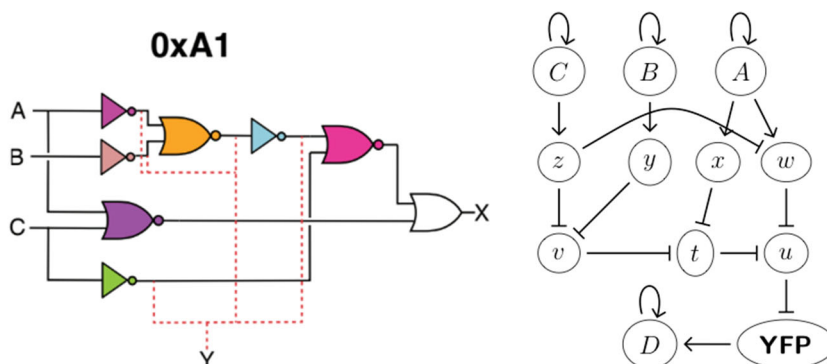


Figure 4. (Left) Original Circuit Design with Glitch, (Right) Alternative Design Produced by DSGRN.

Additionally, DSGRN parameters can be used to diagnose circuit malfunction. Circuit malfunctions can be identified as DSGRN dynamical signatures and the DSGRN parameters associated with these signatures can be identified. Several informal observations were made about the connection between certain DSGRN parameters and failure points on specific networks and used to hypothesize conceptual failure modes, such as spurious upregulation of a gene product by the cellular environment. Although this work did not proceed further in the project, there exists formal diagnosis methods that may allow one to codify the information contained in DSGRN parameters associated with malfunctioning phenotypes.

3.2.3 DSGRN Functionality outside of Gene Regulatory Networks

The Murray lab, at the California Institute of Technology (Caltech), built two strains of a microbe, each of which had a toxic kill switch introduced into its genome. Both strains produced a toxin and an antitoxin, the toxin for themselves and the antitoxin for the other strain. The intention was to create a homeostatic mixed population with a controllable proportion of each strain. The Duke Team explored the possibility of using DSGRN to model extracellular regulation as a window into the population dynamics of this system. The predictions of two models were compared to data from the Murray lab, shown in Figure 5. Both models correctly predicted more monocultures than bicultures, however only one model predicted sensitivity to initial conditions, which was also seen in the data. This indicates that it is a better model of the observed phenomena. This work also predicted that population oscillations should be seen if the system were grown in a chemostat. This is the first demonstration that DSGRN can be used outside of the purview of genetic regulatory networks.

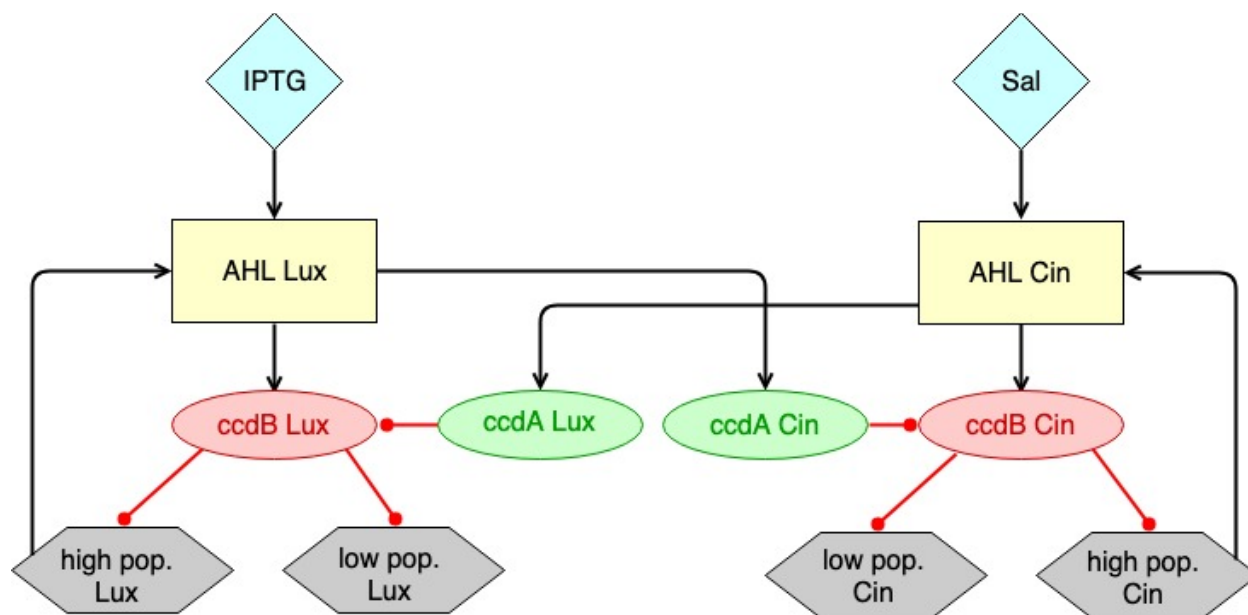


Figure 5. DSGRN Model of the Toxin-Antitoxin Two Population System.

3.2.4 DSGRN: Designing Circuits that Function Robustly Across Growth Conditions

3.2.4.1 FAST OFF Switch

An SD2 challenge was introduced to design and build a two-inducer circuit. The idea of the two-inducer design was to create 2-input circuits with novel phenotypes beyond logic that could respond to environmental stimuli such as chemical inputs. The team was given the idea of designing a regulatory network where the introduction of the second inducer acts as a “FAST OFF” switch. In other words, the addition of the second inducer would cause the fluorescence signal to cease far more rapidly than it otherwise could through protein decay. The team explored two possible branches of this project, a two-inducer system with activating inducers and a two-inducer system with repressing inducers.

A design of any regulatory network has what the team calls a “local logic” at each node in the network. The local logic indicates which of multiple regulatory inputs is dominant with respect to the others. It is deemed a type of logic because it can be expressed as a truth table. For example, consider a node with two activating regulators at the promoter region in which both inputs are required for transcription to occur. This is represented as AND logic, since either activator alone cannot produce output; e.g. fluorescence. On the other hand, if either input is powerful enough to trigger transcription on its own, this is described as OR logic. The negations, NAND and NOR, apply when there are two repressors, and the logic IMPLY and its negation NIMPLY are relevant when there is both an activator and a repressor impinging on the same node. The design process has to consider both local logic, which describes the circuit parts available for builds, and “global logic”, which is the logical behavior of the circuit as a whole. Therefore, there can be NOR parts or gates (local logic) within a NAND circuit (global logic).

The six original circuits tested in the Yeast States Challenge Problem had a phenotype of standard logical functions and were constructed from four strains, each from Clustered Regularly Interspaced Short Palindromic Repeats (CRISPR) logical NOR gates using guided RNA (gRNA) as the regulation molecule. Each strain represented an input state of the truth table with constitutive expression for the presence of a gRNA and a lack of promoter for the absence of gRNA. Three changes were planned to this paradigm. First, as required, one would construct networks that were sensitive to chemical input rather than being constitutively expressed, changing the desired phenotype from a truth table to a FAST OFF switch, and include an activating regulation instead of only repressing regulation.

This was an exciting design challenge for DSGRN, since other design software available in the program (e.g. Cello) provide single topologies with limited parts for limited phenotypes. The flexibility of DSGRN allowed a novel design process. This included the ability to model mixed regulation, both activation and repression, at a single promoter region (local logic). This is the standard IMPLY logic along with its negation, a new modeling construct at the topology level of synthetic design.

The two-inducer system with activating inputs required construction of new parts, i.e. new builds. In particular, the labs at Duke and UW explored the possibility of using plant transcription factors embedded into the yeast genome as sensors for chemical inputs. The plan was to combine the plant transcription factors with the CRISPR NOR gates from the previous project to construct circuits with nonstandard topologies, which are typically constructed from NOR and NOT gates only and so involve only repressing regulation.

The initial DSGRN design search across network space for a FAST OFF switch resulted in 1636 networks that met the design criteria. To narrow the field by better capturing robustness, a custom-built additional constraint was added that a specific initial condition had to lead to the desired behavior. However, there were 1608 networks that fulfilled this criterion, so further pruning was necessary. A new algorithm developed by the Mischaikow lab (Rutgers) was used to test robustness not only at design specification parameters, but also at neighboring DSGRN parameters, to check robustness at the failure of one component. This favors networks with redundant regulation. Additionally, using feedback from the UW Biofab, all networks were filtered out with more than two inputs at any node to reduce build complexity and filtered out those in which one node acted as an activator at one node and a repressor at another, a clear impossibility for the parts given to work with. This left 23 designed networks that fulfilled all the criteria. Finally, given a final piece of lab input, restricting the designs such that every pair of activating regulators must operate with the same local logic at a node (either AND or OR) and every pair of repressing regulators must operate with the same local logic at a node (either NAND or NOR). Leaving 5 networks, all of which were built to test DSGRN predictions. It was hypothesized that (1) the networks had a clear rank ordering in terms of performance; (2) it could be identified whether a plant transcription factor could dominate the effect of a CRISPR gRNA or vice versa; and (3) a more complex network with redundancy would outperform a network with a simpler build, as shown in Figure 6.

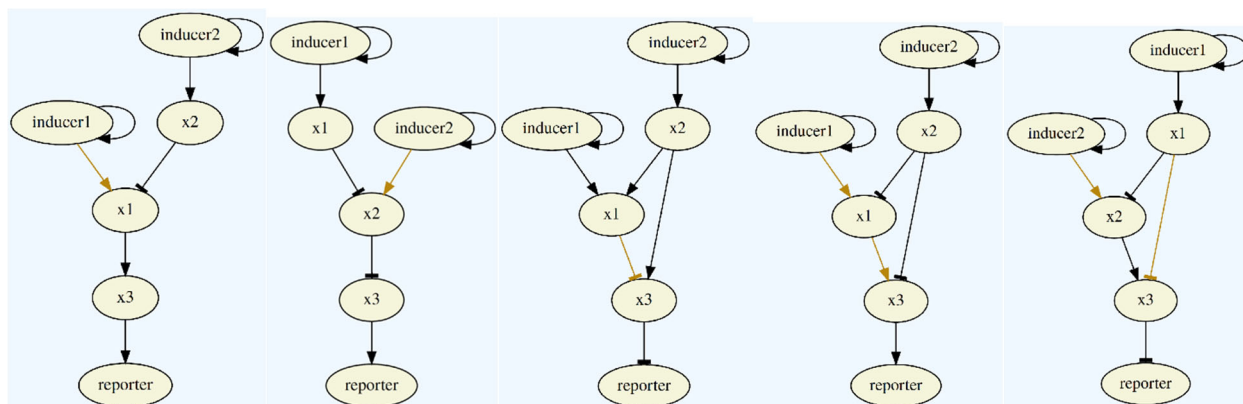


Figure 6. Two Inducer FAST OFF Designs – The Repressors Dominate the Activators to Achieve the Desired Behavior in the Third and Fifth Networks, While Activators Dominate the First, Second and Fourth Networks.

A consequence of the two-inducer design project is that an important discovery was made about the encoding of a biological phenotype in the language of DSGRN graph objects. A phenotype can be encoded either as a property of the state space dynamics for a single DSGRN parameter, or it can be encoded as a property of the DSGRN parameter graph itself. This choice impacts the representation of inducers in the regulatory network model. In particular, the choice of whether to represent an inducer as a node with no in-edges or as a node with a self-edge or implicitly in the network entirely depends on the choice of DSGRN phenotype representation. The discovery has been formally written up with an illustrative example. The relationship between the two DSGRN phenotype representations is not yet well understood in general.

Further discoveries were made about the expression of DSGRN parameters in plain language for communication with biologists. In particular, the standard local logic functions that had been used for describing had to be translated into the DSGRN parameters representing semi-algebraic regions of high-dimensional, real-valued parameter space. These parameters then had to be connected to requirements on the constitutive expression (or not) of the gene product at each node in the regulatory network. This resulted in a common language between the computationalist using DSGRN and the experimentalist creating the build.

Despite the great success in the modeling process, the build for the individual plant transcription factor parts did not succeed, likely due to technical issues with either the DNA sequences or insertions into the yeast cells. At this point, a mitigation strategy was adopted, which was to redesign global logic circuits using the previous CRISPR NOR gates only, where some of the strains were altered to respond to the chemical inducers doxycycline-hyclate and beta estradiol.

3.2.4.2 Synthetic Circuit Design Integrated with the Round Trip

During the course of this second design process, the common language facilitating explanation of DSGRN design requirements corresponding to build constraints was codified into a software package (the DSGRN Design Interface) with clear input and output files that do not require the intervention of a DSGRN expert. It is anticipated that the software is simple enough for a naïve user to devise and run a logic circuit design problem for choosing a network to build.

The Python library `dsgnrn_design_interface` is completely coded, tested, documented and a tutorial is provided. It is available as a public GitLab repository, https://gitlab.com/breecummins/dsgnrn_design_interface.git. The key features of the software are as follows:

- Simple user input is provided. Seven plain language inputs are specified in a Java Script Object Notation (JSON) file, including certain build constraints determined by experimentalists. The user is not required to understand the many specific parameters required by the underlying software packages, which have been defaulted to sensible values for logic circuit design.
- Simple user output is provided. The output includes networks and statistics files in custom formats for DSGRN and also includes a Synthetic Biology Open Language (SBOL) document for machine-readability of the design. An output Jupyter notebook is also provided that provides a quick look at the topology for the top networks, for users who wish to browse.
- Two separate installation methods are provided. One is a local installation through a conda environment. This installation has succeeded on Ubuntu 18 and Mac OS X Mojave. Additionally, there is a Dockerfile provided that facilitates installation on Ubuntu, Mojave, and Catalina.

The DSGRN Design Interface software was used in a true test of the design-build-test-learn loop called the Round Trip, a cross-team project in the automation of experiment submission, and execution, and data collection, curation, and analysis, as shown in Figure 7. Using data from the Yeast States Round Trip experiments, it was discovered that circuit performance was poorest in the global logic OR and NOR circuits. Two circuits were chosen for topology redesign through DSGRN. Thousands of networks were screened for the appropriate global logic and build constraints, and the top 10 or so were presented to experimentalists and modelers jointly for final choice on which builds to pursue. One new topology was chosen for OR and one for NOR, see Figure 8. The final build choices included the two original topologies tested earlier in the program, but now rendered inducible, and the two new topologies.

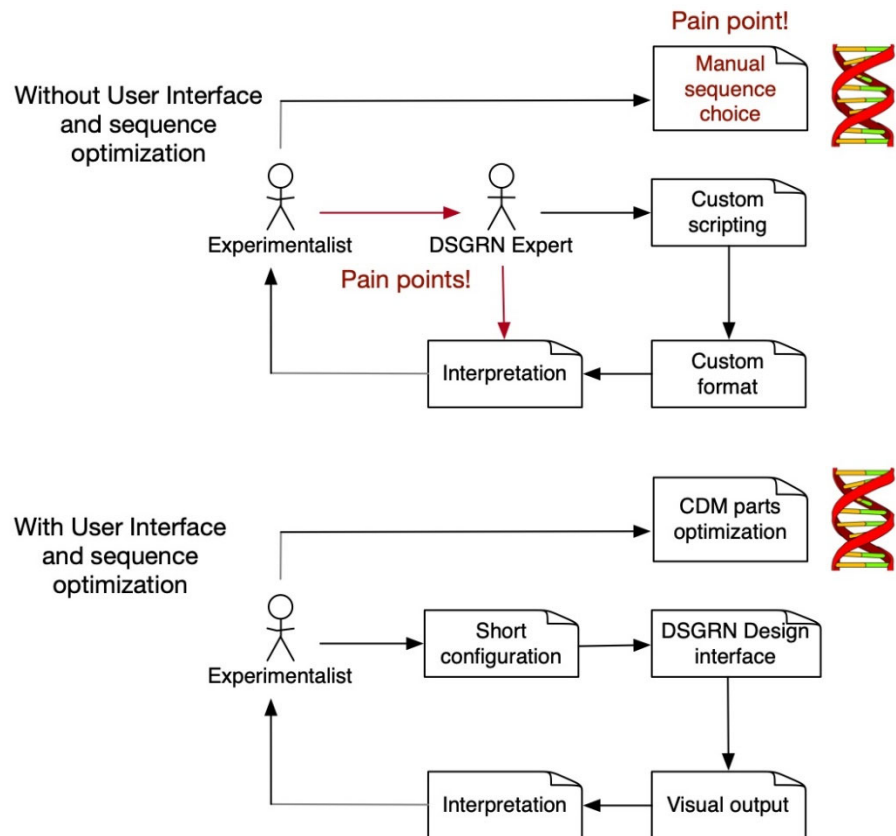
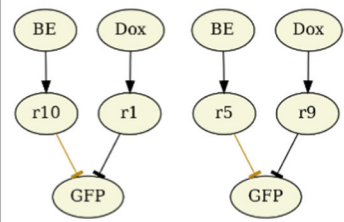
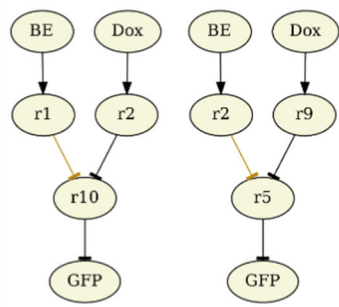


Figure 7. DSGRN Design Interface and Combinatorial Design Model Improvements to Process.

	NOR circuit		OR circuit	
	CDM low design	CDM high design	CDM low design	CDM high design
CDM score difference	1.49		2.54	
Simple topologies				
Topology Robustness Score	0.08		0.07	

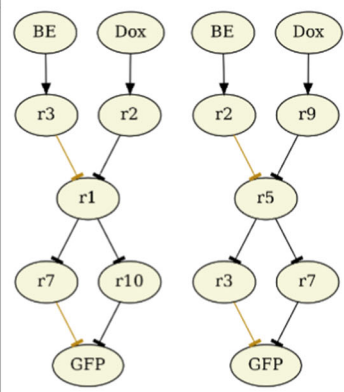
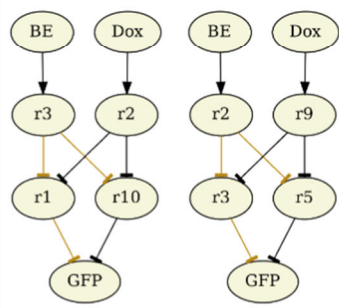
CDM score difference	2.66		2.65	
DSGRN topologies				
Topology Robustness Score	0.22		0.28	

Figure 8. (Top) Original Simple Circuits, (Bottom) DSGRN+CDM Circuit Designs with Redundancy.

Each topology was assigned a “good” build and a “bad” build using the machine learning tool Combinatorial Design Model (CDM), developed at Netrias. CDM takes a network topology and assigns a specific biological part to each node in the network based on ML optimization using preliminary dosage response data. The highest performing and the lowest performing collection of parts was chosen for the good and bad builds respectively. This is an example of the synergism between DSGRN and different tools in the SD2 program.

DSGRN predicted that the DSGRN redesigned OR and NOR topologies with their added redundancy should outperform the simplest circuit topologies used in previous experiments. CDM predicted specific fold-change values for the original NOR circuit and also predicted increased performance of the “good” builds over the “bad” builds. This design paradigm was integrated

with the lab management tools Terrarium, Aquarium, and DASi from the UW Biofab as a precursor to the automated Round Trip. A manuscript is currently under revision at Oxford University Press (OUP) Synthetic Biology.

3.2.5 SD2 Tool Integration

During the course of the various SD2 design challenges, integration was attempted with various aspects of the infrastructure and the design tool Cello, all of which remain uncompleted for various reasons. The team attempted development of a design tool Agave app, the development of a DSGRN Network Database, and the integration of `dsgrn_design_interface` with the logic circuit design tool Cello. Although these projects were never completed, each one of them provided insight for other projects or future research directions.

Currently, DSGRN builds a Structured Query Language (SQL) database for each regulatory network that is indexed by DSGRN parameters. This is convenient for phenotype or dynamical signature searches within a network, but not between networks, which would be desirable to answer questions about network evolution. In that case, it would be more convenient to have a single database that is indexed by networks. Mohammed Eslami at Netrias prototyped such a structure in a graph database with the Gremlin query language. It not only had the advantage of connectivity between networks, but also reduced the size of the stored data. This product was intended for release and public hosting, but the database was never populated due to two theoretical barriers.

The first barrier is uniform indexing of the networks. In order to search across networks, it is necessary to have only one copy of the network in the database. This means that it is necessary to have an isomorphism check on each network before inserting it into the database. As the size of the database grows, isomorphism checking eventually becomes infeasibly slow. Solution to this problem was sought through a procedure called “canonicalization”. Canonicalization means defining a standard way of representing a network topology as a string and using string comparisons instead of graph isomorphism checks for insertion into the database. This requires choosing standard node names and node orders in a consistent manner for any possible regulatory network. Work on this problem was started, but not completed. It remains an interesting avenue of research to avoid redundant computation.

The second barrier is the unknown connections between DSGRN parameters across networks. It is intuitively clear that a parameterized subnetwork of a regulatory network must somehow constrain the dynamical behavior of the parent regulatory network. It is work in progress (manuscript in preparation) to codify how this happens and a future direction of research would be to connect the parameters of more disparate networks. This knowledge would allow the direct connection between DSGRN parameters in a DSGRN network database.

