Machine-guided design and evolution of biological systems: from the protein to the genome scale

Machine-guided design and evolution of biological systems: from the protein to the genome scale

A DISSERTATION PRESENTED
BY
GLEB KUZNETSOV
TO
THE DEPARTMENT OF BIOPHYSICS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
BIOPHYSICS

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
MAY 2018

Thesis advisor: Professor George Church Gleb Kuznetsov

Machine-guided design and evolution of biological systems: from the protein to the genome scale

Abstract

Evolution has shown that mutation and selection over billions of years can produce complex molecules and organisms that thrive in a diverse range of environments. As biological engineers, we would like to systematize the navigation of genetic landscapes to find solutions to urgent health and technological needs. In this thesis, I approach the engineering of biological systems from the perspective of design. I illustrate the view of design as an iterative framework of satisfying engineering constraints while discovering and testing degrees of freedom in biological systems. Beginning at the genome scale, I describe a software framework for encoding design rules for recoding genomes and its application to the design of an *E. coli* strain using only 57 of 64 codons. The genome is being assembled and tested in 50-kilobase segments and we have verified that over 50% of the recoded genome design can functionally complement. Where design rules break down, we leverage DNA synthesis and genome editing to generate targeted diversity and update the design rules. Next, I describe how a model-guided approach that prioritizes mutations to test can augment adaptive laboratory evolution. A 63-codon genomically recoded organism that we previously engineered suffered from impaired fitness and we used our approach to discover a minimal set of high-impact edits that recover 59% of the fitness defect. Finally, I discuss ongoing work to augment design and evolution of proteins by training machine learning models that learn from and guide high-throughput mapping of fitness landscapes. I describe lessons learned in a proof-of-concept study mapping the fitness landscape of the green fluorescent protein and implications for engineering of other proteins. The unifying contribution of this dissertation is a demonstration at multiple scales of how to systematically integrate DNA synthesis, sequencing, high-throughput assays, and computational methods to interrogate biological systems and learn design principles that expand our engineering capabilities.

Contents

To my parents,
for enabling this epic journey through life.

Acknowledgments

Toward the end of my undergraduate days as an EECS major, I began joking that if I had $10 million and could do anything, I would commit to the monastic pursuit of biology. The first profound realization I had was that life is too short to make a medium amount of money. An even more profound realization was that studying biology is anything but a lonely pursuit.

The best teams I have worked with in my life have been in the lab of my advisor George Church. I am grateful to George for taking a chance on a confused software engineer who wanted to give the biology thing a try. It can't be said enough that George Church is not only one of the smartest and most visionary individuals, but he is also one of the kindest.

The people that George has attracted to his lab set the tone for the nearly six years I have spent here. I have been fortunate to be mentored by, collaborate, and in turn mentor people who have become some of my best friends.

I'm indebted to Adam Marblestone for introducing me to folks in the Church Lab. When I asked for him to keep an eye out for anyone who might want a coding buddy, he responded with a dozen glowing introduction emails. Less than a month later I found myself in George's lab.

Marc Lajoie and Dan Goodman took me in when I joined the lab. I can't thank them enough for their kindness and patience as they welcomed me to the lab and provided a crash course in science. Both Marc and Dan were skilled in translating the biology world so a computer scientist could understand. Marc taught me how to pipette and had me editing genomes that same day. Perhaps more importantly, he recruited me to all of his soccer teams. Dan and I spent hundreds of hours coding together, as he taught me bioinformatics tricks and I showed him the power of Git and unit tests. I would end up working closely with Marc and Dan over the years to come and hope to do so again in the future.

My collaboration with Gabe Filsinger started with him asking me to teach him how to do MAGE and we soon found we made a great team. His creativity, curiosity, and work ethic is infectious and when I had nearly set aside the work on model-guided genome optimization, the energy he brought to the project gave it new life and together we saw it through to the end.

I have been fortunate to mentor several brilliant undergraduates: Woody Ahern, Changping Chen, and Kevin Chen. All of them were very talented and made important contributions to our work.

The trifecta we have formed with my current team of Surge Biswas and Pierce Ogden has made the past two years one of continued growth and excitement. Surge's ability to jump between thinking like an academic and entrepreneur, his mathematical wizardry, and his courage to try any experimental technique has been inspiring. Likewise, it's been a most excellent experience sending it with Pierce every day in the lab, whose creativity and experimental design and debugging skills know no limits. His focus on the big picture and endless stream of ideas have saved me from many a local minima.

And of course there have been many others in the lab with whom I've worked, learned from, and shared great discussions over the years, including Anik Debnath, Nikolai Eroshenko, Justin Feng, Alex Garruss, Chris Gregg, Eric Kelsic, Matthieu Landon, Dan Mandell, Michael Napolitano, Alex Ng, Nili Ostrov, Lexi Rovner, Max Schubert, Seth Shipman, Sam Sinai, Dima Ter-Ovanesyan, Dave Thompson, and many, many others. It's a big lab!

My friends outside of the lab (and their couches) have been endlessly supportive. Over the years, I've shared adventures, ideas, and dreams with Tom Alcorn, Ashvin Bashyam, JaDee Bodell, Lawrence Chan, Andy Goessling, Evan Guze, Jason Hoch, Allen Lin, Chris Mather, Harry McNamara, Zviad Metreveli, Will Nathan, Cinjon Resnick, Alex Rothberg, Ian Rousseau, Luke Schiefelbein, Dustin Smith, Antony Speranza, Matt Talpe, Ted Tomlinson, Kevin Tsai, Josh Wang, Bryant Watson, and Gordon Wintrob.

I could not have asked for a better graduate program than Biophysics, made possible by the tireless support from Michelle Eva and Jim Hogle. Financial support for my graduate work was provided by the NIH biophysics training grant and a Department of Defense NDSEG fellowship.

The Church lab could not function without the efforts of John Aach, Nick Conway, and Emma Taddeo. John is an extremely clear thinker who has always provided immediate, thoughtful feedback to any idea or piece of writing. Nick makes it a priority to go out of his way to make sure everyone has what they need. And somehow he also finds time for us to hack on code together. Emma has been immensely helpful and positive for the smallest to the most annoying requests.

Finally, I'd like to thank the members of the qualifying and dissertation committees for very helpful advice in navigating the landscape of the dissertation process and especially developing the later work on machine-guided protein design: Ryan Adams, Leonid Mirny, Michael Baym, Mike Springer, Eugene Shakhnovich, and David Liu.

The final and biggest thank you is reserved for my family. My mom's passing from cancer about a year before I joined the Church lab affected my family deeply. Together with my dad, my brother and my uncle we have somehow found a way to support each other. My mom had the strongest spirit of anyone I've known and her joy for life and strength through the end remains an inspiration. My dad has shared much wisdom and been supportive since as far back as I can remember, despite his continued confusion over why I would leave a secure job at Google to become a graduate student. I am very proud of my brother who faced the challenge of my mom's passing while still a teenager and who took her standards to heart as he doubled his focus on his studies. And my uncle, my mom's twin brother, has been there throughout my life, teaching me to question things objectively and always showing great enthusiasm for whatever I was up to at the time. Finally a special thank you to my cousin Ellen Gelfand and her family for their love and support over the years and also letting me crash on her couch when I moved back to Boston to give biology a try.

Life has been a fantastic adventure so far and I look forward to things to come.

1 Introduction

Humans have been biological engineers for thousands of years. Long before Darwin, we began leveraging evolution to domesticate plants and animals. Over the past century, the advent of molecular biology enabled the development of biologic drugs and industrial enzymes. Today, with computational design core to practically all fields of engineering and the ability to read and write DNA in high-throughput, it might seem that we should have unlocked the full potential of biotechnology by now. And yet, the ability to rationally engineer biological systems remains elusive and we continue to depend heavily on screening and laboratory evolution. Biology arises from the the stochastic interactions among an immense number of atomic and molecular interactions, making it uniquely difficult to accurately simulate *in silico*. This thesis takes the perspective of embracing the empirical nature of biological engineering to develop a design discipline appropriate to the complexity of biological systems.

In chapter 2 I explore how design rules for recoding genomes can be encoded in software and empirically tested through the design, synthesis, and assembly of an entire genome. The nearly-universal genetic code is one of the fundamental examples of modularity in biology that defines how DNA encodes the information to create proteins. The apparent redundancy of the genetic code obscures the fact that DNA further encodes regulatory information that affect transcription and translation and a design-forward approach can help guide testing our understanding of the rules. I created a software tool, the Genome Engineering Toolkit (GETK), to enable the design of a synthetic *E. coli* using only 57 of 64 codons. In addition to maintaining amino acid sequences, GETK encodes rules for preserving RBS-binding sites and mRNA secondary structure based on biophysical prediction tools and empirical data derived from related work in our lab[1]. Lessons learned from design failures were used to guide targeted experiments to debug and update the design rules.

Sometimes there is tension between a design objective and physiological balances in an organism. Implementing design constraints in engineered organisms can come at the cost of reduced fitness. In chapter 3, I describe how we used multiplex genome editing, whole-genome sequencing, and predictive modeling to discover a minimal set of detrimental mutations impairing the fitness of the first Genomically Recoded Organism. Notably, this work illustrates that a model-guided targeted editing approach can be more effective than using adaptive lab evolution alone for identifying causal mutations while minimizing additional collateral mutations. This perturb, learn, and repeat pattern of engineering is another pillar of design of biological systems.

Interpreting whole-genome sequencing data for dozens or hundreds of genomes was a bottleneck in the above project and others in our lab. Processing this data often requires assembling an ad hoc computational pipeline and is prohibitive to non-bioinformaticians. In chapter 4 I describe Millstone, a software platform that automates many steps of analysis and comparison of hundreds of engineered or evolved genomes and provides a graphical user interface to make this capability generally accessible.

In chapter 5, the final part of this thesis, we apply model-guided methods to the task of engineering proteins. Here, we can generate upwards of 10^5 genotype-phenotype pairs, allowing us to learn and navigate a protein's fitness landscape in pursuit of design objectives. Through a proof-of-concept study navigating the fitness landscape of the green fluorescent protein, we gain insights into how machine learning can improve on evolution or screening alone. We see that we can rapidly learn the constraints that determine the protein's function and experiment with using a machine learning model to generate diverse variants while maintaining function and optimizing on brightness. In turn, the protein domain provides a substrate for thinking about generalization in machine learning, an important concept as the field has been swept by large neural network based models whose remarkable efficacy is still under intensive investigation.

Despite the overwhelming complexity of biology, evolution has produced countless examples of modularity and stable solutions to challenges across a variety of environments. This lends support to my conviction that a more principled approach to biological design is possible. Models guide the design of interrogative libraries and make the most of limited screening capacity. Through a series of studies described in this thesis, I have explored how design and modeling coupled with experimental explorations allow us to engineer the stuff of life.

The future of empirically-grounded biological design will benefit heavily from increases in the length of high-throughput DNA sequencing and synthesis. As we obtain more data, we can direct our models toward better representing the complex physical principles that predict the function of biological systems. I believe we are still a bit early in our march towards a day when we can design biological systems entirely *in silico* from first principles. The path to such a future, and immediate opportunity, lies in focusing our efforts on computationally augmenting the highly-parallel capabilities of molecular biology and lab evolution.

2

Rule-based genome design and testing toward a 57-codon genome

The contents of this chapter were adapted from:

Ostrov, N*, Landon, M*, Guell, M*, **Kuznetsov, G***, Teramoto, J, Cervantes, N, Zhou, M, Singh, K, Napolitano, MG, Moosburner, M, Shrock, E, Pruitt, BW, Conway, N, Goodman, DB, Gardner, CL, Tyree, G, Gonzales, A, Wanner, BL, Norville, JE, Lajoie, MJ, and Church, GM *Design, synthesis, and testing toward a 57-codon genome.* (2016) Science, 353(6301), 819–822.

Acknowledgements:

This project was a heroic cross-disciplinary effort jointly led with NO, ML, MG, and MJL. My contributions were entirely computational and I include discussion of assembly and testing for completeness. I led design and implementation of the Genome Engineering Toolkit (GETK), with contributions from MJL and DBG. Together with MJL, we used GETK to design the *rE.Coli-57* genome and order the first iteration of all DNA. NO, ML, MG, and MJL designed the assembly pipeline and led experimental validation. I worked closely with ML, NO, and MJL to iterate on genome design as the genome was assembled and debugged. I would like to further acknowledge NO and ML for creating many of the manuscript figures from which this chapter is adapted.

In this chapter, I describe a method for rule-based genome design and its application to the creation of a synthetic *E. coli* genome utilizing only 57 of 64 codons. This organism promises unprecedented industrial capabilities through genetic isolation and expansion of the available biochemical repertoire for protein synthesis. The success of genome-scale engineering projects can be de-risked through computational design. In particular, I describe my contribution of creating the Genome Engineering Toolkit (GETK), a software package that facilitates genome-wide codon reassignment that respects known biophysical constraints. At first glance, the problem of recoded organism design would appear to require a straight-forward interpretation of the genetic code, leveraging its inherent degeneracy to replace forbidden codons with any synonymous alternative. However, synonymous is not the same, and changes in amino acid space may disrupt regulatory information encoded in primary DNA sequence[2]. Thus we created GETK to encode known biophysical rules and generate genome designs that satisfy the recoding objective while anticipating and inviable designs. We have experimentally validated 63% of recoded genes by individually testing 55 segments of 50 kilobases each. We observed that 91% of tested essential genes retained functionality with limited fitness effect. We demonstrate identification and correction of lethal design exceptions, only 13 of which were found in 2229 genes.

2.1 Background

Genetically modified organisms (GMOs) play increasingly critical roles in healthcare, agriculture, and production of a number of human consumables. Modifications in commercially implemented GMOs are typically limited to heterologous gene expression and evolution under optimizing selection. Synthetic genomes that differ radically from any known organism may expand potential applications.

There has been considerable interest in creating minimal[3] and recoded[4] genomes, but the principles of genome design are not yet understood well enough to design them from scratch. While *in vivo* genome engineering strategies may reduce the risk of creating nonfunctional genomes[4,5], rational design will be indispensable for restricting the search space to create viable genomes with a desired function.

Whole-genome synonymous codon replacement provides a mechanism to construct organisms exhibiting genetic isolation and enhanced biological functions. Once a codon is replaced genome-wide and its cognate tRNA is eliminated, the genomically recoded organism (GRO) can no longer translate the missing codon[4]. Genetic isolation is achieved since DNA acquired from viruses, plasmids and other cells would be improperly translated, rendering GROs insensitive to infection and horizontal gene transfer. In addition, proteins with novel chemical properties can be explored by reassigning replaced codons to incorporate nonstandard amino acids (nsAAs) which function as chemical handles for bioorthogonal reactivity and enable biocontainment of GROs[6,7,8].

Building on previous work that demonstrated single stop codon replacement[4], we set out to construct an organism with seven codons replaced genome-wide, yielding a virus-resistant, biocontained bacterium for industrial applications. The number of required changes far surpasses the capabilities of *in vivo* editing and requires DNA synthesis of large stretches of redesigned genome.

Multiple groups have tackled the enormous task of encoding detailed genetic and biophysical information into software as computational design rules. In particular, RNA folding software[9,10] and ribosome strength calculators[11] have enabled design of synthetic DNA parts. However, in order to redesign an entire genome, it is necessary to 1) simultaneously take into account effects of multiple known biological principles underlying the structure and control elements of entire chromosomal sequences, 2) provide as output synthesizable DNA fragments satisfying the assembly strategy ready to be assembled, and 3) incorporate data learned from empirical testing of designs to update rules and iterate on designs.

2.2 Design of a 57-codon *E. coli* Genome

The Genome Engineering Toolkit (GETK) was created with the motivating goal of generating an *E. coli* genome design in which all 62,214 instances of seven different codons (5.4% of all E. coli codons) were synonymously replaced (Figure 2.1). While several synthetic genomes have been previously reported[12,13,14,15,16,3,17,18], a functionally altered genome of this scale has not yet been explored (Figure 2.1 C).

Previous work suggests that codon usage alterations can affect gene expression and cellular fitness in multiple ways[19,20,2,21,22,23,24]. However, parsing the individual impact of each codon remains difficult. Moreover, the number of modifications required to replace all instances of seven codons throughout the genome is far beyond the capabilities of current single-codon editing strategies[4,25]. Although it is possible to simultaneously edit multiple alleles using MAGE[26] or Cas9[27], these strategies would require extensive screening using numerous oligos and likely introduce off-target mutations[5]. With plummeting costs of DNA synthesis, financial barriers for synthesizing entire genomes are greatly reduced, allowing for an almost unlimited number of modifications independent of bi-

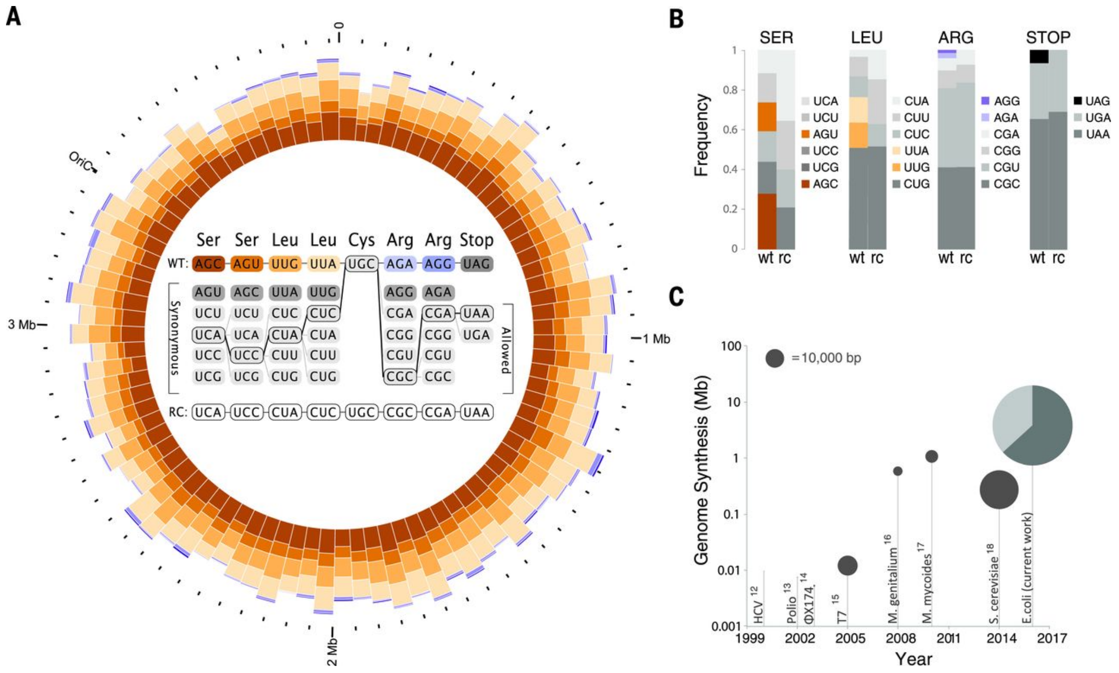


Figure 2.1: A 57-codon E. coli genome. (A) The recoded genome was divided into 87 segments of ~50-kb. Codons AGA, AGG, AGC, AGU, UUA, UUG, UAG were computationally replaced by synonymous alternatives (center). Other codons (e.g. UGC) remain unchanged. Color-coded histograms represent the abundance of the seven forbidden codons in each segment. (B) Codon frequencies in non-recoded (wt; E. coli MDS42) versus recoded (rc; *rE.coli-57*) genome. Forbidden codons are colored. (C) The scale of DNA editing in genomes constructed by de novo synthesis. Plot area represents DNA editing as the number of modified bp compared to the parent genome. For the current work, dark gray represents percent of genome validated in vivo at time of publication (63%). Wt, wild-type.

ological template. Here, we developed computational and experimental tools to rapidly design and prototype synthetic organisms.

In choosing codons for replacement, UAG (Stop) was selected because it was previously replaced genome-wide[4]. AGG and AGA (Arg) are among the rarest codons in the genome, minimizing the number of changes required. Other codons (AGC (Ser), AGU (Ser), UUG (Leu), UUA (Leu)) were chosen such that their anticodon is not recognized as a tRNA identity element by endogenous aminoacyl-tRNA synthetases. We also considered mischarging of newly introduced aaRS/tRNA by endogenous aminoacyl-tRNA synthetases upon codon reassignment. Lastly, we confirmed all chosen codons are recognized by a different tRNA than their synonymous codons, so that both codons and cognate tRNA could be eliminated.

In order to minimize synthesis costs and improve genome stability, we based our 57-codon genome on the reduced-genome *E. coli* MDS42[28]. The computational tool GETK automated synonymous replacement of all forbidden codons occurrences in protein-coding genes while satisfying biological and technical constraints (Figure 2.2, Figure 2.3, Table A.1). Primarily, we preserved amino acid sequences of all coding genes, and adjusted DNA sequences to meet synthesis requirements (e.g., removing restriction sites, normalizing regions of extreme GC content and reducing repetitive sequences). Alternative codons were selected to minimize disruption of biological motifs such as ribosome binding sites (RBS) and mRNA secondary structure[4,5], and the relative codon usage was conserved in order to meet translational demand[29,30]. If no acceptable synonymous codon was found, the constraints were relaxed until an appropriate alternative was identified.

Forbidden codons were uniformly distributed throughout the genome, averaging ~17 codon changes per gene. Essential genes[31], which provide a stringent test for successful codon replacement, contained ~6.3% of all forbidden codons (3,903 of 62,214). Altogether, the recoded genome required a total of 148,955 changes to remove all instances of forbidden codons and adjust the primary DNA sequence for synthesis and assembly.

We parsed the recoded genome into 1256 synthesis-compatible overlapping fragments of 2-4-kb. These were used to construct 87 segments of ~50-kb each, which are convenient for yeast assembly and shuttling. Notably, intermediate 50-kb segments are also easier to troubleshoot than a full-size recoded genome or 3548 individual genes. Importantly, we estimated that each segment would contain on average only ~1 potentially lethal recoding exception[4,5].

Computational Genome Design Procedure and Design Rules

GETK is a general software tool for genome-wide codon replacement in a prokaryotic genome. While existing software enables codon optimization of individual genes and operons as well as introduction of manual genome changes (32, 39), no tool currently exists that can perform genome-wide recoding in an automated fashion.

Several challenges arise when choosing synonymous codons to replace forbidden codons. First, to ensure biological viability, it is important to maintain the fundamental features of the parent genome, such as GC content and regulatory elements encoded by the primary nucleotide sequence. Additionally, when forbidden codons fall in overlapping gene regions, these overlaps must be carefully split in a manner that avoids introducing non-synonymous mutations or disrupting regulatory features. Finally, the computational design scheme must be compatible with the experimental tools being used for genome construction.

To address these needs, GETK was designed according to a rule-based architecture, where user-specified rules, encoded as python functions, serve as constraints that the software must navigate in finding suitable synonymous codon replacements. The set of rules implemented for the *rE.coli-57* genome design is described in Table A.1. Notably, while the implementation described below was customized for this project, the software was written in a modular manner so that it can be extended to more general applications.

The recoding software requires two major inputs (Figure 2.2): 1) a genome template, in the form of an annotated Genbank file. In the current work, the template was *E. coli* MDS42 genome (GenBank: AP012306.1). 2) A list of codons to be synonymously replaced throughout the genome, termed "forbidden codons". Here we replaced seven codons: AGA, AGG, AGC, AGU, UUG, UUA and UAG. The software then automatically replaces all instances of forbidden codons, choosing synonymous codons that allow the resulting sequence to best adhere to biological and technical rules described below. The output is a Genbank file of the final genome design.

The rules guiding the design of *rE.coli-57* can be divided into two major categories: 1) Preserving biologically relevant motifs and 2) Satisfying synthesis and experimental constraints.

The automated computational design pipeline carries out the following steps (Figure 2.2):

1. Apply forbidden codon replacement in all instances of gene overlaps, considering biological constraints.
2. Apply remaining forbidden codon replacement in each gene independently, considering biological constraints. See discussion of graph search-based codon selection below.
3. Apply technical rules considering synthesis and assembly constraints. Modifications are made to satisfy DNA vendor constraints, such as removal of specific restriction enzyme sites and homopolymer sequences, and balancing of GC content. When multiple alternatives exist (e.g. synonymous codon swaps to remove restriction enzyme site), we chose the alternative that best satisfies design rules (minimal biological motif disruption).
4. Partition the genome into ~50 kb segments, then partition each segment into 2 - 4 kb synthesis units. This final step generates the individual fragments for *de novo* DNA synthesis.

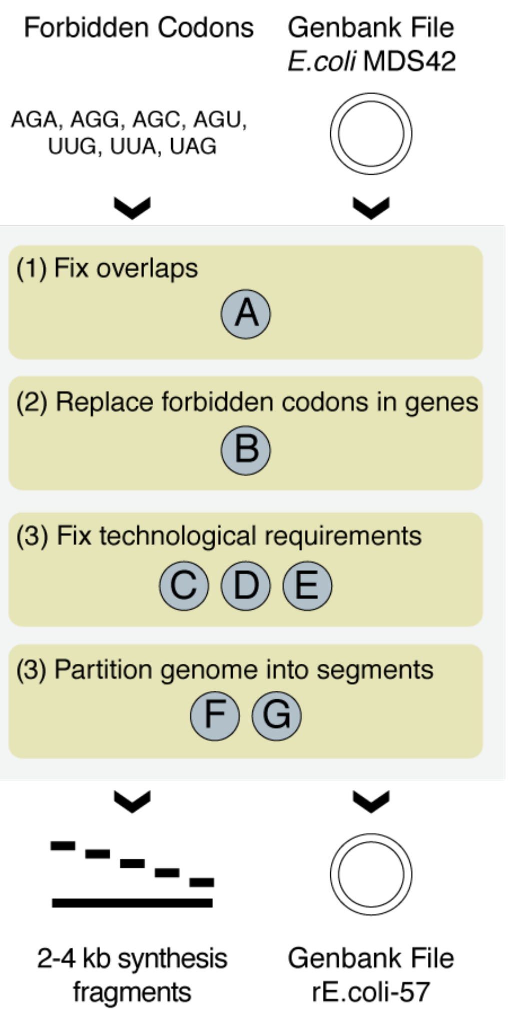
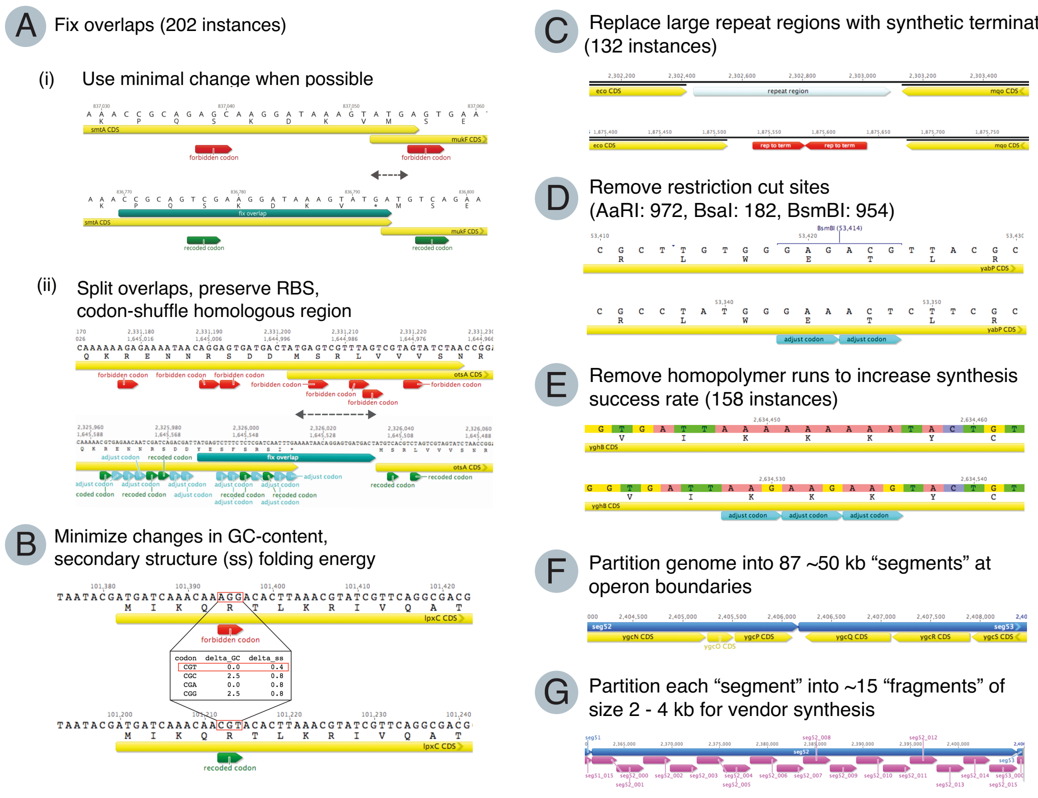

Forbidden Codons
Genbank File
E.coli MDS42
AGA, AGG, AGC, AGU, UUG, UUA, UAG
(1) Fix overlaps
A
(2) Replace forbidden codons in genes
B
(3) Fix technological requirements
C
D
E
(3) Partition genome into segments
F
G
2-4 kb synthesis fragments
Genbank File
rE.coli-57

Figure 2.2: Overview of the computational pipeline for genome recoding. The software accepts as input a genome template (GenBank file) and a list of codons to be replaced. User-defined rules, both biological and technical (A-G), are then applied to generate a new recoded genome (Genbank file). Synthesis-compatible 2 – 4 kb sequences are generated. Rules A-G are illustrated in Figure 2.3 and further explained in Table A.1.

Graph search-based codon replacement algorithm

Each forbidden codon has ~4 synonymous codon alternatives (Figure 2.1A). We optimized for a recoded genome design that is minimally disruptive with respect to rules that quantify deviation



A Fix overlaps (202 instances)
(i) Use minimal change when possible
(ii) Split overlaps, preserve RBS, codon-shuffle homologous region
B Minimize changes in GC-content, secondary structure (ss) folding energy
C Replace large repeat regions with synthetic terminators (132 instances)
D Remove restriction cut sites (AaRI: 972, BsaI: 182, BsmBI: 954)
E Remove homopolymer runs to increase synthesis success rate (158 instances)
F Partition genome into 87 ~50 kb "segments" at operon boundaries
G Partition each "segment" into ~15 "fragments" of size 2 - 4 kb for vendor synthesis
forbidden codon
recoded codon
adjust codon
fix overlap
repeat region
rep to term

Figure 2.3: Examples of applying GETK rules. See Table A.1 for complete list of rules and guidelines.

from the wild-type sequence (e.g. secondary structure, GC content, RBS motif strength). With ~17 forbidden codons per gene, an exhaustive comparison of all possible codon modifications rapidly approaches computational intractability, with 4^{17} possible designs.

Thus, the GETK recoding algorithm seeks a solution that respects each rule up to a threshold, rather than seeking a global minimum. To find a satisfactory solution, the genome-recoding problem is represented as a graph which is traversed using an algorithm based on depth first search. Nodes in the graph represent a unique alternative gene sequence. Sibling nodes in the graph differ in the choice of a single specific codon. Children of a node are all possible changes to the next downstream codon. Each node is assigned a score quantifying deviation from wild-type with respect to each of the scorers. Each score is a quantitative measure of deviation away from wild-type sequence in the respective score profile for a 40-nucleotide window centered around that codon. As long as all scores are below the thresholds for their respective profiles, a node will be expanded and pursued. If all nodes at a level violate the threshold, the algorithm backtracks to an earlier node and chooses a different branch. If the algorithm cannot find a solution for a particular gene, the threshold constraints are loosened and the search is restarted.

This algorithm ensures that the selected changes minimize disruption of biologically important motifs. A variant of the same algorithm is applied at each stage of the pipeline including forbidden codons, replacements and adjustments for DNA synthesis requirements.

2.3 Assembly and testing of *rE.coli-57*

Each ~50-kb recoded segment, carrying on average ~40 genes and ~3 essential genes, was assembled and tested for functionality individually (Figure 2.4). Each segment was assembled in *S. cerevisiae* and electroporated directly into E. coli on a low copy plasmid. Subsequent deletion of the corresponding chromosomal sequence provided a stringent test for recoded genes functionality since errors in essential genes would be lethal.

Thus far, we performed chromosomal deletions for 2229 recoded genes (55 segments), accounting for 63% of the genome and 53% of essential genes. Encouragingly, 99.5% of recoded genes were found to complement wild-type without requiring any optimization. Moreover, the majority of chromosomally deleted strains exhibited limited fitness impairment (<10% doubling-time increase)

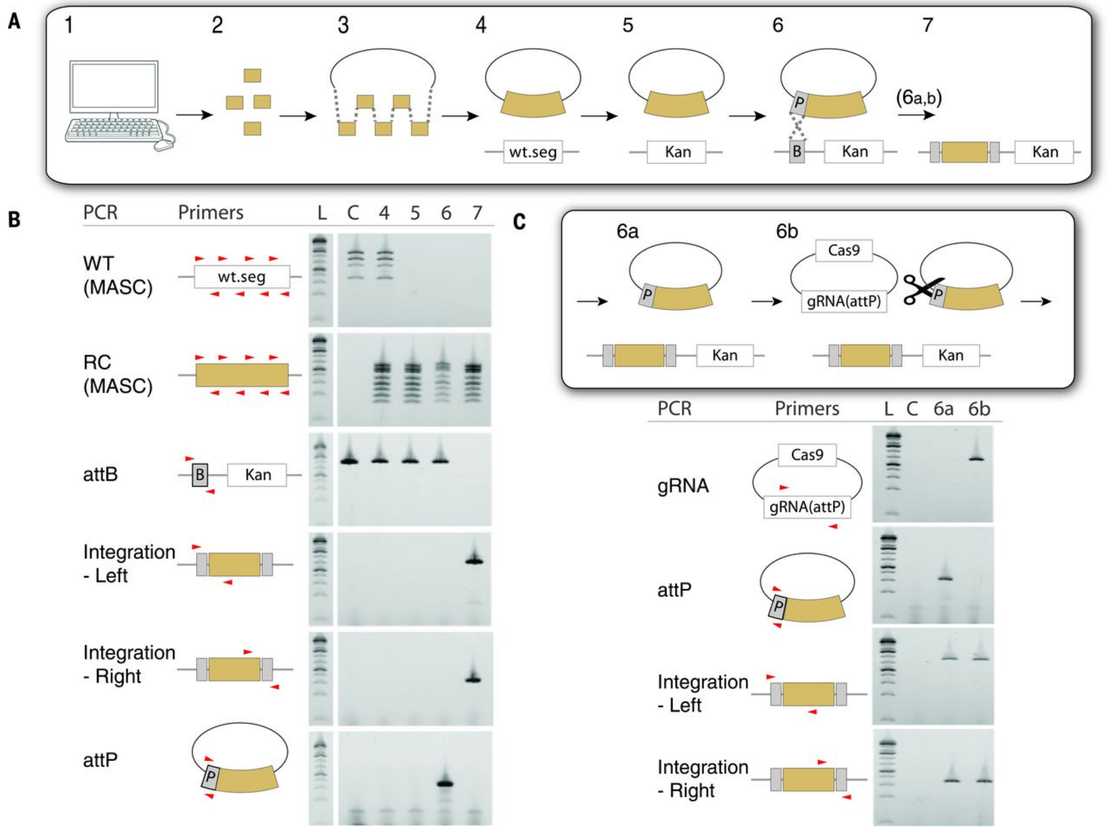


Figure 2.4: Experimental strategy for recoded genome validation. (A) Pipeline schematics: 1) computational design; 2) de novo synthesis of 2- to 4-kb overlapping recoded fragments; 3) assembly of 50-kb segment (orange) in S. cerevisiae on a low copy plasmid; 4) plasmid electroporation in E. coli (wt.seg - non-recoded chromosomal segment); 5) wt.seg is replaced by kanamycin cassette (Kan) such that cell viability depends solely on recoded gene expression; 6) ?-integrase-mediated recombination of attP and attB sequences (P- episomal, B- chromosomal); 6a,b) elimination of residual vectors (see (C)); 7) single-copy integrated recoded segment. attL-attR sites shown in gray. Chromosomal deletions were performed in E. coli TOP10. (B) PCR analysis of steps 4-7 (L- GeneRuler 1-kb plus ladder, C- TOP10 control). Numbers correspond to schematics in (A). PCR primers shown in red. (C) Cas9-mediated vector elimination: residual vector carrying recoded segment is targeted for digestion by Cas9 using attP-specific guide RNA (gRNA).

(Figure 2.5A).

Severe fitness impairment (>1.5-fold increase) was observed in only two strains. The causal genes were mapped by systematically removing wild-type genes, followed by measurement of strain fitness (Figure 2.5B-C). We found fitness impairment in segment 21 was caused by insufficient expression of the recoded fatty acid biosynthesis operon rpmF-accC. Specifically, codon changes in upstream yceD gene were found to disrupt the operon promoter. Fitness was improved when yceD codons were altered via MAGE to preserve the overlapping promoter (Figure 2.5C). In segment 84, analysis suggested three genes caused impairment of fitness, including the recoded gene ytfP which contained a large deletion. Finally, RNA-Seq analysis of 208 recoded genes suggested the majority of genes exhibit limited change in transcription level (Figure 2.5D), with only 28 genes found to be significantly differentially expressed.

2.4 Debugging *rE.coli-57* and Extending GETK Rules

To debug 50-kb segments that failed to complement, we performed an iterative process of deleting portions of the corresponding region on the chromosome in order to identify the single gene or subregion that could not be complemented as designed or synthesized. Figure 2.6 illustrates this process for Segment 44, where we localized the lethal error to the redesigned accD gene. Using Sanger sequencing, we compared the trouble region to the design. In several instances, a synthesis error (e.g. deletion of up to several hundred of bases) was identified, and new synthesis product was ordered. In the case where sequencing confirmed the assembled segment matched the design, as with accD in segment 44, we proceeded with our troubleshooting pipeline for design exceptions. First, RBS strength and mRNA folding were analyzed to pinpoint the cause of expression disruption[2,21,24]. We further used degenerate oligos to prototype viable alternative codons. Based on these insights, a new recoded sequence was computationally generated (Figure 2.6B) and introduced into the recoded segment via lambda Red recombineering. Viable clones were selected upon chromosomal deletion.

Each design exception was identified and characterized as shown in Figure 2.6. In order to computationally redesign lethal recoded genes, we updated our algorithm in two specific ways: 1) we refined scoring of mRNA secondary structure and 2) we added a new rule that conserves predicted internal ribosome pausing motif strengths. These updates to the algorithm were based both on MAGE experiments as well as results from other recoding work in our lab.

Specifically, the initial design used mRNA secondary structure score calculated based on a sliding window of 40 bp around the codon of interest. This simplified heuristic did not take into account the increased importance of mRNA secondary structure near the RBS and start codon of the gene[19,21,32]. We thus updated our algorithm to score mRNA secondary structure as a skewed interval that is -30 to +100 nucleotides relative to the codon of interest. Notably, for codons in the first 100 nucleotides we center the window at the start of the gene.

The second update (conserving predicted strength of internal ribosome pausing motifs) is based on observations suggesting that internal Shine-Delgarno motifs are a significant driver of translation dynamics[33]. In the initial genome design, we did not take into account potential disruption or unintended introduction of ribosome pausing motifs. In order to improve our codon selection algorithm, we treated each codon as putatively being part of a ribosomal pausing motif. Specifically, an AUG start codon was computationally inserted 10 bases after the codon of interest to simulate an RBS and generate an expression score (a proxy for motif strength). To calculate the RBS score, we use the ribosome binding site calculator[11]. We then compare the relative change in predicted expression between wild-type and recoded sequence to generate the final RBS score.

We tested the new design by λ-recombineering. Lastly, the viable recoded accD gene was Sanger-sequenced. Interestingly, all viable clones were found to carry a specific accD sequence of that had the N-terminal end of the improved design and the C-terminal end of the initial (lethal) design, underscoring the significance of N-terminal optimization for successful synonymous codon replacement[19,21]. Such recombination events, which are expected due to the high degree of homology between the two gene versions, effectively shuffle the sequences and increase the search space of viable recoded codons.

2.5 Discussion

Progress toward constructing *rE.coli-57* provides crucial insights into design challenges in recoding genomes. While the number of individual design exceptions encountered were relatively few, we found that testing libraries of degenerate oligos via MAGE alone was insufficient to finding a solution. MAGE is effective for finding a viable alternative to individuals codons, but was unable to yield a solution for multiple codon changes simultaneously. The MAGE results were useful for updating GETK rules, in addition to design principles learned from a parallel project in the lab[1]. In the accD case, the solution was ultimately a combination of an updated GETK design and an adventitious recombination event, highlighting the role and opportunity for integration between computational design and empirical investigation.

Another challenge not directly explored yet will be synthetic effects of combining multiple recoded regions. Individual design flaws may have a neutral or slightly negative effect, but it remains to be determined what the effect of combinations of these changes will result in. Here again we will need to use both design and clever experimental design to make progress.

The utility of GETK is exhibited in the 99.5% of recoded designs proving viable. Further, our ability to resolve the accD design exception (Figure 2.6) through extending GETK illustrates that the software can be tuned based on empirical results. Many segments remain to be constructed and this will require further tuning GETK rules and incorporating additional rules that capture known biophysical principles.

The *rE.coli-57* genetic code will remain unchanged until all codons and respective tRNAs and release factors are removed (e.g. tRNA genes argU, argW, serV, leuX, leuZ, release factor prfA). Only then can the strain be tested for novel properties and introduced with up to 4 orthogonal nsAAs. Once complete, a genetically isolated *rE.coli-57* will offer a unique chassis with expanded synthetic functionality broadly applicable for biotechnology.

The code used to design the *rE.coli-57* genome is available at `https://github.com/churchlab/recoli57`. GETK code is available at `https://github.com/churchlab/getk`.

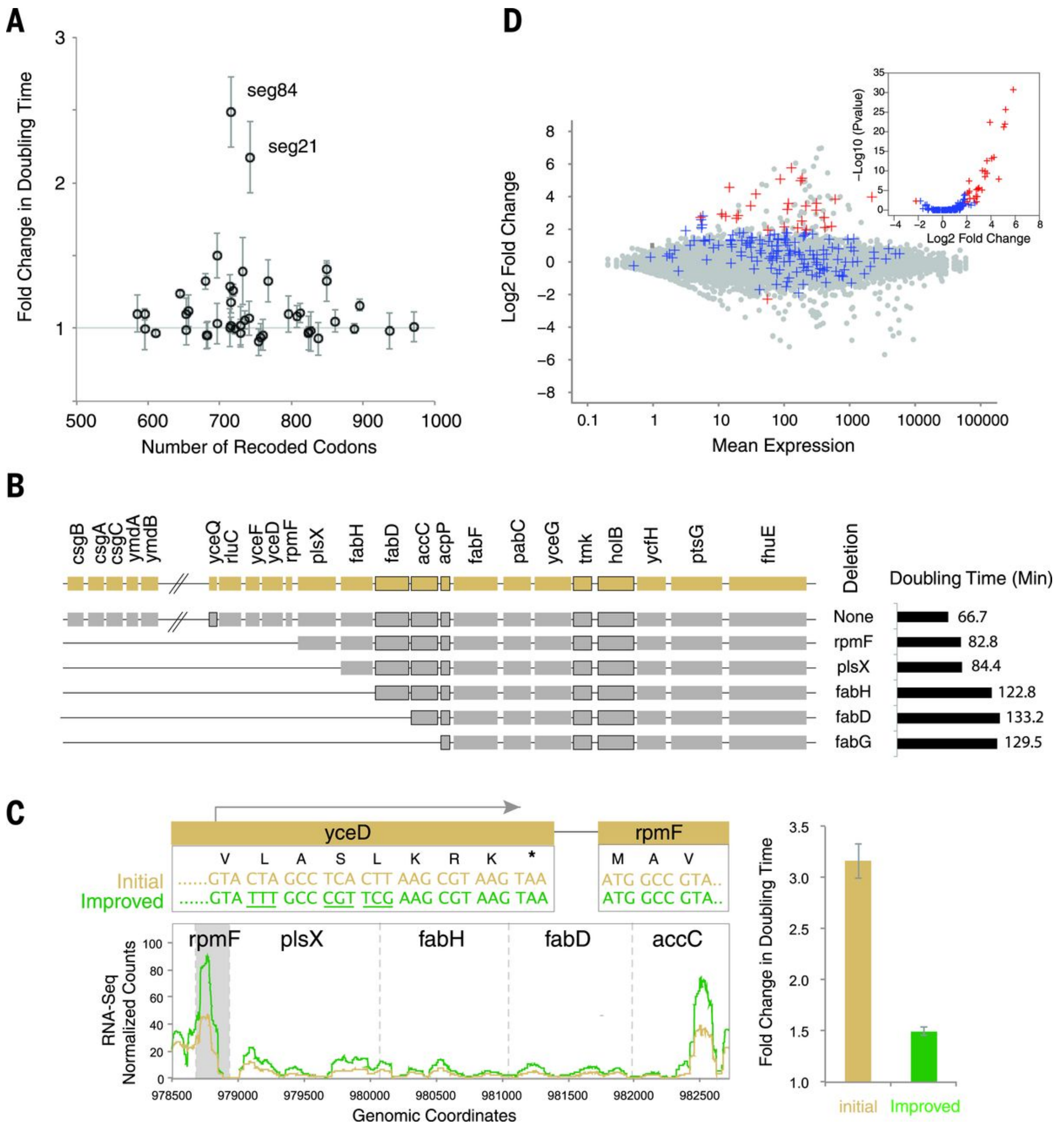


Figure 2.5: Phenotypic analysis of recoded strains. (A) Recoded segments were episomally expressed in the absence of corresponding wild-type genes. Doubling time shown relative to non-recoded parent strain. (B) Localization of fitness impairment in segment 21. Chromosomal genes (gray) were deleted to test for functional complementation by recoded genes (orange). Decrease in doubling time was observed upon deletion of rpmF-accC operon. Essential genes are framed. (C) Fine-tuning of rpmF-accC operon promoter resulted in increased gene expression and decrease in doubling time (normalized counts represent mean scaled sequencing depth). Orange: Initial promoter. Green: Improved promoter. (D) RNA-Seq analysis of 208 recoded genes (blue, segments 21, 38, 44, 46, 70). Wild type gene expression shown in gray. Differentially expressed recoded genes shown in red (absolute log2 fold-change >2, adjusted p-value <0.01). Fold-changes represent the difference between expression of each gene in a given strain and the average expression of the same gene in all other strains. Inset: P-value distribution of recoded genes.

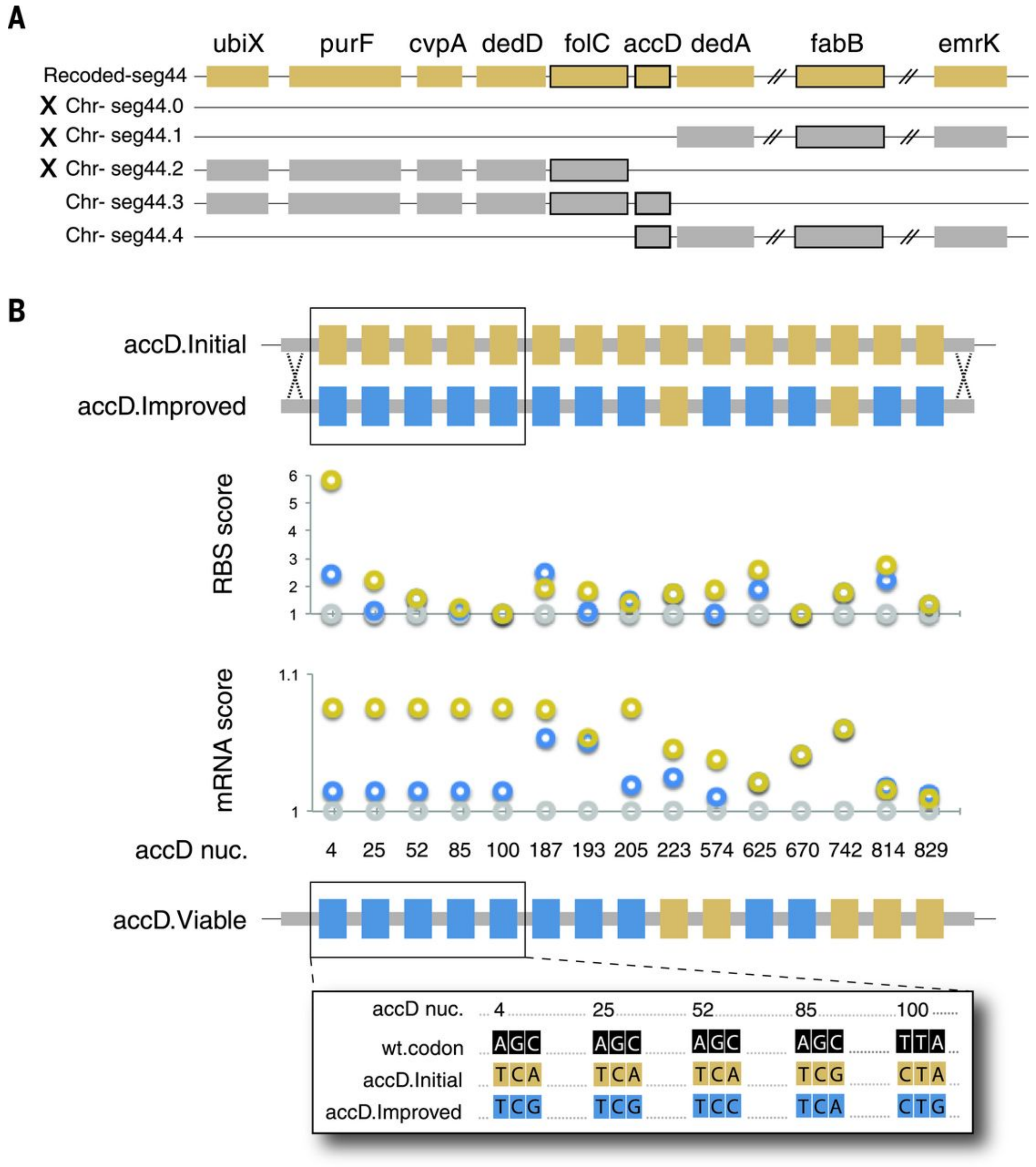


Figure 2.6: Troubleshooting lethal design exceptions. (A) Recoded segment 44 (orange) did not support cell viability upon deletion of the corresponding chromosomal sequence (Chr-Δseg44.0). The causative recoded gene accD was identified by successive chromosomal deletions (Chr-Δseg44.1-4. 'X' – nonviable). Essential genes are framed. (B) ?-recombination was used to exchange lethal accD sequence (accD.Initial, recoded codons in orange) with an alternative recoded accD sequence (accD.Improved, alternative codons in blue). mRNA structure and RBS motif strength were calculated for both sequences. Wt shown in gray. accD nuc- the first position in each recoded codon. The resulting viable sequence (accD.Viable) carried codons from both designs. Full sequences are provided in fig. S11. mRNA and RBS scores - ratio between predicted mRNA folding energy (kcal/mol) [9] or predicted RBS strength [11] of recoded and non-recoded codon.

3
Optimizing complex phenotypes through model-guided multiplex genome engineering

The contents of this chapter were adapted from:

Kuznetsov G*, Goodman DB*, Filsinger GT*, Landon M, Rohland N, Aach J, Lajoie MJ, Church GM. *Optimizing complex phenotypes through model-guided multiplex genome engineering.* (2017) Genome biology, 18:100.

Acknowledgements:

The project was conceived by MJL and GK. GK and MJL performed the 50-cycle MAGE experiment and whole-genome sequencing. GK and DBG developed computational methods and performed computational analysis. GK and GF iterated on modeling results and experimentally validated top alleles. GF performed all protein production assays.

In the previous chapter, we saw how encoding design rules in software can allow generating an update genome design for an *E. coli* using only 57 of 64 codons. We experimentally validated entire 50-kb segments and found that most design changes are permitted. However, the accumulation of many changes or collateral mutations as a result of the engineering process can potentially result in a fitness impaired-organism. While adaptive laboratory evolution (ALE) has traditionally been used to optimize engineered or evolved organisms, there is a need for a method of identifying a minimal set of high-impact tweaks in order to minimize collateral changes and avoid breaking design constraints. In this chapter, we present a method for identifying genomic modifications that optimize a complex phenotype through multiplex genome engineering and predictive modeling. We apply our method to identify six single nucleotide mutations that recover 59% of the fitness defect exhibited by the 63-codon E. coli strain C321.ΔA. By introducing targeted combinations of changes in multiplex, we generate rich genotypic and phenotypic diversity and characterize clones using whole-genome sequencing and doubling time measurements. Regularized multivariate linear regression accurately quantifies individual allelic effects and overcomes bias from hitchhiking mutations and context-dependence of genome editing efficiency that would confound strategies based on enrichment alone.

3.1 Background

Genome editing and DNA synthesis technologies are enabling the construction of engineered organisms with synthetic metabolic pathways[34], reduced and refactored genomes[18,28,15,35], and expanded genetic codes[4,36]. However, genome-scale engineering can come at the cost of reduced fitness or suboptimal traits[18,36] caused by design flaws that fail to preserve critical biological features[36,1], synthesis errors, or collateral mutations acquired during strain construction[4]. It remains challenging to identify alleles that contribute to these complex phenotypes and prohibitive to test them individually. Laboratory evolution has traditionally been used to improve desired phenotypes and navigate genetic landscapes[?] ; however, this process relies on mutations that accumulate across the genome and may disrupt synthetic designs or traits not maintained under selection. In contrast, targeted genome engineering can alter the genome at chosen loci and can be used to target many locations simultaneously[26]. Multiplexed editing creates a large pool of combinatorial genomic changes that can be screened or selected to find high-performing genomic designs. However, as the number of targeted loci considered increases, it becomes difficult to interpret the significance of individual changes. There remains a need for a method to rapidly identify subsets of beneficial alleles from a large list of candidates in order to optimize large-scale genome engineering efforts.

Leveraging recent improvements in the cost and speed of microbial whole genome sequencing, we present a method for identifying precise genomic changes that optimize complex phenotypes, combining multiplex genome engineering, genotyping, and predictive modeling (Figure 3.1). Multiple rounds of genome editing are used to generate a population enriched with combinatorial diversity at the targeted loci. Throughout the editing process, clones from the population are subject to whole-genome sequencing and are screened for phenotype. The genotype and phenotype data is used to update a model which predicts the effects of individual alleles. These steps are repeated on a reduced set of candidate alleles informed by the model, or on a new set of targets. Finally, the highest impact alleles are rationally introduced into the original organism, minimizing alterations to the organism's original genotype while optimizing the desired phenotype.

We applied this method to the genomically recoded organism (GRO) C321.ΔA, a strain of E. coli engineered for non-standard amino acid (nsAA) incorporation[4]. C321.ΔA was constructed by replacing all 321 UAG stop codons with synonymous UAA codons and deleting UAG-terminating release factor prfA. Over the course of the construction process, C321.ΔA acquired 355 off-target mutations and developed a 60% greater doubling time relative to its non-recoded parent strain, E. coli MG1655. An improved C321.ΔA strain would accelerate the pace of research involving GROs and further enable applications leveraging expanded genetic codes, including biocontainment[7], virus resistance[37] and expanded protein properties[23]. We expected that a subset of the off-target mutations caused a considerable fraction of the fitness defect, providing a starting hypothesis for iterative improvement.