A second attempted project was to embed the DSGRN design tool within an Agave app on the Texas Advanced Computing Center infrastructure. The first step was to create an app that only wrapped DSGRN and not the full design tool. This was accomplished with substantial help from the TACC staff due to extensive difficulties with the C program language packages on the TACC infrastructure. The interface app was started but never completed due to other technical difficul-

ties. However, the Dockerfile that was created for the DSGRN Agave app was instrumental to creating the Dockerfile for the `dsgnrn_design_interface` installation.

A third attempted project was to integrate DSGRN design with the Cello design tool at MIT. The team successfully prototyped a hand-off of a DSGRN design specification to the back-end Cello modeling, but then lost the contact at MIT. Support after that would have been minimal. In addition, the team was informed that what had been done was not truly Cello integration, but integration with a Hill function modeler and optimization tool on the backend of Cello. It was suggested that a new script should be written, and due to lack of support the project was abandoned. The required parts optimization for design that was hoped for from Cello was subsequently provided by CDM as described above.

Despite the unfortunate end to this project, a valuable tool was gained which was a SBOL2 document template for a DSGRN design specification. This is a machine-readable document that is automatically produced by `dsgnrn_design_interface` and can facilitate future integration with other toolsets. A successful unit test passing a network from DSGRN to Cello to Terrarium (an automated build tool) via SBOL documents was performed, which included beta testing pySBOL2. A consequence of the SBOL document design for Cello integration was a more formal understanding of the relationship between DSGRN parameters and local logic types, OR, AND, etc. The earlier description of the common language for DSGRN users and experimentalists is due in part to the intricate process of developing this structured document.

3.2.6 Network Discovery

Many important cellular processes are controlled by transcriptional regulation of gene products. However, in most cases the gene interactions are unknown, and the number of experiments required to collect sufficient evidence for a hypothesis is very large. The reduction of experiments through in silico techniques is very valuable and this field is still in its infancy. During the course of the SD2 program, the team worked on a computational toolchain for the discovery of small core oscillator networks that control large scale cellular oscillations such as the cell cycle and the circadian rhythm.

The tool chain, called the Inherent Dynamics Pipeline, https://gitlab.com/biochron/inherent_dynamics_pipeline.git, is composed of three primary components, an algorithm for hypothesizing important gene products (nodes), an algorithm for hypothesizing important pairwise regulatory interactions (edges), and an algorithm for assessing the global performance of collections of pairwise interactions that form strongly connected networks. This last step is based on the capabilities of DSGRN and is used to refine and re-prioritize the proposed nodes and edges for experimental interventions. It has been shown in previous publications that the first two steps of node and edge identification and ranking perform well. The primary addition to this technology is the network dynamics checking step at the end of the Inherent Dynamics Pipeline and the iterative process of hypothesis reduction.

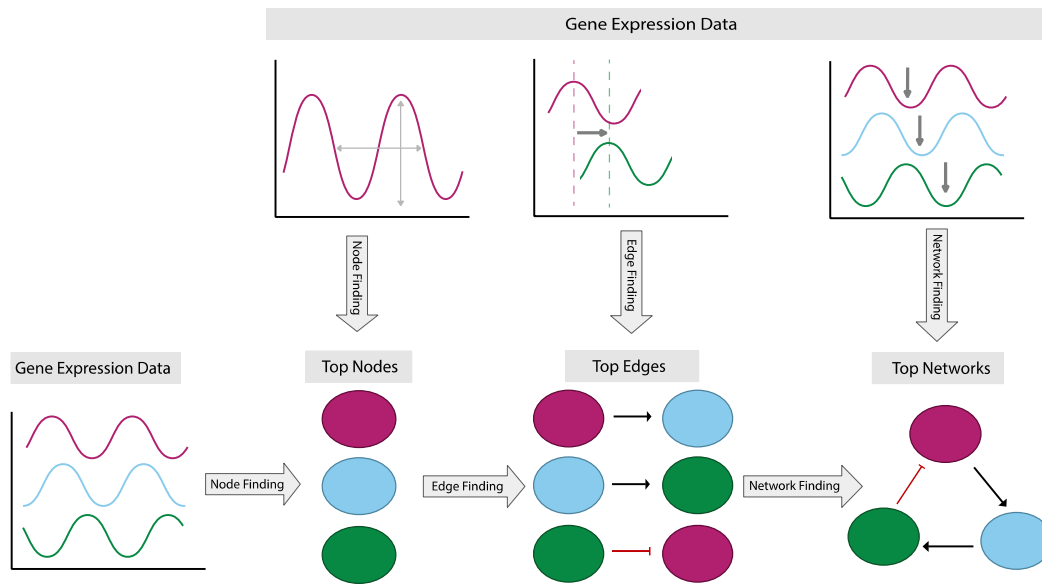


Figure 9. Schematic of the Inherent Dynamics Pipeline.

The test case data consisted of three distinct time series datasets from a simulated network and two replicate experimental time series from yeast cells with a synchronized cell cycle, as shown in Figure 9. It was observed that in both cases, the re-prioritization of nodes and edges according to participation in validated global network models that meet high performance criteria corrects misrankings that occur earlier in the pipeline. Under the restricted conditions of the test data, the team was able to distinguish between high probability actors and low probability actors in the core oscillators, enabling the team to provide straightforward and accurate advice to experimentalists seeking to understand the driving force behind an oscillating cellular phenotype. The opportunity to use this technique on SD2 challenge problems did not present itself, but the manuscript introducing this open source software to computational systems biologists was accepted to Public Library of Science (PLOS) Computational Biology.

3.3 Tools for Automated Data Pre-Processing, Normalization and Quality Control

3.3.1 Transcriptomics Processing

In Q0 to Q3, pipelines were setup for processing transcriptomics data. This included gathering, processing, and annotating both RNA-seq and microarray data to support analysis. Both internal SD2 RNA-seq data and external RNA-seq and microarray data were processed to support research in several early challenge problems: Yeast-Gates, Cell Growth, and Cell States. Since much of the initial research in these projects relied on comparing a diversity of data sets, it was especially important to be able to pull all the raw data, process it in a consistent manner, gather the same metadata, annotate using the same versions, and normalize the data together. This also included some initial analysis like differential expression and plotting.

3.3.2 Processing RNA-seq Data

It was required to analyze RNA-seq data in the beginning of the program, so temporary RNA-seq processing pipelines were built. This included scripts that would organize all the input data, build job descriptions for all the inputs, submit each of these jobs to the TACC infrastructure, and then check when the jobs were complete to repeat these steps for the next stage of processing.

As SD2 RNA-seq processing pipelines were being developed by pipeline groups, the Duke Team tools evolved to use those standardized pipelines. Processing was also setup for annotating these data sets, differential expression using Differential Expression Sequencing (DESeq), and creating plotting tools for analyzing results.

3.3.3 gRNA Quantification for CRISPR circuits

Figure 10 displays aspects of a small RNA-seq of yeast circuits. Figure 10A depicts an OR gate design which contained the DNA sequences for four gRNAs, though only one to three would be expressed depending on the input state. Figure 10B shows the distribution of total reads in all small RNA-seq files generated in SD2. Figure 10C shows the percent of reads for each or both read files that passed QC. Finally, Figure 10D displays the heatmap showing the number of reads per million for all possible gRNAs found in yeast containing OR gates with an input state of 11. Expected gRNAs for an OR gate with an input state of 11 are r3 and r6. Each row represents a sequenced yeast OR gate strain with an input state of 11.

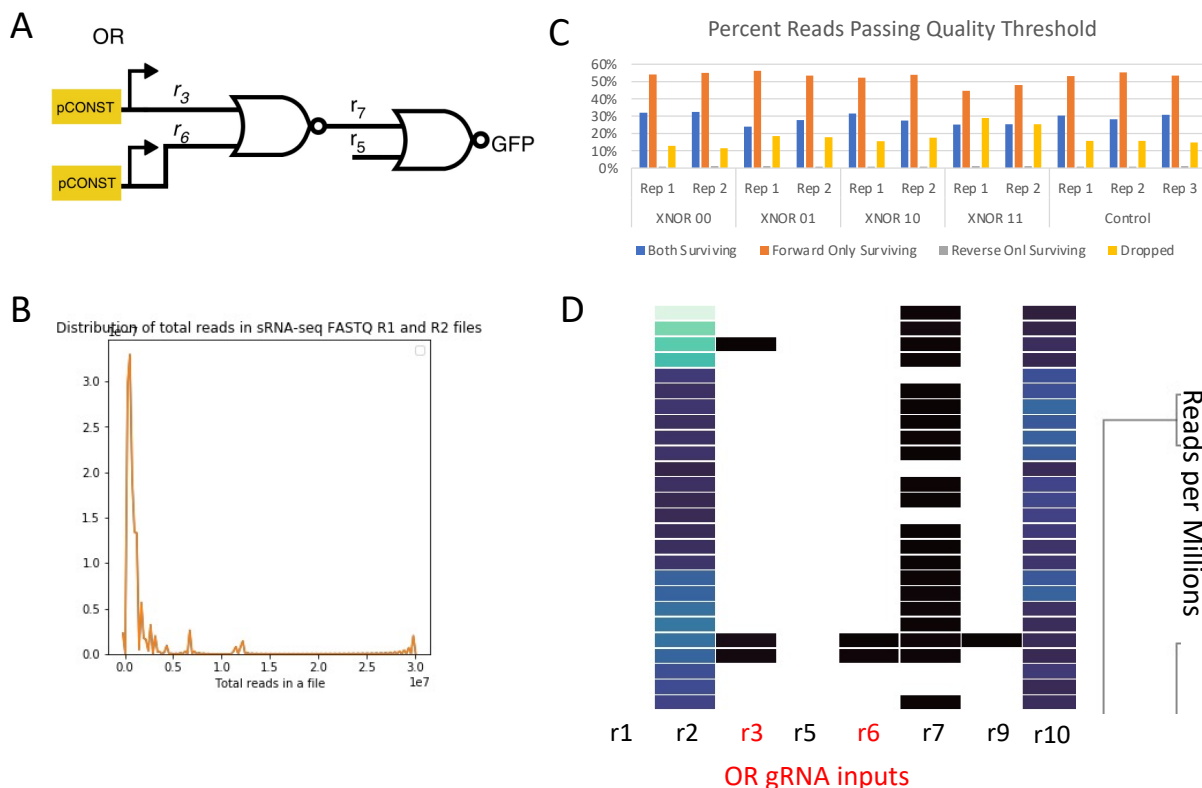


Figure 10. Small RNA-seq of yeast circuits. A) OR gate, B) Distribution of total reads, C) Percent of reads that passed QC, D) Heatmap for input state of 11.

As most of the yeast circuits used several gRNAs within the circuit, see Figure 10A, determining the abundance of the gRNAs would help with investigating circuit functionality. Paired-end small RNA-seq was performed at Ginkgo, yet there was not a small RNA-seq processing pipeline. Therefore, the team built a small RNA-seq processing pipeline using python and deployed it on TACC using Abaco and Tapis. The pipeline only quantifies reads that match to gRNA within the yeast circuits. The Levenshtein distance metric was used, also known as fuzzy string matching, to find reads that matched to gRNA sequences. The Levenshtein distance is a metric to measure how apart are two sequences of words and the result is a similarity score, ranging from 0 (completely different) to 1 (the same). This method was chosen to account for inaccuracies in sequencing. A similarity score of 0.8 was used to identify reads that mapped to gRNA sequences. The pipeline was tested on q0 small RNA-seq data from Ginkgo, however, most of the sequencing results had very small number of total reads making it hard to compare across experiments, as shown in Figure 10B. Quality Control was performed on the RNA-seq data resulting in only ~30% of both forward and reverse reads surviving QC, ~55% of only forward surviving and ~2% of only reverse surviving, see Figure 10C. Therefore, only the forward reads were analyzed in the pipeline. Additionally, gRNA abundances from the pipeline could only be compared within a sample, and not across. A normalization method was needed for this to be possible. Small RNA-seq reference beads were included in the experiments from Ginkgo, however, due to quality issues, the reference beads were not usable to normalize gRNA abundances.

Regardless of a normalization method, the presence/absence of gRNAs in the small RNA-seq data could be used to assess circuit function. Based on presence/absence of gRNAs, it was found that the yeast strains containing the OR gate were both missing expected gRNAs and containing unexpected gRNAs, as shown in Figure 10D. When these results were compared to DNaseq results and to annotations of the OR gate from the Gander 2017 paper and in SynBioHub, disagreements were found between all four sources of information, see Figure 11. In addition, the OR gate strains failed across all conditions tested. Two hypotheses were presented 1) the OR gate lacks any kind of robustness or 2) the OR gate strains are not representative of an OR gate. Hypothesis 2 was favored as it was found that multiple versions of the OR gate were designed, and it was possible that a different strain was sent for Deoxyribonucleic Acid Sequencing (DNaseq) and/or small RNA-seq relative to the strain annotation deposited on Synthetic Biology Hub (SynBioHub) and presented in Gander 2017. The next step was to compare all OR design versions with the SynBioHub annotation, the DNaseq results and the small RNA-seq results to determine the correct OR strain for each, however, this did not occur.

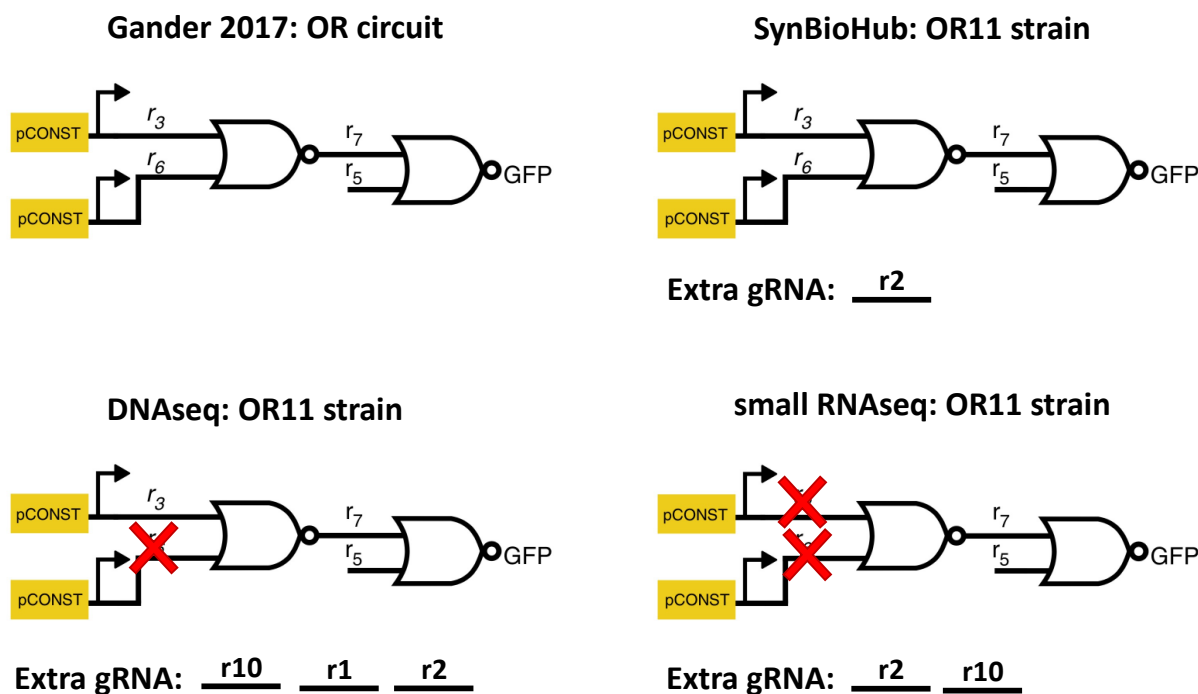


Figure 11. Expected and Unexpected gRNAs for the OR Gate with an Input State of 11 from Various Resources, Where the Red “X” Indicates the gRNA is Missing from the Resource.

3.3.4 Processing Microarray Data

To be able to analyze certain external data sets of interest, an ability to process microarray data was needed.

The Duke Team implemented code to run the raw CEL files through summarization and normalization. Two summarization and normalization algorithms were used, DNA-Chip Analyzer

(dChip) and Robust Multiarray Analysis (RMA). They output a tsv file with genes by all samples, a tsv file of time series for each experiment with genes by time, and plot of distributions of gene expressions per experiment.

The next step is annotation, which includes attaching gene names, symbols, species database (Saccharomyces Genome Database (SGD), Mouse Genome Informatics (MGI), etc) ids. Currently included are affymetrix yeast s98 and 2.0 probe sets. Also included is the SGD data for use to convert between systematic name, common name, and SGD ID, which can be used on RNA-seq data sets. This outputs the original tsv data with columns added for gene names, symbols, ids.

Using an annotated tsv file, probes for affymetrix yeast s98 or 2.0 microarrays are cleaned out. Probes are removed if: they do not map to an SGD ID, they do not map to a systematic name (e.g. YKR009C), the probe maps to multiple genes, there are "questionable probes" (e.g. probes with `_<X>_` at suffix, where X = s,f,g,x,i,r). Note that there is not a check or get rid of gene ids that are duplicated; it is left to the user to decide how to deal with them. But most methods cannot handle them, so they should be removed by dropping all of them or using some selection criteria (e.g. min expression, max expression, etc). This outputs the original tsv data with rows removed, and info about what was removed in the header.

Differential Expression analysis was set up by processing scripts to run differential expression analysis on microarray data using limma in R.

Worked on: 2017-2019

Challenge Problems: Yeast Gates, Immortality, Cell Growth, Cell States

Code: https://gitlab.sd2e.org/gda/transcriptomics_process

3.3.5 RNA-seq & Microarray Data: Normalizing Transcriptomics across platforms

The team previously generated separate lists for RNA-seq data and microarray data, because these sets cannot be normalized and analyzed together. To normalize and analyze the sets together, a method was developed for picking invariant sets of genes, which will be used to normalize data across microarray and RNA-seq data.

Given an invariant set of genes is wanted whose rankings/values can be used to renormalize data across experiments, assume the invariant set should:

- Have the same gene expression rankings across all samples for all experiments.
- Be invariant to time, so time points are treated as replicates within a data set.
- Have genes with expression levels that cover distribution of gene expression levels of the entire data set (ie. some genes that are expressed lowly, some that are expressed highly, and some that have intermediate expression levels).

- The more genes there are, and with a more even spread of genes across possible expression levels, the resolution of the normalization will be finer.
- as more samples are added from more experiments, the invariant subset will become smaller.

The first step is to make the set of genes invariant to time by removing all genes that have dynamic expression with a data set. The notion of time is removed from the data sets by treating all the time points as repeated samples. For each gene, the standard deviation and the mean is computed across all the time points and remove all genes that have a standard deviation above a set percent of their mean expression levels.

The software starts with all the samples, computes rankings, and removes the gene with the largest change in ranking, repeat; (ie if the change was big enough to overlap changes detected in another gene). This leaves a set that meets the requirements of an invariant set (no ranking change for each gene, and the set of genes should cover a variety of expression levels). While this method does filter on variation in a way, it only does so relative to other genes with similar expression levels (sets of genes that would swap rankings).

This method has been tested on three data sets, one RNA-seq and two microarrays. The genes are first selected to have a standard deviation up to 1/10 of their mean expression. The gene list is next pruned by removing the gene with the biggest rank change across all samples in all experiments, loop until there are no more rank changes. The 13 remaining genes have the same rank order in every sample across all experiments, as shown in Figure 12.

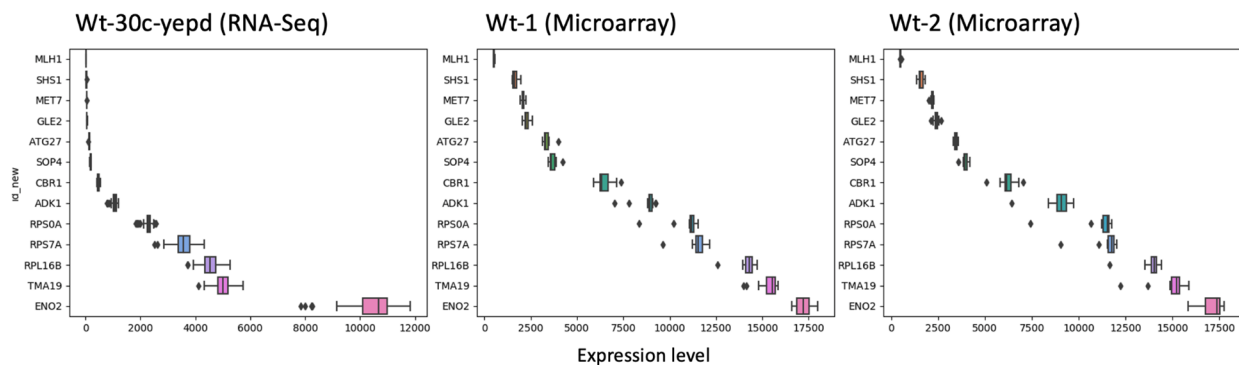


Figure 12. Set of 13 Invariant Genes Found Across RNA-seq and Microarray Experiments.

The results looked very promising, but there was no experimental validation of the results.