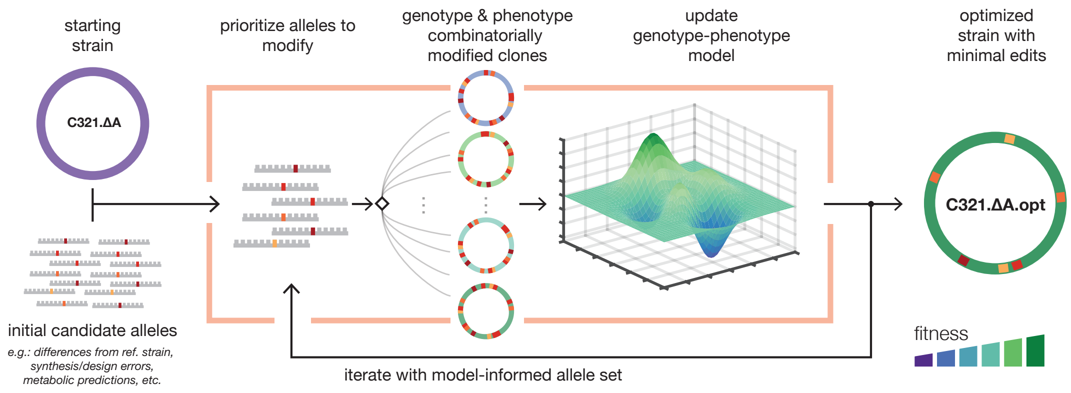

starting strain
C321.ΔA
initial candidate alleles
e.g.: differences from ref. strain, synthesis/design errors, metabolic predictions, etc.
prioritize alleles to modify
genotype & phenotype combinatorially modified clones
update genotype-phenotype model
optimized strain with minimal edits
C321.ΔA.opt
fitness
iterate with model-informed allele set

Figure 3.1: Workflow for improving phenotypes through model-guided multiplex genome editing. First, an initial set of target alleles (hundreds to thousands) is chosen for testing based on starting hypotheses. These targets may be designed based on differences from a reference strain, synthesis or design errors, or biophysical modeling. Multiplex genome editing creates a set of modified clones enriched with combinations of the targeted changes. Clones are screened for genotype and phenotype, and predictive modeling is used to quantify allele effects. The workflow is repeated to validate and test new alleles. Beneficial alleles are combined to create an optimized genotype.

3.2 Results

To select an initial set of candidate alleles (Figure B.1), we first used the genome engineering and analysis software Millstone[38] to analyze sequencing data from C321.ΔA and to identify all mutations relative to the parental strain MG1655. Millstone uses SnpEff[39] to annotate affected genes and predicted severity of each mutation. We further annotated each coding mutation with the growth defect of its associated gene's Keio collection knockout strain after 22 hours in lysogeny broth (LB_22)[40]. Based on this analysis, we identified 127 mutations in proteins and non-coding RNA as the top candidates responsible for fitness impairment. Our candidate alleles included all frameshift and non-synonymous mutations, mutations in non-coding RNA, and synonymous changes in genes with LB_22 < 0.7. We partitioned the targets into three priority categories according to predicted effect ([41]Additional file 2 and Additional file 3).

MAGE introduces combinations of genome edits with approximately 10-20% of cells receiving at least one edit per cycle[26]. To generate a diverse population of mutants enriched for reversions at multiple loci, we performed up to 50 cycles of MAGE in three lineages. The first lineage used a pool of 26 oligonucleotides targeting only the highest category of mutations, the second lineage targeted the top 49 sites, and the third lineage targeted all 127 (Figure B.1).

We sampled a total of 90 clones from multiple time points and lineages during MAGE cycling, including three separate clones of the starting strain. We then performed whole genome sequencing and measured doubling time for each clone. Millstone was used to process sequencing data and to report variants for all 90 samples in parallel. We observed fitness improvement across all three lineages with a diversity of genotypes and fitness phenotypes across the multiple time points (Figure 3.2 and Figure 3.3 a,b). Clones selected from the final time point recovered 40-58% (mean 49%) of the fitness defect compared to MG1655 and had between 5 and 15 (mean 10.2) successfully reverted mutations. Of the 127 targeted mutations, 99 were observed in at least one clone, with as many as 19 successful reversions in a clone from the 127-oligo lineage. Additionally, we observed 1,329 unique *de novo* mutations across all clones (although only 135 were called in more than one clone), accumulating at a rate of roughly one per MAGE cycle in each clone (Figure 3.2 d,e). This elevated mutation rate was caused by defective mismatch repair (ΔmutS), which both increases MAGE allele replacement frequency and provides a source of new mutations that could improve fitness.

The combinatorial diversity produced by sampling at regular intervals between consecutive rounds of multiplex genome engineering generates a dataset well-suited for analysis by linear regression. Initially, we made a simplifying assumption that doubling time is determined by the independent effects of individual alleles and employed a first-order multiplicative model that predicts doubling time based on allele occurrence (Methods and Supplementary Note 1). As model features, we considered the 99 reversions and 135 *de novo* mutations that occurred in at least two clones. Multivariate linear regression was used to fit the model, with feature coefficients indicating the predicted effect of the respective allele. We considered several priors in selecting our specific modeling strategy: 1) we expected a small number of alleles to contribute significantly to fitness improvement; 2) the continuous passaging nature of our experiment may allow hitchhiker alleles to become associated with causal alleles. Thus we chose to use elastic net regularization[42], which adds a weighted combination of L1 and L2 terms to the objective function. To limit overfitting, we performed multiple

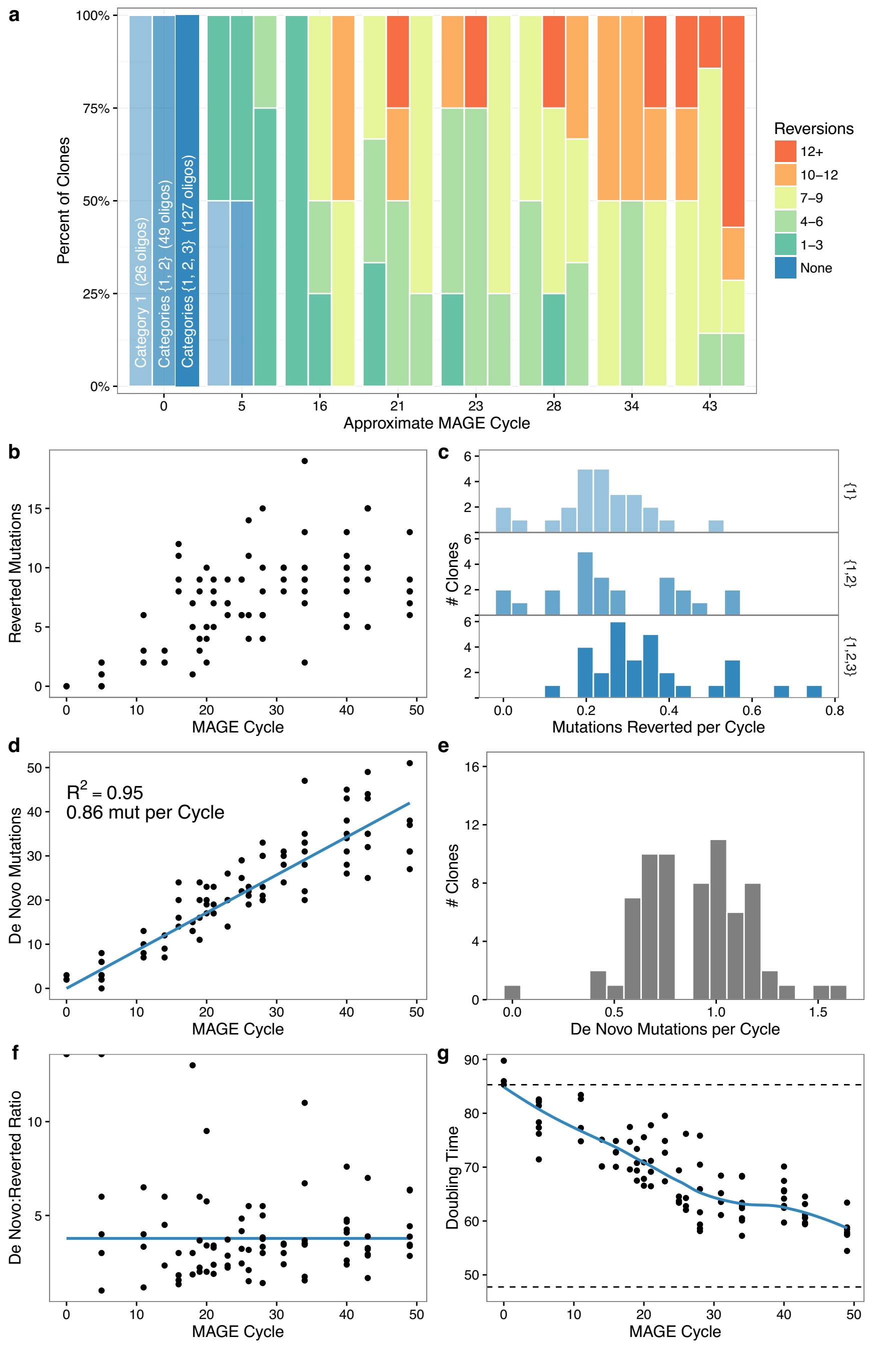


Figure 3.2: Mutation dynamics over many cycles of MAGE allele reversion. (a) Increase in combinatorial diversity and reversion count versus number of MAGE cycles. (b) Number of reversions per clone vs MAGE cycle. (c) The rate of reversions per MAGE cycle among the different allele categories, showing a higher rate per cycle for cells exposed to all 127 oligos. (d) The number of *de novo* mutations per clone over successive MAGE cycles. (e) Rate of de novo mutations per MAGE cycle. (f) The average ratio between number of de novo mutations and reverted alleles per MAGE cycle remains constant throughout the experiment. (g) Doubling time (min) improvement per clone from the C321.ΔA starting strain (top dotted line) towards the ECNR2 parent strain (bottom dotted line). Blue line is a LOESS fit.

rounds of k-fold cross-validation (k=5) and selected alleles that were assigned a non-zero coefficient on average. The analysis of the data obtained over 50 cycles of MAGE identified four targeted reversions and four de novo mutations that had the greatest putative effect on fitness (Figure 3.3 c,d and [41]Additional file 4).

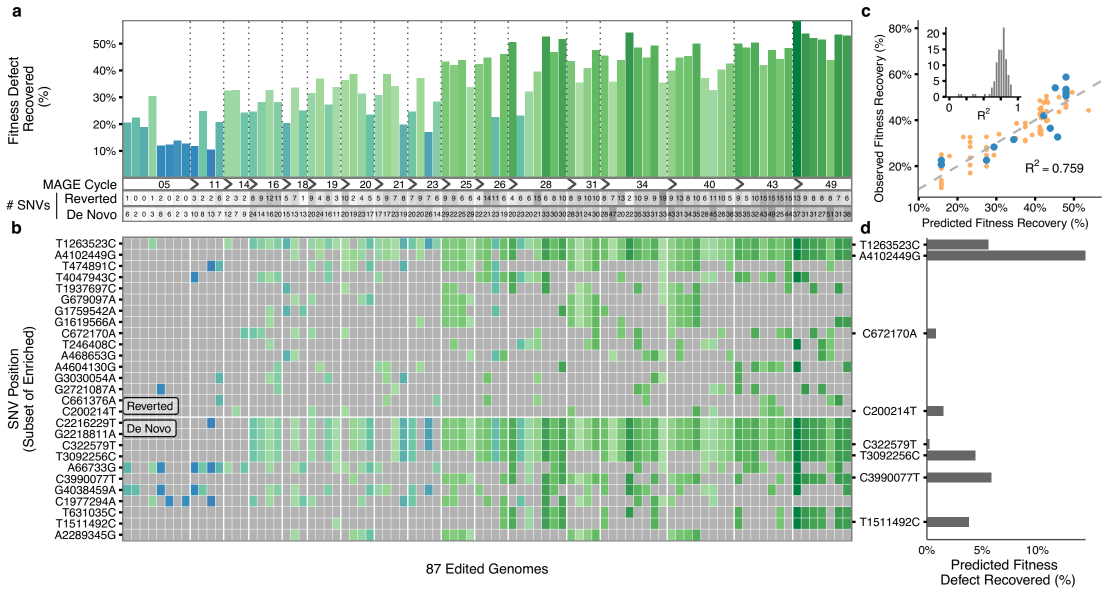


Figure 3.3: Genotypic and phenotypic diversity in 87 clones sampled across 50 MAGE cycles enabled model-guided prioritization of top single nucleotide variants (SNVs) for further validation. (a) Percent of C321.ΔA fitness defect recovered across MAGE cycles (shown with bar color and height). The number of SNVs reverted or introduced are shown below. (b) Presence of targeted reversions and *de novo* mutations in each clone colored according to fitness. A subset of the most enriched mutations are shown, ordered by enrichment (full dataset available in [41]Additional file 10). (c) Example model fit using top 8 alleles as features with 15 samples left out as a test set (blue points) and used to evaluate R2. Training points are plotted in orange. The inset shows distribution of R2 values for 100 different simulations with 15 random samples left out to calculate R2 for each. Example fit was chosen to exemplify a median R2 value from this distribution. (d) Average model fit coefficients for top 8 alleles assigned non-zero values over repeated cross-validated linear regression (Methods) indicate their predicted contribution to fitness improvement.

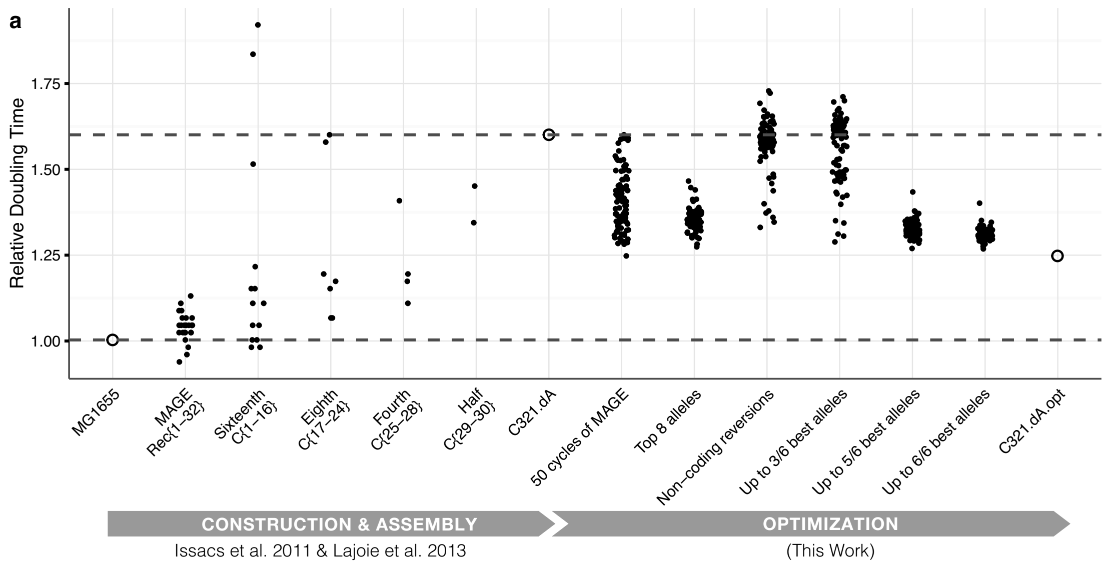
To validate the eight alleles prioritized in the 50-cycle MAGE experiment, we performed nine cycles of MAGE using a pool of eight oligos ([41]Additional file 4) applied to the starting C321.ΔA strain. We then screened each clone using MASC-PCR (Methods) and measured doubling time

(Figure B.2). Modeling revealed strong effects for two reversions (hemA-T1263523C and cpxA-A4102449G) and one *de novo* mutation (cyaA-C3990077T), along with weaker effects for two additional reversions (bamA-C200214T and leuS-C672170A). These mutations are discussed in Supplementary Note 2. A clone with all five of these mutations was isolated and measured to have recovered 51% of the fitness defect exhibited by C321.ΔA. The three remaining de novo mutations did not show evidence of improving fitness despite being highlighted in the initial modeling, illustrating the importance of subsequent validation of model-selected alleles.

To identify mutations that further improved the fitness of C321.ΔA, we extended our search to off-target mutations occurring in regulatory regions using smaller pool sizes. We identified seven non-coding mutations predicted to disrupt gene regulation[1] (Methods and [41]Additional file 5). Applying nine rounds of MAGE followed by linear modeling identified the reversion C49765T, a mutation in the -35 box of the folA promoter, which recovers a predicted 27% of the fitness defect (Figure B.3).

To test whether any of the designed UAG-to-UAA mutations caused a fitness defect in the C321 background, we followed the same procedure with 20 previously recoded UAA codons predicted to have a potentially disruptive effect ([41]Additional file 6). We tested reversion back to UAG in a prfA+ variant of C321 capable of terminating translation at UAG codons. We observed no evidence of a beneficial fitness effect from any individual UAA-to-UAG reversion.

Finally, we used MAGE to introduce the best six mutations ([41]Additional file 7) into the original C321.ΔA strain (Methods), creating an optimized strain C321.ΔA.opt that restores 59 +/- 11% of the fitness defect in C321.ΔA (Figure 3.4 a). This rationally designed strain recovered the same amount of fitness as the fastest clones obtained through 50 rounds of MAGE and substantial passaging, which resulted in 6-13 reversions and 31-38 *de novo* mutations. (Figure 3.4 a). Whole genome sequencing of the final strain confirmed that no UAG codons were reintroduced. Nine additional de novo mutations arose, but these are predicted to have a neutral effect ([41]Additional file 8). We characterized UAG-dependent incorporation of the nsAAs p-acetyl-L-phenylalanine (pAcF) in C321.ΔA.opt using sfGFP variants with 0, 1, and 3 residues replaced by the UAG codon and confirmed that C321.ΔA.opt maintains nsAA-dependent protein expression (Figure 3.4 b). C321.ΔA.opt has been deposited at AddGene (Bacterial strain #87359).



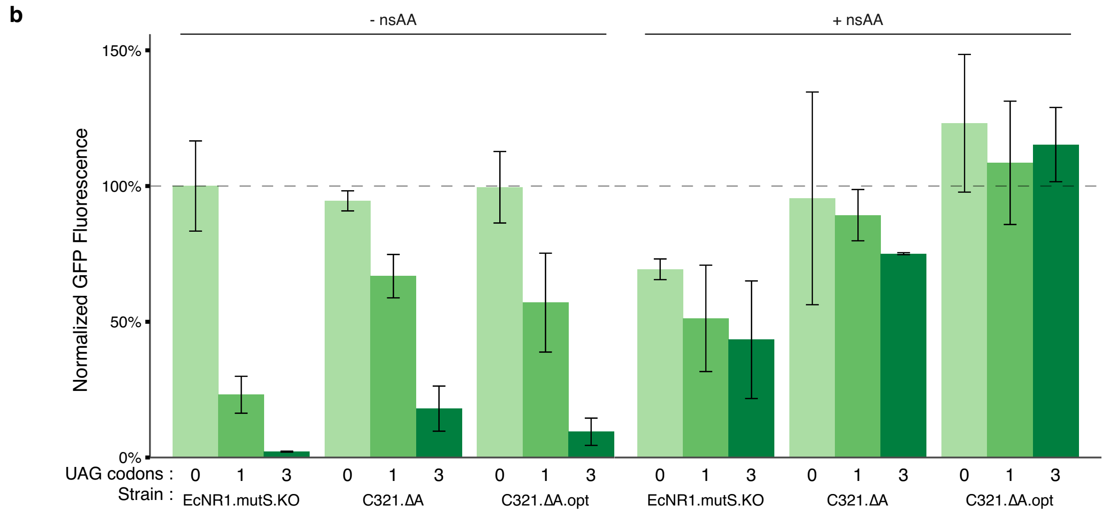


Figure 3.4: Construction and characterization of final strain C321.ΔA.opt (a) Doubling time of clones isolated during construction and optimization of C321.ΔA. Strain C321.ΔA.opt was constructed in seven cycles of MAGE in batches of up to three cycles separated by MASC-PCR screening to pick clones with the maximum number of alleles converted (see Methods). The two dotted horizontal lines correspond to the relative doubling times for the original GRO and the wild-type strain. (b) Testing nsAA-dependent protein expression using the nsAA p-acetyl-L-phenylalanine (pAcF) in sfGFP variants with 0, 1, or 3 residues replaced with UAG codons. Normalized GFP fluorescence was calculated by taking the ratio of absolute fluorescence to OD600 of cells suspended in Phosphate Buffered Saline (PBS) for each sample and normalizing to the fluorescence ratio of non-recoded strain EcNR1.mutS.KO expressing 0 UAG sfGFP plasmid.

To address the remaining fitness defect, we first examined potential interactions among the six alleles identified. We characterized the fitness of 359 clones with intermediate genotypes generated during the construction of the final strain (Figure 3.4 a). We applied linear regression with higher order interaction terms (Figure 3.5 a) and observed that combinations of mutations tended to produce diminishing returns[43], suggesting that additional beneficial alleles would only contribute marginally to fitness (Figure 3.5 b). A set of relatively weaker mutations may contribute to the remaining fitness defect, although we cannot exclude the possibility that the combination of 321 designed UAG-to-UAA mutations contributes to the global defect as well.

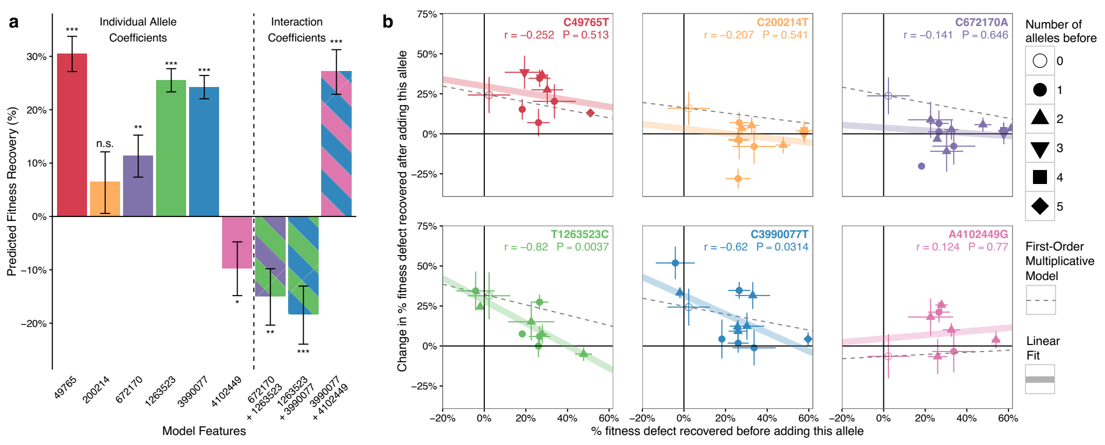


Figure 3.5: Interactions among top six alleles show evidence of epistasis. Genotypes and fitness measurements were obtained from 359 intermediate clones generated during the construction of the final strain containing the six best alleles ([41]Additional file 7). Each clone was genotyped using MASC-PCR and doubling time was measured during allele validation experiments and final strain construction. (a) Individual model coefficients for the top six alleles, as well as three significant interaction terms identified during combinatorial construction. These values are from a linear model with interaction terms between each pair of alleles. The bars signify the standard error of the mean of the model coefficients, and the significance codes for a non-zero effect size are '***' : $p < 0.001$, '**' : $0.001 < p < 0.01$, '*' : $0.01 < p < 0.05$, 'n.s.' not significant. All three interactions coefficients remain significant after a family-wise error rate (FWER) $\alpha = 0.05/C(6, 2) = 0.003$. (b) Each data point represents the amount of fitness recovered when adding the allele specified to an identical starting genotype background. Horizontal error bars correspond to the standard deviation of fitness defect among all clones with this starting genotype. Vertical error bars represent the standard deviation of all differences between clones with and without the respective allele. For each plot, the thick colored line represents a simple linear fit through the points, corresponding to the r and p values given in each plot. The dotted line corresponds to the predicted fit for a simple multiplicative model of fitness where the allele always recovers a constant percent of the remaining fitness defect regardless of the background. For all alleles except A4102449G (pink), adding the allele to C321 showed a recovery of the fitness defect (>0 on the y axis), with the percentage of defect recovered decreasing as other alleles are also reverted, consistent with a first-order multiplicative model. In some cases the fitness improvement drops more rapidly than predicted by the multiplicative model (i.e. points below the dotted lines), suggesting diminishing returns epistasis. This is supported by the negative-coefficient interaction terms in panel a. In the case of A4102449G there appears to be a negative effect with the mutation alone, but an increase in the presence of other alleles, suggesting possible sign epistasis.

Simulating effects of experiment design parameters and modeling technique on power of predicting causal mutations

To evaluate the possibility that our modeling procedure did not detect all effects among alleles tested, we performed *in silico* simulations of a simplified version of our experiment and investigated our ability to detect fitness effect with varying numbers of underlying causal mutations. We found that in the idealized case of no epistasis, we would detect over 90% of total fitness effect given our experimental design parameters (Figure 3.6 e). Because simulation can be a powerful tool for design of experiments, here I describe the implementation of the simulation and insights learned about experimental parameters and comparisons of modeling methods. A Jupyter notebook containing the simulation code can be found at `https://github.com/churchlab/optimizing-complex-phenotypes`.

The simulation parallels our 50-cycle MAGE experiment and allows exploring the relationship among experimental design parameters including number of oligos tested and number of clones sampled for genotyping. The simulation also investigated different number and effect-size distributions of causal mutations. For a given combination of parameters, we sample a distribution of underlying mutation effects, which are distributed in effect size according to a power law distribution. The total fitness effect across all mutations is capped at a 50%, comparable to the C321.ΔA context. We then performed, *in silico*, iterations of MAGE separated by competitive expansion and bottlenecking of the population. We sample clonal genotypes from this simulated population and calculate phenotypes using the underlying mutation effects. We then perform predictive modeling with the simulated genotype-phenotype data and evaluate precision and recall relative to the true mutation effects. We also compare our regularized linear modeling strategy to univariate linear regression (as is used in GWAS) and enrichment of mutations in the final population. We made simplifying assumptions of no *de novo* mutations, no epistatic interactions among mutations, no measurement noise, and equal recombination efficiency for all mutations.

The simulations show that the predictive power of multivariate linear modeling requires a diverse set of genotype-phenotype pairs. Here, selection acts between MAGE cycles during expansion and bottlenecking of the population (re-growth to mid-log and sub-sampling for next round of MAGE). Sampling clones for genotyping at regular intervals over the course of MAGE cycling allows obtaining the needed genotype-phenotype diversity. We initially tuned the simulation parameters of recombination efficiency, distribution of fitness effects, and intervals between MAGE cycles by sampling clones until we observed fitness improvement and mutation accumulation distributions (Figure 3.6 a) that were representative of our real data (Fig. 2). Sampling only from the final time point (Figure 3.6 b), or simulating without selection (Figure 3.6 c), resulted in a lack of phenotypic diversity and subsequently reduced predictive modeling power.