Worked on: 2018

Challenge Problem: Yeast States

Collaborators: Mohammed Eslami, Netrias

Code: https://gitlab.sd2e.org/gda/ge_xexp_norm

Results: [sd2e-community/shared-q1-workshop/adeckard/cp-yeaststates/xexp_norm](#)

3.3.6 Quality Control Reporter for RNA-seq

QC information about RNA-seq data and preprocessing is spread through a variety of files in a variety of formats. Since the QC data is essential for further analysis of RNA-seq data, it needed to be available. This tool was developed to gather data from several tools and make a report of QC data that would be analysis ready.

The data sources include output from sortmeRNA, Trimmomatic, Star, Sailfish, and fastQC. For each program, a JSON file describes all the output files that need to be checked to exist and all files that need to have data extracted from them. For extracting data, the JSON describes what files need to have data extracted, a regular expression for finding the data in the output file, and if the value that is extracted will generate a warning for certain values.

The program takes in a JSON configuration and a location for the RNA-seq output, and then it uses the JSON configuration to determine what files to check and what fields to extract from the files. For extracting the data from the files, it executes to regular expression and stores the results. It then generates a QC report, in JSON format and saves it with all the files and fields it checks. For each file it reports if the file exists, its size, and line count. For each field, the field title and the values are reported, and any flags for the data values.

The results of the QC reporter were used in determining the quality of data, or if data was missing, to be used in downstream analysis.

Worked on: 2018

Challenge Problems: Yeast Gates, Cell States

Code: [sd2e-community/shared-q1-workshop/adeckard/wg-etl-omics/qc_reporter_tool/](#)

3.3.7 Flow Cytometric Data Analyses: Removing Dead Cells from the Analyses (Autogater)

The group used flow cytometry to measure circuit function within the SD2 program. Flow cytometry allows for single-cell measurements of green fluorescent protein (GFP) expression (the output of yeast circuits); however, flow cytometry does not distinguish between live and dead cells. Separating out healthy cells from dying and dead cells, and any potential debris, is an important first step in analysis of flow cytometry data. While gating of debris can be conducted using measured optical properties, identifying dead and dying cells often requires utilizing fluorescent stains to identify cells that should be excluded from downstream analyses. Current methods of differentiation between live and dead cells rely on adding stains such as Sytox, a nucleic acid stain that stains cells with compromised cell membrane such as dead cells. Staining methods allow researchers to distinguish between live and dead cells at the cellular level, but they can be time consuming and expensive. Other methods used to measure cell death are based on growth on solid or liquid mediums, such as Colony Forming Units (CFUs). There are two main limitations to CFUs. First, CFUs do not measure cell death directly, but rather provide a growth inhibition measure that can be used as a proxy to measure cell death. In doing so, they fail to capture

some viable cells such as those that remain alive but are unable to divide. The second limitation to CFUs is that they only provide a population-based estimate of the percentage of live and dead cells and cannot determine if any individual cell is alive or dead. Therefore, the team sought to develop a method that can screen live/dead cells in flow cytometry data without the use of stains. This method is based on developing a model that learns different forms of death and then using that model to predict if a cell is dead or alive.

As with any machine learning method, data was needed to train the model on. Five protocols were developed for killing cells five different ways which were to be used to kill four different yeast strains and bacteria strains. This data would be used to train and test a model on. Additionally, three different live/dead screening methods were to be used for subsequent comparison to the ML method. Lastly, these experiments were to be done at Duke, BioFab at UW and Strateos to examine generalization of the method. Strateos performed several experiments treating yeast with different concentrations of ethanol and screened live/dead cells using Sytox and their flow cytometer. At Duke, initial experiments were performed treating different yeast strains with the various killing methods to develop the protocols. After establishing the protocols, a single yeast strain was treated with various ethanol concentrations (0, 5, 10, 12.5, and 20%), with samples taken over time for Sytox + flow cytometry and CFUs. Samplings occurred every 30 minutes for 6 hours. This set of experiments lacked flow data for the control experiments (0% and 80% ethanol) and contained several outliers in the CFU data, therefore, a new set of experiments were designed for treating yeast with ethanol. A coarser sampling was performed, and more ethanol concentrations tested based on analysis of previous data and conversations between experimentalists and the data scientists building the model. The new sampling frequency was 0, 0.5, 3 and 6 hours and the ethanol concentrations were 0, 5, 10, 12, 15, 20 and 80%, as shown in Figure 13. Following these experiments, a new set of experiments were designed that treated cells to various temperatures (25, 30, 35, 40, 45, 50, 55, and 65C) with samplings at 0, 0.5, 1, 2, 3, 4, 5 and 6 hours. Samples were taken for Sytox + flow cytometry and CFUs. All data were uploaded to TACC and run through Data Converge. Autogater was found to perform equally as well as stain-based methods. This work led to a white paper and preparation for a manuscript.

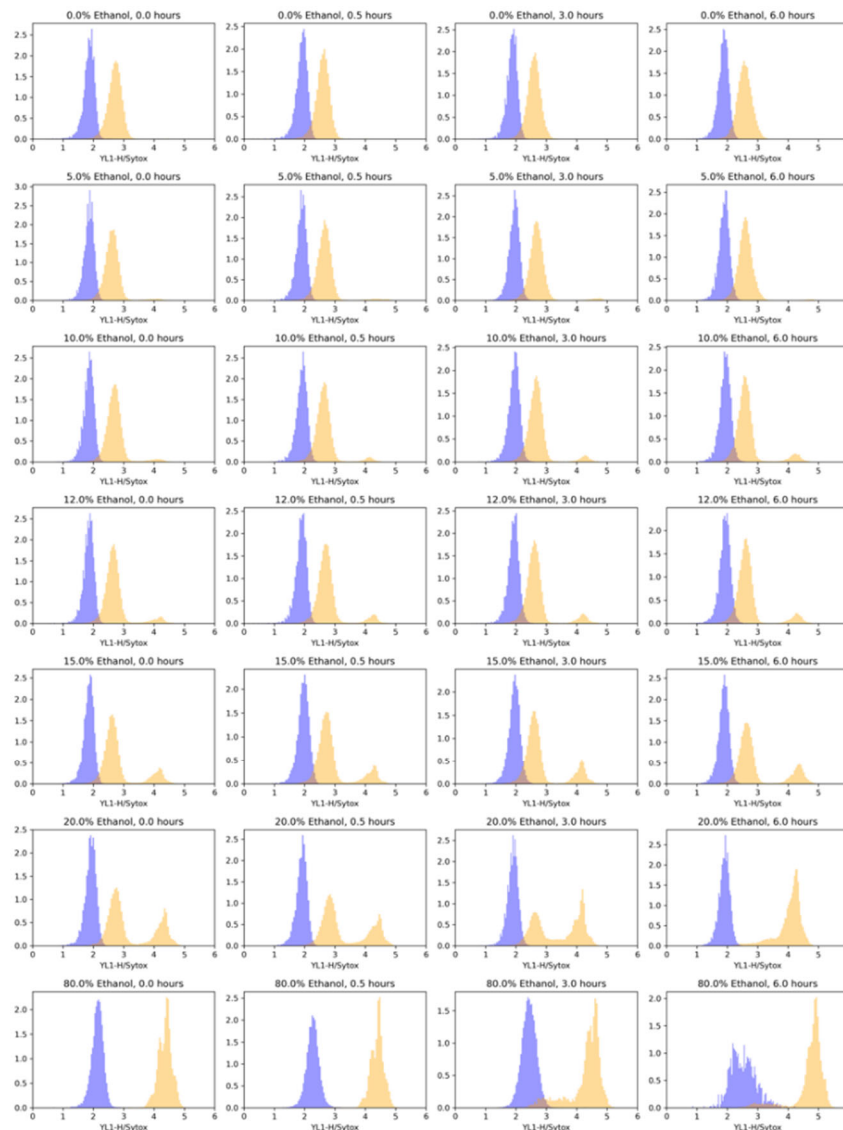


Figure 13. *Flow Cytometry Data from Yeast Cells Killed with Various Ethanol Concentrations; Rows are Different Ethanol Concentrations, Columns are Time Points, Purple Distributions are Cells not Stained with Sytox, and Yellow Distributions are Cells Stained with Sytox.*

3.3.8 Protein Design CP, Data Processing Pipelines

The Protein Design CP sought to use the high-throughput output of synthetic protein design tools, construction platforms, and experimental assays, to train models that improve the design of synthetic proteins, optimizing either for specific protein properties or functions. The first major thrust was to use existing data collected from earlier protein stability assays—a protocol developed in the Baker Lab—to develop models that could predict protein stability using primary sequence and/or other biophysical features of the synthetic designs. The protein stability assay, described in Rocklin *et al.* (2017), combines parallel oligo library synthesis, yeast surface display, and next generation sequencing to measure a proxy of protein stability. This is achieved by subjecting the synthetic proteins—displayed on the surface of yeast cells—to one or more digestive

enzymes (trypsin or chymotrypsin) at varying concentrations and inferring an ec_{50} value by measuring the concentration at which one sees a 50% drop in baseline levels of expression. This platform can generate very large datasets (on the order of 10^4 - 10^5 synthetic proteins per experiment), enabling the training and testing of stability-prediction models.

Initially a design, build, test, train, predict loop was envisioned in which models trained on experimental outputs would either be used to directly generate new designs, iteratively improve on design sequences, and/or down select to the most favorable designs generated by the Rosetta modelling platform (<http://www.rosettacommons.org>) or others. Designs would be built and tested, and the results would be used to retrain, and hopefully improve, model performance (e.g., reduce error, increase generalizability, etc.). To facilitate the execution of this loop and ensure that data was made machine-learning ready, i.e., properly formatted, cleaned, normalized, etc. a host of tools for data processing, management, sharing, and versioning were developed.

3.3.9 Versioned Dataframe Repository

One of the fundamental challenges was ensuring that attributes of individual synthetic designs were properly tied to a unique design in a way which enabled modelers to accurately combine data sources, e.g., metadata, experimental output, or new derived protein features. It was realized early on that this CP data was not, and would not be static, and that even previously processed data would need to be updated as improvements were made to various preprocessing steps and models. The need for data provenance, a common data standard, and a common source of shared data across teams led to the creation of the Versioned Dataframe Repository (VDR). This software package—which the Duke Team helped conceptualize, test, and debug, but which was primarily built by members of the TwoSix team—enabled various teams to contribute new datasets that were useable program-wide, indexed in a standard way to ensure compatibility with existing dataframes, and commented with meaningful descriptions of the data, together with creation timestamps and contact information of the dataset’s creator.

One example of the need to update existing dataframes was the need for dataset normalization across experiments. Necessarily, models would be ingesting stability measurements from experiments run by different labs, in slightly different conditions. Therefore, to ensure valid model training, a normalization technique was needed that could remove systematic differences between experiments. This would help data to be comparable across existing experiments and all future experiments. To address this issue, a simple linear model was developed using so-called ladder proteins: proteins which appeared in the original Rocklin experiment, and which would be rebuilt and assayed in each new stability assay experiment. The model worked directly on the ec_{50} values and mapped ec_{50} measurements in each experiment into the ec_{50} space of the original Rocklin experiment, which was treated as the base space. This model was implemented in a modular Python codebase which was eventually integrated in the Protein Stability Assay Processing Pipeline.

The Protein Stability Assay Processing Pipeline consists of a sequence of data preprocesses steps that had previously been owned by several performers and required numerous handoffs via the VDR. In collaboration with the Duke Team, members of the TwoSix team, and members of the Baker Lab, TA4’s Ethan Ho spearheaded the effort to integrate and couple the data processing tools needed to produce the finalized, machine-learning-ready, stability assay dataframes pro-

duced by experimental assay of new designs. This pipeline eliminated the need for multiple performers to individually process experimental outputs, and thereby saved significant performer time and effort, and improved the reliability of the data products.

3.3.10 Build Quality Control Unit

In the Yeast States project, it was discovered genome sequencing data from certain yeast circuit strains did not agree with their expected strain designs. Therefore, it was decided to add a Quality Control step between the Build and Test units of the DBTL loop. This would allow one to detect failed builds earlier in the loop, so that they can either be rebuilt or redesigned. Additionally, adding a QC check also improves resource usage. Figure 14 shows how Build QC connects to other resources used or built in SD2. The end goal for the Build QC Unit was to provide a single metric of similarity between the expected build and what was built. The Duke Team worked with the SynBioHub team on data representation and with Joshua Urrutia on building the application on TACC. An experimental request was authored for running DNaseq on a set of yeast strains. Sequencing of strains though did not continue so this project was put on hold.

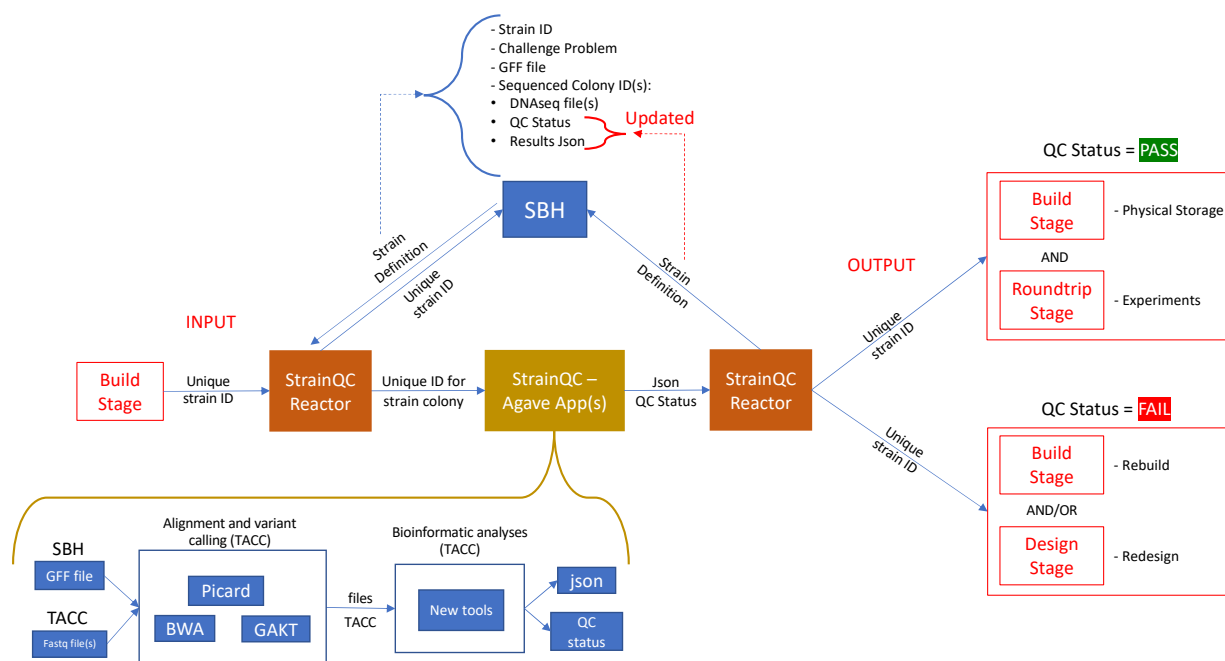


Figure 14. Initial Design of Build QC.

3.3.11 Signal Prediction

Flow cytometry was used extensively to measure output from yeast circuits within the SD2 program. Properly interpreting flow cytometry data collected from cells expressing a fluorescent molecule requires the use of a positive and a negative control. Traditionally this is done manually by a human, where the controls are used to gate/threshold the data taken from experimental samples. This gating/thresholding is usually done on the channel the respective fluorescent molecule has the strongest signal on. This manual method is subject to human error and biases and is not scalable. The Signal Prediction method was initially developed by Dan Bryce of SIFT and fur-

ther extended by Duke as an alternative method to the traditional method of manual inspection and use of controls to interpret experimental samples. The Signal Prediction tool develops a random forest classifier to predict whether an event within a flow cytometry sample corresponds to a high (the positive control) or a low (the negative control) signal, in the same sense as digital logic. In the case of most SD2 experiments, the positive controls are cells that constitutively express GFP and the negative controls are cells that do not express GFP due to the lack of the GFP's coding sequence. Using these controls, the Signal Prediction tool develops the classifier for each pair of high/positive and low/negative controls. The tool trains the classifier on all flow cytometry events for the high/positive and low/negative controls, labeling each respective event as either 0 (low) or 1 (high), and then uses it to predict for each event within each sample whether the event is low or high. Compared to the traditional method that defines a linear threshold to a single flow cytometry channel measuring GFP to separate low and high, the Signal Prediction tool constructs a multi-dimensional non-linear threshold (represented as a random forest). The classifier not only includes the threshold method, but also incorporates all flow cytometry channels, such as forward scatter, side scatter, and other color channels.

The Signal Prediction tool builds a controls model from cleaned flow cytometry data to improve predictions on experimental samples. Figure 15A shows flow cytometry data from a positive and negative control before and after Signal Prediction cleaned the data. The red distribution is the original flow cytometry data, and the blue distribution is the cleaned flow cytometry data. The cleaned flow cytometry data is used to build a model for subsequent prediction on experimental samples. Figure 15B shows predictions made by a model built from the original control flow cytometry data (left column) and a model built from the cleaned control flow cytometry data (right column) on two experimental samples, here named Circuit 1 (top row) and Circuit 2 (bottom row). For each circuit, two replicates were measured. The orange distributions are data predicted to be off and the blue distributions are data predicted to be on. Shaded areas represent the standard deviations of the replicates.

The initial design of the Signal Prediction tool was only as good as the data collected from the positive and negative controls. In several experiments, bimodality was observed in GFP-positive cytometer channels for the positive and negative controls, see Figure 15A. This caused ambiguity in determining what events were high and low by the classifier as sub populations of events in one control were consistent with sub populations in the opposite control, shown in Figure 15B. The team updated the Signal Prediction tool to overcome this bimodality in the positive and negative controls, removing the poor control data by using the classifier's training/test split of the controls during cross-validation to identify the probability of each label predicted in the test set of each cross-validation fold. The control data was then "cleaned" where for each control type, a threshold on probability was applied that dropped events found below the threshold but maintained a minimal total event count of 10,000 events. A new classifier is then constructed on the remaining "clean" data and used it to predict signals for the experimental samples, as seen in Figure 15B. The resulting model self-assesses the probability that each control data point should be used in a respective group of labeled points and keeps only the most probable.

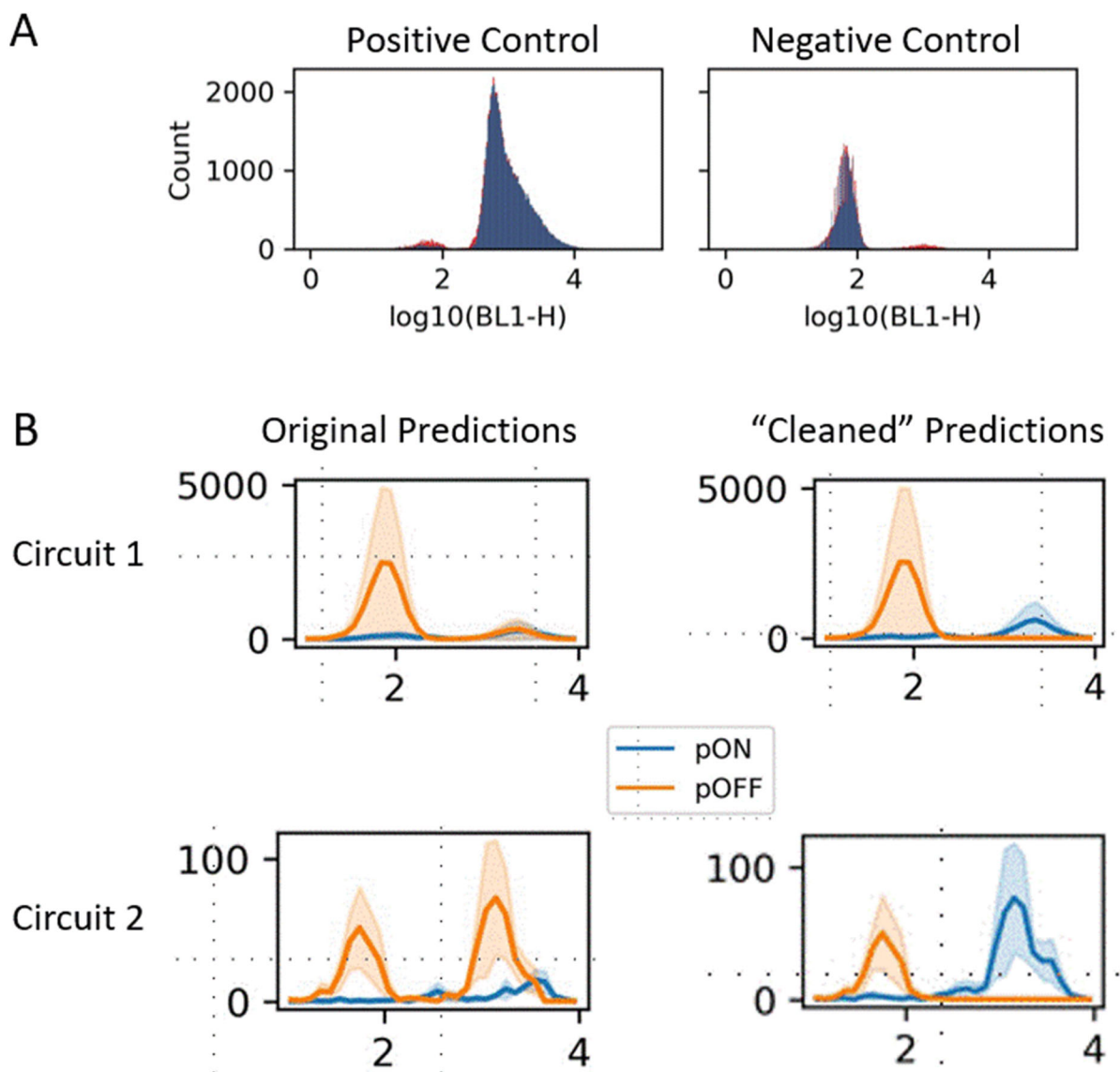


Figure 15. *The Signal Prediction tool builds a controls model from cleaned flow cytometry data to improve predictions on experimental samples.*

3.3.12 Replicate Checking

When analyzing the RNA-seq data, problems in the data can lead to poor results. In addition to looking over QC data from processing steps, it is also needed to analyze differences in expression levels between replicates. Software was developed for grouping experiments, analyzing the replicates, and flagging experiments and replicates for use in analysis. This provides information on what replicates can be used in analysis, and additionally why certain replicates or experiments are not usable. Different thresholds can be supplied, and spearman and pearson correlation are both run, as different types of analysis are more sensitive to different types of variation.