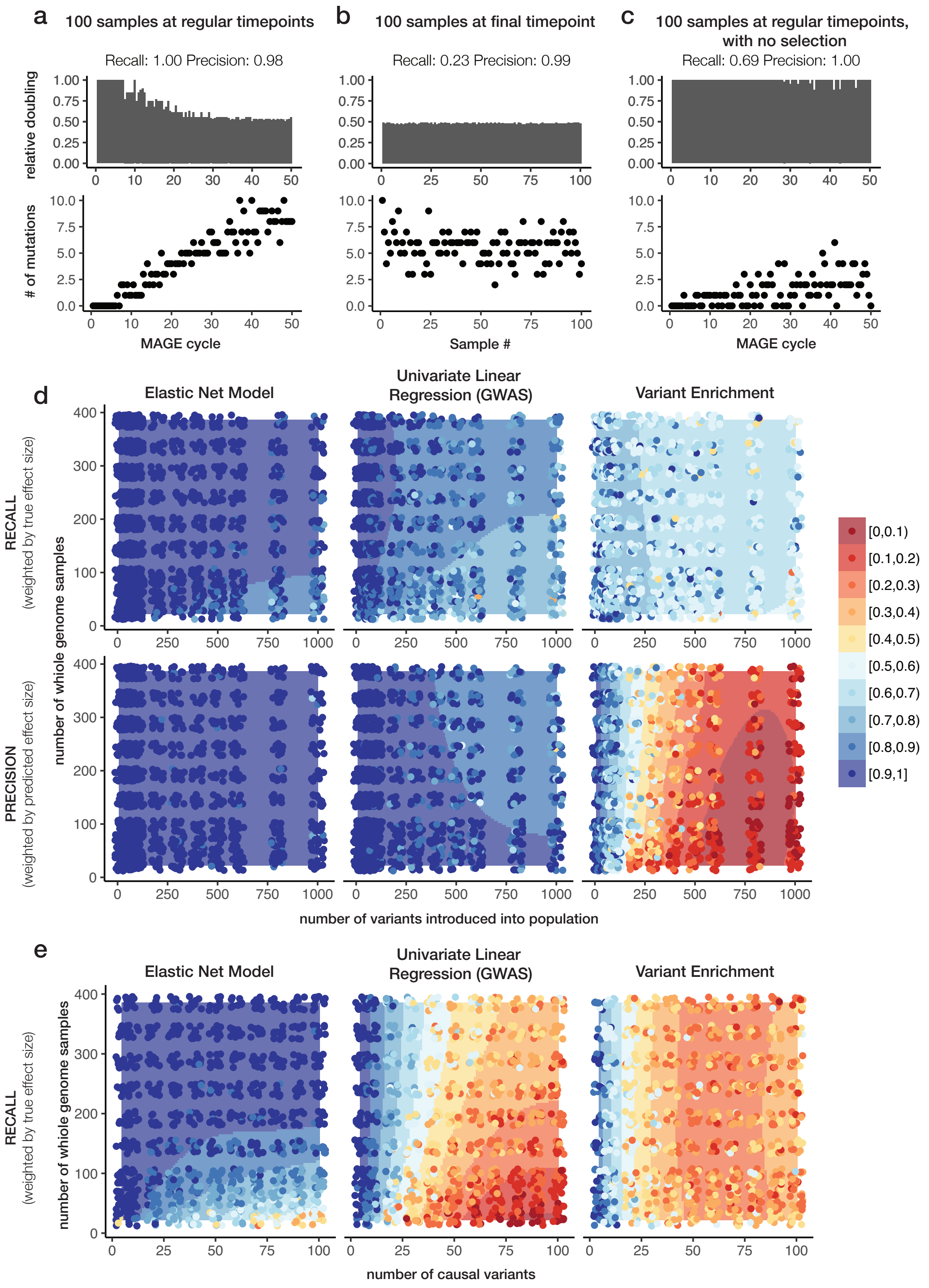
Using the simulations, we assessed the predictive capabilities of elastic net-regularized linear regression across different numbers of variants considered and number of whole genome samples, and compared our model to univariate linear regression (GWAS) and enrichment (Figure 3.6 d). As evaluation metrics, we used variants of recall and precision that are weighted by underlying mutation effect and predicted mutation effect, respectively. We found that for a variety of parameter combinations, regularized linear modeling achieved higher recall and precision, and conclude that our approach yields better results than univariate regression or enrichment. By modeling the quantified effects of combinations of mutations, regularized linear modeling with elastic net can more effectively discriminate between causal mutations and hitchhikers. Our simulations also show that while recall decreases as the total effect is distributed among greater numbers of individual mutations, it increases with the number of samples sequenced (Figure 3.6 e). Sequencing up to 200 clones was simulated to capture at least 91% of causal effect for as many as 100 causal SNPs. Precision remains consistently high at different combinations of parameters.

We expect variations of our simulation strategy to be useful for design of other experiments and provide the simulation code at `https://github.com/churchlab/optimizing-complex-phenotypes`.

3.3 Discussion

In summary, we used an iterative strategy of multiplex genome engineering and model-guided feature selection to converge on six alleles that together recover 59% of the fitness defect in C321.ΔA relative to its wild-type ancestor. This method allowed us to quantify the effects of hundreds of individual alleles and then rationally introduce only the minimal set of beneficial genetic changes, reducing unintended effects from additional off-target mutations.

Our approach reveals several problems inherent to simply using enrichment to rank allelic effect. Our data show that alleles enriched over rounds of selection are not necessarily well-correlated with fitness. Allele enrichment may be affected by differences in editing efficiency, competition among beneficial alleles through clonal interference, and genetic drift. Combinatorial targeted editing overcomes these obstacles by allowing the measurement of each allele in many genetic backgrounds, so that linear modeling can quantify its average individual effect.



a
100 samples at regular timepoints
Recall: 1.00 Precision: 0.98
b
100 samples at final timepoint
Recall: 0.23 Precision: 0.99
c
100 samples at regular timepoints, with no selection
Recall: 0.69 Precision: 1.00
relative doubling
# of mutations
MAGE cycle
Sample #
MAGE cycle
d
Elastic Net Model
Univariate Linear Regression (GWAS)
Variant Enrichment
RECALL (weighted by true effect size)
PRECISION (weighted by predicted effect size)
number of whole genome samples
number of variants introduced into population
[0,0.1)
[0.1,0.2)
[0.2,0.3)
[0.3,0.4)
[0.4,0.5)
[0.5,0.6)
[0.6,0.7)
[0.7,0.8)
[0.8,0.9)
[0.9,1]
e
Elastic Net Model
Univariate Linear Regression (GWAS)
Variant Enrichment
RECALL (weighted by true effect size)
number of whole genome samples
number of causal variants

Figure 3.6: Simulation illustrates predictive strength of elastic net model at different experimental design parameter combinations. Results of simulating varying experimental design parameters. (a) Doubling times and mutation distribution of clones sampled regularly over 50 cycles of simulated MAGE approximates what was observed in real data (Figure 3.2, Figure 3.3). (b) Sampling from final time point results in phenotypically homogenous population and reduced predictive modeling performance. (c) Running the simulation without selection results in insufficient propagation of effective alleles for predictive accuracy. (d) Precision and recall across different settings of the experimental design parameters of total variants introduced versus number of whole genome samples collected. A comparison among our elastic net linear modeling strategy and univariate linear regression (GWAS) and enrichment is shown. Points indicate results of simulation at corresponding parameters (10 replicates per parameter combo; separated by jitter). Fill color is interpolated using a LOESS regression. (e) Comparing number of genomes sampled vs number of underlying causal variants. Elastic net model predicted to achieve recall > 0.913 at at 200 whole genome samples for as many 100 causal effect SNPs.

Further, measuring mutation effects in multiplex makes it experimentally tractable to explore a much larger set of mutations. We observed evidence of positive epistatic interactions between some alleles (Figure 3.5 a, left), which would be harder to identify through singleplex editing strategies. These findings demonstrate the utility of multiplex genome engineering and predictive modeling for studying epistasis.

A similar model-guided approach could be used to augment other multiplex genome modification techniques, including yeast oligo-mediated genome engineering[44] or multiplex CRISPR/Cas9-based genome engineering in organisms that support homology-directed double-stranded break repair[44,45]. Biosensors tied to selections or screens[46] can extend this method to optimize biosynthetic pathways in addition to fitness. The rapidly declining cost of multiplex genome sequencing[47] will allow this method to scale to thousands of whole genomes, increasing statistical power and enabling the use of more complex models. While we use column-synthesized oligos in this study, chip-based oligo synthesis enables scaling up the number of genomic sites targeted, allowing thousands of alleles to be tested simultaneously[48,49,50]. Our simulations suggest that the predictive power of this method can support larger number of mutations than we tested with a modest increase in genomes sampled (Figure 3.6 d). Finally, making genomic changes trackable[51,52] for targeted sequencing could further increase the economy, speed, and throughput of this approach.

Efficiently quantifying the effects of many alleles on complex phenotypes is critical not only for tuning synthetic organisms and improving industrially relevant phenotypes, but also for understanding genome architecture. While our method is used here to identify and repair detrimental alleles to improve fitness, it will also enable rapid prototyping of alternative genome designs and interrogation of genomic design constraints. Iteratively measuring and modeling the effects of large numbers of combinatorial genomic changes in parallel is a powerful approach to navigate and understand genotype-phenotype landscapes.

3.4 Methods

Media and reagents

All experiments were performed in LB-Lennox (LBL) medium (10 g/L bacto tryptone, 5 g/L sodium chloride, 5 g/L yeast extract) with pH adjusted to 7.45 using 10 M NaOH. LBL agar plates were made from LBL plus 15 g/L Bacto Agar. Selective agents were used at the following concentrations: carbenicillin (50 μg/mL), chloramphenicol (20 μg /mL), gentamycin (5 μg/mL), kanamycin (30 μg/mL), spectinomycin (95 μg/mL), and SDS (0.005% w/v).

Strains

The construction and genotype of engineered E. coli strain C321.ΔA was previously described in detail[4]. Here, before improving fitness, we constructed strain C321.ΔA.mutSfix.KO.tolCfix.Δbla:E by further modifying C321.ΔA to introduce the following changes: 1) the mutS gene was reinserted into the C321.ΔA strain in its original locus, and MAGE was used to disable the gene by introduction of two internal stop codons and a frameshift, and 2) the carbenicillin-resistance marker bla was swapped for gentamicin resistance marker aacC1 in the lambda red insertion locus. Several control assays were performed in EcNR1.mutS.KO, a non-recoded by MAGE-enabled strain similar to EcNR2[26]. All genomic positions reported in the manuscript are in the frame of MG1655 K12 (Genbank accession NC_000913.2). The final C321.ΔA.opt strain has been deposited at AddGene (Bacterial strain #87359).

Millstone, software for multiplex genome analysis and engineering

Millstone[38] was used throughout the project to rapidly process whole genome sequencing data and identify variants in each sample relative to the reference genome, to explore variant data, and to design oligonucleotides for MAGE. The Millstone analysis pipeline takes as input raw FASTQ reads for up to hundreds of clones and a reference genome as Genbank or FASTA format. The software then automates alignment of reads to the reference using the Burrows-Wheeler Aligner (BWA-MEM) followed by single nucleotide variant (SNV) calling using Freebayes. Millstone performs variant calling in diploid mode, even for bacterial genomes. This helps account for paralogy in the genome and results in mutation calls being reported as "homozygous alternate" (strong wild-type), "heterozygous" (marginal), or wild-type, along with an "alternate fraction" (AF) field that quantifies the fraction of aligned reads at the locus showing the alternate allele. Marginal calls were inspected on a case-by-case basis using Millstone's JBrowse integration to visualize raw read alignments. Millstone provides an interface for exploring and comparing variants across samples. After initial exploration and triage in Millstone, we exported the variant report from Millstone for further analysis and predictive modeling. In follow-up analysis, we determined empirically that $0.1 < AF < 0.7$ indicated a variant call was marginal in our data.

Identifying off-target mutations for reversions

For the 50-cycle MAGE experiment, we considered only mutations occurring in regions annotated as coding for a protein or functional RNA. Using Millstone annotations of predicted effect and Keio knock-out collection annotation of essentiality[40], we defined three priority categories according to expected effect on fitness ([41]Additional file 2). A total of 127 targets were allocated to the three categories to be used for the 50-cycle MAGE experiment.

For a separate experiment, off-target mutations in regulatory regions were selected based on the criteria of predicted regulatory disruption of essential genes and several non-essential genes with particularly strong predicted disruption. Regulatory disruption was determined based on calculating change in 5' mRNA folding or ribosome binding site (RBS) motif strength for mutations occurring up to 30 bases upstream of a gene. We calculated mRNA folding and ribosome binding site (RBS) motif disruption as described in[1]. Briefly, the minimum free energy (MFE) of the 5-prime mRNA structure was calculated using Unafold's hybrid-ss-min function[9] (T=37 °C), taking the average MFE between windows of RNA (-30, +100) and (-15, +100) relative to the start codon of the gene. Mutations that caused a change in MFE of the mRNA of over 10% relative to the wild-type context were prioritized for testing. To predict RBS disruption, the Salis RBS Calculator[11] was provided with sequence starting 20 bases upstream of the gene ATG and including the ATG. Mutations that caused a greater than 10-fold change in predicted expression were included for testing. Finally, we also considered mutations that overlapped promoters of essential genes based on annotations from RegulonDB[53].

The 20 UAG-reversion targets were chosen when UAGs occurred in essential genes, introduced non-synonymous changes in overlapping genes, or disrupted a predicted regulatory feature as above.

Multiplex automated genome engineering

Single-stranded DNA oligonucleotides for MAGE were designed using Millstone's optMAGE integration `https://github.com/churchlab/optmage`. Oligos were designed to be 90 base pairs long with the mutation located at least 20 base pairs away from either end. We used the C321.ΔA reference genome (Genbank accession CP006698.1) for oligo design to avoid inadvertently reverting intentional UAG-to-UAA changes. OptMAGE avoids strong secondary structure (< -12 kcal mol-1) and chooses the sense of the oligo to target the lagging strand of the replication fork[26]. Phosphorothioate bonds were introduced between the first and second and second and third nucleotides at the 5-prime end of each oligo to inhibit exonuclease degradation[26]. All DNA oligonucleotides were purchased with standard purification and desalting from Integrated DNA Technologies and dissolved in dH2o.

MAGE was performed as described in[26], with the following specifications: 1) Cells were grown at 34 °C between cycles. 2) We noted that C321.ΔA exhibits electroporation resistance so a voltage of 2.2 kV (BioRad GenePulser, 2.2 kV, 200 ohms, 25 μF was used for cuvettes with 1mm gap) was chosen based on optimization using a lacZ blue-white screen. 3) Total concentration of the DNA oligonucleotide mixture was 5 μM for all electroporations (i.e., the concentration of each oligo was adjusted depending on how many oligos were included in the pool).

The 50-cycle MAGE experiment was carried out in three lineages, with oligo pool sizes of 26, 49, and 127 consisting of oligos from priority categories 1, 1,2, and 1,2,3, respectively ([41]Additional file 2). Note that we originally began with just two pools –the top 26 and all 127 oligos –, but after 5 MAGE cycles the lineage exposed to all 127 oligos was branched to have a separate lineage with only the 49 category 1, 2 oligos in order to obtain more enrichment of the higher priority targets. In order to prevent any population from acquiring permanent resistance to recombination, we toggled the dual-selectable marker tolC at recombinations 23, 31, and 26 for the three lineages, respectively, as described in[54]. Briefly, an oligo introducing an internal stop codon in tolC was included in the recombination, and after at least 5 hours of recovery, cells were selected in media containing colicin E1, which is toxic in tolC+ E. coli. In the subsequent recombination, an oligo restoring tolC function was included in the pool after which cells were selected in the presence of 0.005% SDS (w/v).

Validation MAGE experiments composed of 10 or fewer oligos were carried out for up to 9 MAGE cycles, as we expected adequate diversity based on previous experience with MAGE efficiency.

Whole genome sequencing

Genomic DNA (gDNA) preparation for whole genome sequencing of 96 clones (only 87 considered in manuscript because sequencing analysis revealed that 9 cultures were polyclonal) was performed as in[54]. Briefly, gDNA was prepared by shearing using a Covaris E210 AFA Ultrasonication machine. Illumina libraries were prepared for pooled sequencing as previously described[55]. Barcoded Illumina adapters were used to barcode each strain in a 96-well plate. All 96 genomes were sequenced together on a single lane of a HiSeq 2500 PE150 ([41]Additional file 9). Alternative inexpensive WGS library preparation methods have since become available[47].

WGS data was processed to identify clonal genotypes in Millstone and then exported for further analysis ([41]Additional file 10). Demultiplexed .fastq reads were aligned to the MG1655 reference genome. SNVs were reported with Millstone, as described above. During analysis, marginal calls were visually confirmed by examining alignments using Millstone's JBrowse integration.

Multiplex allele-specific colony PCR (MASC-PCR)

MASC-PCR was used to assess successful reversions in validation experiments of <= 10 targeted mutations and typically performed for 96 clones in parallel. The protocol was performed as previously described[4]. Briefly, two separate PCRs, each interrogating up to 10 positions simultaneously, were performed on each clone to detect whether the C321.ΔA or reverted allele was present at each position. For each position, the two reactions shared a common reverse primer but used distinct forward primers differing in at least one nucleotide at the 3' end to match the SNV being assayed specifically.

Positive and negative controls were included when available to aid in discriminating cases of non-specific amplification.

Measuring fitness

Fitness was determined from kinetic growth (OD600) on a Biotek H-series plate reader. Cells were grown at 34 °C in 150 μL LBL in a flat-bottom 96-well plate at 300 rpm linear shaking. To achieve consistent cell state before reading, clones were picked from agar plates or glycerol, grown overnight to confluence, passaged 1:100 into fresh media, grown again to mid-log (3 hours), and passaged 1:100 again before starting the read. OD measurements were recorded at 5 minute intervals until confluence. Doubling times were calculated according to tdouble = c * ln(2) / m, where c = 5 minutes per time point and m is the maximum slope of ln(OD600). The maximum slope was determined using a sliding window linear regression through 8 contiguous time points (40 minutes) points rather than between two predetermined OD600 values because not all of the growth curves were the same shape or reached the same max OD600. The script used for analyzing doubling time is available at `https://github.com/churchlab/analyze_plate_reader_growth`.

Predictive modeling of allele causality

Choosing alleles for subsequent validation was framed as a feature selection problem. We used predictive modeling to prioritize features. Both targeted reversions introduced by MAGE and *de novo* mutations were considered.

For most analyses, we used a first-order multiplicative allele effect model, where each allele (reversion or *de novo* mutation) is represented by a single feature and the fitted coefficient corresponding to that feature represents the allele's effect on doubling time. To find coefficient values, we fit a linear model where genotypes (WGS or MASC-PCR) predict the logarithm of doubling time. Alleles corresponding to features with the most negative coefficients were selected for validation in smaller sets. An additive model was also tested and yielded similar results, as previously noted by others[43].

While we anticipated the possibility of epistatic effects among alleles tested, a first-order model of the 50-cycle MAGE experiment already had 239 features (99 reversions + 140 *de novo* mutations observed at least twice) and 87 samples, so we omitted higher-order interaction terms to avoid overfitting due to model complexity. We discuss implications of this independence assumption and other details of our allele effect modeling strategy in Supplementary Note 1.

Elastic net regularization[42], which includes both L1 and L2 regularization penalties, was used in model-fitting. L1 regularization enforces sparsity, capturing the assumption that a handful of alleles will explain a majority of the fitness effect. L2 regularization prevents any one of a subset of highly correlated alleles from dominating the effect of those alleles, balancing the tendency of L1 to drop subsets of highly co-occurring alleles.

Accordingly, the elastic net loss function used follows from Zou and Hastie[42]:

$$L(\lambda_1, \lambda_2, \beta) = |y - X\beta|^2 + \lambda_1|\beta|_1 + \lambda_2|\beta|^2$$

where

$$|\beta|_1 = \sum_{j=1}^{p} |\beta_j|$$

$$|\beta|^2 = \sum_{j=1}^{p} \beta_j^2$$

And the coefficients are estimated according to:

$$\hat{\beta} = \arg\min_{B} L(\lambda_1, \lambda_2, \beta)$$

Elastic net regression was performed using the ElasticNetCV module from scikit-learn[56]. This module introduces the hyperparameters

$$\text{alpha} = \lambda_1 + \lambda_2$$

$$\text{l1\_ratio} = \frac{\lambda_1}{\lambda_1 + \lambda_2}$$

and uses k-fold cross validation ($k = 5$) to identify the best choice of hyperparameters for a given training dataset. We specified the range of l1_ratio to search over as [.1, .3, .5, .7, .9, .95, .99, 1], which tests with higher resolution near L1-only penalty. This fits our hypothesis that a small number of mutations are responsible for a majority of the fitness effect. For alpha, we followed the default of allowing scikit-learn to search over 100 alpha values automatically computed based on l1_ratio.

To address the under-determined dataset in the 50-cycle MAGE experiment (more alleles than individual clones), we performed 100 repetitions of scikit-learn's cross-validated elastic net regression procedure, and for each repetition, we randomly held-out 15 samples that could be used to evaluate the model fit by that iteration. The model coefficient for each allele was then calculated as the weighted-average across all 100 repetitions using the prediction score on the 15-held out samples as the weighting factor. Only model coefficients with a negative value (some putative fitness improvement) were considered in a second round of 100 repeats of cross-validated elastic net regression, again with 15 samples held-out in each repeat to evaluate the model fit. The weighted-average coefficient values over this second set of 100 repetitions were used to determine the top alleles for experimental validation in a 9-cycle MAGE experiment. While this method reproducibly reported the alleles hemA-T1263523C, cpxA-A4102449G, and cyaA-C3990077T, alleles with weaker predicted effects were detected more stochastically, depending on the randomized train-test split, even with 100 repetitions. We expect that sequencing additional clones, as well as further tuning of our modeling method for detecting weak effects may be warranted in future studies.

To evaluate the results of the 9-cycle MAGE validation experiments, we used unregularized multivariate linear regression. With <= 10 parameters and ~90 clones, only a single iteration of cross-validated regression applied to the full dataset was required to assign predicted effects without requiring the testing of individual alleles.

Elastic net-regularized multivariate regression was compared to univariate linear regression for our data (Supplementary Note 1,[41]Additional file 11).

Final strain construction

C321.ΔA.opt was constructed by adding the six alleles identified by the optimization workflow ([41]Additional file 7) to C321.ΔA.mutSfix.KO.tolCfix.Δbla:E. A total of seven cycles of MAGE were required, with a MASC-PCR screening step every three cycles to select a clone with the best genotype so far (Figure 3.4 a), minimizing the total number of cycles required. Three cycles of MAGE were performed using oligos targeting all six alleles. Ninety-six clones were screened by MASC-PCR, and one clone with 3/6 alleles (C49765T, T1263523C, A4102449G) was chosen for the next round of MAGE. Three more rounds of MAGE were performed on top of the clone with 3/6 alleles using only the three remaining oligos. MASC-PCR identified a clone with 5/6 alleles (C49765T, C200214T, C672170A, T1263523C, A4102449G). One more round of MAGE was performed using the remaining oligo and a clone with all six alleles was obtained. Additional off-target mutations acquired during construction as identified by whole genome sequencing of the final clone are listed in [41]Additional file 8.

Characterizing non-standard amino acid incorporation

nsAA incorporation was measured as previously described[4]. 1-UAG-sfGFP, and 3-UAG-sfGFP reporters were produced by PCR mutagenesis from sfGFP ([41]Additional file 1: Supplementary Note 4), and isothermal assembly was used to clone 0-UAG-sfGFP (unmodified sfGFP), 1-UAG-sfGFP, and 3-UAG-sfGFP into the pZE21 vector backbone[57]. We used the pEVOL-pAcF plasmid to incorporate the non-standard amino acid p-acetyl-L-phenylalanine. Reagents were used at the following concentrations: anhydrotetracycline (30 ng/μL), L-arabinose (0.2% w/v), pAcF (1 mM).

4

Millstone: software for multiplex microbial genome analysis and engineering

The contents of this chapter were adapted from:

Goodman DB*, **Kuznetsov G***, Lajoie MJ, Ahern BW, Napolitano MG, Chen KY, Chen C, Church GM. *Millstone: software for multiplex microbial genome analysis and engineering.* (2017) Genome biology, 18:101.

Acknowledgements:

Millstone was designed by DBG, GK, and MJL. DBG prototyped early forms of the NGS analysis pipelines and GK designed the software architecture. GK and DBG wrote the bulk of the code, with important contributions from BWA, KYC, and CC. MJL and MGN tested and contributed to design of advanced features.

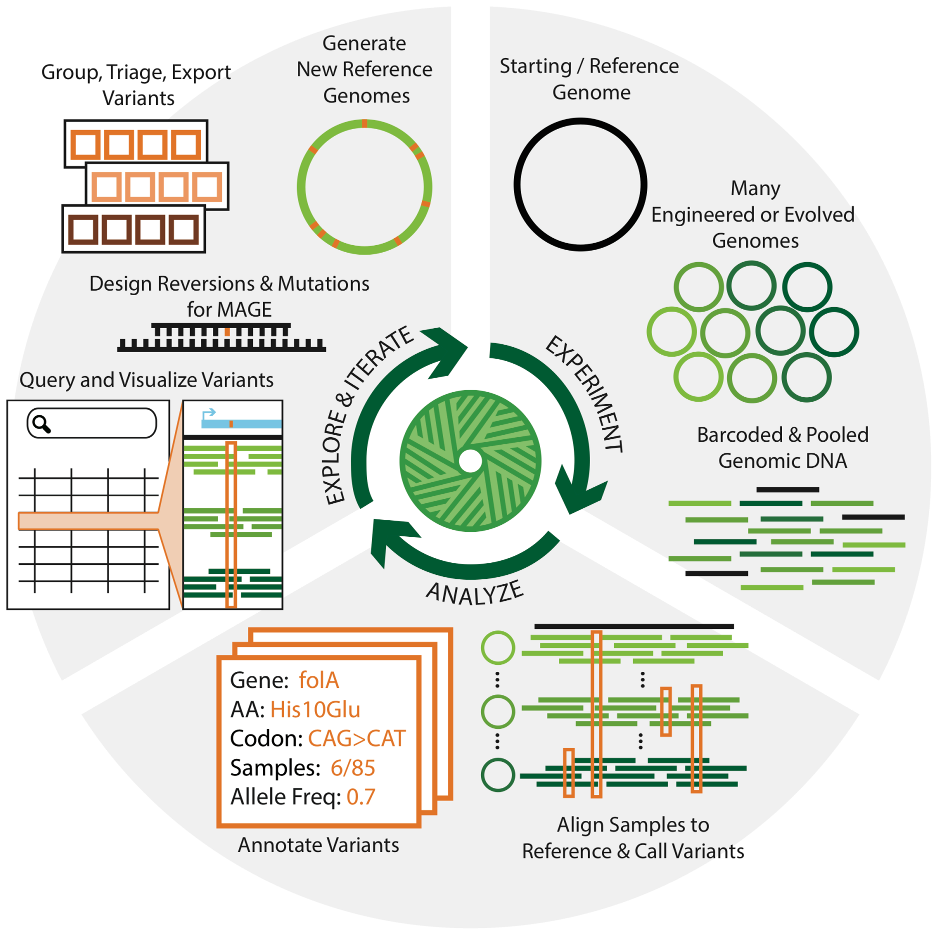
Inexpensive DNA sequencing and advances in genome editing have made computational analysis a major rate-limiting step in adaptive laboratory evolution and microbial genome engineering. In the previous chapter, we saw how whole genome sequencing and comparison of hundreds of engineered clones allowed us to systematically identify targeted mutations for improving the fitness of the first Genomically Recoded Organism. That effort relied heavily on Millstone, a web-based software platform that we built to enable a number genome of engineering projects in our lab[4,54,7,1]. Millstone automates genotype comparison and visualization of engineered and evolved genomes. In this chapter, I describe Millstone's features and how it can be used in engineering and evolution projects.