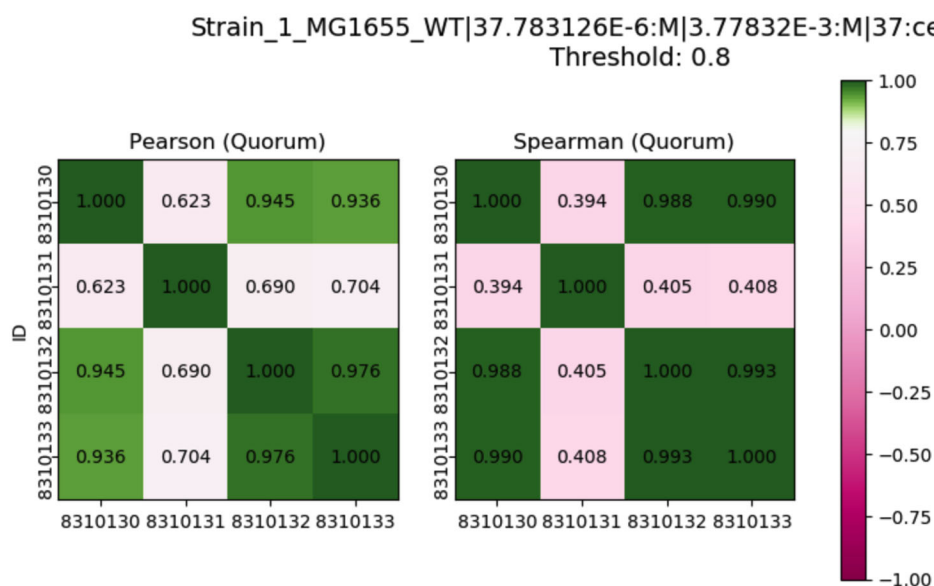
The user can specify how to group samples by experiment, e.g. "strain", "Isopropyl β -D-1-Thiogalactopyranoside (IPTG)", "Arabinose", "temp", "timepoint". Then each group is analyzed by computing pearson and spearman correlation on the quantification for each gene. Spearman considers only the ranked list of genes, while pearson considers the values for each gene. This is computed pairwise between the replicates.

The user sets a threshold for the correlations, which the software uses to categorize how good the replicates are for each experiment. For further analysis, just use replicates that are marked as "good", and do not use replicates marked as "bad" or "contra". Categories are:

- all good: all were above the threshold, so all replicates have a high degree of correlation between each other. all replicates are marked as "good".
- quorum: there is one set of replicates that have a correlation above the threshold, but other sets do not correlate. It is assumed the replicates in agreement are correct, and are marked as "good", and the other replicates are marked as "bad".
- no quorum: there are more than one pair of replicates that correlate, but the sets don't correlate to each other. It is assumed there is a contradiction between the different sets of replicates, so all replicates are marked as "contra".
- all bad: all replicates were below threshold. all replicates are marked as "bad".

The team also compared results to flags that were added to the data set to indicate if there were issues discovered with a sample. These flags indicate a biologist checked the data and saw there were problems with the samples. NA means there was no problem detected, otherwise there was some issue.

The software outputs a table summarizing the results for each set of replicates, and a plot and table to visualize the results for users, see Figure 16.



	Flags	Replicate	Spearman Replicates Label	Pearson Replicates Label
8310130	nan	0	good	good
8310131	Large raw r1/r2 size discrepancy AND low alignment percentage	1	bad	bad
8310132	nan	2	good	good
8310133	nan	3	good	good

Figure 16. Plot of QC Data for RNA-seq Replicates, Showing the Visualization of the Results for a Replicate and Comparing it to a Flag of Upstream QC Checks.

Worked on: 2019

Challenge Problem: Yeast States, Novel Chassis

Collaborators: Josh Urrutia, TACC

Code: https://gitlab.sd2e.org/gda/qc_rna_reps

Results: sd2e-community/shared-q1-workshop/adeckard/qc_rna_reps

3.4 Tools for automated data aggregation and acceleration of Design, Build, Test, Learn loop

3.4.1 Build Request and Build Request Parser

To aid in the automated DBTL workflow initiative, a semi-structured document for holding build information, such as part names, DNA sequences, and build intent, that could be ingested by Aquarium as well as read by a human was built. A Google Sheet template was created and named Build Request. This document was used for designing and building the plant transcription factor parts and for projects in the Novel Chassis group. Along with the Build Request, the Build Request Parser was built, which takes in a Build Request document and parses information for subsequent uploading to SynBioHub. This project was handed off to Jake Beal of BBN Technologies for further development.

3.4.2 Data Converge

Before Data Converge, anyone wanting to access data in the SD2 program, mainly analysts, had to either individually write code or ask someone to write code for them that could query the program's MongoDB (a noSQL database). As the program progressed, data would change causing previous query code to be obsolete and requiring new or updated code. Furthermore, due to analyst-specific querying, different datasets would be used in analyses causing issues with the comparison of results between different analyses. Tracking changes or problems in the data was also issue prone. Overall, these problems caused large spans of time, typically months, to be spent between when data was available to actionable results. DC was created to help remedy several of these issues.

DC provides a flexible bridge between changing, complicated data sources and multiple data consumers that require structured, consistent data, see Figure 17. DC handles data provenance tracking and maintains the same data structure even when database structures change, so analysts don't have to worry about updating their analyses. The main team responsibilities were developing querying code and formatting code for flow cytometry and CFU data. On top of formatting the measurement data from a flow cytometer, metadata was also extracted from each flow cytometry file, exposing additional features for analysis, such as number of events captured and flow volume for subsequent cell density estimates. Plate reader, RNA-seq and TASBE processed flow cytometry data was also handled by DC, this work being handled by Anastasia Deckard of GDA. DC was deployed as a TACC application and executed automatically when new data was available or manually when an update on data was needed.

Worked on: 2020-2021

Challenge Problem: All Recent

Developers: Anastasia Deckard, Rob Moseley. Data: George Zheng, Niall Gaffney, Mark Weston. TACC: Shweta Gopaulakrishnan, John Fonner. Testers: Bree Cummins, William Jordan, Robert Goldman.

Code: https://gitlab.sd2e.org/gda/data_converge

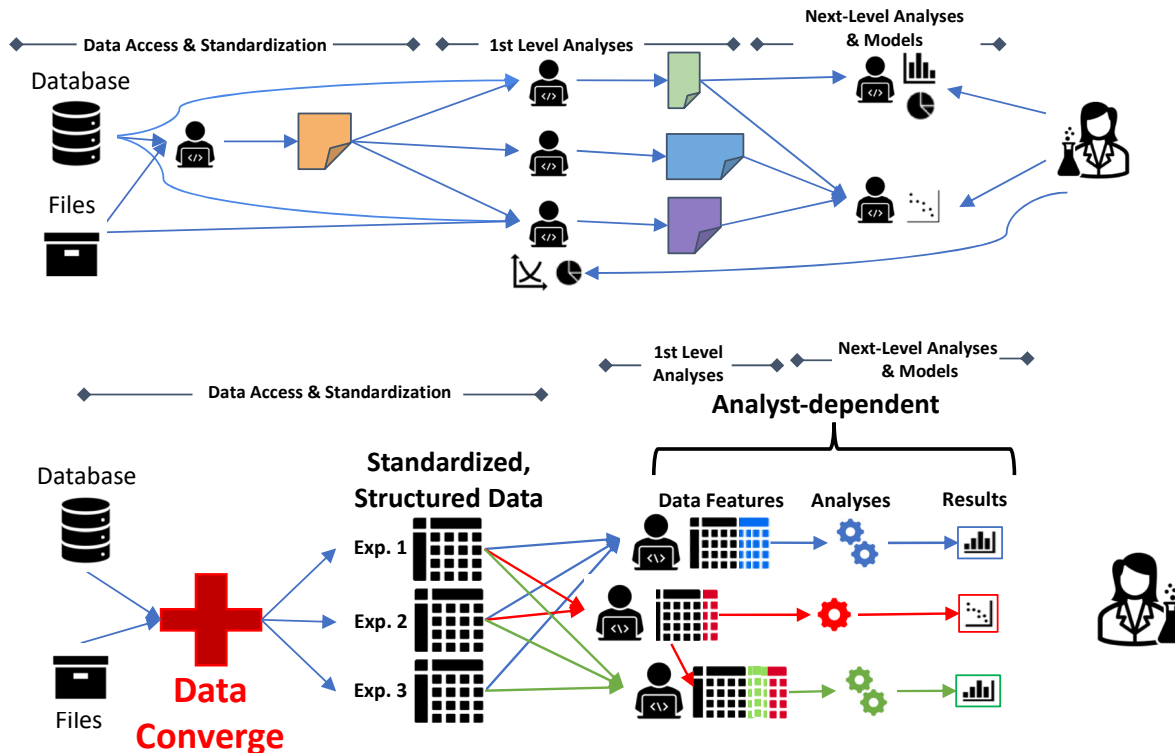


Figure 17. (Top) Previous Method for getting from Data to Results, (Bottom) Integration of Data Converge into the Method.

3.4.3 Precomputed Data Table

After DC was in production, it was realized that analyses were still manually executed, which was still causing a bottleneck between raw data and actionable results. Manual execution of analyses can also negatively impact reproducibility due to lack of provenance for both analysis code and input data. The Precomputed Data Table was developed by Duke University and George Zheng of Netrias, see Figure 18. The PDT improves analytical reproducibility by enabling automated and consistent execution of stereotyped analyses of different degrees of algorithmic complexity, as well as providing versioned, organized, analysis-ready, precomputed data for custom analyses. Additionally, the PDT provides users with rapid, consistent analysis of data of different types and from different experiments and the ability to automatically add additional data features (results from analyses) for higher-level analyses and modeling. Integrating the PDT into an automated experimental/computational system, like a DBTL loop, enables an increase in the rate at which the system can be iterated as actionable results are available within hours of data availability and more advanced analyses can be applied sooner. The combination of DC and the PDT cut down the time between data and actionable results from weeks to days to hours to days.

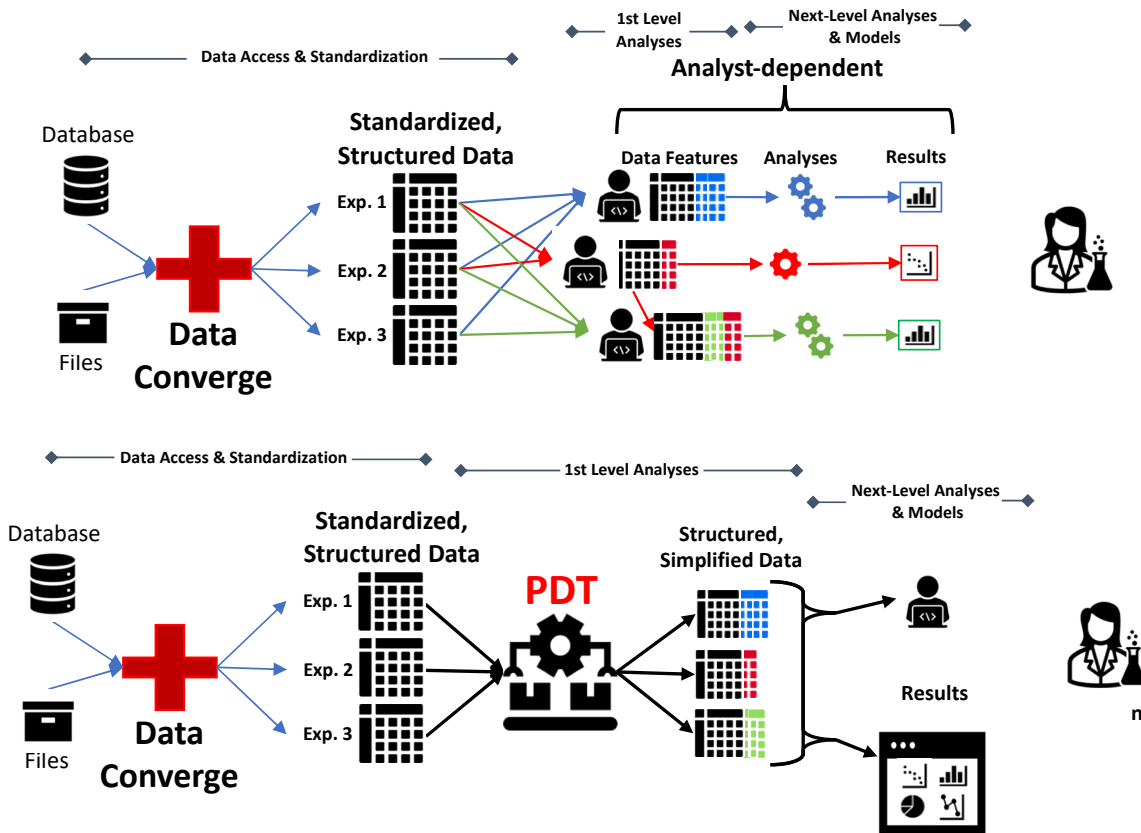


Figure 18. (Top) Method for getting from data to results with Data Converge integrated, (Bottom) Integration of Precomputed Data Table into the method.

3.5 Tools for Data Analyses and Automation

3.5.1 Performance Metrics

As with any experiment, it is important to analyze the circuit performance for these experiments. A package called *Performance Metrics* was developed with measures of performance for both aggregate and per sample values.

The metrics for aggregate performance compare the full ON and OFF groups for each grouping of samples. The goal was to compare the distributions of each group to analyze the fold difference between groups. This is accomplished by comparing the left versus right hand side of the distributions via percentiles and standard deviations, e.g. one could compute the difference between the 50% percentile of the ON group and the 50% percentile of the OFF group, as shown in Figure 19. Similarly, one can analyze the results on a per sample basis. In particular, for each individual ON, one looks at its difference (or ratio) to the median of the OFF states and then vice versa for each of the OFF samples.

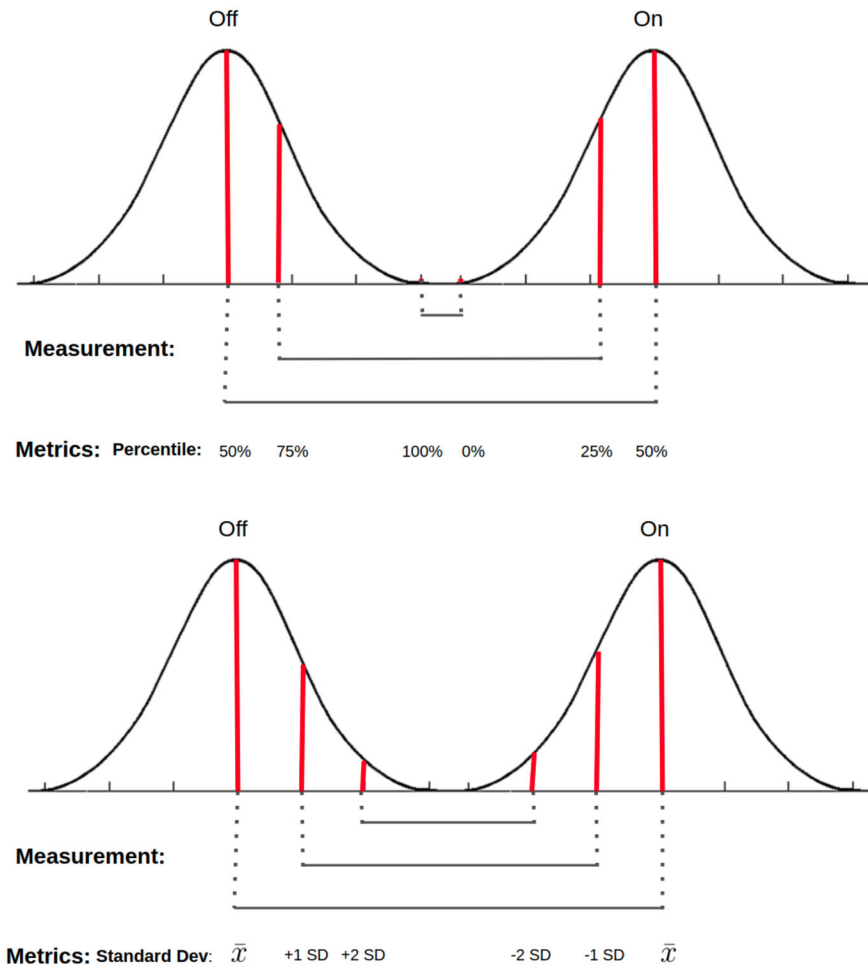


Figure 19. *How the metrics are computed using percentiles (top) and standard deviations (bottom). The normal distribution is used purely for illustrative purposes - no assumptions are made about the distribution of the data.*

For each metric one takes the difference and ratios of the left/right sides of the distributions for comparison and record the values at different thresholds. It also computes per sample metrics, where value is computed for each value in the OFF/ON group compared to a summary of the opposite state's distribution. For each individual ON, one looks at its difference (or ratio) to the median of the OFF states and then vice versa for each of the OFF samples.

To perform these computations, the package accepts a configuration file as input so users can easily utilize the tools without deep prior knowledge of python. The configuration file requires the column name of the experiment output, information about the ON and OFF group, as well as a dictionary of the groupings to run the metrics on. The program can also interpret ON/OFF states by setting "max", "min", and the program will find the highest and lowest values in a column to determine the groups.

After running the pipeline, a folder containing all of the results outputs to a user specified location. These results include tsv files with aggregate and per metrics as well as histograms of the fold change data and boxplots of the ON versus OFF groups.

The tabular output gives information about: `group_name`: which group (consisting of the combinations of the columns listed in the `group_cols_dict`) is being analyzed, `off_count`: number of values associated with the OFF state, `on_count`: number of values associated with the ON state, `num_std(/percentile)`: Number of standard deviations away from the mean used to compute metrics (/percentile used for metrics), `off_agg`: distribution value associated with either percentile (q) or mean + `num_std`*standard deviation, `on_agg`: distribution value associated with either 1-percentile (q) or mean - `num_std`*standard deviation, `diff`: difference between `on_agg` and `off_agg`, `ratio`: ratio between `on_agg` and `off_agg` (`on_agg` / `off_agg`).

The plots show summaries of the performance. The histograms of fold change data show for each `group_cols` combination listed in the config, the number of groups that have certain ratios of ON/OFF. A comparison of ON and OFF distributions, for each `group_cols` combination listed in the config, shows stacked boxplots comparing the distribution of ON values vs. the distribution of OFF values.

This is deployed as a TACC app and has a TACC reactor to automatically run it. The python package has been built, documented, and tested. Users can install the package locally using either anaconda or docker, and it also has a fully deployed agave app that runs automatically on TACC.

Worked on: 2019-2021

Challenge Problem: Yeast States

Code: https://gitlab.sd2e.org/gda/perform_metrics

Results: sd2e-projects/sd2e-project-48

3.5.2 Data Diagnosis

A package called Diagnose was developed to perform diagnostic tests for both experimental and experimental performance. These tests look for variables that are associated with variations in performance and identify which values of the variables are associated with good or bad performance. Additionally, tests identify if there appears to be any unintentional dependence between variables, e.g. one type of circuit is always put in the same well for experiments.

More specifically, three tests for performance and one test for dependence were developed that can be run either pre or post experiment. The first performance test is the Kruskal-Wallis H-test for categorical variables. This completes a non-parametric analysis of variance for the categorical variables grouped by a user selected variable. This test outputs several plots and tsv files for researchers to further analyze the results. Then one can compute the spearman correlation coefficients for continuous variables to measure the association between the performance variable and each continuous variable. These correlations and scatter plots are output between each pair of variables for further analysis. Additionally, the Kruskal-Wallis H-test is performed to analyze the

parts of the circuit. This is similar to the analysis of the categorical variables except now one is interested in studying whether different parts of the circuit are associated with better performance. Finally, there is the dependence test for randomization. This is primarily to study whether or not the experimental design is correct and thus ideally should be run prior to the experiment but can also be run afterwards to look for improvements for the next experiment. This test determines if the dependent variables were properly randomized by running a Chi-Squared Test for Independence between all pairs of categorical features. If the experiment was properly randomized then none of the pairs should be dependent. Together, all tests can provide researchers plenty of information on how their experiment did and where to revise it for future iterations.