4.1 Introduction

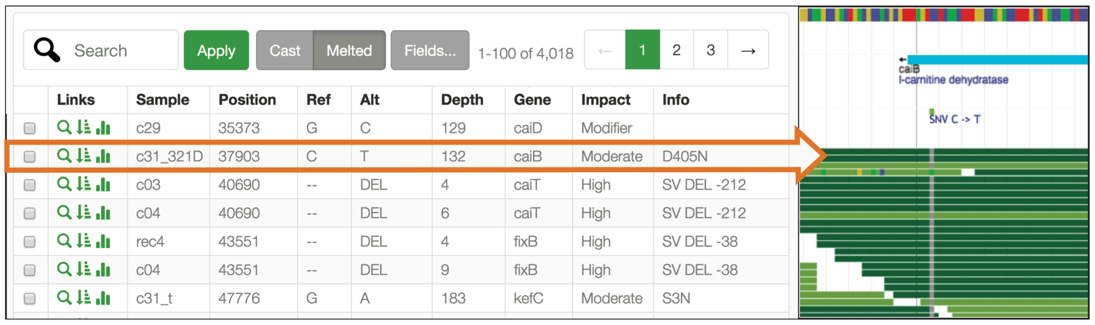
Microbial populations can harbor a staggering amount of genomic diversity, enabling them to evolve and adapt to diverse environments. Adaptive laboratory evolution uses this process to generate strains that are useful for biotechnology or for answering fundamental biological questions[59]. In addition to harnessing natural variation, biologists can generate targeted genomic diversity in a population of cells and then screen or select for phenotypes of interest[26]. The decreasing cost of reading and writing microbial genomes has made it possible to generate billions of combinatorial genomic variants per day at specific loci[26,25,60] and to sequence entire *E. coli* genomes for less than 25 USD per sample[47,61] (Supplementary Note 1).

Computational analysis is increasingly a bottleneck when mapping whole-genome data to phenotypes across many samples. Going from raw DNA sequencing reads to annotated variants requires the integration of a large number of disparate tools, usually assembled into an *ad hoc* pipeline by individual labs and followed by time-intensive manual confirmation of variants. There remains a critical need for an integrated solution capable of comparative analysis among multiple genomes and supporting interactive querying and data visualization, collaboration, genome versioning, and the design of additional mutations or reversions (Table C.1, Supplementary Note 2).

To address this need, we developed Millstone, a web-based software platform that supports an iterative process of multiplex mutation analysis and genome engineering. Millstone automates read alignment and variant calling using a hybrid reference-based and *de novo* assembly approach, then allows researchers to explore and compare mutations among genomic samples, and finally creates updated reference genomes and designs new genomic edits for subsequent rounds of experiments (Figure 4.1 a, Figure C.1). Serving as both a genomics pipeline and a platform for exploring a


Group, Triage, Export Variants
Generate New Reference Genomes
Starting / Reference Genome
Many Engineered or Evolved Genomes
Design Reversions & Mutations for MAGE
Query and Visualize Variants
EXPLORE & ITERATE
EXPERIMENT
ANALYZE
Barcoded & Pooled Genomic DNA
Gene: folA
AA: His10Glu
Codon: CAG>CAT
Samples: 6/85
Allele Freq: 0.7
Annotate Variants
Align Samples to Reference & Call Variants

b



	Links	Sample	Position	Ref	Alt	Depth	Gene	Impact	Info
☐		c29	35373	G	C	129	caiD	Modifier	
☐		c31_321D	37903	C	T	132	caiB	Moderate	D405N
☐		c03	40690	--	DEL	4	caiT	High	SV DEL -212
☐		c04	40690	--	DEL	6	caiT	High	SV DEL -212
☐		rec4	43551	--	DEL	4	fixB	High	SV DEL -38
☐		c04	43551	--	DEL	9	fixB	High	SV DEL -38
☐		c31_t	47776	G	A	183	kefC	Moderate	S3N

Figure 4.1: Millstone enables rapid iterative multiplex genome analysis and engineering. (**a**) To use Millstone, a researcher uploads a reference genome and next-generation sequencing reads for many individual genomic clones, for example from long-term evolution or targeted genome editing. Millstone performs alignment and variant calling for both single nucleotide variants and structural variation and then assigns predicted effects based on reference genome annotations. A unified data model stores sample genotype, phenotype, and variant annotation data. Variants can then be queried, filtered, and grouped into sets for export, triage, and analysis. These variant sets can be used to design oligonucleotides to recreate or revert mutations of interest, or used to generate new versions of the reference genome. (**b**) A combined screenshot of the Millstone analysis and alignment visualization views (condensed and cropped for clarity). A custom query language allows searching and filtering over the data. As variant calls sometimes require visual inspection and comparison, Millstone's variant analysis view provides programmatically-generated links to visualizations of the relevant read alignments in JBrowse[58].

whole-genome sequencing data, Millstone provides a powerful user-friendly interface that allows researchers to investigate individual variants through interactive filtering and alignment visualization (Figure 4.1 b).

4.2 Results

Millstone was built in response to challenges encountered during the construction of the genomically recoded organism (GRO) $C321.\Delta A$[4], a strain of *E. coli* in which all 321 UAG stop codons were replaced with a synonymous UAA. Multiplex automated genome engineering[26] (MAGE) was used to introduce sets of 10 mutations into 32 strains and conjugative assembly genome engineering[25] (CAGE) was used to hierarchically combine redesigned regions into a chromosome with all 321 UAGs recoded (Figure 4.2 a, green). We sequenced 68 intermediate clones to confirm the designed changes but our initial analyses were slow, error-prone, and lacked the ability to visualize and compare evidence for mutations among samples. Millstone solved these issues, allowing us to identify and track the 3127 designed and off-target mutations across all strains. Finally, by iteratively applying mutations directly to the initial reference genome and re-aligning reads, Millstone allowed us to generate a new $C321.\Delta A$ reference sequence which incorporated 355 additional off-target mutations that had accumulated during strain construction. (Figure 4.2 a, green and orange).

Millstone's ability to rapidly generate clonal genotypes from whole genome sequencing reads enabled a follow-up project to improve the fitness of the GRO. The final strain from Lajoie et al. demonstrated incorporation of proteins containing non-standard amino acids, but suffered from an impaired growth phenotype, which we hypothesized was due to a subset of the 355 off-target mu-

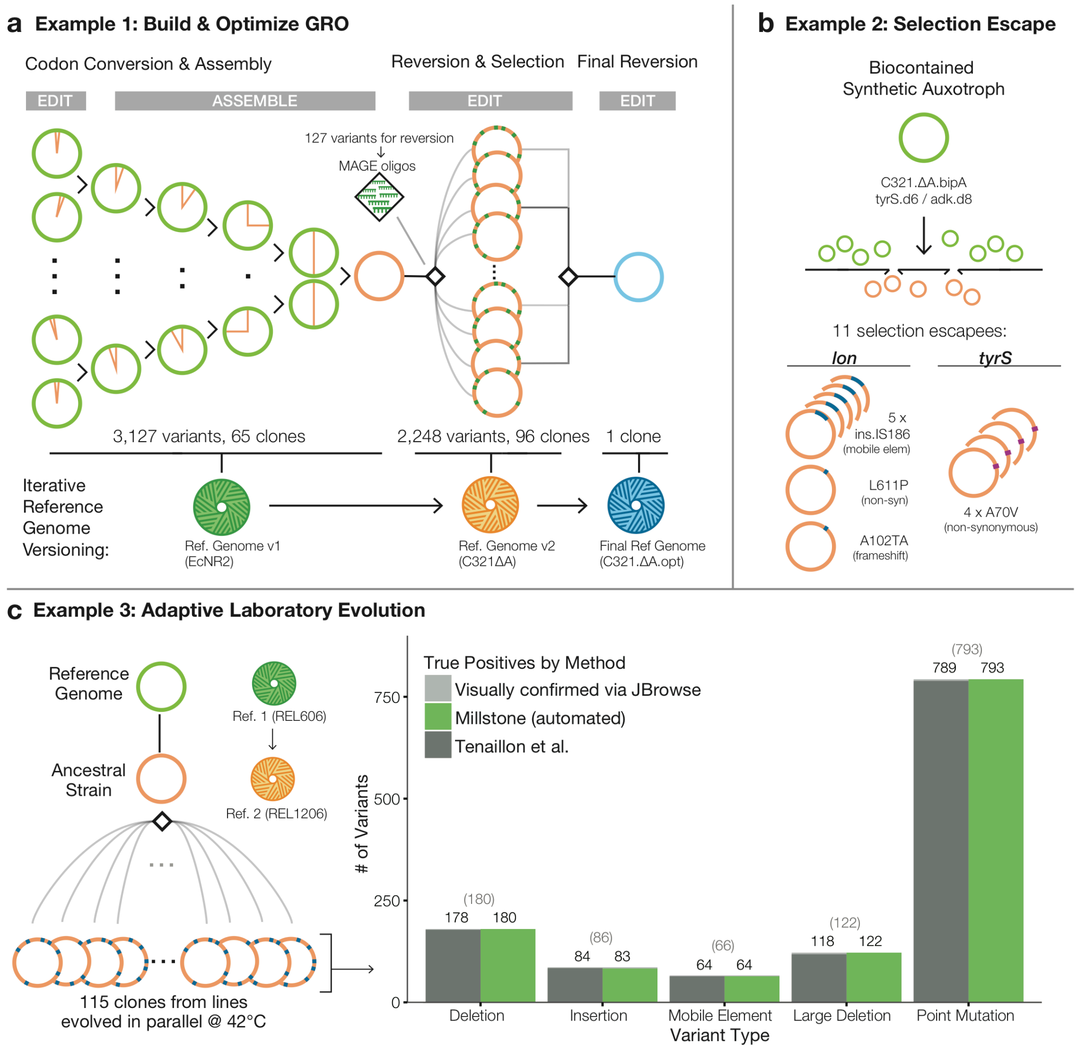


Figure 4.2: Millstone accurately detects genomic variants and can iteratively version genomes. (**a**) Millstone was used to analyze genomic clones involved in generating and rationally optimizing a genomically recoded organism. MAGE[26] and CAGE[25] were used to generate the *C321.ΔA* strain of *E. coli*[4]. With sequencing data from these strains, Millstone confirmed the designed mutations, identified and annotated off-target mutations, and generated a new reference genome. Further reversion of variants was performed with MAGE to improve the strain's fitness (*Kuznetsov et al., submitted*), and a final reference genome was generated. (**b**) Analysis of 11 escapee clones from a biocontainment selection with a synthetic non-standard amino acid (nsAA) auxotrophy[7] identified two escape mechanisms, either mutation of *tyrS* or disruption of *lon.* (**c**) Millstone can also be used for Adaptive Laboratory Evolution studies. We employed Millstone to analyze mutations across 115 clones in the Tenaillon et al.[62] high temperature evolution experiment. Millstone was used to create a new reference genome for the ancestral strain from REL606, the closest available reference genome, and called variants against this new reference. Millstone reports 99.2% of SNVs, deletions, and mobile elements found by the Tenaillon pipeline, as well some not identified in the original study (Table C.2).

tations. We developed an iterative method for systematically optimizing strain fitness through predictive modeling and multiplex testing of reversions[41]. Millstone was used throughout this process: first, to rank high-effect candidates for reversion, then to design oligonucleotides for MAGE, and finally to report variants from whole genome sequencing of 96 edited clones (Figure 4.2 a, orange). Once the final subset of effective reversions was identified and used to construct a faster-growing GRO, Millstone was also used to produce a final reference genome for the improved strain.

Millstone's *de novo* assembly and genotype comparison features were crucial in a project to engineer a biocontained version of the GRO which is dependent on a non-standard amino acid (nsAA) for survival[7]. A major challenge in engineering biocontainment, and in selection more generally, is diagnosis of escape mechanisms. In *Mandell et al.*, plating of early versions of the biocontained GRO on non-permissive media revealed individual clones that could survive without the essential nsAA. We performed whole genome sequencing on 11 escapee clones and several controls and used Millstone to identify loci enriched for mutations across escapees. This led to the discovery and validation of two primary mechanisms of escape: a single off-target nonsynonymous mutation in the redesigned *tyrS* gene occurring in 4/11 clones and disruption of the *lon* protease in the remaining 7 clones. Millstone revealed several modes of *lon* disruption: a frameshift (1/7), nonsynonymous substitution (1/7), and insertion of a mobile element upstream of the gene (5/7) (Figure 4.2 b). To identify and precisely map these mobile elements and other structural variants, Millstone combines a local *de novo* re-assembly approach with coverage-based deletion calling (Figure C.3). Rapid analysis of escapee clones allowed us to identify and validate the key mechanisms of escape from biocontainment, so that further modifications lowered escape rates by at least 5 orders of magnitude (undetectable escape with detection limit 2.2e-12 escapees per c.f.u.).[7].

Millstone can also be used to analyze genomic variation in samples undergoing adaptive laboratory evolution (ALE). In Tenaillon *et al.*[62], 115 lines of *E. coli* were grown at 42 C in parallel for over 2,000 generations in an attempt to identify convergent evolutionary responses to this environmental challenge (Figure 4.2 c). This impressive effort required a custom sequencing analysis pipeline consisting of over half a dozen tools, followed by manual validation and visual confirmation of all 1331 variants. We reanalyzed the raw data from this project in Millstone and identified 99.7% of SNVs and 98.9% of structural variants and mobile element insertions. Millstone further discovered 8 SNVs, 4 large deletions, and 2 mobile element insertions that were not reported in the original work (Figure 4.2 d, Table C.2). On an Amazon Web Services EC2 instance, the entire process from sample upload to variant triage across all 115 strains took a single day (Table C.3).

4.3 Discussion

New technologies for constructing, screening, and selecting microbial genomes now allow for increasingly complex functional genomics studies and bioengineering endeavors. As the sequence constraints of the genome come into focus, the promise of designing new organisms to address medical and material challenges is becoming a reality[63]. The path forward requires rapid construction and characterization of successive versions of redesigned genomes[18,36], and computational genome design and analysis tools will increasingly become integral to this process. Researchers who already have raw sequencing data can use Millstone to identify and explore mutations. We have reduced the barrier for other labs to use Millstone by making the software deployable on Amazon Web Services

(AWS). Documentation and an online demo are available at `http://churchlab.github.io/millstone`.

5
Toward machine-guided protein engineering

Acknowledgements:

This chapter describes joint, unpublished work in progress with Surge Biswas and Pierce Ogden. The three of us have participated in all experiments and computational modeling.

In chapter 3, we demonstrated the efficacy of a model-guided approach to genome-scale optimization. Inexpensive DNA synthesis and next-generation DNA sequencing allow generating ~100-1000 whole-genome/phenotype pairs which can be interpreted by simple models to guide further engineering. Focusing on a single protein, we can further leverage the same DNA synthesis and sequencing capabilities to develop assays that measure on the order of 10^5 variants in a single experiment, thus providing an even stronger foundation for machine-guided methods to leverage.

In this chapter, I review relevant background and describe lessons learned from proof-of-concept work using machine learning to navigate the fluorescence landscape of the green fluorescent protein (GFP). The key technological question is whether and how machine learning can be effectively combined with modern DNA sequencing and synthesis capabilities to augment design and optimization of proteins. By measuring thousands of genotype-phenotype pairs for a protein and building a model predicting function from sequence, we may be able to overcome limitations of and augment structure-based design and directed evolution. The genotype-phenotype models we build serve the dual role of approximating the fitness landscapes of proteins, allowing us to gain insight into their evolution and engineering potential.

In our GFP proof-of-concept work, both modeling and experimental design have been moving targets. I have done my best to fit a report of our progress to a linear narrative, and I apologize in advance to the reader for any non sequiturs in the discussion of work so far.

5.1 Background

Proteins are molecular machines that play critical roles in cells, including catalyzing reactions, sensing, signaling, and providing structure. Over evolutionary time, nature has generated a diverse set of proteins capable of performing a wide range of functions and enabling life to exist across a broad spectrum of environments. We are still unable to build nanomachines capable of sophisticated molecular manipulations from scratch, but we can modify existing proteins to achieve therapeutic and industrial goals. In the process, we gain insight into the biophysical properties and evolutionary history of proteins that promises to broaden the space of applications from tuning these nanomachines.

The function of a protein is determined by its structure, chemical properties of its side chains, and the environment in which it acts. Abstractly, we can think of this as a function that maps a given sequence and environment to some biological activity:

$$f(\text{sequence, environment}) = \text{biological function}$$

In protein engineering, we are seeking a solution to a specific biological function in a given environment, and the challenge can be thought of as finding solutions to the inverse function of $f()$:

$$f^{-1}(\text{biological function, environment}) = \text{set of sequences}$$

The primary strategies for protein engineering include building and testing mutants guided by structure, directed evolution of existing proteins[64], semi-rational approaches that augment directed evolution with targeted mutagenesis[65], as well as advances in *de novo* protein design[66]. Prediction of protein structure and function from sequence[67] is another active area research[68], leaving protein engineering efforts limited by assay development. Most engineering objectives require identifying more than one modification and the difficulty of predicting interactions among mutations is likely a major contributor to the difficulty of protein engineering[69]. Insight gained from empirically measuring and modeling many such interactions, consequently mapping the fitness landscape of a protein, may advance the rate at which proteins can be engineered.

5.1.1 Fitness Landscapes

The relation $f(\text{sequence, environment}) = $ biological function is often referred to as the fitness landscape[70]. Because the number of possible amino acid sequences scales exponentially with protein length, obtaining an exhaustive assessment of a fitness landscape quickly becomes infeasible, even for a peptide with just ten residues.

Paraphrasing De Visser and Krug's summary of the history of conceptualizing fitness landscapes[71]: Sewall Wright painted an early picture of the rugged terrain describing the relationship between genotype and phenotype[70]. Several decades later, John Maynard Smith crystallized the importance of networks of connected paths through fitness landscapes for facilitating adaptive evolution[72]. Kauffman and Weinberger suggested one strategy for modeling the topology of the landscape concept through their NK model of tunable roughness[73]. However, there remains a disconnect between intuitive but abstract notion of a fitness landscape and interpretable models based on data. The intractable task of assaying every possible genotype means a complete empirical landscape for any protein is unattainable.

Intelligent interrogation of fitness landscapes, such as through assaying combinatorially complete sets of small numbers of mutations[74], has provided insights into the nature of fitness landscapes, and in particular into one of their most salient features: epistasis. Confirmation of the occurrence of diminishing returns, sign epistasis, reciprocal sign epistasis, and other non-additive effects of mutations confirm the richness of fitness landscapes but also reinforces the daunting challenge of a complete analytical understanding.

More recently, advances in DNA sequencing, DNA synthesis, and mutagenesis methods have made deep mutational scanning[75] a fashionable pursuit[76,77,78,79]. Notably, these high-throughput assay methods trade-off accuracy for scope and begin to reveal global features of fitness landscapes. Deep mutational scans can be used to improve engineered protein function, as illustrated by Whitehead et al. who used deep mutational scanning to identify mutations to significantly improve an influenza hemagglutinin-binding protein that had been computationally designed and optimized through directed evolution[80]. This is just one of a growing number of illustrations that fitness landscapes, whether obtained through evolution or systematic interrogation, contain information that can be leveraged to enhance protein engineering.

5.1.2 Machine Learning and Design of Experiments

Computational and statistical methods are needed to interpret data from empirical characterization of fitness landscapes, allow prediction of fitness at unobserved points, and guide further exploration of the landscape. While there are many examples in literature describing statistical methods for quantifying the effects of individual mutations and identifying epistatic interactions, methods for interpolating along and navigating a landscape are less well-characterized. Specifically, there are two challenges: 1) Identifying a modeling method that can best represent the nonlinear relationship $f(\text{sequence, environment}) = \text{biological function}$ and 2) Identifying a strategy for efficiently sampling additional points in pursuit of an engineering goal.

Previous work leveraging machine learning methods to engineer proteins have used variations of linear regression[81,82], Gaussian Process (GP) regression[83,84], and other kernel-based methods to model the relationship between sequence and function. The work by Romero et al. using GP regression with a consensus structure-based kernel worked quite well given their context of having on the order of hundreds of well-characterized mutants derived from shuffling of homologous proteins. However, being a nonparametric model, exact inference with GP regression scales with $O(N^3)$, where N is the number of training points, making them impractical for applications beyond a few thousand training points[85]. Additionally, the structure-based kernels used by GP-based methods may not capture indirect interactions among non-contacting residues.

In short, we would like a model that scales well with large datasets and can represent nonlinear interactions among any subset of residues. The methods of multilayer neural networks, colloquially known as deep learning, have seen a recent resurgence due to quick iteration enabled by commoditization of GPUs and availability of large datasets, demonstrating their remarkable ability to obviate much feature engineering and outperform previous methods in a number of tasks[86]. Neural networks are attractive in their potential as universal function approximators[87], theoretically capable of representing any nonlinear function given a sufficient number of nodes and training data, though a strong caveat of course. Deep learning has been demonstrated to outperform previous methods on a variety of tasks in the biology domain, including predicting effects of mutations on splicing[86,88], predicting protein-nucleic acid binding[89], and protein contact map prediction[90]. However, simpler methods, noteably linear regression, have repeatedly demonstrated their efficacy as a starting point for fitting data and lend themselves more readily to interpretability even while neural network approaches may be more predictive in the same domain[91]. Thus a principled exploration of appropriate complexity for every new domain is warranted.

Complementing methods for quantitatively representing fitness landscapes are methods for sampling against them. As biology remains a fundamentally empirical pursuit, there is much value to a method for intelligently filling the screening capacity of any round of experiments to balance exploration with exploitation. Strategies for doing so are formalized as methods of Bayesian optimization, as reviewed by Ghahramani[92]. Referred to by related terms of Bayesian decision theory and active learning, Bayesian optimization uses a representation of uncertainty in model predictions to guide prioritization of new observations.

The Gaussian process method used by Romero et al. comes with a built-in representation of uncertainty for new data points[83]. The authors computationally sampled possible variants relative to their model, balancing exploration and exploitation to select variants to physically create and characterize. This ultimately led to the creation of a cytochrome P450 variant more thermostable than any reported by DNA shuffling or directed evolution[83]. For deep neural networks, a variety of methods have been proposed for quantifying uncertainty[93,94].

5.1.3 Fluorescent Proteins as a Test Domain

An ideal application to develop this new methodology of high-throughput characterization of fitness landscapes would be a well-characterized protein with interesting diversity of functions and existing high-throughput assay methods. Fluorescent proteins satisfy this requirement with their diversity in fluorescent properties, including intensity and peak excitation-emission spectra[95]. And they are naturally amenable to the high-throughput screening technique of fluorescence activated cell sorting (FACS).

The family of fluorescent proteins from various aquatic creatures consist of ~238-residue polypeptide chain that forms into an eleven-stranded beta-barrel structure encasing a single helix in the center, of which three amino acids cyclize to form an excitable chromophore[96]. The most commonly used variants are those of the jellyfish *A. victoria* green fluorescent protein (avGFP), the coral-derived red fluorescent proteins, and more recently discovered *B. lanceolatum* LanYFP and its engineered derivatives. Variations in fluorescence intensity and peak excitation-emission spectra result from variations in the maturation and local environment of the chromophore and overall stability. Many mutations have been well-characterized, including those at residues 65-67 that form the chromophore. These universally consist of the amino acids XZG, where X is variable, Z is an aromatic amino acid, and G is a universally-conserved glycine. Novel protein variants continue to be reported[97,98]. These and other recent engineering efforts have typically used semi-rational approaches that combine structural knowledge and directed evolution. For our purposes, fluorescent proteins represent a balance between existing knowledge that can serve as positive control, and remaining questions that may be addressed by further interrogation of the fitness landscape.

Recently, Sarkisyan et al. reported exploration of the local fitness landscape of avGFP using FACS and next generation sequencing (NGS)[78] . The authors generated ~50,000 mutants of GFP using a single round of error-prone mutagenesis covering 90% of single mutants accessible by single-nucleotide mutations, 2% of double mutants, and vanishingly small fractions of higher-order mutants. This conceptually but rich dataset contained numerous examples of epistasis in the local vicinity of avGFP, and formed the starting point for our own investigations.

5.2 Results

5.2.1 Exploring generalization in a local fitness landscape

We began our explorations with a deeper dive into the local fitness landscape dataset of 50,000 GFP variants published by Sarkisyan et al.[78]. The authors of that study experimented with fitting several different models predicting fluorescence from sequence. Specifically, they tested a linear model ($r^2 = 0.7$), a linear model with a sigmoid function applied to the output ($r^2 = 0.84$), and a simple neural network with two hidden layers composed of one and two nodes with non-linear transformations, respectively, feeding into a final output node ($r^2 = 0.935$). The reported variances were computed by fitting each model architecture on five random iterations of 90-10 train-test splits. We reproduced these results and further experimented with more complicated architectures, but were unable to improve the fit significantly. We were surprised that a fundamentally additive model (with a non-linearity on the output) performed the best. Scaling up to models with more parameters and greater capacity may squeeze out incremental improvements to the fit data, but our ultimate purpose was to augment design.

To evaluate new designs, a machine learning model must have the capacity to generalize beyond training data. A model's potential to evaluate previously unseen data is approximated by its performance on held-out test data. However, this makes the assumption that unseen data will come from the same *data distribution* as that of the test data, which is by no means guaranteed, and often not true. A random train-test split is a natural treatment of existing data in many domains. However, if the ultimate goal is design, a random train-test split may not adequately select for a model that is capable of generalizing to novel design modes.

In the protein engineering domain, it is not immediately apparent which characteristics clearly define a data distribution. We enumerated several natural characteristics to consider: edit distance to a reference point (e.g. wild-type avGFP), phenotype distribution (e.g. dark vs bright), and identity of individual mutations. Focusing on these properties, it became clear that the published Sarkisyan dataset, generated by error-prone PCR from a single avGFP wild-type, had certain properties that by no means necessarily represent the space of possible fluorescent proteins. For example, Sarkisyan et al. observed a generally bimodal distribution in the fluorescence of their 50,000 mutants, with most mutants being near wild-type, or non-fluorescent, leaving relatively few intermediate values. Most of their mutants were within a small hamming ball (mean 3 mutations) from the avGFP wildtype. Considering edit distance together with brightness, they observed that most avGFP variants with greater than five mutations were non-fluorescent. Finally, the mutations observed covered 90% of all possible single amino acid mutations reachable by a single nucleotide mutation; however, we calculated this to be only 40% of all possible single amino acid mutations. With the exception of brightness, all of these characteristics are a direct result of the mutagenesis process used.

To further explore whether a model trained data from a local fitness landscape could generalize, we took a design approach and chose as an initial design objective to generate GFP variants that are functional but maximally distant in sequence space from the wild-type starting point. In addition to being a conceptually simple metric, edit distance can serve as a design constraint in itself. Design often requires navigating multiple constraints, and maintaining function while increasing edit distance would be a sufficiently interesting and conceptually simple objective to focus on. From the Sarkisyan dataset, we knew that this design objective was not trivial, as most mutants with greater than five mutations were no longer fluorescent. A model would have to do better than random at predicting function from sequence as a baseline. Further, the objective of maximizing hamming distance while maintaining function is not entirely dissimilar to at least one real problem in biotechnology: deimmunization of a protein therapeutic–where a drug's susceptibility to immune response must be minimized while maintaining function. As a first step, we asked how much progress experimental methods alone could make toward the design objective.

5.2.2 Mapping the local superfolder GFP landscape through Break-Fix Evolution

We devised an experimental strategy we refer to as Break-Fix evolution to enrich for diverse mutants that maintain function. The main idea is to perform mutagenesis followed by alternating rounds of positive selection for bright mutants and negative selection for dim mutants. We considered a simpler strategy of repeated mutagenesis under high-selection only. However, we were concerned that maintaining high-selection only would allow wild-type and low-order mutants to persist and dominate the library.