The software uses a flexible configuration format in JSON, which allows a user to tell the program how their data is organized. The input is a dataframe with samples in rows and variables in columns. The user specifies their score/performance column. The user can group on a column name, e.g. group by gate to see performance per gate. different stakeholders will be interested in different groupings, like "lab", "strain_id", or "gate". The user can specify the path to data frame with variables and performance metric/score, path to output directory, what performance metric to analyze, and what variables to treat as groups. Analysis will be run separately on each variable in each group. The user also specifies what variables to run for the categorical and continuous analysis.

To improve automation, the software will look at all the variables and values to determine what it can analyze. The software will determine if there are enough values to analyze for each variable. The software will automatically make one variable (column) to group all your samples together, so that it will analyze all your samples together, so you do not need to add another variable for this. For the categorical analysis, the program will also analyze binned continuous variables. It will make a new column for each continuous variable called "variable BIN" with the values in five bins. Any variables or valuables that can't be used for analysis will be logged to output.

For each of these analyses, the program will generate tabular output for use in downstream processing and visualizations for users to review.

This is deployed as a TACC app and is automatically run by the PDT. A fully documented and tested python package has been built that is available in the SD2 git repo. Users can install the package locally using either anaconda or docker, and it also has a fully deployed agave app that runs automatically on TACC. The package accepts a configuration file as input so users can easily utilize the tools without deep prior knowledge of python if they wish to run it locally or they can run it through TACC's User Interface.

Worked on: 2019-2021

Challenge Problem: Yeast States

code: <https://gitlab.sd2e.org/gda/diagnose>

results: [sd2e-projects/sd2e-project-48](https://gitlab.sd2e.org/projects/sd2e-project-48)

3.5.3 Yeast States Analysis Pipeline

For analysis in Yeast States, the tools Data Converge, Perform Metrics, and Data Diagnose are used. These tools are automated into the SD2 system and run automatically when data finishes being ingested and initial processing. This pipeline has run 17 experiment references for Yeast States, for both flow cytometry and plate reader data. Performance Metrics first computes how well the samples are performing by comparing the median of measurements for the ON versus OFF states, as shown in Figure 20.

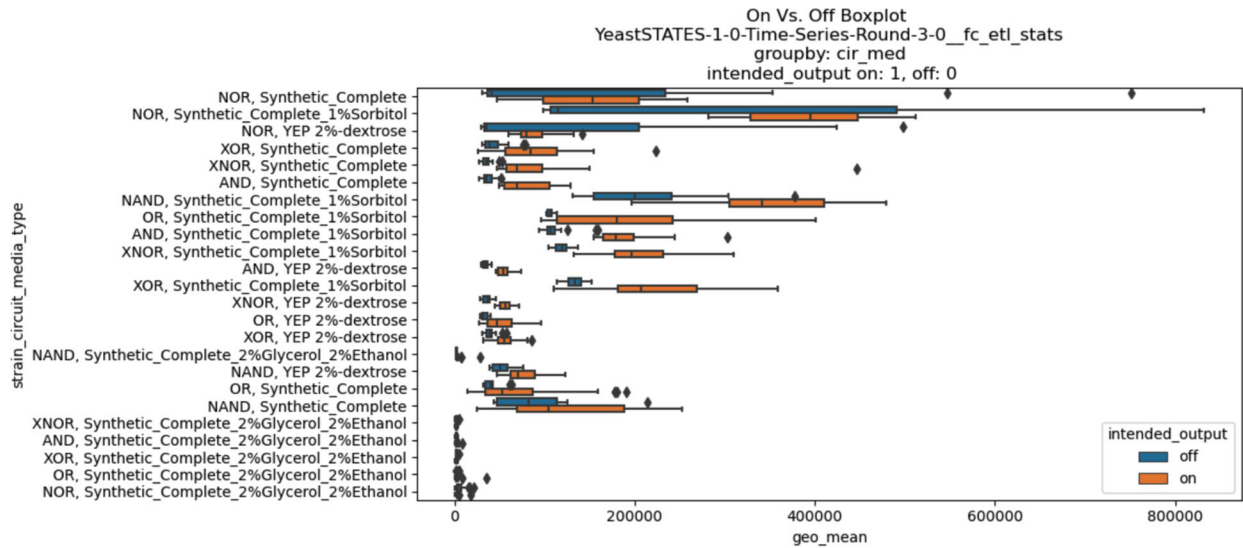


Figure 20. Perform Metrics plots of ON vs. OFF states for groups of data, sorted by their performance score.

Next, Data Diagnose uses the sample metrics and analyzes the metadata to determine what variables are associated with the differences in performance, as shown in Figure 21.

Performance Distributions for
timepoint = 64.0
with corrected KW p-values

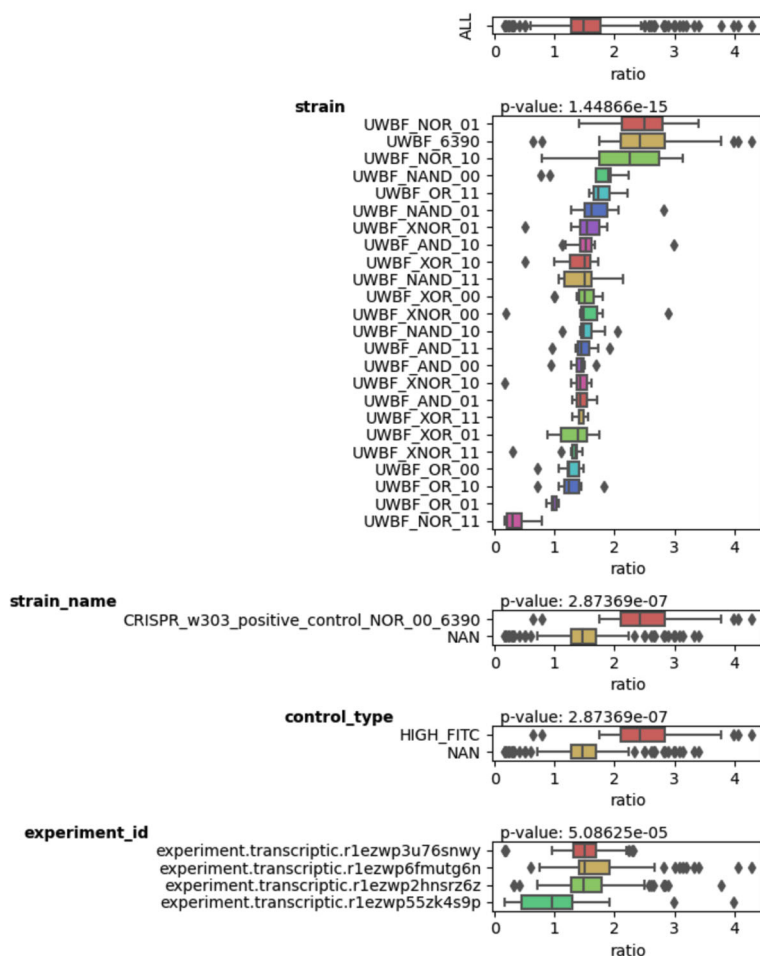


Figure 21. Data Diagnose showing the top variables that associate with differences in performance. The plots are ordered from most significant to least significant, top to bottom. Only the first few variables are shown in this plot.

This information from Performance Metrics and Data Diagnose is then used to determine what in the experiment worked well or poorly, and what should be run in the next experiment.

Worked on: 2019-2021

Challenge Problem: Yeast States

Results: sd2e-projects/sd2e-project-48, sd2e-projects/sd2e-project-33

3.5.4 Circuit Scoring

A primary focus of the Montana State University group in SD2 is the DSGRN-aided design of synthetic logic circuits and DSGRN predictions of their performance. The circuits are built with the intention of producing fluorescence signals for a specified logical truth table, such as AND, OR, etc. To assess the predictions of DSGRN on synthetic biology logical circuit performance, a way of analyzing experimental data was needed that gave a reliable indicator of circuit performance in terms of the desired logical truth table. That is, given a binary input state, such as 101 (chemical inducer 1 present, chemical inducer 2 absent, chemical inducer 3 present), the method requires a specific output state representing either an “ON” state or an “OFF” state. These states are measured in fluorescence, with sufficiently high fluorescence being ON and sufficiently low fluorescence being OFF. The measurement types available were plate reader, which gives the total fluorescence of a population of cells, and flow cytometry data, which returns a distribution of fluorescence measured at the single cell level. The information content from flow cytometry is much higher than from a plate reader and this data type was focused on for analysis.

The SD2 program began with the study of a 3-input logic circuit called “Rule 30”. The first task of the analysts in the program was to assess the performance of the Rule 30 network. In addition to having the desired ON and OFF states correctly associated to each input state as determined by the truth table, a high-performance circuit will also have a wide separation between the flow cytometry distributions between ON and OFF states. There are several techniques to measure the distance between two distributions. The methodology that was chosen is called the “earth mover’s distance”, which measures the minimum mass movement from one (normalized) distribution required to form the other distribution.

The earth mover’s distance is a pairwise distance and the distances between all pairs of the experimental flow cytometry distributions for a single circuit must be combined in order to assess the performance of a synthetic circuit as a whole. The combination method that was chosen is a graph theoretic technique called the normalized cut. Each experimental circuit sample is transformed into a weighted, complete graph by taking the collection of input states to be the set of nodes of the graph. For example, Rule 30 is a 3-input logic circuit and has $2^3 = 8$ binary input states, shown in Figure 22. Every pair of nodes in the graph is connected by an edge that is weighted by the earth mover’s distance. The normalized cut technique optimizes the division of the graph into two disconnected pieces by choosing the maximum dividing edge weights normalized by graph size. The decomposition into two pieces means that the pairwise distances within each partition are low compared to the distances between the partitions. This corresponds to a group of closely packed OFF flow cytometry distributions, and a similarly packed group of ON flow cytometry distributions. The weights of the discarded edges provide a relative measure of absolute distance between the ON and OFF histograms.

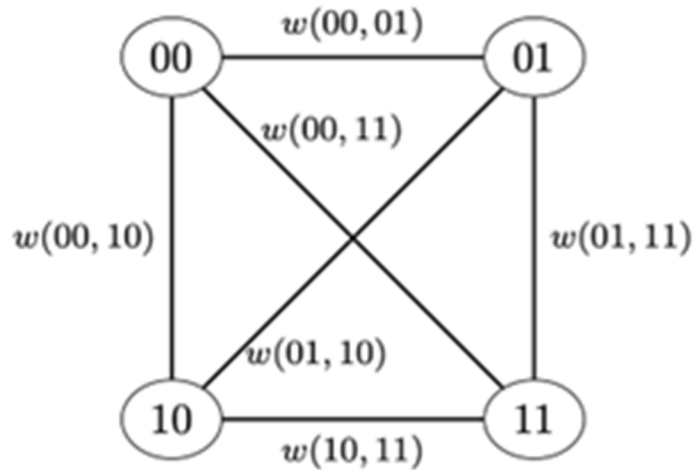


Figure 22. Cut Scoring as a Weighted Complete Graph.

Although this method worked well for Rule 30, it was discovered that there is a bias when the technique is used for 2-input logic circuits. The truth tables associated to these circuits include the well-known AND and OR logical expressions and their negations, and have 4 input states each, 00, 01, 10, and 11. The normalized cut technique was biased away from the exclusive OR function, XOR, and its negation XNOR. As the program moved away from Rule 30, the 2-input logic circuits became the focus of the Novel Chassis and Yeast States challenge problems and it became necessary to revisit the circuit scoring method. The team developed what was called the “average cut” circuit scoring method, where instead of normalizing by graph size, the maximum average of the removed edges was used. This effectively removed the bias in the 2-input circuits and agreed with human-curated circuit performance, as seen in Figure 23.

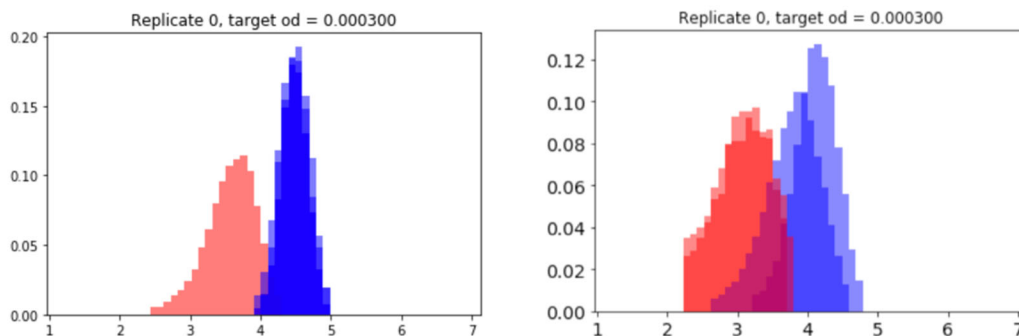


Figure 23. (Left) A correctly performing OR circuit that is scored as OR under both the normalized cut and average cut scores. (Right) A correctly performing XNOR circuit that was misclassified as OR under the normalized cut score but was correctly identified under the average cut score.

The next algorithmic improvement was the utilization of controls in the average cut circuit scoring. Since the average cut method only measures the cost of partitioning states into two groups, there is no way to assess the possibility of having a constant truth table; i.e. assessing when all

flow cytometry distributions are coincident. This is a common circuit failure type and it is important to be able to distinguish it. The use of experimental controls provided this ability. In principle, the negative and positive controls define the boundaries of the logic circuit responses. By taking the distance between the controls, one can normalize each of the weights in the circuit graph, leading to comparable results between experiments. This control-normalized scoring also allowed the development of an empirical score for a constant truth table. The fact that this score was empirically determined rather than analytically determined remains a weakness of the method and impedes generalizability.

The circuit scoring software implementing this algorithm is fully debugged, tested, documented, and publicly released, https://gitlab.com/breecummins/circuit_scoring.git. During SD2, the software ran on the Stampede2 cluster at Texas Advanced Computing Center and interacted with the SD2 data_converge reactor that provides standardized data products. The software at various points in its development has been used to evaluate a number of SD2 datasets, including Rule 30, the Novel Chassis NAND circuit, six 2-input logic circuits from Gander et al. (2017) in three widely disparate batches of experiments run at two different labs, and a recent OR/NOR redesign experiment.

The key features of the circuit scoring software include the following:

- User input is supplied through a structured JSON configuration file with general arguments for flexible dataframe manipulation.
- Output is returned in a Python pandas dataframe for easily manipulated results in visualization software such as Jupyter notebooks.
- The software supports generalized dataframe merging and row filtering.
- The comparison of truth table scores is enabled across aggregated experiments.

3.5.5 Cell Growth & Cell States Transcriptomics Analysis

The common goal of the Cell Growth (formerly Immortality challenge problem) and Cell States challenge problems was to look at states that exist in yeast under different conditions, especially for finding transcriptionally similar or dissimilar states. For CP-Immortality, the state of interest investigated was the encapsulated state in *Saccharomyces cerevisiae* to states that occur in other experimental conditions.

The first step was to gather, clean, and annotate Microarray data sets from a variety of sources: Nagarajan, Knitjeborg, Vos, Boender, Bristow, and Orlando, see Table 1. The transcriptomics pipeline was used for adding annotations and cleaning data sets. These data sets included 68 conditions with 440 total samples and ~6000 genes.

Table 1. Experiment Data Used in Analysis for Cell Growth and Cell States.

Conditions	Samples	Author	Platform
Encapsulated	BatchLog, BatchStationary, Chemostat, Pre-encapsulated, Encapsulated (1, 2, 5, 17 days)	Nagarajan 2014 GSE21187	Affymetrix 2.0
55 Chemostat Conditions	Combinations of conditions	Knijnenburg 2009 GSE11452	Affymetrix S98
Anaerobic retentostat	2, 9, 16, 22 days	Boender 2011 GSE22574	Affymetrix S98
Aerobic retentostat and chemostat	1.5, 3.5, 8, 16 days	Vos 2016 GSE77842	Affymetrix S98
WT and CLB mutant at 30c	~0.5 to 4.5 hours	Orlando 2008 GSE8799	Affymetrix 2.0
cell cycle mutants at 30c	18 - 360 minutes	Bristow 2014 GSE49650	Affymetrix 2.0

As the data sets were not normalized together, similarity in conditions were looked for using rank order correlation. The encapsulated state was also analyzed to find differentially expressed genes. 961 genes were found to be differentially expressed in Nagarajan 2014, and this subset was not highly correlated with other conditions, as seen in Figure 24.

CP-CellGrowth: state using subset of periodic

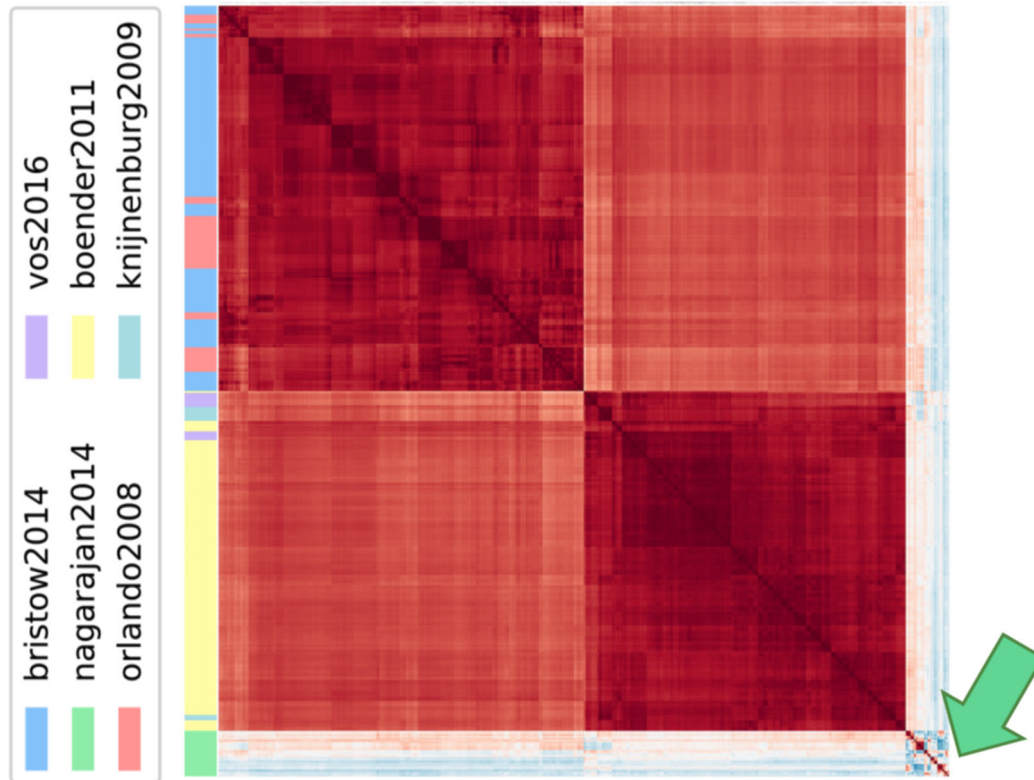


Figure 24. *Heatmap of the Correlations between Differentially Expressed Genes.*

Worked on: 2017-2018

Challenge Problems: Immortality, Cell Growth, Cell States

Collaborators: Katie Clowers, Gingko

Code: [sd2e-community/shared-q1-workshop/adeckard/cp-yeaststates](https://github.com/sd2e-community/shared-q1-workshop/adeckard/cp-yeaststates)

Results: [sd2e-community/shared-q1-workshop/adeckard/cp-immortality_corr](https://github.com/sd2e-community/shared-q1-workshop/adeckard/cp-immortality_corr)

3.5.6 Cell State Markers, Max Separation

The goal of this project was to find differentially expressed "sentinel" genes that uniquely identify different cell states to be used as fluorescent gene fusions. The general form of this algorithm is for each gene, use all replicates (and treat all time points as replicates), measure the difference in expression between conditions, find the genes that have the largest difference in expression between the conditions, find a set of genes that has a unique pattern of high and low for each of the conditions.

The first version of this algorithm used differential expression to measure the difference of a gene in different conditions. For differential expression algorithms, limma was used for microar-

ray and DESeq for RNA-seq data. Lists of potential reporters were selected using differential expression measurements to maximize separation between conditions and minimize variability between replicates and time points. Sets of genes were selected that generated a "truth table" of low/high expression that distinguish the conditions. This version was able to find sets of genes that could distinguish between three different conditions in microarray data and also in RNA-seq data.

For the next version of finding markers, a 'max_separation' algorithm was developed that looks for the biggest separation in the data between all conditions. Then the top separated genes were taken and automated finding high/low groups and a scoring system that finds combinations of genes that can distinguish conditions, and then ranks them by the minimum max separation of the genes. The scoring system is generalized to handle finding markers for larger numbers of conditions (e.g. two genes can distinguish up to four conditions, three genes can distinguish up to eight conditions, etc.). This returns a ranked list of the sets of genes that would be best able to distinguish a set of conditions, and includes information about how well they perform.

This method was tested on both RNA-seq and microarray data. For example, this was used on a set of time series data from 3 relevant cell growth states (provided from the Haase lab, and processed with the pipeline):

WT = wild-type cells in yeast extract peptone dextrose (YEPD) 2% glucose @ 300C (rich medium)

WT37d = wild-type cells in YEPD 2% glucose @ 370C (high temperature)

Rho0 = rho0 cells grown in YEPD 2% glucose @ 300C. (metabolic impairment)

This data was processed through the transcriptomics pipeline and then analyzed with the max separation algorithm, which returned a rank list of sets of genes that could be used to distinguish the conditions, and also the truth tables for the gene states and conditions, shown in Figure 25.

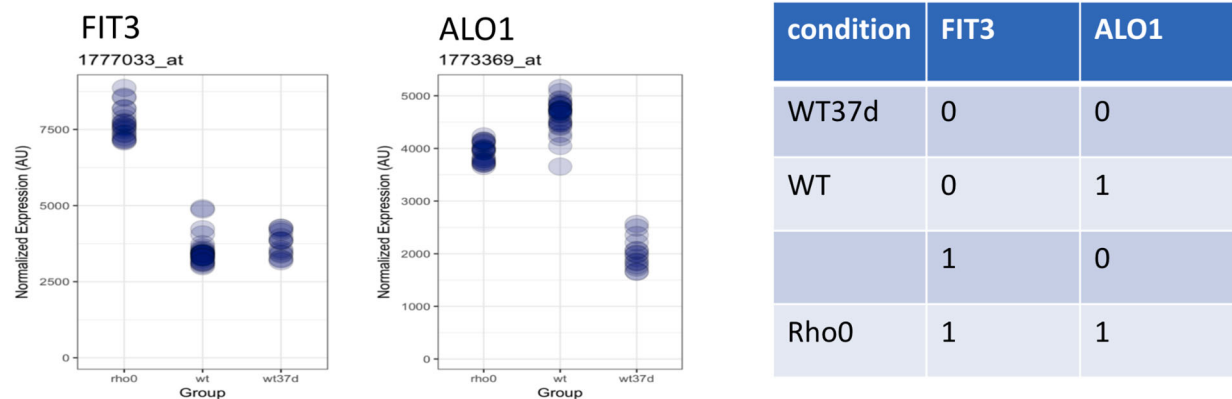


Figure 25. One set of possible "sentinel" genes that uniquely identify different cell states to be used as fluorescent gene fusions.

The results looked very promising. No markers using fluorescent gene fusions were built from these lists, so there was no experimental validation of the results.

Worked on: 2018-2019

Challenge Problems: Cell States, Yeast Gates, Yeast States

Code: https://gitlab.sd2e.org/gda/ge_max_separation

Results: sd2e-community/shared-q1-workshop/adeckard/cp-yeaststates/markers201806/, sd2e-community/shared-q1-workshop/adeckard/cp-yeaststates/max_separation_201907

3.5.7 Enhanced Gene Regulation Inference Technologies

The Duke Team's earliest efforts in the SD2 program aligned closely with the original proposal: to improve the functionality, efficiency, and usability of in-house tools that were designed to infer regulatory relationships and build functional gene regulatory network models in a semi-automated way using data collected from high-throughput experiments. In one effort towards these goals, the team improved the Local Edge Machine (LEM) network inference tool, which had been created by members of the Haase lab, but which lacked a robust, efficient, extensible, and broadly accessible implementation. During the first few quarters of the SD2 program, the team worked to expand a version of LEM written in Python to ensure compatibility with TACC HPC resources, including a robust, push-button parallelization scheme that takes advantage of the trivial parallelizability of the algorithm. These efforts hardened the implementation of LEM, making it scalable to inference on large network problems, which had previously been possible only with complicated, custom scripting that required pre-allocation of compute resources—which had made the existing implementation prone to job failure on HPC clusters. Additionally, the team integrated LEM into the TACC infrastructure as an Agave application which provided early feedback to TA4 on the defined application process. This codebase has been pushed to a public open-source repository (<https://gitlab.com/biochron/lempy>) which contains extensive documentation and examples, thus increasing the tool's potential impact by removing barriers to its use by other non-SD2 researchers. Over the course of the program the team built a semi-automated network inference pipeline that incorporated methods to identify core gene regulatory network control variables, propose potential models of interaction between the variables (using LEM), and build and test functional network models for consistency with measured data (using the DSGRN). The development of this larger network inference pipeline required extensive code development, which was nearing completion by the end of the program.

3.5.8 Immortality Challenge Problem Proposal

During Q1 2018, the Duke Team contributed to the development of the Immortality CP. The goal of this challenge problem was to understand pathways responsible for the yeast quiescence state, during which cells stop dividing, and resource use is limited. Specifically, the team was interested in the state yeast cells enter when colonies experience physical encapsulation in alginate beads. If initial efforts to unravel the cellular response to physical encapsulation were successful, the next stage of the challenge problem was to drive cells to this 'immortal' quiescence state without the need for encapsulation by using synthetic circuits and/or external stimuli. Specific tasks for the proposed CP included predicting nutrient conditions that lead to a gene expres-

sion state akin to the immortal state, identifying encapsulation-specific gene expression patterns, and then designing experiments to measure the dynamics of the transition of growing cells into the encapsulated state at the level of the transcriptome, proteome, phosphoproteome, and metabolome. The team developed and proposed analysis methods applicable to the Immortality CP using gene regulatory network inference tools, and differential expression methods, before any experimental data was made available to help guide what experiments to perform and what data to collect. One analysis plan was the use of LEM to propose models of interaction between gene regulators using time series RNA-seq data collected as cells transitioned into and out of the encapsulated state.

This CP made it to the proposal and early experimentation phase, with specific CP phases, tasks and approaches being outlined by the interested performers. Some experiments were conducted by Katie Flowers at Ginkgo Bioworks, generating Biolog plate reader measurements. However, after Q1 2019 because of issues with bead integrity and the limited feasibility of experimental designs, this CP was supplanted by other efforts.

3.5.9 Perovskite Data Processing Pipelines: Accounting for Temperature Variation Across 96-Well Plates

At a high level, the Perovskite CP aimed to explore the experimental Perovskite “recipe” space to increase the pace of discovery of novel perovskite crystals with useful properties. This goal led to the creation of a variety of software and hardware tools to automate the proposal, execution, and analysis of experiments. Many of these experiments used 96 well plates that were held at (roughly) constant temperatures. During Q1 2019 it was observed that there was a spatial dependence of well temperatures, leading to systematic variation in the actual temperatures of the wells in each plate. Since temperature can be a key parameter controlling experimental outcomes, correcting for deviations from the expected, nominal temperature became an important CP goal. That said, directly reducing the spatial dependence of temperature might have been difficult, requiring changes to some aspects of the experimental platforms. The Duke Team’s approach was to develop a model to adjust nominal temperature to actual temperature inside each well post-experiment, and thus improve the data features ingested by downstream ML models that were being trained to predict experimental outcomes and characterize the experimental parameter space.

Using accurate temperature measurements from a small number of experiments, multiple spatial-temperature-adjustment models were developed including isotropic and anisotropic Gaussian models that were either dependent on or independent of nominal temperature. Working with CP leads, a temperature-adjustment module was implemented based on these models that was easily adapted to the post-experiment data processing pipelines being developed in the Perovskite CP. The module ingested a dataframe of experimental results that encoded well sample features and applied one or more temperature adjustment models to add improved temperature estimates to these standardized dataframes. The team further streamlined the fitting of the temperature-adjustment models to enable other performers to rapidly retrain and reprocess existing experimental outcomes if new, accurate temperature data became available.

3.5.10 Data Sharing Initiative: Improving the Reproducibility and Generalizability of Published Results

In Q2-Q3 2019, the DARPA SD2 program engaged with data generators from outside the program in a data sharing initiative. SD2 performers were paired with outside labs who provided data to be analyzed using SD2 technologies. The assigned data partner, Mark Schnitzer's lab, a performer in the DARPA Neural Engineering System Design (NESD) program, collected and provided neural activity data in the form of calcium fluorescence levels on a large set of neurons in mouse hippocampus while the mouse explored a rectangular domain. The task was to better understand how the brain organizes neural activity in response to spatial inputs.

An earlier study on neural activity during spatial navigation (Giusti *et al.* (2015)) concluded that neuronal correlations showed underlying geometric organization but were not entirely consistent with the low-dimensional geometry of the stimulus (i.e., the spatial arena). Using topological data analysis tools, the team was able to confirm this finding using this new data modality being produced by Mark's lab. This TDA method happens to be invariant to a very large class of transformations of the data, and so the generalizability of this method to a different representation of neuronal activity demonstrates one of the original motivations for using it over classic methods for analyzing correlations in neuronal activity.

By leveraging previously unavailable TDA technologies, and computational resources made available by the SD2 program, the team was able to extend the Giusti analysis method to many more cells than could have previously been analyzed with this TDA approach. Because of computational limitations, the numbers of neurons that could be modelled had been limited to less than 100, but Dr. Schnitzer's data demanded at least twice that many neurons be modelled, and in some cases many times more than that. Newly developed algorithms and improved implementations as well as HPC resources enabled computations of topological features in the correlation structures for many more neurons that had been analyzed previously. Surprisingly, it was observed that the qualitative characteristics one would expect were not invariant to the number of neurons, and in fact qualitatively changed when analyzing sufficiently many neurons. Moreover, it was observed that the conclusions of Giusti *et al.* (2015) and the underlying organization of the activity depended on the particular subset of hippocampal neurons being analyzed.

This data sharing initiative benefited greatly from use of SD2 resources. For example, the Versioned Dataframe Repository tool enabled groups to easily share and communicate the content of various transformations of the data. While the tool was developed in the context of a totally different data set, its flexibility provided a way to rapidly share and justify updates to the data which allowed focus on the analysis rather than the wrangling.

On the analysis side, one of the major challenges faced was the tension between the time given for this initiative and the large number of hypotheses that emerged. Although full exploration of all these hypotheses couldn't occur, by using rapid development technologies on top of an HPC infrastructure, the data was able to be quickly explored in meaningful ways which led to many additional questions.

3.5.11 Topological Features of Protein Structure for Machine Learning Models

Starting in Q1 2018 and extending through the end of the program, a major focus of the teams SD2 research efforts occurred within the context of the Protein Design CP. In this challenge problem, the primary goal was to build data-driven tools that improve the *de novo* design of synthetic proteins. The initial focus was on developing models to improve the prediction of protein stability, which could then be experimentally estimated using experimental techniques developed by Gabe Rocklin in David Baker's lab.

Much effort focused on using TDA methods to generate novel, shape-based features of protein structures for use in downstream stability-model predictions. The computational demands of the challenge problem motivated significant improvements to software implementations of TDA technologies and the creation of new technologies as well as novel data-driven computational approaches. In summary, topology-based data visualization code was developed for the Protein Design CP, a code base for performing topological feature extraction from Protein Databank (PDB) files, a state-of-the-art and fully transitioned implementation of a popular topological feature generation tool called Persistence Images, showcased the use of data-driven feature extraction methodology known as CDER to learn interpretable, discriminating topological features, and created a suite of software tools—collectively called TopOpt—for automated TDA model selection through simultaneous topological feature generation and model hyperparameter optimization.

One of the first major challenges was developing analysis pipelines that could efficiently generate and store topological representations of proteins that were specified in PDB files. Early in the program, SD2 experimentalist collaborators had already created a massive collection (on the order of hundreds of thousands) of *de novo* designed synthetic mini-proteins, which were represented as labelled atomic point clouds in individual PDB files. Code was developed that ingests a PDB and computes the 0-,1-, and 2-dimentional persistent homology (PH) diagrams using weighted alpha filtrations. The output is a set of 2-dimensional point clouds that incorporate both atomic locations and identities in a compact representation of the multiscale connectivity, holes, and voids in the tertiary structure of the protein. These diagrams were stored as JSON files tied to the original PDB file.

Over a few weeks of model conceptualization and implementation the team then developed a data-driven ML method employing a novel feature-learning technology called CDER to learn topological/structural features of proteins associated with protein stability. A small number (6-21) of completely data-driven, interpretable, shape-based features was found that could be used on their own to achieve 85-90% of the stability-prediction classification performance as state-of-the-art models trained on ~100 expert-curated biophysical features. Moreover, by providing a quantified, interpretable characterization of protein structure, these features revealed the intuitive biological insight that more stable designs have an increased number of smaller voids and fewer large voids than unstable designs.

While exploring the use of TDA methods to predict protein stability, it was found that several published methods of transforming topological features into machine-learning-ready vectors required *ad hoc* choices of parameters which may not be able to be made ahead of time in a princi-

pled way. Also, some methods proved inefficient to compute on the massive, designed protein datasets.

The SD2 innovations that made a real difference were two technology solutions which promise to enhance the application of TDA methods to ML problems across domains and beyond the SD2 program. Software was developed to automatically optimize both the choices of parameters determining the representation of TDA features and the choice of ML model and its hyperparameters. This tool was called TopOpt. Importantly this tool reduces uncertainty and makes model selection systematic and automatic, thereby reducing the human effort required for complex model development and exploration. Components of the tool were optimized, reducing compute time on HPCs by as much as 135-fold as compared to publicly available versions of these component methods. These tools were also hardened, making them scalable, feature-rich, and user-friendly. By the end of the program, the team had completed implementation and began the process of transitioning the tool into a widely used open-source ecosystem called scikit-TDA. This transition will make the tools easily accessible to the broad community of researchers and modelers using TDA methods.

To showcase the utility of the TopOpt tool outside the SD2 program, several outside collaborators were invited to apply the tool to their data: 3D scans of bones exhibiting a range of porosities. Thus, within the SD2 program, the team showcased the use of this tool to automatically optimize model hyperparameters for models that predict synthetic protein stability using topological features, while separately outside collaborators applied it to models designed to predict the so-called trabecular number of bone tissue using topological features. The results of these studies were published in Motta *et al.* (2019).

3.5.12 Topological Data Analysis: Protein Stability

Application of persistent homology to identifying protein stability and geometric features of electron density fields requires the ability to efficiently compute and analyze persistent homology over numerous data sets. With this in mind, the following tools (<https://github.com/marciogameiro/ProteinPersistence>) were developed:

1. A Python library (with C++ backend using Computational Geometry Algorithms Library (CGAL) and Persistent Homology Algorithm Toolbox (PHAT)) that performs persistent homology code directly from PDB files for team 10 researchers at Hackathon.
2. A python-accessible tool to compute persistence diagrams from cubical electron density data.
3. Code to extract multiple subsets of atomic structures in proteins and efficient computation of persistent homology on these subsets.

The goal was to use the persistent homology information as input to standard statistical and machine learning techniques. The techniques typically require that the input data can be represented as vectors as this allows for standard computations associated with linear algebra. Persistence diagrams do not have a natural vector structure and thus there have been several different theoretical efforts to vectorize the persistent homology information. To make use of these ideas a variety of methods were implemented to vectorize persistence diagrams including modifications to

the original ideas either for the sake of efficiency or the greater flexibility to extract particular features.

Using protein structure data from the PDB, the teams developed computational tools were applied to a wide variety of proteins. Computations showed that modified version of the persistence diagram vectorization provides good linear fit and suggested the ability to identify protein fold structure within regions of the persistence diagram. Efforts to identify these regions and test their impact on the question of predicting protein stability were begun. In particular, the following types of computations were carried out.

1. Computed dual persistence diagrams, by applying linear regression to vectorized persistence diagrams, to protein folding data and to Rocklin data (protein stability data).
2. Extracted features from persistence diagrams from the Rocklin data set and started to work on the new dataset of 85k proteins.
3. Used the models trained on the 85k dataset to predict stability of the untested 200k dataset.
4. Performed parameter optimization for topological data analysis feature extraction.
5. Used these optimized features to predict stability using Machine Learning.
6. Started analyzing feature selection methods to select and combine TDA feature with Rosetta features for better predictions.

The results of these computations, as carried out by the Rutgers group, were frustrating. Simple analysis of the data strongly suggested that quantification of protein stability contained complementary information to that expressed by the original analysis of the Rocklin data set. However, efforts to capitalize on this information were not successful. Nevertheless, with the technology developed, the team explored several other avenues of analysis of protein structure including the following.

1. Developed methods to extract information from the multiple persistence diagrams to be used for machine learning. Preliminary results along this line improved stability prediction from 0.54 (14% error) to 0.48 (13% error) Root Mean Square Error (RMSE).
2. Examined whether using persistence diagrams bases on specific atoms, e.g. carbon, in proteins would provide alternative information for predicting folding or stability of proteins. While this clearly provides less information than working with the complete molecular structure, it was surprising how much information can be extracted from this information alone.

A fundamental theoretical question is to understand the limits of persistent homology as a tool to predict protein stability. Being based on algebraic topology, persistent homology depends on limited geometric information, thus an answer to this question could lead to an understanding of how much geometry plays a role in protein stability. Using Mapper, another basic tool from topological data analysis to explore the distribution of shapes of proteins, did not obtain conclusive results.

Finally, efforts with regard to the electron density fields were minimal, but did devise pseudo-code for key steps to analyze via discrete Morse theory algorithms (see the discussion on Conley

complexes below) the geometry of scalar spatial data, e.g. electron density fields, when expressed using the data structure of cubical complexes.

3.5.13 Responding to Experimental Surprise: Data Driven Models for Identifying New Informative Experiments

A second major research thrust in service of the overarching Protein Design CP goal emerged Q4-2018/Q1-2019. In collaboration with Hugh Haddox, Gabe Rocklin, and TwoSix Labs, the team conceptualized a data-driven approach to identify limitations in the Rosetta modelling framework and potentially discovered previously unknown principles of stable protein design using the Rocklin-designed mini protein library. A ML method was developed to identify unstable designs which were proximal to highly stable designs in biophysical feature space. It was hypothesized that these “unexpected unstable designs” would be highly informative since their instability may have been due to ostensibly small, but highly destabilizing errors in the amino-acid sequences proposed by Rosetta’s design systems. Thus, by interrogating over one-hundred biophysical features of approximately 15,000 previously published protein sequences and searching for disparities between ML model stability predictions and the associated experimentally measured stability scores, the research hoped to focus the subject matter expert’s attention on the most informative regions of biophysical feature space. The hypothesis was that the resulting ranking by disparity between model predicted stability and measured stability would reflect the extent to which the existing set of hand-curated biophysical features and Rosetta’s existing design methodologies failed to account for important (perhaps subtle) determinants of stability.

A random-forest classifier was used to rank designs according to the disparity between their experimental stability scores and the stability scores of their “closest” neighbors, as determined by the trained model. By using a random forest model to measure proximity in the biophysical feature space, the relative importance of those features are implicitly increased, which were strongly associated with protein stability and discounted those features which were less discriminating when determining proximity between designs, since it was not entirely known which features, or which combination of features were the important determinants of stability.

The data-driven approach taken suggested 21 of approximately 15,000 of the original designs to re-assay for stability following exhaustive single-amino-acid mutations. Using simple scripting, a design file containing exhaustive single-site mutations of these 21 designs was generated, which yielded 14,846 designs (including 382 ladder proteins needed for normalization of experimental results) to be built and assayed using the high-throughput Rocklin stability assay. Excitingly, for most of the original 21 designs, there were single amino-acid mutations that rescued the design, making it stable. After careful consideration of the mutations which rescued stability, Hugh Haddox observed that many of these stabilizing mutations were found to relieve energetically unfavorable steric clashes that Rosetta had designed in the protein core, suggesting a physical inaccuracy in Rosetta’s energy function. This observation led to re-tuning Rosetta’s energy function parameters, which resolved the bias and substantially improved Rosetta’s performance in a standard set of benchmarks used to quantify Rosetta’s physical accuracy.

This initial investigation has since revealed additional limitations to the parameters within Rosetta’s modelling framework and has led to a major improvement to Rosetta’s underlying model energy function. This has implications for all aspects of Rosetta-based investigations and de-

signs-not only in *de novo* design of stable mini proteins, but in design of protein-protein binders, protein-DNA binders, etc. Since Rosetta is an established standard used by many labs and investigators around the world, the impact of these refinements is substantial.

3.5.14 Identification of DNA-binders from Primary Sequence

In the final months of the SD2 program, in collaboration with TwoSix Labs, a hypothesis-driven flow chart of increasingly challenging queries was developed whose goal was the development of ML models that could both predict and improve the design of targeted DNA-binders. A full realization of this goal could have enabled, among other things, the *de novo* design of sequence-specific DNA binders that are orthogonal to the host genome. Such designs could support manufacturing components for synthetic circuit design with minimal disruption to normal host function. The approach taken was to mitigate risk of failure, while making iterative improvements to evermore capable models, starting with the most basic question: can one predict whether a naturally occurring protein binds to DNA or not. The published literature contained numerous models which purported to predict DNA binding from primary sequence and other biomolecular features and/or phylogeny. It was discovered that many of these models had been built on standard datasets which contained numerous problems. For one, over 80% of DNA-binding protein sequences in the previously published benchmark test set were also in the benchmark training set. Also, the benchmark training set contained duplicate entries for identical monomers making up homodimers and homotrimers, and some entries in the benchmark training set were not protein sequences at all but were DNA sequences instead. The benchmark dataset was also very small compared to the amount of relevant available data. Finally, numerous studies utilized complicated modern, deep learning models without benchmarking performance against simple baseline models, this effort was motivated to compare the reported performance of published models against simpler, perhaps more interpretable models.