At this point, we decided to switch to the superfolder GFP[99], a descendant of avGFP differing at 14 amino acid residues and optimized over a series of studies[100,101,99] for increased brightness and stability. Our primary motivation to do this was to obtain a deep measurement of a second mode of GFP, distinct from avGFP, that could serve as a complementary dataset for training.

We decided to employ a simplified version of the Sarkisyan high-throughput assay without loss of generality toward our primary goal of testing whether machine learning can enable design in the protein domain. We limited mutagenesis to a region corresponding to only 163 of 238 amino acid residues of the full GFP (residues 39 - 201), a region spanning 489 nucleotides that would comfortably fit within 600-bp constraint set by the the longest commercially available short-read technology (MiSeq paired-end 300), with margin for the known reduced quality of the second read. Sarkisyan et al. employed a neat restriction enzyme + barcoding trick to obtain the full 714 nucleotide sequence, which added a bit of complexity to the experiment. Our chosen region entirely covers the chromophore residues (65 - 67) and most important residues previously reported in literature[96], as well as several of each type of secondary structure feature (beta sheets, turns, etc.) Further, we reasoned that in order to achieve our design objective of functional but distant mutants, it would be sufficient to train a classification model on binary data and we sorted into 2 bins after every mutagenesis round (and 4 bins for the first round to better map the immediate landscape of sfGFP). This decision had the additional benefit of expanding the library size we could analyze with the same sequencing capacity.

We performed two rounds of Break-Fix, which involved a total of four rounds of mutagenesis with FACS into 4 bins after the first mutagenesis and into just the low and high bin for the rest. The input to each round alternated between the low-fluorescence and high-fluorescence mutant library from the previous round. Cells in all bins were sequenced.

Upon processing the FACS + NGS data, we obtained a dataset of 260,314 genotypes with a binary dark or bright label. We performed several quality control sanity checks of our data, including confirming that function was disrupted by internal stop codons and mutations to known highly conserved residues[96]. We were surprised to find that the proportion of functional mutants vs edit distance was not significantly different than that attained by Sarkisyan et al. (Figure 5.1), illustrating the challenges of implementing desired constraints via directed evolution alone. A DNA shuffling approach[100] may have yielded more distance+functional diversity, and experiments are under way at time of writing.

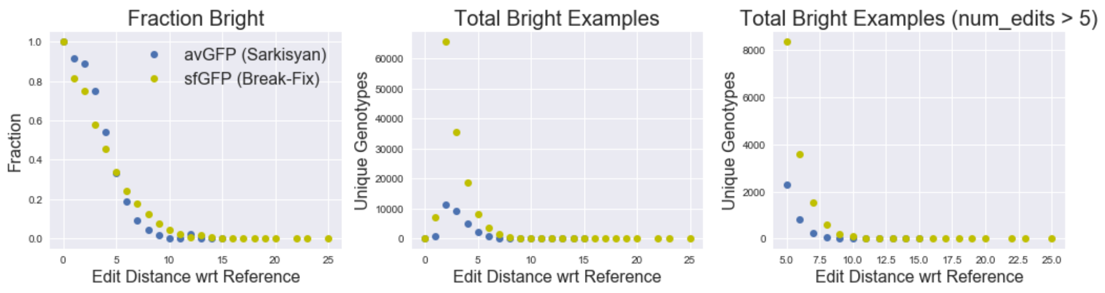


Figure 5.1: Comparison of brightness vs edit distance between the Sarkisyan avGFP (blue) and Break-Fix Evolution sfGFP (yellow) datasets. The larger number of genotypes imaged gave more bright examples within five mutations of the respective reference. However, despite the Break-Fix selection strategy intended to increase further mutants, the proportion of bright mutants at higher edit distance was not significantly increased.

Thus, we proceeded to investigating how we could use a model trained on the Sarkisyan and

Break-Fix data to design higher edit distance functional GFP mutants that simple directed evolution approaches could not access.

5.2.3 Model Comparison

To design sequences, we needed to choose a model that would generalize well beyond the local avGFP and sfGFP fitness landscapes. Using these datasets, we compared various model architectures. In addition to random splits, we experimented with different train-dev-test splits (where dev refers to data held-out from training epochs but used for model selection). Figure 5.2 shows a comparison among three models we focused on throughout this work across three dataset splits and five metrics. The three models shown are a linear model, composite residues model (to be explained later), and a fully-connected feed-forward neural network (FNN) with 3 hidden layers composed of 100-30-10 nodes respectively. For each model, we further compared two versions that differ by whether the output node activation is linear or sigmoidal, following the observation that data tended to be bimodally distributed, as noted by Sarkisyan et al.[78]. We evaluated the model on three different dataset split designs: 1) *sarkisyan_90_10*: a 90-10 random split of the Sarkisyan data, 2) *sarkisyan_breakfix*: train on Sarkisyan and test on Break-Fix and 3) *sarkisyan_epistasis*: a Sarkisyan avGFP data split where training data consists of all single mutants and doubles that fit an additive model and test dataset composed of doubles with non-additive epistatic behavior (Figure 5.3).

We chose the feed-forward neural network (FNN) with sigmoid output to sample against in the design of our initial library for synthesis. The linear model with a sigmoid output outperformed all other models on *sarkisyan_90_10* split. The FNN was competitive with the linear model on the

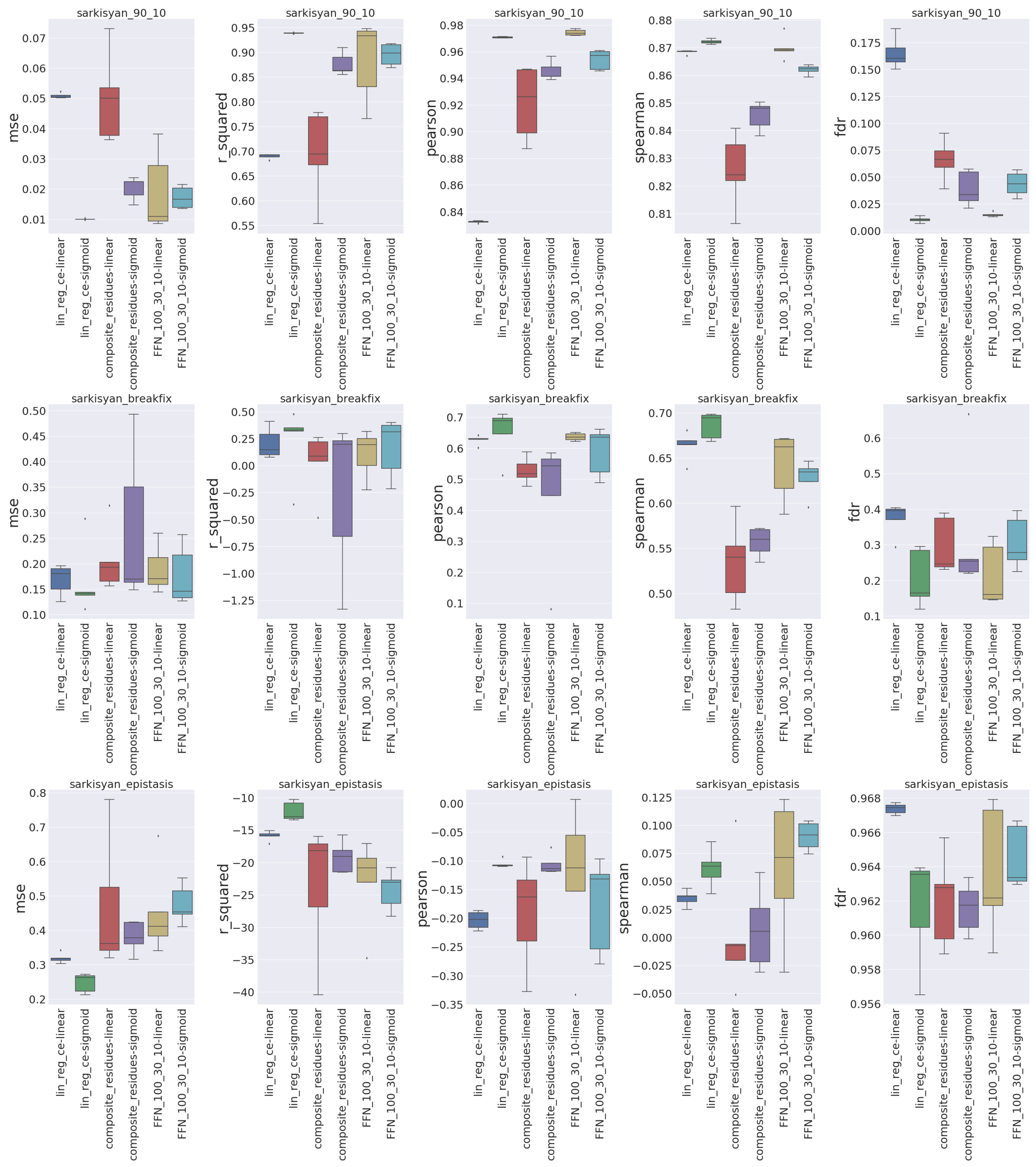


Figure 5.2: Comparison of model performance across different train-test splits. Three different models, each with two output node variations (linear or sigmoid) are applied to three different dataset splits. We ended up selecting the feed-forward neural network (FNN) with sigmoid output for our initial designs, focusing only on r^2 at the time. In retrospect, a linear model with sigmoid output was likely more justified at this point.

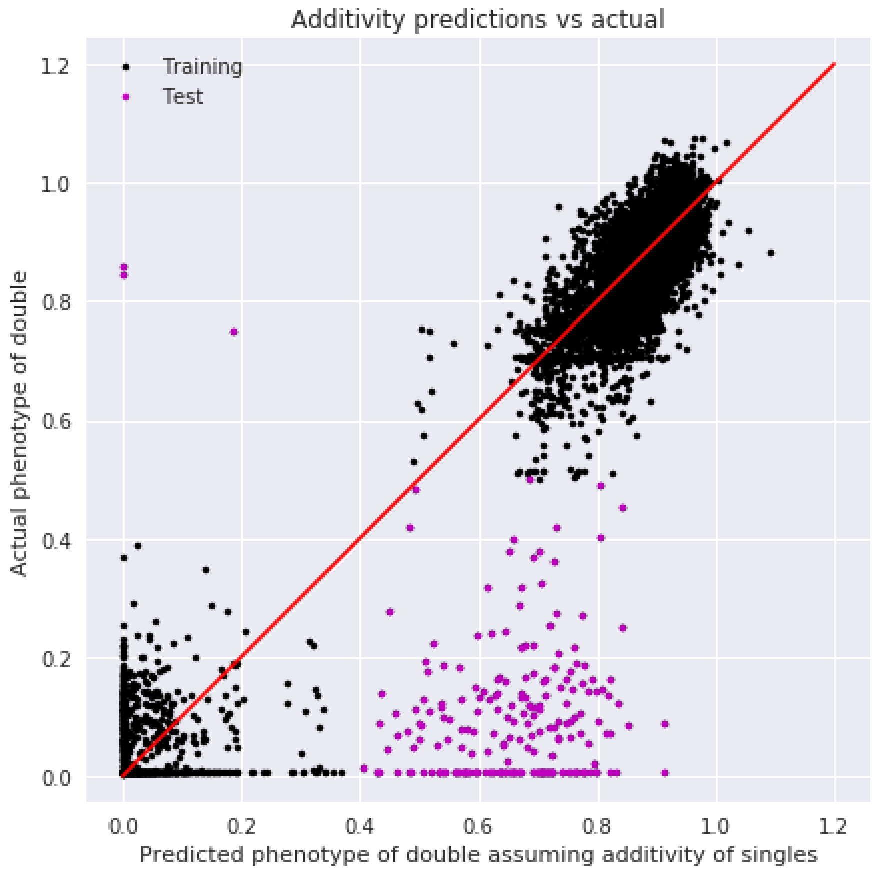


Figure 5.3: Data split for *sarkisyan_epistasis* dataset. Sarkisyan avGFP data split where training data consists of all single mutants and doubles that fit an additive model and test dataset composed of doubles with non-additive epistatic behavior.

sarkisyan_breakfix train-test split. On *sarkisyan_epistasis*, all models performed quite poorly, but the FFN weakly out-performed others according to Spearman correlation. The composite residues model did not perform well on this dataset, but would perform well on later training data. It is included for comparison.

In retrospect we found that this model selection procedure did not robustly justify selecting the FNN over the linear model, especially given lessons learned from subsequent analysis and experiments. Still, designs using this model would show interesting results.

5.2.4 Designing far but functional sequences

We proceeded with using the FNN discriminative model trained on avGFP and sfGFP data to design sfGFP variants that were distant but still functional. Because our design objective was high edit distance and maintaining functionality (but not necessarily improving on wild-type), and a large portion of our Break-Fix sfGFP data was collected in a binary mode, we trained a binary classifier using the previously chosen 100-30-10 neural network architecture.

To guide design, we had to consider the synthesis capacity for the proposed variants. The ideal output would be 10^5 full-length sequences that would fill our NGS + FACS screening capacity. However, the present price for synthesis of individual variants beyond ~200 nucleotides is \$.07 per basepair, or on the order of ~\$40 for our designs, including accessory cloning regions. Thus we were limited to synthesizing up to ~100 variants per iteration.

We asked whether we could instead leverage inexpensive microarray synthesis of ~200-mer DNA oligos (~\$0.0001 per basepair) upon which commercial full-length gene synthesis is increasingly based. A key insight was that full-length gene synthesis price is driven by the general need to isolate low-error target designs, whereas we hypothesized that our model-guided and pooled screening approach could tolerate a pooled library of near-miss variants. We devised a biased-combinatorial gene assembly approach from microarray chip oligos that would have allowed us to inexpensively synthesize a library of hundreds of variants, and some number of recombinations. However, we continue technical development on this approach and details are beyond the scope of this thesis.

In the meantime, we devised a validation pipeline to measure sequence *neighborhoods*. Synthesis of 100 full-length sequences would leave three-to-four orders of magnitude below our FACS + NGS screening capacity. To fill this capacity with interesting information, we can then perform error-prone PCR on the individual sequences and perform FACS + NGS on the resulting library of sequence neighborhoods.

We used the neural network classifier as the discriminator in a sampling strategy to generate distant variants that were predicted to be bright. Our sequence candidate design procedure was based on a simulated annealing approach. Starting with wild-type sfGFP, a random mutation is proposed. The mutated sequence is then passed through the discriminative model and a continuous score between 0 and 1 was assigned by taking the value just prior to the final sigmoid output node. This score is compared to the score of the parent sequence before the mutation, and using the Metropolis-Hastings criterion, the proposed mutation is accepted if a) its score was better than its parent OR b) a random number is above a threshold proportional to the magnitude of the score decrease. This allows for occasional negative steps through the landscape with the motivation of avoiding local minima. Repeating this procedure, a lineage of mutants is generated. We generated 1000 such lineages with terminal members having as many as 50 mutations relative to wild-type. This was more lineages than we needed for synthesis, but this would give us extra sequences to prioritize based on other metrics.

We proceeded to choose 96 candidates for full length synthesis. To prioritize these, we generated confidence intervals on the predicted brightnesses by scoring each sequence multiple times with different random settings of weight dropout at inference time[102] to make our discriminative neural network approximately Bayesian[94]. The jury is still out on whether this is a valid means of approximating uncertainty but conceptually this strategy appeared to be a reasonable criteria to prioritize sequences while addressing some of the noise in the predictions. We chose 48 sequences at least 20 mutations from sfGFP predicted to be bright with 95% confidence using this method. We also sampled 48 sequences from the same lineage with approximately half as many mutations from wild-type, and also predicted to be bright with high confidence.

The 96 synthesized sequences were received, individually cloned, and visualized on a plate (Figure 5.4). We found that 43 maintained some level of fluorescence at the GFP peak excitation. Outsourced full length synthesis turnaround time is typically less than one week from Twist at time of writing, allowing rapid feedback initial feedback on designs.

We used error-prone PCR with the designed variants as inputs to generate libraries of design *neighborhoods*. We grouped the 96 designs into six pools according to brightness in order to simplify library generation and allow enriching for assaying functional neighborhoods. The elements of the neighborhoods were then quantified for fluorescence by FACS + NGS. During FACS, we sorted cells into four equal-sized bins evenly distributed along the log-GFP dimension with the boundary between the two top bins aligned with the center of the wild-type sfGFP distribution. From bucketed NGS reads, we inferred fluorescence for a given genotype using the distribution of observations genotype among the different bins to infer fluorescence, as in[78]. The continuous measurements of individual proteins would allow us to build regression models that could be used to improve quantitative brightness. The resulting dataset consisted of 308,548 unique genotype-phenotype quantitative measurements, though many were singletons. Unlike the Sarkisyan and Break-Fix datasets,

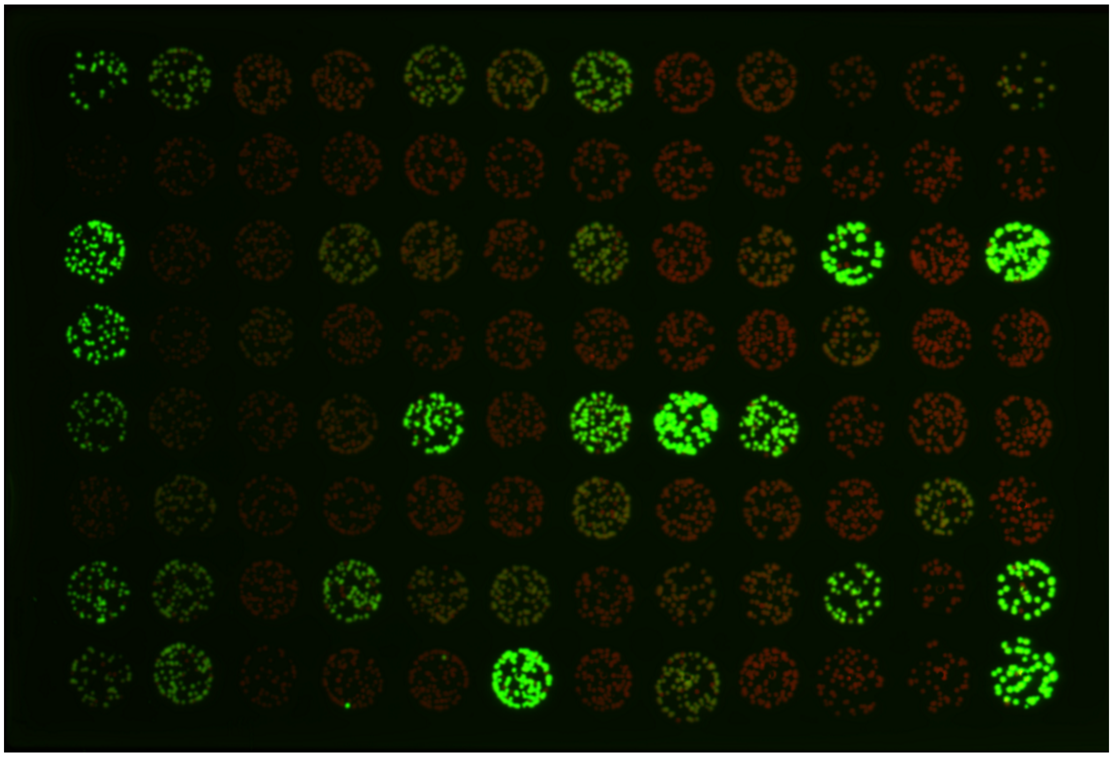


Figure 5.4: 96 Sequences Designed by Simulated Annealing The 96 sequences designed by simulated annealing with respect to the discriminative model and optimized for increasing edit distance while maintaining function. These were cloned into separate backbones and imaged at both GFP and mCherry excitation/emission settings and the images overlaid. 43 maintained GFP function at a detectable level.

fluorescence measurements were more evenly distributed across different quantitative levels, which would have interesting implications for modeling.

5.2.5 Assessing model-predicted designs

Figure 5.5 shows that while the fraction of sequences maintaining function falls off with edit distance, the functional mutants still covered a range of edit distances, including a functional mutant 39 mutations removed from wild-type sfGFP.

Inspecting the functional mutant with 39 mutations, we found that many of the mutations

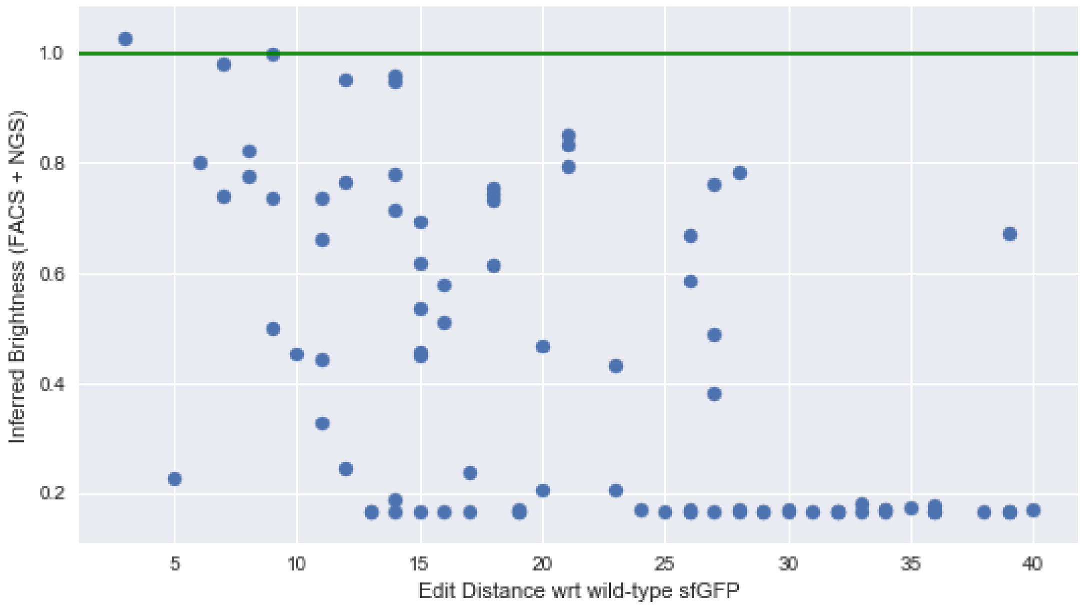


Figure 5.5: Brightness vs edit distance for model-designed variants Green line indicates wild-type fluorescence. Fluorescence was inferred from FACS+NGS with 4 bins.

resided on the outer barrel. We found that outer barrel mutations were enriched and inner barrel mutations were depleted in the designs (Figure 5.6). This matches general *a priori* intuition about the GFP protein, where the inner barrel environment is important for the formation and maintenance of the chromophore environment. As our design generation process optimized for distance while maintaining function, it is likely that many of the mutations were neutral.

To more generally quantify the triviality of a mutation at a given position, we returned to the Sarkisyan and Break-Fix mutagenesis datasets and calculated the entropy at each position from the distribution of amino acids observed at the position across the dataset. Specifically, we chose a low brightness threshold to distinguish dark and bright mutants and calculated the positional entropy within each set. We then took the ratio of bright to dark and interpreted this as the independent

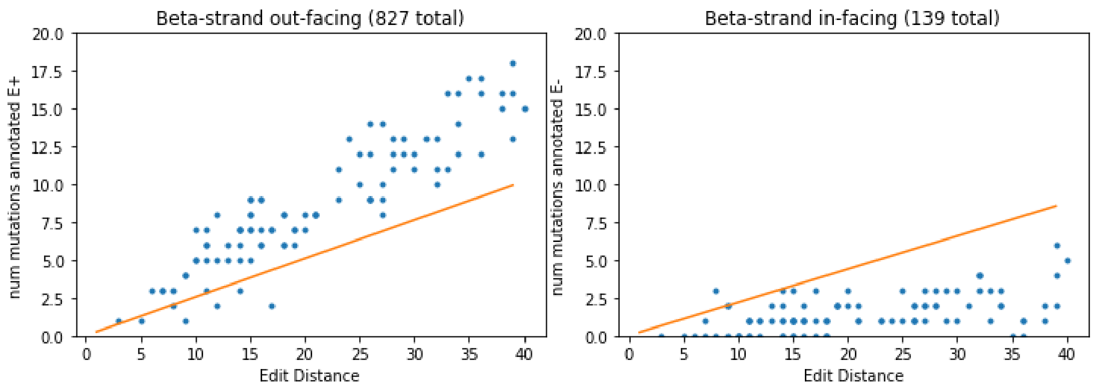


Figure 5.6: Outer barrel mutations enriched and inner barrel mutations were depleted. Each of the dots corresponds to one of the 96 designs. The orange line indicates the number of mutations of the respective type if mutations were chosen by chance, with the slope representing the fraction of positions annotated by the respective outer or inner barrel annotation for secondary structure.

permissibility of mutability of a given position in the protein. We found that the positional entropy ratio was correlated between the Sarkisyan and Break-Fix datasets (Figure 5.7).

This analysis of positional triviality hints at a deep principle for exploring protein fitness landscapes. Positions with low positional entropy ratio between bright and dark are independently recalcitrant to mutation. At present, we are designing experiments that further stress test these nontrivial positions.

5.2.6 Optimizing brightness using a regression model

Using the dataset of continuous fluorescence measurements from the neighborhoods of the initial 96 machine-designed sequences, we compared regression models of various architectures that predict quantitative brightness from sequence (Figure 5.8). Notably, the neighborhood data was not distributed bimodally as in the Sarkisyan and Break-Fix dataasets and instead was clustered around

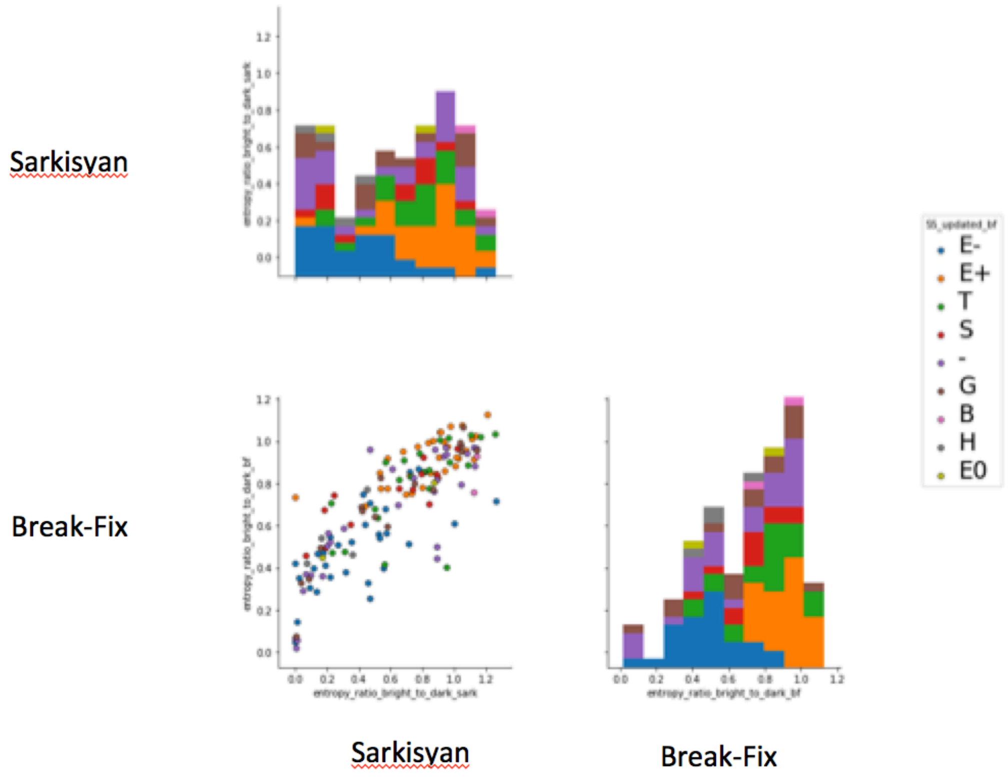


Figure 5.7: Positional entropy ratio as a quantified measure of mutability. Sarkisyan (avGFP) vs Break-Fix(sfGFP). Bottom right compares the datasets directly and shows positional entropy is correlated between the two GFP modes separate by 14 mutations. Colors annotated in legend indicate the secondary structure annotation for the respective residue (H: α-helix, B: Isolated β-bridge, E+: outer-barrel, E-: inner-barrel, E0: barrel orientation undefined, G: 3-10 helix, I: π-helix, T: Turn, S: Bend, -: Other). Upper left and bottom right show the distribution of position entropies, colored by secondary structure annotations.

modes covering a range of brightnesses between dim and wild-type sfGFP brightness. The three datasets were 1) *nbrhoods4_90_10*: a 90-10 random split of the 96-designs neighborhoods data, 2) *nbrhoods_tr_dev_sark_test*: train and dev on neighborhoods data and test on Sarkisyan and 3) *nbrhoods_tr_sark_dev_test*: train from neighborhoods data and dev and test from Sarkisyan. We used the last dataset to make the final model decision because the Sarkisyan data was a sufficiently different mode from the training data and serve as proxy for assessing generalization.

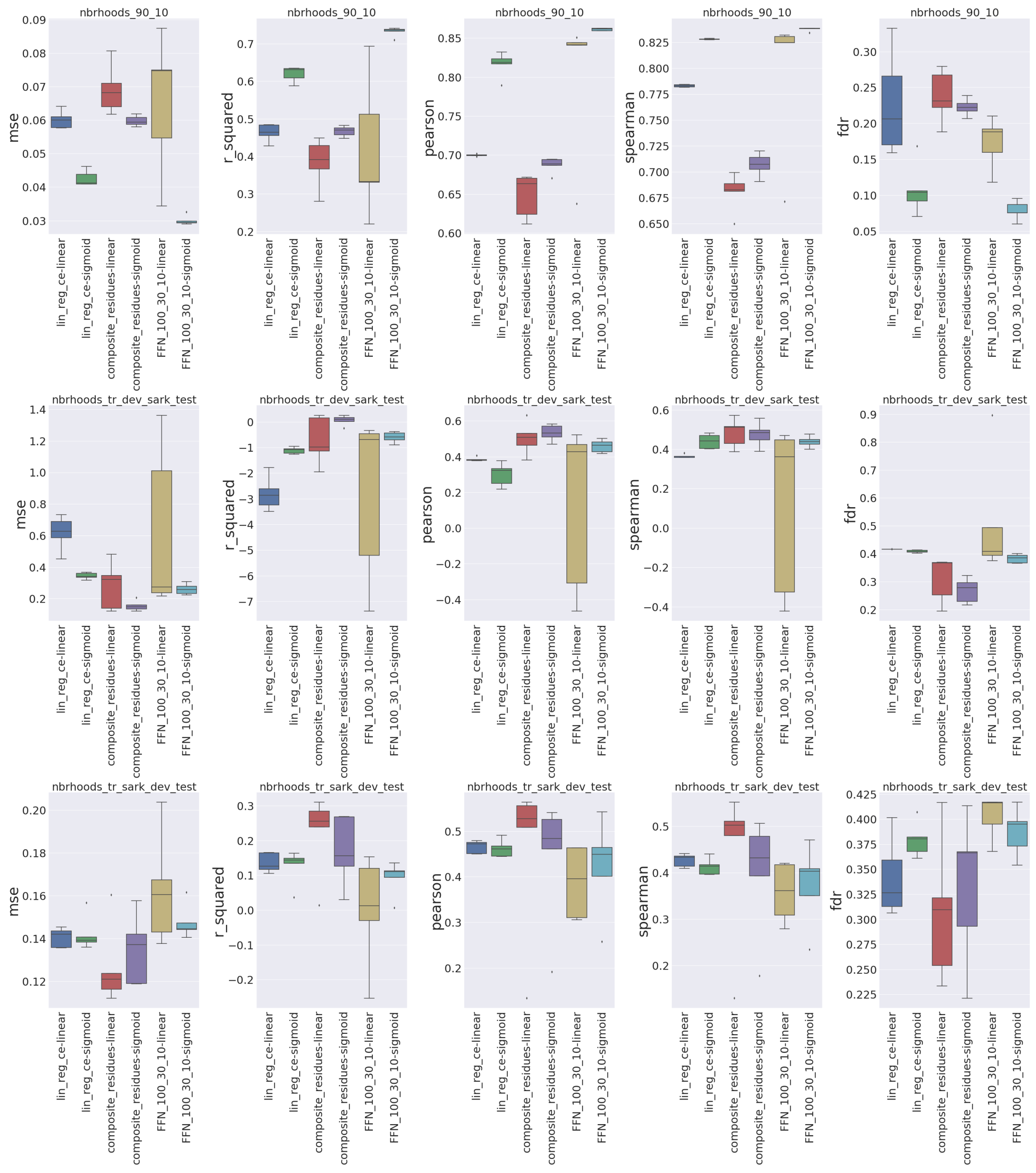


Figure 5.8: Comparison of model performance on designed neighborhoods measurements. The same models are compared as in Figure 5.2. We selected the composite residues model with with linear activation for further design as it showed the highest predictive performance and lowest FDR.

Here we found that the optimal model was what we refer to as the *composite residues* architecture (Figure 5.9), which captures the prior that certain residues should be considered together when any of them are mutated. A permutation layer contains weights that represent learned associations among these residues. We found that a linear activation on the output node was optimal, which matched the observation of the data being more uniformly distributed from low to high as compared to the bimodal distributions in the Sarkisyan and Break-Fix datasets, where a sigmoid activation on the output performed better.

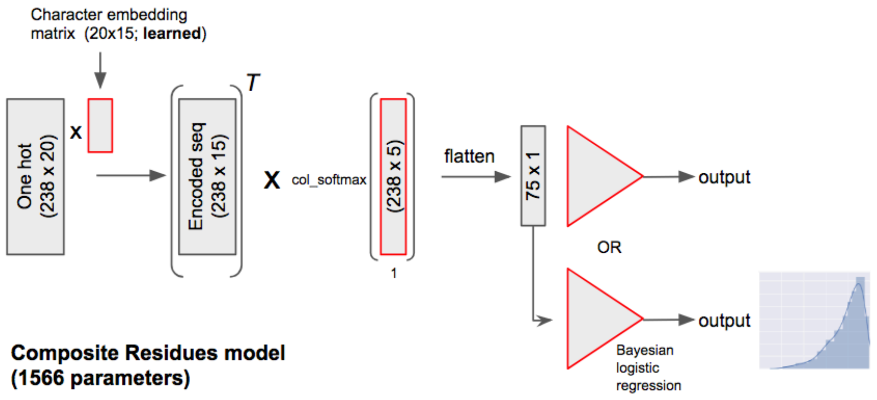

Character embedding matrix (20x15; **learned**)
One hot (238 x 20)
x
Encoded seq (238 x 15)
T
X
col_softmax
(238 x 5)
1
flatten
75 x 1
output
OR
output
Bayesian logistic regression
Composite Residues model (1566 parameters)

Figure 5.9: Composite residues model architecture. We designed the composite residues model architecture to capture the prior that certain residues should be considered together for mutation. The key component is a permutation layer that learns which subset of residue interactions to consider most strongly for prediction.

We observed that the composite residues model performance varied depending on the numerical random seed used during training. We trained ten iterations of the model from different seeds and tested several ensemble models, specifically comparing a mean, median, and min ensemble. While the mean and median ensemble performance as assessed by Pearson R was best, the min-ensemble false-discovery rate (FDR) was significantly lower. We thus proceeded with the min-ensemble of ten instances of the composite residues model. We continue to explore variations on ensembles and boosting.

We used the model to generate sequences using two general strategies: 1) The simulated annealing approach for the first set of designed variants and 2) A greedy search algorithm that sequentially picks the best model-predicted mutation to make at every step. As seed sequences, we used wild-type sfGFP as well as nine other bright sequences from the neighborhood exploration of the first design set. We designed 70 sequences optimizing for brightness. We also designed 24 sequences that were optimized for further maximizing distance from wild-type sfGFP.

The synthesized sequences were transformed and imaged (Figure 5.10). The fluorescence of the brightest clones was quantified (Figure 5.11) and found to be near wild-type, achieving the optimization objective over the first round of designed sequences that maintained fluorescence, though it was reduced relative to wild-type (Figure 5.5). Several clones were measured to have fluorescence above that of wild-type sfGFP.

5.3 Discussion and Future Work

In the present work, we explored how high-throughput mapping of protein fitness landscapes can be employed to learn design principles and optimize protein function. We performed our study with the green fluorescent protein (GFP) where a well-established FACS + NGS assay can be used to measure 10^5 genotype-phenotype pairs per experimental cycle. We found that a model trained on local mutagenesis data could be used to design functional sfGFP mutants having up to 39 mutations from the wild-type. We mutagenized and measured the local neighborhoods of 96 designed variants. A regression model trained on this data was used to design variants verified to be close to and

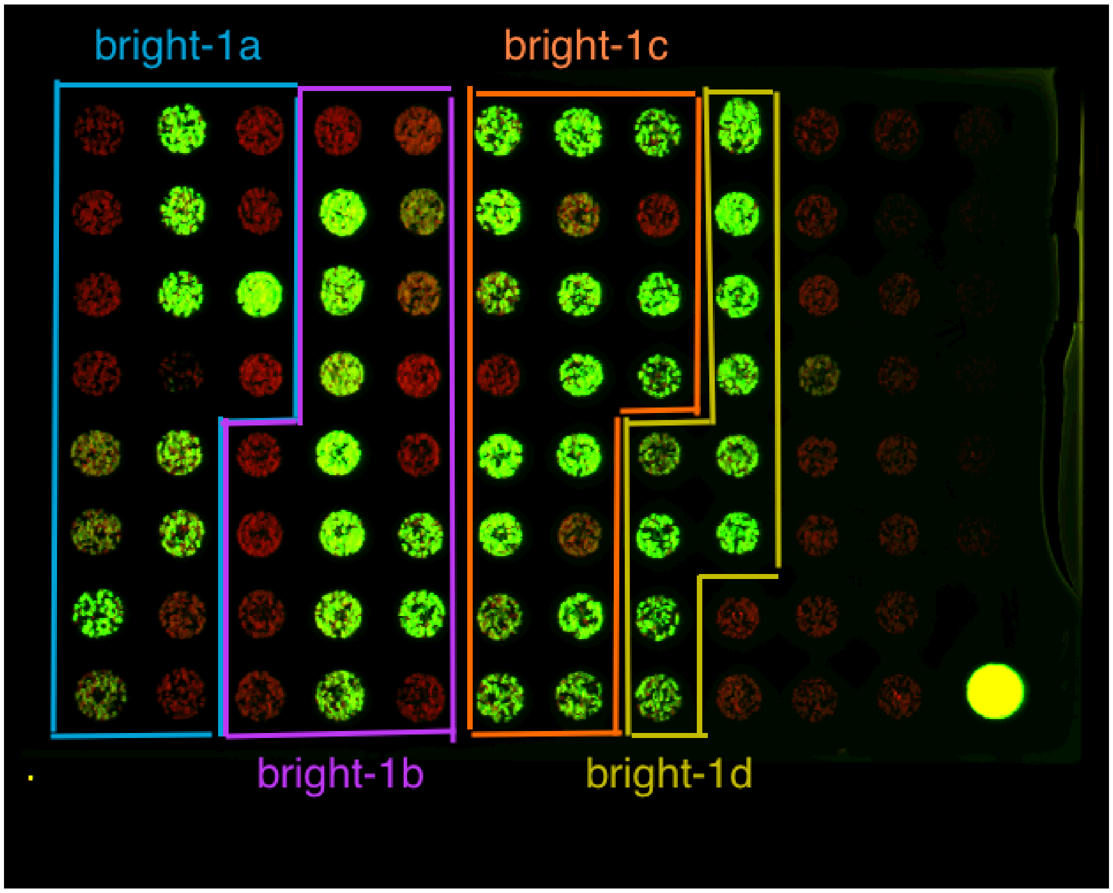


Figure 5.10: 94 Sequences Toward Optimizing Brightness We designed 94 sequences using simulated annealing and greedy optimization. This image shows the overlay of GFP and mCherry channels. The control in the bottom right was at a different OD and is not comparable in this figure. *bright-1a*: 20 sequences (2 per seed) ED = 15 from seed by 1000 parallel searches that terminate at 15 mutations. *bright-1b*: 20 sequences (2 per seed) ED = 15 from seed using kernel reward attracting to ED = 15 and allowing 4000 steps so toggles some mutations under termination *bright-1c*: 20 sequences by greedy search (saturation at each step and take best step) 10 ED = 15 from seed (pick brightest predicted) 10 ED = 5 (same lineages; conservative hedge). *bright-1d*: 10 sequences edit distance 5 brightest from each set 1a (conservative hedge).

sometimes exceed the reference sfGFP brightness.

Our work is motivated by the need to systematize traditional protein engineering methods of directed evolution and structure-guided mutagenesis. Screening capacity is one of the biggest limitations of protein engineering. Random or targeted mutations are often restricted to a narrow space

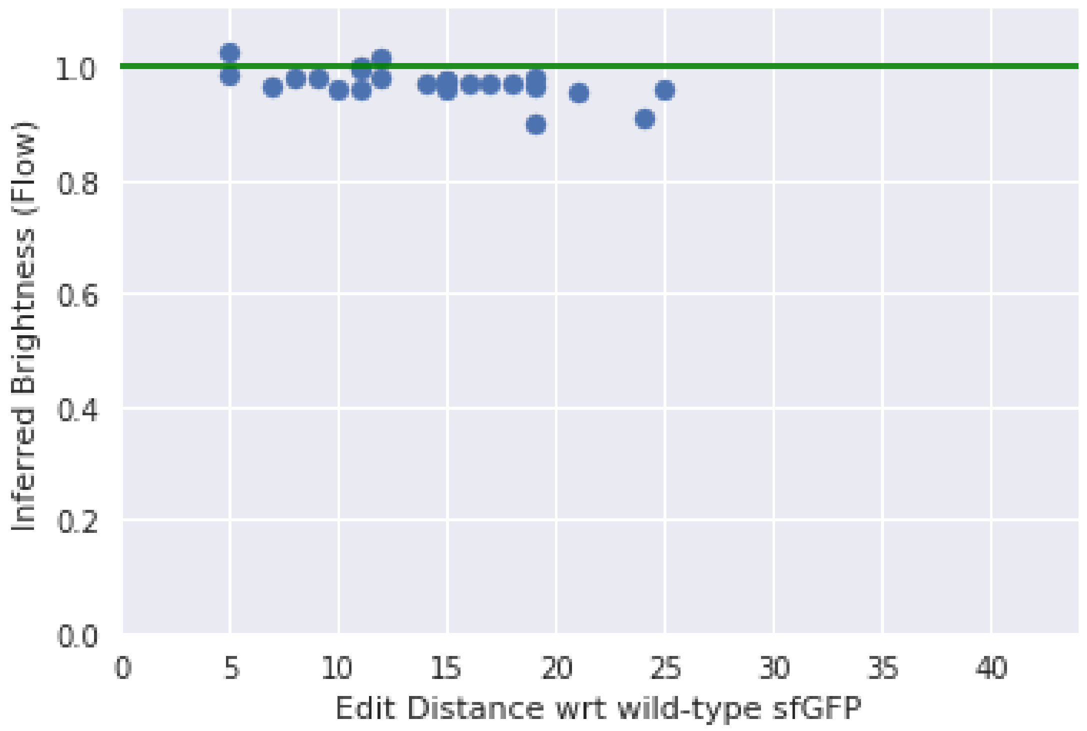


Figure 5.11: Quantified brightness for optimized clones. Green line indicates wild-type fluorescence. Fluorescence was inferred from FACS+NGS with 4 bins. **Right: Distribution of edit distance among mutants with non-zero green fluorescence.**

in order to ensure capacity is filled with mutations with a high prior expectation. With increasing synthesis and sequencing capacity, it becomes possible to leverage iterative experimentation to empirically design mutagenesis while optimizing for screening variants, neighborhoods, and trajectories predicted to be bright.

5.3.1 Design principles for machine-guided protein engineering

Our results so far hint at several design principles for engineering proteins guided by machine learning. First, we showed the efficacy of bootstrapping on an initial random mutagenesis followed by high-throughput characterization. While limited to a local exploration, this data can reveal design constraints and serve as medium for training initial models. From the Sarkisyan avGFP data and Break-Fix sfGFP data generated by variations of error-prone PCR, we were able to train a model that successfully generated functional design removed by twenty more or mutations, where nearly all mutants above ten mutations from error-prone PCR were non-functional.

Another design principle that emerges is the remarkable efficacy of simple addictive models at the begging of iteration. Our success at achieving the far-but-still-functional design objective may be owed to the broadly additive nature of many mutations, which applies strongly to many residues in GFP and may also apply to other proteins. An additive model can be a surprisingly effective predictor. We observed that a linear regression model with a sigmoid output performed best on various train-test splits and combinations of the Sarkisyan avGFP data and Break-Fix sfGFP mutagenesis data. Previous engineering work has demonstrated the efficacy of model that don't take into account interaction terms, although with smaller data sizes[82]. We were surprised that even with greater amounts of data it is difficult to beat a linear model without interaction terms.

However, adding model complexity can also be beneficial as the training data distribution shifts from local perturbations to different distributions enabled by design. Using the neighborhood measurements of the initial designed set of 96 variants, we found that the composite residues model, intentionally architected to capture the prior that the model should learn to consider certain residues together, had better performance. Throughout this work, we came to understand the significant role of the dev dataset in model selection and more generally the central role of designing data distributions in machine-guided protein engineering.

5.3.2 Designing data distributions

The protein engineering domain allows the design and testing of thousands of variants in a single experiment. This control over the design of data distributions is a unique and powerful aspect of this domain and perhaps more important than the design of model architectures. Directed evolution can find solutions when they are local to the starting point, navigating a space of points sharing a similar distribution. However, as edit distance from the reference starting point increases, new idiosyncrasies of the distant space arise. We found that a linear model does well predicting the effect of combinations of mutations in the local space, but the more complex composite residues model, which takes into account the importance of the residue proximity in three-dimensional space, was more effective for more distant modes. However, the ability to train models of increasing complexity in turn relies on the availability of appropriately held-out data. Thus designing data distributions is both challenging but also a great opportunity.

Methods for generating data variation carry their own nuances and biases. Error-prone PCR from a single starting variant explores a specific sliver of the possible fitness landscape. This is generally limited to amino acid mutations within a single nucleotide mutation and few functional mutants are observed beyond a hamming ball of ~5 mutations from the starting point. While an error-prone dataset can be effective for revealing permissiveness of individual positions to mutations, there are many other experimental perturbations that can yield interesting data. DNA shuffling allows diverse variants to be combined. Selection allows evaluating library numbers beyond screening capacity. For particular residues of interest, targeted mutagenesis can be employed. And of course synthesis of specific variants. These techniques have been employed for decades and now with the availability of inexpensive sequencing, they become further amenable to a model-guided approach. Beyond generating individual sequences that a model would like to explore or exploit, it is possible to predict the fitness of entire neighborhoods or trajectories. Thus a library of variants can be designed that is predicted to have a high chance of containing interesting mutants and whose information content would further accelerate the model's ability to discern desirable sequences.

Reiterating, the core value of being able to design experiments that collect specific data distributions is the ability to craft dev sets that facilitate effective model selection. There is yet significant work to be done to systematize methods for design of experiments.

5.3.3 Multi-objective optimization

Proteins as medicines or industrial tools must not only be optimized for their specific task, but must also act safely and reliable in their destined environments. As therapeutics, proteins must be limited in toxicity, while also having favorable pharmicokinetic properties. Optimizing for these early in the development pipeline before getting to expensive clinical trials is a tremendously valuable proposition of much interest in the field[103].

Selecting for multiple properties using directed evolution is particularly difficult. A serial development strategy may be employed where an initial library is pruned based on performance against a first objective. And the remaining members are screened for performance against a second objective. The intersection of sequences that satisfy both objectives may be very limited. A machine-guided approach could be used to separately learn predictive models for performance in the separate assays.

Then the models could be used to generate candidate sequences that are predicted to be optimal in both assays. The proposed intersection library could be screened both separately in a combined a assay. I hypothesize that a model-guided approach may be essential to generate the diversity needed for a successful screen for a molecule satisfying multiple objectives in many applications.

In the context of our GFP proof-of-concept, our foray into multi-objective optimization was limited to maximizing edit distance while maintaining function: only one of these objectives was assay-limited. A next step in our work is developing the capability to optimize against multiple assay-determined objectives. Mastering this capability may allow us to solve problems where evolution alone would fail.

5.3.4 On to other proteins

While we have developed our proof-of-concept using GFP, the methods and design principles extend broadly to other protein systems. Any protein problem that can be expressed as a sufficiently high-throughput assay coupled to next-gen DNA sequencing can be augmented by our machine-guided approach. Limitations and challenges are determined by the difficulty of engineering the assay and the size of the DNA space.

In assay development, there is typically a trade-off between throughput and quality of individual measurements. Machine learning methods may be able to leverage noisy data that is correct on average. Thus rather than collecting individual quantitative measurements, relative enrichment or pooled values may be enough to provide hints for subsequent targeted exploration.

The DNA space is constrained by technological limitations of DNA sequencing and synthesis.

Next-generation DNA sequencing can provide billions of reads but is currently limited to ~600 nucleotides. Molecular biology tricks, including the clever use of barcodes, can be used to push these limitations. Inexpensive DNA synthesis at scale is currently limited to ~200 nucleotides. The price of longer synthesis is in part driven by the need for screening for near-perfect clones. High-throughput genotype-phenotype mapping assays may tolerate and even benefit from imperfect libraries, allowing leveraging imperfect longer synthesis.

A

Supplemental Information for Chapter 2

Table A.1: Genome Design Rules

Category	Fig S5.#	Rule	Motivation	Implementation
Biological	A	Fix gene overlaps: Perform minimal synonymous codon swaps required to properly recode both overlapping genes.	Forbidden codons may fall in the overlapping region of two genes. Sometimes it may be possible to remove forbidden codons through synonymous swaps alone. In other cases, in order to avoid introducing non-synonymous mutations or disrupting regulatory motifs such as ribosome binding sites (RBS), it is necessary to separate the genes first so that codons in each gene can be replaced independently.	Use synonymous codon swaps (Genbank annotation: *adj_base_ov*) to avoid introducing non-synonymous changes in overlapping genes.
				Use computational RBS motif strength prediction (*38*) to maintain RBS motif.
				In short gene overlaps, attempt to minimize editing, for example reduce 4 nucleotide overlap to 1 nucleotide
		If necessary - separate by duplicating overlapping regions [202 instances]		If minimal overlap fix does not preserve RBS motif, separate the overlap by copying the overlapping sequence and 15-20 base pairs upstream, to preserve native RBS (Genbank annotation: *fix_overlap*)
		Reduce homology between duplicated regions through	To separate overlapping genes we duplicate the sequence, creating two tandem	Perform synonymous codon swaps in

		non-disruptive shuffling of copied region	paralogous regions. These two paralogs have the potential to recombine spontaneously which could cause a disruptive change in either the upstream or downstream gene. We attempt to prevent this spontaneous recombination by shuffling the codons of the upstream paralog, thus maintaining the native nucleotide sequence of the N-terminus of the downstream gene and 15-20 bases upstream. This region has shown to be important for mRNA folding and translation initiation	copied regions to reduce homology while maintaining regulatory motifs. (Genbank annotation: *adj_base_ov*)
	B	Preserve 5-prime mRNA secondary structure of genes	Gene expression is affected by mRNA secondary structure	Use thermodynamics-based secondary structure prediction (*37*) to compare mRNA free energy (ΔG) of wild-type and recoded sequence. Minimize ΔG change across 40 bp windows centered at modified codons.
		Preserve GC content	Related to DNA stability, mRNA secondary structure.	Maintain GC content when choosing among alternative codons. Minimize ΔGC across 40 base pair windows centered at modified codons.
		Rebalance codon usage	Preserve codon usage bias for remaining 57 codons in order	Ensure selection of alternate codons

			to preserve expression dynamics that are dependent on aa-tRNA availability.	is consistent with global distribution of codon choice; both for recoding and heterologous expression.
Synthesis	C	Remove repetitive (REP) sequences [132 instances]	We found REP regions to be over-enriched in DNA fragments that failed the repetitiveness metric for commercial synthesis and/or failed during synthesis. Hypothesizing that these REP elements were used as transcriptional terminators, we replaced REP sequences with synthetic transcriptional terminators.	Replace each REP sequence with unique terminator sequence drawn from orthogonal set(*57*). Note that not all REPs were deleted as some were tolerated for DNA synthesis. (Genbank annotation: *rep_to_term*)
	D	Remove restriction sites needed for synthesis [AarI: 972 instances, BsaI: 182 instances, BsmBI: 954 instances]	DNA synthesis vendor constraint.	Disruption of restriction enzyme motifs using synonymous codon swaps. (Genbank annotation: *adj_base_RE*)
				Preserve functional RNA (e.g. rRNA) secondary structure when necessary(*58*). If outside of coding regions, change single nucleotides to avoid disrupting annotated regulatory motifs. (Genbank annotation: *adj_base_RNA*)

	E	Remove homopolymer runs [158 instances]	DNA synthesis vendor constraint: remove sequence of more than 8 consecutive A, C, T or more than 5 consecutive G	In coding sequence, we performed synonymous codon swaps. In intergenic sequence, minimal nucleotide changes were performed that avoid disrupting annotated regulatory motifs. (Genbank annotation: *adj_base _hp*)
	NA	Rebalance GC content extremes	DNA synthesis vendor constraint: 0.30 < GC < 0.75.	If coding sequence contains very high/low GC content, use synonymous codon swaps to normalize GC content. (Genbank annotation: *adj_base_GC*)
				If intergenic sequences contains high/low GC content, introduce minimal nucleotide changes to avoid disrupting annotated regulatory motifs. (Genbank annotation: *adj_base_GC*)

	F	Partition genome into 87 50-kb "segments" at operon boundaries	We make sure to avoid splitting operons so that segments remain modular and can be redesigned independent of each other. (See Manuscript and Suppl. Methods for choice of segment size).	Allow ±5-kb variability in segment size to find partitioning that keeps whole operons together. (Genbank annotation: *segment*)
	G	Partition each "segment" into ~15 synthesis-compatible fragments of 2 to 4-kb with 50 bp overlaps between adjacent fragments	We used 2 to 4-kb as the primary synthesis unit, as offered by vendors. 50 bp overlaps enable homologous-recombination-based assembly in *S. cerevisiae.*	Choose partitioning to minimize secondary structure at 50 base pair overlaps to maximize success rate in yeast assembly. (Genbank annotation: *synthesis_frag*)