Over the course of study, a new benchmark dataset was created for training and testing DNA-binding models which was much larger than the existing set and did not suffer from its limitations. Also, new evaluative benchmark tasks have been created that provided a better assessment of how models would perform on real-world protein annotation tasks. A simple new model for the prediction of DNA-binding proteins has been proposed and found that, on the improved datasets, it outperformed two previously published models, in some cases drastically. Finally, the research challenged the models to predict DNA-binding across taxa and across kingdoms and found that even simple baseline models can outperform previously published models built with complex architectures. These results were recently published in a leading bioinformatics journal, the Oxford Academic journal *Bioinformatics* (Zaitzeff *et al.* (2021)).

Following the success of the first level of this project's risk-mitigation model hierarchy (i.e., distinguish DNA-binder vs. non-DNA-binder), the team briefly endeavored to develop models capable of distinguishing sequence-specific binding. However, the program ended before the team was able to fully realize the vision of develop models capable of predicting specific DNA-sequence binding. This remains a major thrust of future research efforts due to its importance in synthetic and systems biology.

3.6 Automating Scientific Knowledge Extraaction

3.6.1 Support Data Acquisition For Mathematical Modelers

To get data to modelers and make it easier to use, a variety of data sources were accessed, to help in standardizing data, and formatting it into data frames. Data currently comes from three sources: Integrative Model of Mobile Media Use and Need Experiences (IM3UNE), LocaleDB, and public web service application programming interfaces (APIs). In addition to making additional data available and easier to query through IM3UNE, python queries were created with examples to make it easier to access this diversity of data sources and to use the data.

IM3UNE and LocaleDB are databases that are loaded with data from other external data providers. The data providers used by IM3UNE and LocaleDB change their data through time, so processes were developed to keep data access up to date or to find new data providers.

Certain data sets have been made available through web service APIs, which allows for directly querying remote data sources. Very complex queries can be written, similar to the querying functionality available in databases, but using a different syntax. The data sets being used include:

COVID-19 Diagnostic Laboratory Testing (Polymerase Chain Reaction (PCR) Testing) Time Series

COVID-19 State and County Policy Orders

Centers for Disease Control and Prevention (CDC) COVID Vaccination (by county and day)

CDC COVID Community Transmission Level (by county and day)

U.S. Department of Health and Human Services (HHS) Patient Impact and Hospital capacity (by hospital and by week)

This effort has also curated a list of available data sources, and used it to track what sources are used, might be used in the future, as well as changes to the data sources. The list has been updated and extended including data types, sources, and resolution. Description location and access locations have been added to the list. The list was also updated with information about limitations on availability of data (e.g. due to services no longer being provided or licensing), and marked which data sources are available through which providers and which can be accessed from the ASKE Data Access Layer.

A data access layer was built for each data source in python. With python and conda environments, users can quickly and easily install all the packages needed to access all these data sources. A configuration file was created for storing data access information/credentials and functions to build connections to the different data sources. For each database/web service, python scripts were created that handle connecting to different data sources and building common queries, which allows users to query by date range and location without having to learn the details of the different data sources. The team also created target-to-source mappings for data fields to make it easier for analysts by standardizing commonly queried fields such as date, locations, Federal Information Processing Standard (FIPS) codes, etc. This allows the user to query

and ingest data products without worrying about all the different names for date and location fields in the underlying data sets. So far, analysts were mainly interested in having use of the software to query data for them and then give them a package of all the data that meets their requirements.

One data source was LocaleDB, which was built by the University of Pittsburgh team on the ASKE-E project. This is a Postgres Database run within docker, with commands to automate loading data from a variety of sources. This system was used and loaded data in for: county information, COVID cases, deaths; flu vaccinations, and COVID non-pharmaceutical interventions.

The team pulled, cleaned, and prepared data for analysts. For example, Cytidine 5'-triphosphate (CTP) data was used to analyze dynamics mid 2020 where the pandemic was established in the US, but before vaccinations. This data was analyzed for its cleanliness, completeness, and interesting dynamics to identify US states that would be of interest in Susceptible-Exposed-Infectious-Removed (SEIR) modeling tasks at GDA and Galois.

The team met with Mitre about data sharing, and shared:

- 16 dataframes generated by the data access layer. The queries were run for a set date range and location.
- Data source list with updated information. The team has offered to load/query any other data of interest within the system.
- Entity Relationship Diagrams for the IM3UNE and LocaleDB databases, and summaries on all the available fields and record counts contained in the databases.
- Access to the IM3UNE database, so that more complicated queries can be run (beyond the set of queries provided by the data access layer). Created an example script for querying the database in python and loading the data directly into pandas data frames.

3.6.2 Data-Driven Rapid Adaptation of Models

GDA has determined capabilities to select and fuse data, perform variable selection, and return the most relevant variables to modelers. Selection will be driven by (a) locality, (b) data availability, and (c) user query. These include such capabilities as disaggregation, fusion and local dimension reduction to reduce the number of variables and parameters necessary to use multiple data streams together. Imagine an analyst interested in specifying queries and locality and wants to receive results about the data availability and perform selection and fusion to enable modelers to address their queries. This requires two important parts: Models-to-Data and Data-to-Models. In Models-to-Data, the problems are seen as being those related to the data error detection and data error correction. In Data-to-Models, the models are improved from the data by using variable selection/dimensionality reduction.

Methods for adapting models includes discussions of time series data and delay embeddings, which provide a motivation for developing dimension reduction techniques, especially nonlinear dimension reduction tools. This also covers categories of variable selection/dimensionality reduction, their properties, and how to apply them. Feature selection methods aim to reduce the dimension of a feature space by sub-selecting a subset of features to use in model development, in the hopes of improving some aspect of model performance; for example, model accuracy, generalizability, or computational efficiency in training or evaluation. Like feature selection, feature extraction is generally used to reduce the dimension of a model input feature space via a transformation from one feature space into another lower-dimensional one. That said, in principle, the target feature space of the extraction transformation need not be lower dimensional than the starting feature space, as there may be modeling advantages to mapping into a higher-dimensional space. Additional coverage of statistical methods for variable selection, and Aggregation/Disaggregation of Data allow one to reduce the number of variables in a model by aggregating several other variables of the same type.

Examples also provide coverage of where these types of methods could be used in epidemiological models, as well as a selection of specific methods implementations, such as Principal Curves, Local PCA, Multidimensional scaling (MDS), Isomap, etc. that could be used.

Implementations have outlined: dimension reduction, feature selection dimensionality reduction techniques, feature extraction, feature selection, feature importance, sequential feature selection methods, and autoencoders.

GDA's own custom delay embedding functions were described in Python, which efficiently performs embeddings on time series data of arbitrary dimension into a chosen embedding space using user-selected delays and strides. GDA has also implemented in Python an established heuristic method called the false-nearest-neighbor criterion for informing the choice of embedding dimension.

Additionally, GDA has implemented and demonstrated the use of linear regression methods for performing disaggregation of COVID-19 cases and hospitalization data in Jupyter notebooks.

A meeting with the Galois team was held due to a shared interest in comparing model complexity to the model's ability to account for data. The Duke Team conceptualized and ran several simulations using an agent-based model, developed to model Duke University and its surveillance testing and contact tracing program. These simulations were designed to challenge standard SEIR model output. In particular, the team developed:

- A model which incorporated an outside pool of infections which drove infections within the modeled population of agents, and
- A model that incorporated two subpopulations--a vaccinated and an unvaccinated group--which exhibited asymmetric interaction behaviors within and across groups.

Galois planned to use their modeling framework to rapidly deploy increasingly complex SEIR model variants and show iterative improvement fitting to realistic case count time series data. A private git repository was utilized to share model code and the simulated data, and suggested

SEIR models that might be able to recapitulate that data. The team gathered, reviewed and selected real time series data and visualizations.

3.6.3 Metrics and Evaluations On Model Performance

GDA looked at metrics to assess and compare the effectiveness of different models used in ASKE. This included metrics for Goodness of fit, Computational complexity, Model complexity, Algorithmic complexity, Robustness, and Sensitivity. The team examined what types of information about the models must exist to use each type of metric, for example, can the model be treated as a black box or will examination of model internals like parameters or network structure be necessary. The team also implemented some of the metrics and a demo application for applying all the implemented metrics to models.

Consider two types of metrics:

Assessment: computing the metric(s) for a single model.

Comparison: comparing two models and reporting the differences in their metrics.

Divide the metrics into different levels, depending on how much information is required about the model:

Level 1: Requires only output from a simulation of a model.

Level 2: Requires input and output from a model.

Level 3: Requires representation of the model, mathematical framework, and/or simulation code.

For each of these levels, the following metrics were implemented:

Level 1: Requires only output from the model and real observations. Model output dynamics are provided in the specified formats.

- Goodness of fit.
 - actual time series vs a single forecasted time series
 - mae: Mean Absolute Error
 - mse: Mean Squared Error
 - rmse: Root Mean Squared Error
 - r2: R-Squared
 - actual time series vs a set of forecasted time series

- `crps_ensemble`: Empirical estimates of the N continuous ranked probability scores based on K forecasts of N time points / variables.
- `crps_gaussian`: Exact calculation of the CRPS assuming time series forecasts follow Gaussian distributions.
- `crps_quadrature`: Numerically estimated calculation of the CRPS assuming time series forecasts follow a specified distribution.
- `log_score`: the log of the predicted probability of the realized outcome.
- `interval_score`: proper score of forecast accuracy applicable to $(1-\alpha) \times 100\%$ confidence interval forecasts.
- `Weighted_interval_score`: A proper score of forecast accuracy applicable to a collection of $(1-\alpha_k) \times 100\%$ confidence interval forecasts.

Level 2: requires sets of inputs and outputs from a model. Model input and output is provided in a specified format. The model will need to be executed by another system/service and will not be automated in this tool.

- Robustness, Sensitivity. For robustness and sensitivity measures one needs to be able to run the models at different parameter values and measure the resulting outputs, so this requires access to the model to run/rerun. Recommend using the SALib for their metrics.

Level 3: requires representation of the model, mathematical framework, and/or simulation code. These need to be provided in well-defined formats so that results will be comparable. Simplified examples were generated for these by representing the models as graphs.

- Model complexity: parameters
 - `param_count`: sum of parameters.
 - `param_key_set`: compare the set of parameters between two models.
 - `param_values`: align each parameter between two models, and then compare the values of the parameters.
- Model complexity: structure
 - `node_count`: count of nodes in one model.
 - `edge_count`: count of edges in one model.
 - `node_list_diff`: compare two models and find differences in nodes.

- `edge_list_diff`: compare two models and find differences in edges.
- `centrality`: compute degree centrality and compare between two models.

The team also implemented extensible software for running metrics. For each type of data, the required input was defined for running a set of metrics as well as what will be returned. All model metrics that work for a given set of inputs are executed together. The system is extensible for adding new metrics functions. The current preference is to use existing packages where available, and to wrap them for use in automated systems. A list of metrics and their corresponding functions were then built and used to automate running the metrics. A report is returned for all the metrics. The report will be in a JSON file in the form `metric: value` or `dictionary of submetrics: value`. This will allow these results to be used by other analysis/automated systems.

The team met with members from Galois and discussed metrics that would be of interest for different types of models.

3.7 Transitioning Technology to New Stakeholders

3.7.1 National Primate Center: Tulane (NIH)

The aim of the DARPA Synergistic Discovery and Design program was to build a highly collaborative and interdisciplinary socio-technical system (STS) infrastructure to support a research architecture in which experimentalists are geographically distributed but collaborating to perform experiments focused on coordinated hypotheses. In this system, the data and associated metadata are pushed to a central data repository where they are pre-processed, normalized and then delivered to geographically distributed analysts in standardized data frames. This research architecture coupled with the supporting STS is particularly well-suited for automation, giving rise to rapid and highly reproducible results.

Transition Partners. The Tulane National Primate Research Center (NPRC), through support from NIH/Office of Research Infrastructure Programs (ORIP) and National Institute of Allergy and Infectious Diseases (NIAID), is serving as the Nonhuman Primate Coordinating Center, for a program aimed at accelerating research on COVID-19 vaccine and therapeutics. The research program shares a common architecture with the SD2 program, where NPRCs are distributed around the country, are performing harmonized master protocols, with optimized standard operating procedures, and the raw data are to be housed within a centralized study level data center at Tulane. This effort has involved the Coronavirus Vaccine and Therapeutic Evaluation Network (COVTEN), comprised of representatives from all 7 National Primate Research Centers. The standard operating procedures (SOPs) have been developed with participation of 11 subcommittees within COVTEN, and approximately 30 members. The goal of this architecture is to be able to perform and compare studies at any NPRC and to be able to share control groups at other centers. While this structure optimizes study performance, reduces the number of animals potentially needed and enhances rigor and reproducibility, there is an urgent need to be able to harmonize data outputs from different NPRCs, often using different instrumentation, to further optimize the ability to compare studies across Centers, and mine data from diverse locations. This current non-human primate (NHP) CC architecture can facilitate rapid analysis, the deployment of AI/ML tools and enable rigor and reproducibility. However, the current architecture is not suffi-

ciently capable on its own to harmonize the actual data outputs from different Centers. This would require extensive computational tooling and development of an STS to support rapid analysis, and rigor and reproducibility. Thus, transitioning the STS ecosystem to support the NPRC effort will save substantial time in effort for Tulane as the tooling and STS from SD2 need only be adapted rather than developed *de novo* for the NPRC research program.

If the transition of the SD2 ecosystem is successful, the primate centers will have a new rapid response infrastructure that will enable an agile response to the current pandemic as well as future biological threats. Automation will increase the speed and reproducibility of the research which will maximize resources. As well, increased reproducibility will facilitate the shared use of control group animals by multiple centers, minimizing animal resources and expense.

Finally, the development of this STS infrastructure is a solution well-aligned with Rigor and Reproducibility Goals of the NIH Strategic Plan that will have NIH applications beyond the Tulane/NPRC effort.

Successes to date. Automating Flow Cytometric analyses. The team has used SD2 tooling (TASBE) and workflows for analyzing Flow Cytometry Data at Tulane and demonstrated accelerated analysis of gated flow cytometry samples. The SD2 approach is resilient to re-gating, saving time downstream as panels are adjusted and sample throughput increases.

Open-source flow analytics software TASBE (<https://github.com/TASBE/TASBEFlowAnalytics>) supports a variety of gating strategies, including a gaussian mixture model clustering algorithm that allows the configuration of a variable number of components, k , for which one wants to create gates. When compared to hand-gating, this clustering approach will still capture the divisions in the cell populations without relying on specific cutoffs. In these cases, a hand-gating approach will need to either re-draw the gates (average ~20 seconds per gate) or the existing hard gates will be used, potentially dropping relevant experimental data. The time savings increase as complexity of the panel increases, which roughly corresponds to the number of gates and color channels being used. There are also time-savings related to compensation, which takes about 5 minutes per panel, per time point for analysis. TASBE includes built-in support for spectral overlap and fluorescence compensation taking only seconds per sample. As more time panels are run, savings here will increase as well. Finally, TASBE enables normalization of data across instruments, allowing for the application of a single gating strategy across instruments, saving time and increasing reproducibility.

TASBE gating takes some upfront time to define. Gating strategies are implemented using MATLAB code to correspond to the hand-drawn gates flow analysts provide. For a simple pilot project that ran 3 gates (see below), it took ~1 hour to implement and test the full strategy. While this upfront cost is slower than the hand-drawn gates a human might draw, once implemented samples can be run and re-run through the pipeline with no additional user interaction required.

Estimates for Time Savings for Flow Cytometric Analyses (Based on Pilot Studies). Considering a representative NK whole blood 13 color panel typically run at Tulane. With 28 time points and 224 total samples (6-13 per time point), and 45 gates hand drawn for each time point. The total analysis time with gating and compensation is estimated to be 57 hours and 46 minutes. For automated TASBE runs, as discussed above, the component gating will be resilient across

timepoints, eliminating the need to re-draw gates for new timepoints once the initial gating strategy has been implemented. Assuming a high ceiling of ~20 minutes per gate to implement as part of protocol setup and 1 hour to validate, there is an upfront cost of 16 hours. Total analysis, normalization, and ingest for 224 samples will be under 1 hour, for a total end to end time of 17 hours, a 70% savings.

Immediate Goals. Rigor and Reproducibility in Flow Cytometric Analyses. Reproducible results arise not just from the precise execution of experimental protocols, but from standardized approaches that extend through the entire experimental workflow; from well-defined hypotheses to protocol development, experimental execution, data and metadata storage, extract, transform, load (ETL), and pre-processing and then through the production of common data frames and explainable analyses.

The Tulane group has already developed a plan for harmonization of protocols and experimental execution. During September of 2020 a document was compiled entitled; “Harmonized Standard Operating Procedures for COVID-19 Research Conducted at National Primate Research Centers.” This document was a coordinated effort between all 7 NPRCs and was formed in collaboration with the NPRC CoV Vaccine and Therapeutic Evaluation Network, along with Tulane National Primate Research Center’s (TNPRC) Quality Assurance Program. SOPs were developed with subsequently standardized practices amongst subject matter experts from each center in various topical NHP scientific and research methodology-based areas. These areas include inclusion criteria, sample collection, pathology, samples processing, virology, innate and inflammatory response, antibody response, and cellular immunology. These developed SOPs represent a national priority in standardization of all SOPs used for NHP research for COVID-19 vaccine and therapeutic testing and evaluation. This document is considered a living document. As procedures are optimized and additional research methodologies are developed, the SOPs will be adapted and shared amongst the NPRCs. This is one of the tasks of the NHP Coordinating Center managed at TNPRC.

Much of the tooling and infrastructure from SD2 complements Tulane’s current capabilities, sitting downstream of data production and supporting data and metadata storage, ETL, pre-processing, normalization and analysis. This infrastructure is automated, so speed is increased as described above, and reproducibility should be measurable as the infrastructure is deployed in the context of a multi-NPRC effort.

The immediate goal is to examine the speed and reproducibility of the automated Flow Cytometry tools (TASBE) described above using an NPRC relevant antibody panel deployed across multiple instruments and multiple experimental sites. Here one can exercise a portion of the SD2 STS relating to this specific data type, and assess increases in speed, accuracy, and reproducibility of the SD2 systems as compared to the current hand-gating strategy. If the results support the expected value of the combined Tulane/SD2 system, attention will turn to expanding the STS to include all relevant data types and analyses for the NPRC effort. A task and timeline for immediate goals are outlined below.

Long term vision. If the transition of the SD2 ecosystem is successful, the NPRCs will have a new infrastructure that will enable an agile deployment of resources to the current pandemic as well as future biological threats. Moreover, the NPRC/SD2 ecosystem can then serve as a model

for the development of similar infrastructure across other experimental groups (e.g. clinical, small animal) so the U.S. can have a coordinated and rapid response to future biological threats. A successful pilot program would leave the SD2/Tulane group poised to submit applications for a full program to fund the entire effort.

Although the team was successful with initial efforts, the transition eventually failed due to the inability to secure a commitment from the NIH/ORIP to support efforts of the Tulane group to collaborate with SD2 performers in order to establish an NPRC-centric SD2 architecture. The lack of commitment did not appear to reflect a lack of enthusiasm for the effort, but rather bureaucratic constraints. This failure highlights the need for better interagency collaboration.

3.7.2 Xilis, Inc.

Transition Partner and Challenges: Xilis, Inc. (Xilis.com) develops precision therapy technologies and drugs to treat cancer patients. This discovery and design activity is complex and executed on a large scale. Xilis strives for robust communication between its experimentalists and analysts. That is difficult to achieve. Such laboratories are often challenged, for example, to reliably communicate experimental intent from experimentalists to analysts. That information provides the context for efficient, informative analysis. Xilis is also exploring ways to better automate its workflows.

SD2 Performers and Solutions: A former member of the SD2 team from Duke recently joined Xilis and has begun a campaign to transition specific SD2 techniques and technologies to the organization. He is working to integrate several SD2 technologies into operations at Xilis. DataConverge will regularize data frames and enforce version control. It will serve as a flexible bridge carrying complicated and dynamic data between many sources and multiple human and machine data consumers that require structured, consistent data with provenance. The Pre-Computed Data Table is a promising component for Xilis because it standardizes the computation of measurements and their formats for analysis. For example, in SD2, the Pre-Computed Data Table was used to compute a standard growth statistic over time series data. This eliminated variance in the specific growth statistic, algorithm, computational implementation, and resulting measurement values. Finally, a SD2 tool for Quality Control may have value to Xilis because it identifies variables with high or low variance that may explain failures in experiments or in data processing.

Goals: The SD2 team member is applying the communication practices developed in SD2 in his position as a leader and liaison between the Xilis experimenters and analysts, improving the understanding of experimental intent, experimental conditions, and other factors across the cultural divide.

3.7.3 Duke External Quality Assurance Program Oversight Laboratory

Transition Partner and Challenges: The NIH-funded, NIAID and Division of Acquired Immunodeficiency Syndrome (AIDS) (DAIDS), External Quality Assurance Program Oversight Laboratory designs, implements, and executes Quality Control in laboratories that conduct Human Immunodeficiency Virus (HIV)/AIDS research and vaccine trials, internationally. These HIV/AIDS laboratories routinely apply flow cytometry, which measures the prevalence of fluo-

rescent colors in live cells. It is a considerable challenge to compare flow cytometry measurements between laboratories) given the variance introduced by use of different instruments or different calibration of the same instrument, and by differences in the skills and practices of human analysts who manually apply gates (thresholds) to assess measurements. EQAPOL and its client laboratories are also hampered by the time cost of flow cytometry analyses, which is driven in large part by scale of the task: 15-25 marginally distinct fluorescent colors must be distinguished in plots of approximately 20,000 cells.

SD2 Performers and Solutions: Members of the SD2 performer team at Duke, BBN Technologies, and Netrias are working to demonstrate how specific SD2 tools can help EQAPOL solve these problems. The SD2 extensions to TASBE automate the manual gating procedure. Use of standard control beads, introduced to SD2, normalize measurements from instruments that differ in design, because they are different brands and models. The SD2 data ingest workflow will automate the onerous task of cleaning data and standardizing its presentation in data frames. The SD2 infrastructure developed at TACC is available to host data ingest and analyses. Additionally, Netrias and Duke have developed ML/AI powered flow-cytometry tools (Autogator) that may be extended to the automated gating challenge in addition to capabilities for classifying live and dead cells.

Goals: The SD2 performers and their partners from EQAPOL hope that a successful demonstration to NIH will persuade the Institute to increase its current funding to EQAPOL to enable the team to integrate this SD2 tooling into the QC process.

Opportunities: If successful, the SD2 performers are positioned to develop demonstrations of additional capabilities needed by EQAPOL. There are also opportunities for broad dissemination of SD2 technologies to the private sector, including an integration of TASBE with the industry-standard, FlowJo analytics package would both standardize data and automate gating in analyses. Members of the transition teams have connections with the developers of FlowJo. Additionally, preliminary discussions have been initiated with Thermo Fisher, who produces and sells a popular line of flow cytometers along with software support. There was enough initial interest from Thermo Fisher in the technology to begin a discussion with the Duke Office for Translation and Commercialization.

Current Status: TASBE has been demonstrated to accurately gate multi-color panels used by EQAPOL and substantially reduce gating times over manual gating. Currently, SD2 performers are demonstrating normalization across instruments using multi-color antibody panels. Duke subcontracted to Netrias to support the Autogator work and a proposal for future funding to extend Autogator for automated gating approaches was initiated along with TASBE.

3.7.4 Publications.

Members of the Duke Team have authored multiple publications documenting efforts in SD2. Each of these publications will enable others to understand and utilize advances made by the Duke Team in SD2. Many of these manuscripts are still in preparation, so a comprehensive list is not available at the time this document is being prepared.

4.0 Results and Discussion

4.1 Integrated Tools: Pipelines and Automation

The descriptions of tools in Section 3 elaborate on the tools as a stand-alone capability. However, each of these tools was integrated with additional tooling from other SD2 performers. In the Yeast States challenge problem, tooling was integrated into a nearly fully automated Design, Build, Test, Learn loop as shown in Figure 26. Developing each of these tools to be compatible with other tools in the loop was a major contributor to the success of the program.

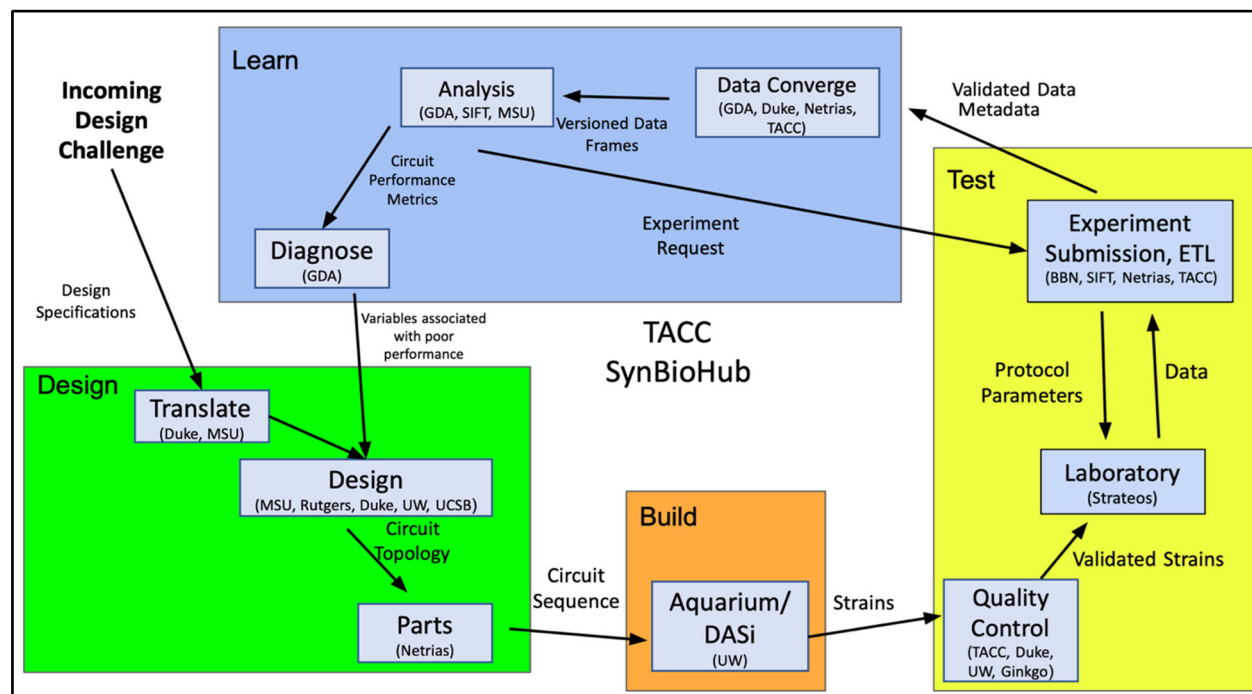


Figure 26. The DBTL Architecture for the Yeast States Challenge Problem.

4.2 Metrics for Success

Efforts were made throughout the program to identify metrics for the success of tools in improving a speed, accuracy, volume etc. Figure 27 summarizes the successes achieved by the program across multiple metrics.

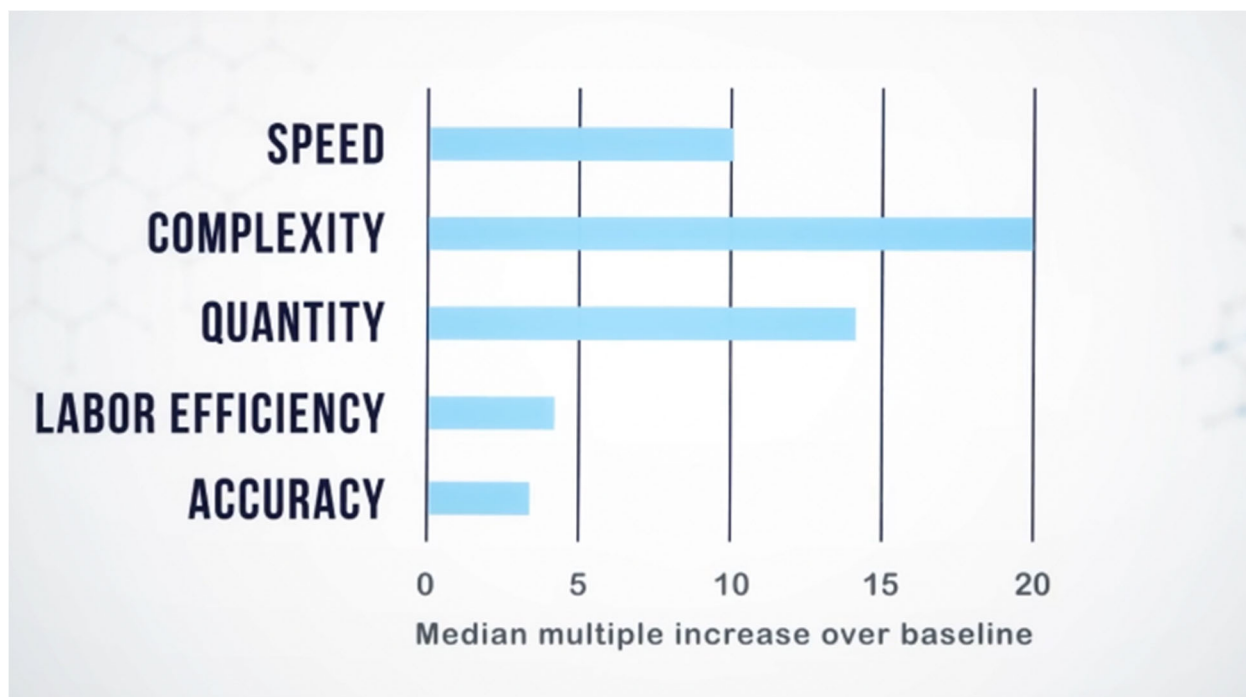


Figure 27. SD2 Program Metrics.

4.3 Purpose built vs. Generalizability: Data Sharing and Transition

One of the successes of this effort that was difficult to quantitate was the generalizability of the tools, pipelines, and architecture constructed. Tools were built to solve problems specific for the particular SD2 challenge problems, yet they were often generalizable for other uses.

Although good quantitative metrics for generalizability were not able to be developed, experiences with the data sharing exercise, where tools were deployed on problems from outside of the program, demonstrated that the tools were easily extensible. The Duke Team worked closely with a group at Caltech to develop models for using synthetic genetic circuits to adjust the proportion of different bacteria in a population. The effort was aimed at improving wound healing by delivering compounds to the wound site in the proper quantities that were produced by friendly bacteria.

The extensibility of the tooling was also learned about while working with transition partners as described in section 3.7.

5.0 Conclusions

5.1 Lessons of COVID-19 Pandemic: SD2 Infrastructure for National Bio-preparedness

The COVID-19 pandemic began during the SD2 program and highlighted the need for methods that accelerate scientific discovery. The current approaches for research that rely on individual investigator-initiated programs are slow in that findings are only shared after the entire experimental cycle is completed and the work is published or discussed in a meeting, as shown in Figure 28. Moreover, increasing funding and adding additional investigators does not increase speed, as each new investigator must traverse the entire experimental/analysis path before sharing findings.

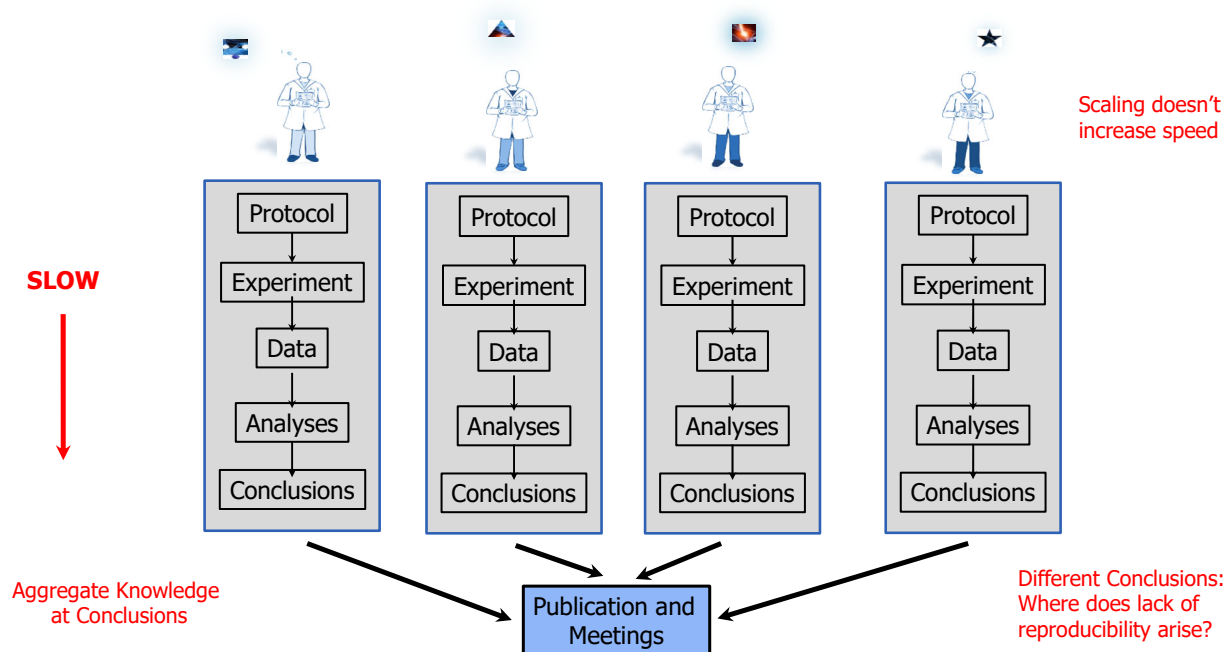


Figure 28. The Investigator-Initiated Model for Discovery.

As an example, members of the Duke Team undertook a modeling study using Duke's COVID-19 data in May of 2021. Contrary to CDC recommendations, the study found that even in a highly vaccinated population, surveillance testing would be an important part of a COVID-19 mitigation program when new variants appeared in the fall. That study was completed in May 2021, but was not published until October of 2021, well after universities had made their plans (Motta et al. 2021).

SD2 results provide an alternative for this investigator-initiated model that accelerates discovery by virtue of sharing knowledge at the level of data, as shown in Figure 29. This architecture provides scalability, as adding researchers increases the accumulation of data available to analysts and the results are rapidly shared within the community.

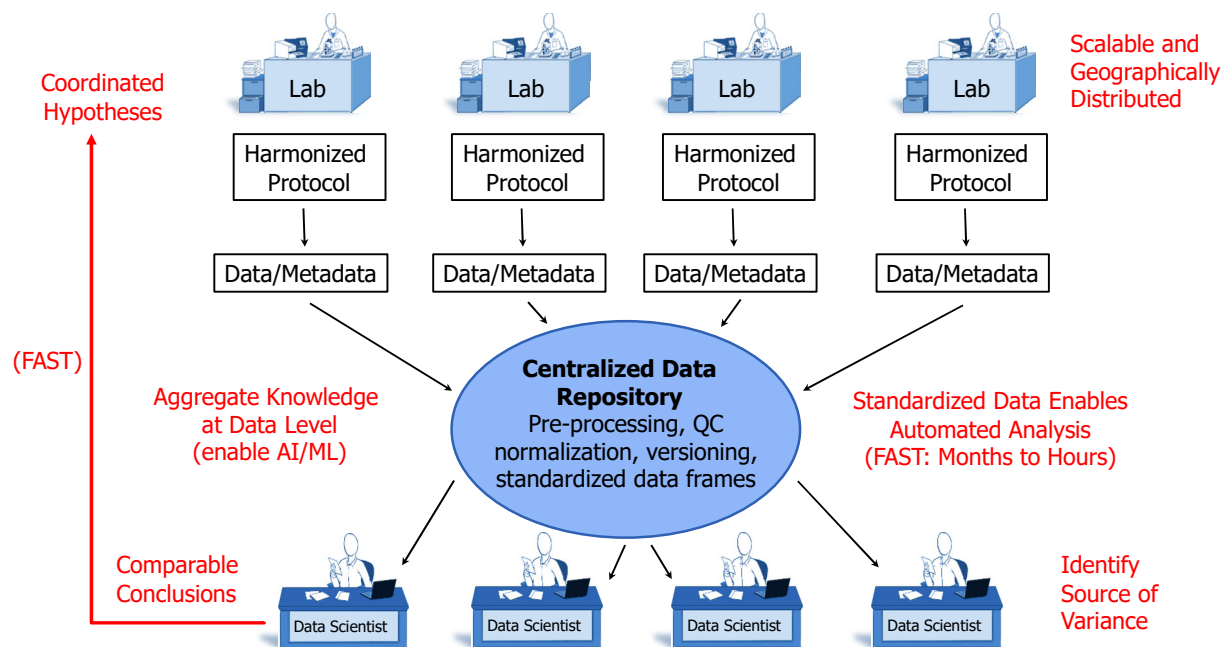


Figure 29. The SD2 Model for Discovery.

The achievements of the SD2 program could be extended to a national infrastructure for bio-preparedness and could be shared jointly by multiple entities including CDC, Food and Drug Administration (FDA), NIH or any group involved in public health responses to pandemics or other biological threats. There are many challenges to scaling and extending such an infrastructure, but the success of the SD2 program suggests this is a viable approach for preparing for the next group of biological threats to public health.

6.0 References

Rocklin GJ, Chidyausiku TM, Goreshnik I, Ford A, Houliston S, Lemak A, Carter L, Ravichandran R, Mulligan 944 VK, Chevalier A, Arrowsmith CH, Baker D. “Global analysis of protein folding using massively parallel design, synthesis, and testing.” *Science*. 2017; 357(6347):168–175. <https://science.sciencemag.org/content/357/6347/168>, doi: 10.1126/science.aan0693.

Alexander Zaitzeff, Nicholas Leiby, Francis C Motta, Steven B Haase, Jedediah M Singer, “Improved datasets and evaluation methods for the automatic prediction of DNA-binding proteins,” *Bioinformatics*, 2021, btab603, <https://doi.org/10.1093/bioinformatics/btab603>.

F. Motta *et al.*, "Hyperparameter Optimization of Topological Features for Machine Learning Applications," 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019, pp. 1107-1114, doi: 10.1109/ICMLA.2019.00185.

Chad Giusti, Eva Pastalkova, Carina Curto, Vladimir Itskov, “Clique topology reveals intrinsic geometric structure in neural correlations,” *Proceedings of the National Academy of Sciences* Nov 2015, 112 (44) 13455-13460; DOI: 10.1073/pnas.1506407112.

Motta, F.C., McGoff, K.A., Deckard, A., Wolfe, C.R., Bonsignori, M., Moody, M.A., Cavanaugh, K., Denny, T.N., Harer, J., and Haase, S.B. (2021). Assessment of Simulated Surveillance Testing and Quarantine in a SARS-CoV-2–Vaccinated Population of Students on a University Campus. *JAMA Health Forum* 2, e213035-e213035.

List of Symbols, Abbreviations, and Acronyms

ACTIV	Accelerating COVID-19 Therapeutic Interventions and Vaccines
AI	Artificial Intelligence
AIDS	Acquired Immunodeficiency Syndrome
API	Application Programming Interface
ASKE	Automating Scientific Knowledge Extraction
BU	Boston University
Caltech	California Institute of Technology
CDC	Centers for Disease Control and Prevention
CDER	Cover-tree Differencing via Entropy Reduction
CDM	Combinatorial Design Model
CFU	Colony Forming Units
CGAL	Computational Geometry Algorithms Library
COVID-19	Coronavirus Disease 2019
COVTEN	Coronavirus Vaccine and Therapeutic Evaluation Network
CP	Challenge Problem
CP-YS	CP Yeast States
CRISPR	Clustered Regularly Interspaced Short Palindromic Repeats
CTP	Cytidine 5'-triphosphate
DAIDS	Division of AIDS
DAL	Data Access Layer
DARPA	Defense Advanced Research Projects Agency
DBTL	Design, Build, Test, Learn
DC	Data Convergence
dChip	DNA-Chip Analyzer
DESeq	Differential Expression Sequencing
DNASEq	Deoxyribonucleic Acid Sequencing
DOD	Department of Defense
DSGRN	Dynamic Signatures Generated by Regulatory Networks
EQAPOL	External Quality Assurance Program Oversight Laboratory
ETL	Extract, Transform, Load
FDA	Food and Drug Administration

FIPS	Federal Information Processing Standard
FlowJo	Not Acronym
GDA	Geometric Data Analytics
GFP	Green Fluorescent Protein
gRNA	Guided Ribonucleic Acid
HIV	Human Immunodeficiency Virus
HPC	High Performance Computers
HHS	U.S. Department of Health and Human Services
IPTG	Isopropyl β-D-1-Thiogalactopyranoside
IM3UNE	Integrative Model of Mobile Media Use and Need Experiences
JSON	Java Script Object Notation
LEM	Local Edge Machine
MBF	Monotone Boolean function
MDS	Multidimensional Scaling
MGI	Mouse Genome Informatics
MIT	Massachusetts Institute of Technology
ML	Machine Learning
mRNA	Messenger Ribonucleic Acid
NESD	Neural Engineering System Design
NHP	Non-Human Primate
NIAID	National Institute of Allergy and Infectious Diseases
NIH	National Institutes of Health
NPRC	National Primate Research Center
NSF	National Science Foundation
ODE	Ordinary Differential Equation
ORIP	Office of Research Infrastructure Programs
OUP	Oxford University Press
PCR	Polymerase Chain Reaction
PDB	Protein Databank
PDT	Precomputed Data Table
PH	Persistent Homology
PHAT	Persistent Homology Algorithm Toolbox
PI	Principal Investigator

PLOS	Public Library of Science
PM	Performance Metrics
QC	Quality Control
RMA	Robust Multiarray Analysis
RMSE	Root Mean Square Error
RNA	Ribonucleic Acid
RNA-seq	Ribonucleic Acid Sequencing
Rutgers	Rutgers, The State University of New Jersey
SBOL	Synthetic Biology Open Language
SD2	Synergistic Discovery and Design
SGD	Saccharomyces Genome Database
SIFT	Smart Information Flow Technologies
SOP	Standard Operating Procedure
SQL	Structured Query Language
STS	Socio-Technical System
SynBioHub	Synthetic Biology Hub (SynBioHub.org)
TA	Technical Area
TACC	Texas Advanced Computing Center
TASBE	Not Acronym
TDA	Topological Data Analysis
TNPRC	Tulane National Primate Research Center
UCSB	University of California, Santa Barbara
UW	University of Washington
VDR	Versioned Dataframe Repository
YEPD	Yeast Extract Peptone Dextrose