B

Supplemental Information for Chapter 3

B.1 Supplementary Figures

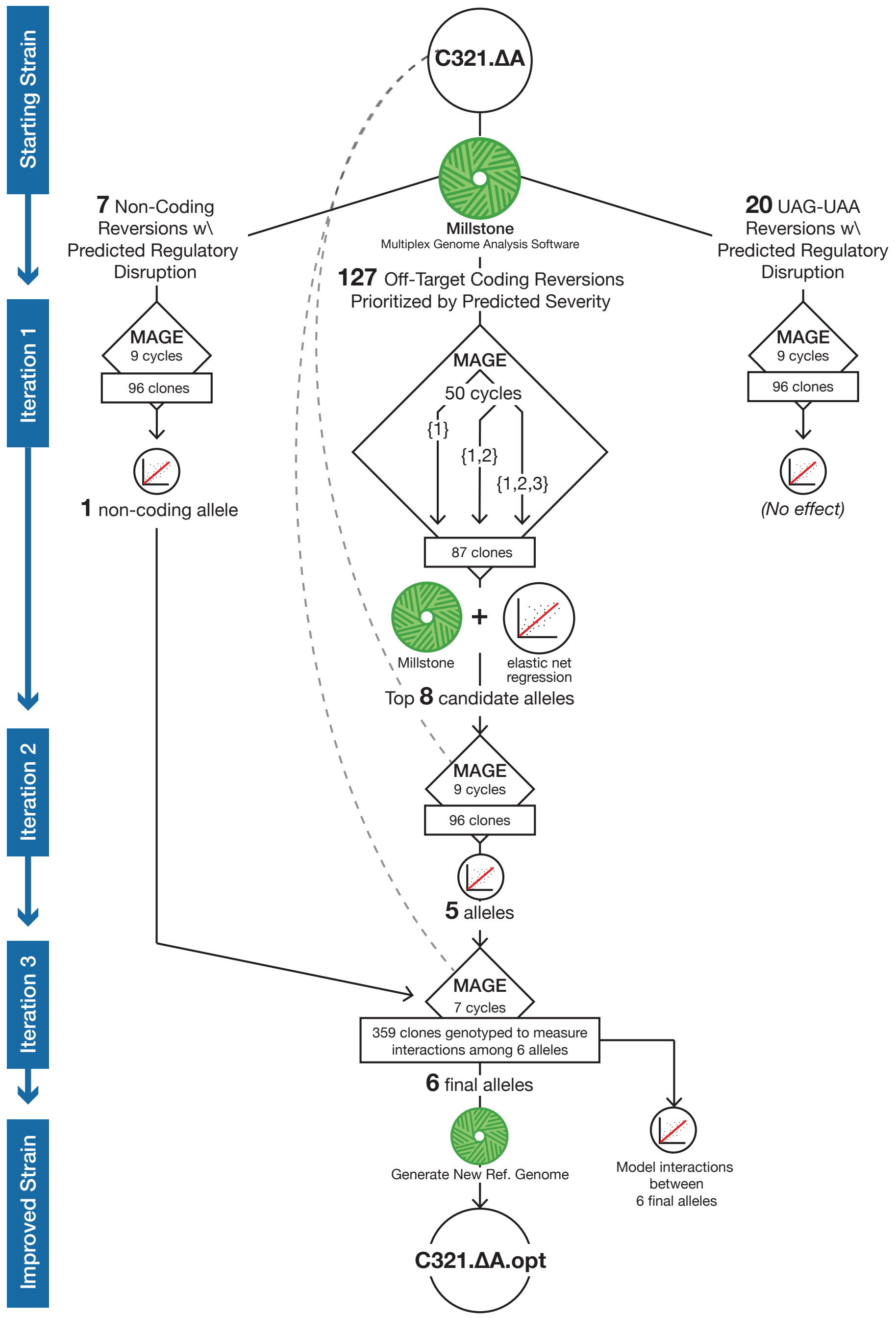


Figure B.1: Detailed Experimental Workflow Depiction of the specific steps used to identify the six alleles that optimized the fitness of C321.ΔA. Millstone (Goodman et al., submitted) was used to annotate mutations in C321.ΔA. 127 prioritized coding mutations were tested in C321.ΔA over 50 cycles of MAGE in 3 lineages. Eighty-seven clones were genotyped by whole genome sequencing and annotated using Millstone, and their doubling times were measured. Modeling by multiple linear regression identified 8 alleles for subsequent validation. After the second iteration, 5 alleles were chosen, with 3 alleles having a significant linear model coefficient and 2 more reversions having subtle effects. In a parallel experiment, a small pool of 7 non-coding mutations was tested, and modeling identified one allele that was found to have a strong effect. In another parallel experiment, 20 UAA-to-UAG reversions were tested on a C321 background with prfA still present, but no reversions were found to affect fitness. The top 6 alleles were combined in a final optimized strain, and clones with intermediate combinations of alleles were used to characterize their interaction effects (Fig. 5).

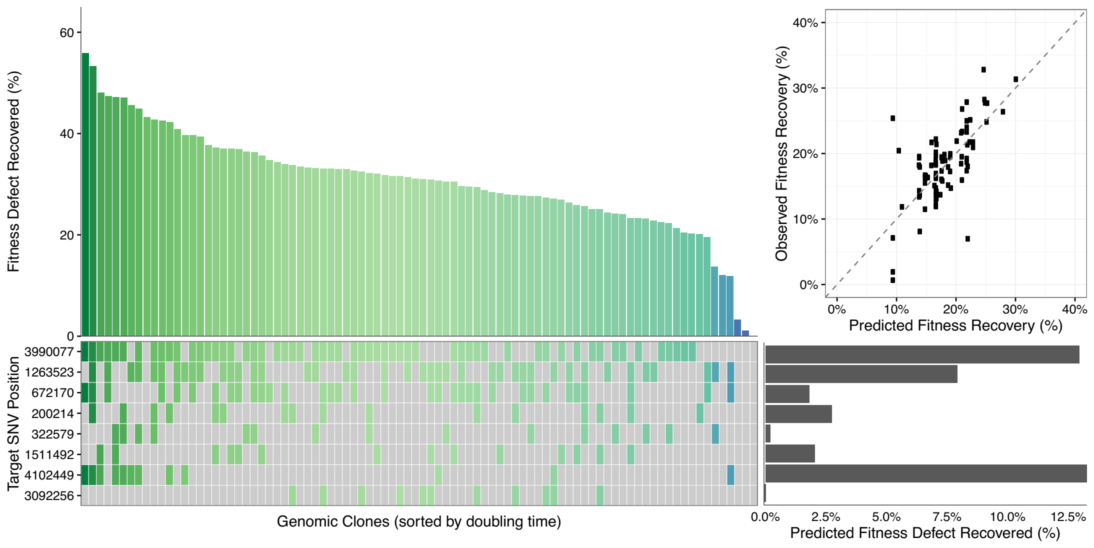


Figure B.2: Empirical testing to validate top eight alleles from 50-cycle MAGE experiment. The top eight alleles (Additional file 4) were tested in the original C321.ΔA background using nine cycles of MAGE. We selected 96 clones from the final population and measured doubling times and performed MASC-PCR to assess genotypes. Clones are sorted by fitness on the x axis and alleles are listed on the y axis in order of enrichment. Linear modeling revealed a strong predicted effect for reversions hemA-T1263523C and cpxA-A4102449G and de novo mutation in cyaA-C3990077T, with weaker predicted effects for reversions leuS-C672170T and bamA-C200214T and de novo mutation T1511492C. For construction of the final strain, we chose to keep the three high-predicted-effect alleles (hemA-T1263523C, cpxA-A4102449G, cyaA-C3990077T) and the two weak-predicted-effect reversions (leuS-C672170T, bamA-C200214T), but we omitted the three weak-effect de novo mutations.

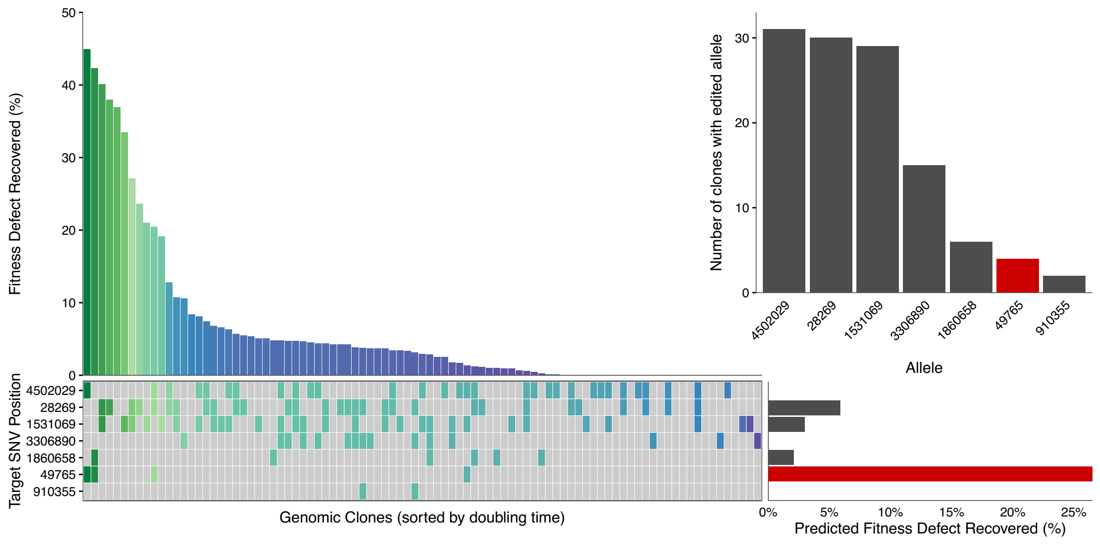


Figure B.3: Empirical testing identifies high-effect non-coding mutation. Genotypes and fitness from testing a set of seven non-coding mutations (Additional file 5). Upper right: the top model-selected allele is not apparent from enrichment alone, and it is later experimentally validated to have a significant effect (Fig. 4).

B.2 Supplementary Note 1: Further discussion of allele effect modeling and feature selection.

Adaptive laboratory evolution (ALE) typically uses enrichment of mutated genes observed across replicate lineages evolved in parallel to select meaningful features [33,34]. We selected multivariate linear modeling regularized by elastic net as an alternative to enrichment as initial tests suggested that it was a better method for predicting SNPs that recovered fitness in our experiments. For some alleles, high model coefficients corresponded to high levels of enrichment. For example, the reversion of mutation hemA-T1263523C had the highest enrichment after 50 cycles of MAGE (occurring in 78 out of 87 clones) and was also selected by modeling for validation, eventually being verified to confer fitness improvement (Fig. 3). On the other hand, when testing the pool of seven non-coding reversions, the single allele selected by the model folA-C49765T (also later experimentally validated) occurred in only 4 out of 96 clones. Meanwhile, three other alleles occurred in over 25 out of 96 clones, but they were not predicted to have a strong effect by linear modeling (Supplementary Fig. 3). There were many other cases where linear regression assigned low coefficients to alleles that were highly enriched. The discrepancy between enrichment and model-predicted effect may be due to differences in MAGE oligonucleotide recombination frequency [35], insufficient time for mutations to achieve enrichment, or stochastic enrichment of passenger mutations in a lineage during MAGE cycling. As noted in Methods, we observed that alleles with weaker effect were not consistently reported and were dependent on the choice of randomized train-test split, an issue that we expect to be remedied by sequencing additional clones and further tuning of the exact implementation of multivariate linear regression with respect to alleles with weak effect. Altogether, however, we found that regularized multivariate linear modeling provided an effective strategy of predicting fitness effect for individual alleles.

We compared our multivariate linear regression strategy to univariate linear regression (Additional file 11). Treating any Bonferroni-corrected p-value of < 0.05 as a reported effect, we found that the two modeling strategies agreed in six of the eight alleles chosen for validation following the 50-cycle MAGE experiment. Based on our simulations (Supplementary Fig. 4), we expect imperfect recall and precision for both elastic net and the univariate model. However, the elastic net model is predicted to generally perform better in both recall and precision, including at the parameters of our experimental design. Thus, we chose to pursue the targets identified by our elastic net regularized multivariate regression strategy. Further, the diminishing returns observed in our analysis of partial strains (Supplementary Fig. 5) suggests that the majority of effects had been recovered in the six SNPs found. We did not pursue independent validation of the additional mutations reported by the univariate analysis.

An important consideration with multivariate linear modeling is whether to include higher-order interaction terms. For our 50-cycle MAGE experiment, we made an assumption that independent allele effects would dominate relative to complex epistatic effects and included only first-order terms in the model. For validation in pools of <= 10 oligos, we typically selected 96 clones and experimented with second- and higher-order models, but also found that the first-order model was typically sufficiently informative of allele effect.

To investigate how higher-order model terms can inform interpretation of epistatic effects, we assembled a dataset of 359 intermediate genotyped clones obtained from validation experiments or from screening during construction of the final strain with all six top alleles (Fig. 5). Interestingly, linear modeling with second-order interaction terms indicated evidence of possible diminishing returns epistasis among certain alleles [36] and also a possible positive epistasis effect between cpxA-A4102449G and cyaA-C3990077T. Alleles that contribute to fitness through a positive epistasis effect could be lost during validation of small numbers of alleles, supporting our validation of high impact alleles in pools.

Even with targeted engineering by MAGE, de novo mutations can play a role in fitness improvement. The mismatch repair-deficient context in which this study was conducted elevates the background mutation rate >100-fold [37] and resulted in the accumulation of four de novo mutations for every reversion (Fig. 2f). We considered de novo mutations in modeling experiment data, but omitted any de novo mutation that was never observed in more than one clone, reducing the number of features corresponding to de novo mutations from 1329 to 135. Linear regression of data obtained over 50-cycles of MAGE identified four de novo mutations with a putative effect. Validation of these alleles determined that three of these were false positives or only beneficial in a specific context. The fourth de novo cyaA-C3990077T, however, showed a strong effect upon validation, demonstrating that linear regression can be an effective strategy for identifying causal de novo mutations and may be generally applicable in laboratory evolution studies.

Additional features beyond allele occurrence can be added to the linear model such as terms that capture prior expectation of an allele's effect. For application to ALE, mutations could be merged according to affected gene and its interacting partners. Higher order terms can be iteratively introduce as the candidate feature set is pruned.

B.3 Supplementary Note 2: Discussion of alleles chosen to construct final strain.

The final strain was constructed by introducing six mutations into the starting C321.ΔA background: five alleles reverted to their MG1655 starting point and one de novo mutation not previously present in MG1655 (Additional file 7).

Four of five reversions (bamA-C200214T, leuS-C672170A, hemA-T1263523C, cpxA-A4102449G) were coding mutations in essential genes prioritized in the highest category of 27 mutations in the 50-cycle MAGE experiment, supporting the strength of the initial prioritization method. The fifth reversion (p-folA-C49765T) was identified in screening noncoding off-target mutations predicted to disrupt gene regulation. Though we did not consider mutations outside of coding regions in our initial prioritization, the later consideration and validation of such a causal mutation demonstrates computational prediction of regulatory disruption as an important strategy in tuning organisms [8]. The sixth mutation was the de novo mutation C3990077T that arose in the background of MAGE-cycling, coding for cyaA (adenylate cyclase), a non-essential gene that nonetheless impacts fitness upon knockout (Keio knockout survival = 0.324). If designed, this mutation would have been prioritized in category 1 (Additional file 2).

We characterized intermediate genotypes created while constructing the final strain (Fig. 5) and determined that three of the mutations (reversions of p-folA-C49765T, hemA-T1263523C and de novo cyaA-C3990077T) had especially strong individual effects. Two reversions (leuS-C672170A, bamA-C200214T) had weaker individual effects that diminished in backgrounds with multiple mutations (Fig. 5b). The last reversion cpxA-A4102449G did not have a strong effect alone, and may even have been slightly detrimental, but appeared to provide a benefit in the presence of cyaA-C3990077T (Fig. 5a). To our knowledge, none of the gene identities or relationships reveal a first-order explanation for the findings of epistasis. However, both cyaA (C3990077T) and cpxA (A4102449G) are implicated in stress response pathways [38] [39].

We suspected that the de novo cyaA-C3990077T is a beneficial suppressor in the C321.ΔA background but not in the non-recoded background. Testing the mutation in EcNR1.mutS.KO revealed a minor detrimental effect on fitness, increasing doubling time by 2.94

C

Supplemental Information for Chapter 4

C.1 Supplementary Figures

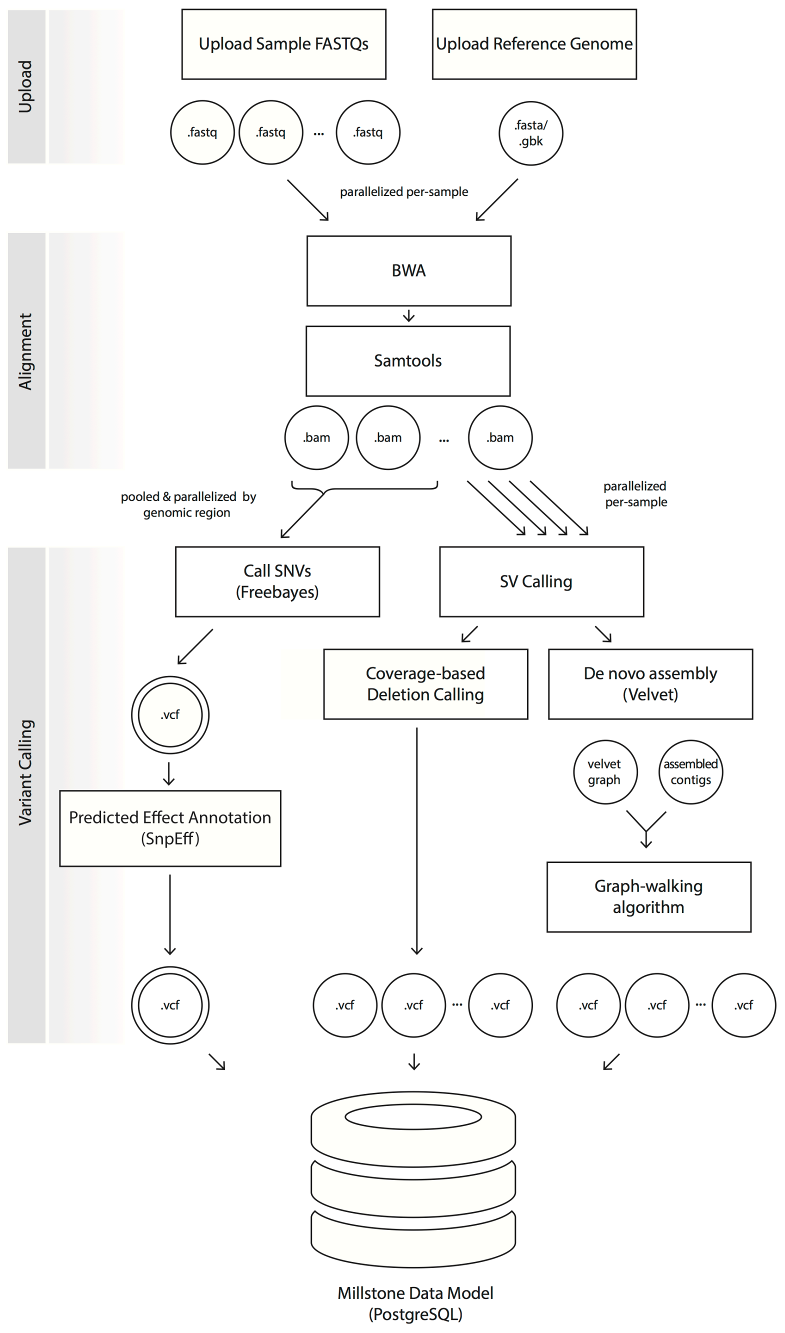


Figure C.1: Millstone Analysis Pipeline. The analysis pipeline efficiently automates the process of identifying single nucleotide variants (SNVs) and structural variants (SVs) from user sample data and storing the information in a data model representation that can be explored using Millstone's variant exploration UI. Once sample FASTQs and a reference genome are uploaded, Millstone uses BWA[104] to align samples to the reference, parallelizing alignments across available processor cores. Once all alignments are complete, Freebayes performs SNV-detection on all .bam files simultaneously, parallelizing across regions of the genome. SVs are identified in parallel in individual samples using two methods: 1) Deletions are detected using sequencing read coverage and 2) novel junctions that indicate insertions and rearrangements are identified using *de novo* assembly of unmapped reads using Velvet[105] followed by a custom graph-walking algorithm to combine assembled contigs and alignment with BWA to place contigs in the genome. All variant callers report their data in the Variant Call Format (VCF) and Millstone parses the VCFs into a single data model representation. Millstone further uses read coverage to identify regions of the genome with poor mapping quality and automatically adds variants that fall into such regions to appropriate VariantSets.

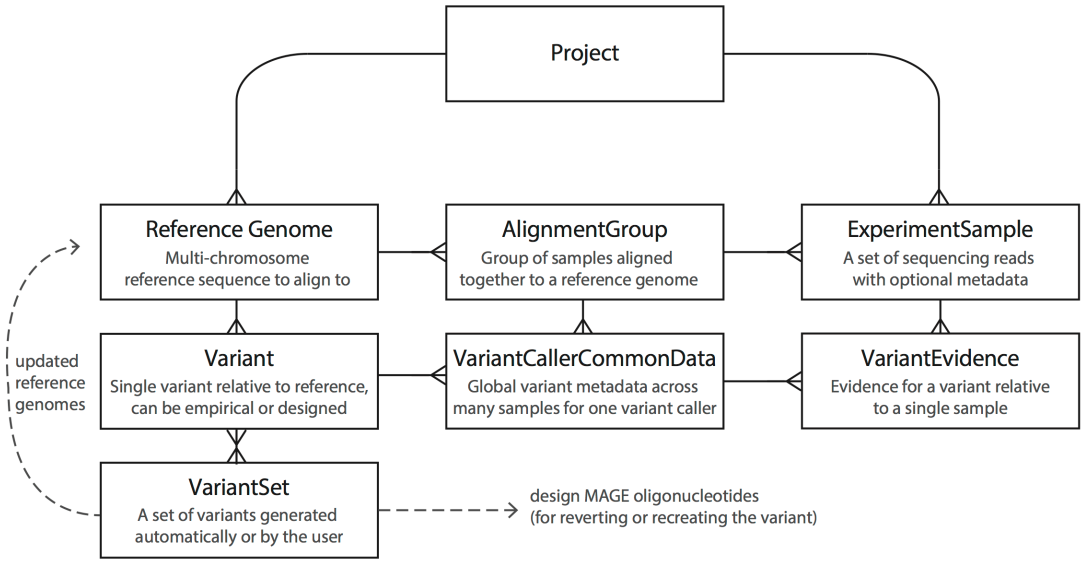
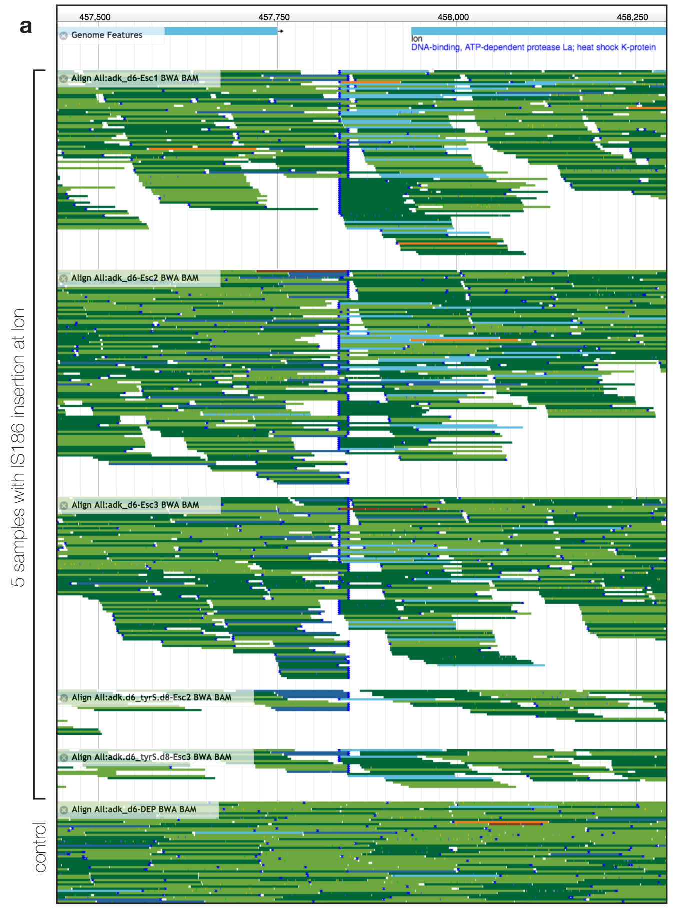
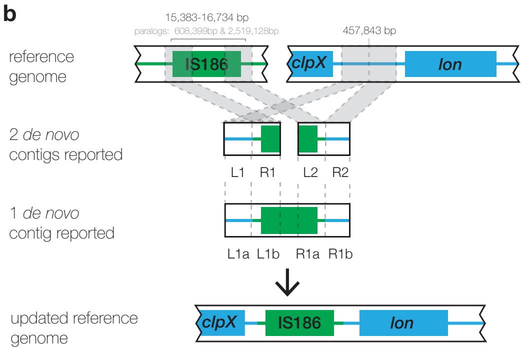

Project
Reference Genome
Multi-chromosome reference sequence to align to
AlignmentGroup
Group of samples aligned together to a reference genome
ExperimentSample
A set of sequencing reads with optional metadata
Variant
Single variant relative to reference, can be empirical or designed
VariantCallerCommonData
Global variant metadata across many samples for one variant caller
VariantEvidence
Evidence for a variant relative to a single sample
VariantSet
A set of variants generated automatically or by the user
updated reference genomes
design MAGE oligonucleotides (for reverting or recreating the variant)

Figure C.2: Millstone Data Model. Millstone's data model was designed to enable project organization, data storage, and to support researcher operations including uploading data, running analysis pipelines, exploring the resulting data, and generating actionable outputs. Users upload *ReferenceGenomes* (e.g. Genbank or FASTA genome sequences) and *ExperimentSamples* (e.g. FASTQ) to a Project. The *AlignmentGroup* model stores data from an alignment of multiple *ExperimentSamples* against a specific *ReferenceGenome. Variants* represent both user-specified designed mutations and those emprically identified by variant callers. *Variants* are the most important primitive in Millstone, and serve as the unit of operation for analysis and design tasks. *Variants* are defined relative to a specific *ReferenceGenome*. The *VariantCallerCommonData* model relates a given Variant to any *AlignmentGroups* the mutation was called in and stores metadata provided by the variant calling tool (e.g. Freebayes). The *VariantEvidence* model further stores evidence for the occurrence of a specific Variant in a specific ExperimentSample. *VariantSets* allow the user to group Variants and take actions on groups. The VariantSet concept is very similar to tags in other software contexts and a Variant can belong to more than one VariantSet. Operations enabled by VariantSets include filtering in the exploration view, exporting subsets of variants, printing MAGE oligos, and generating new versions of reference genomes. The full data model is declared in the source code: `https://github.com/churchlab/millstone/blob/master/genome_designer/main/models.py`.

a


457,500
457,750
458,000
458,250
Genome Features
lon
DNA-binding, ATP-dependent protease La; heat shock K-protein
Align All:adk_d6-Esc1 BWA BAM
Align All:adk_d6-Esc2 BWA BAM
Align All:adk_d6-Esc3 BWA BAM
Align All:adk.d6_tyrS.d8-Esc2 BWA BAM
Align All:adk.d6_tyrS.d8-Esc3 BWA BAM
Align All:adk_d6-DEP BWA BAM
5 samples with IS186 insertion at lon
control

b



c

Sample	Contig #	Mean Coverage	Contig Size	Left/Right Junction	Genome Position	Gene
adk.d6-Esc3	1569	45.9	360	L1	457848	*lon*
	1569	45.9	360	R1	16733	*IS186*
	673	38.8	532	R2	457835	*lon*
	673	38.8	532	L2	15388	*IS186*
adk.d6-Esc2	571	76.0	2145	L1b	608398	*IS186*
	571	76.0	2145	L1a	457848	*lon*
	571	76.0	2145	R1a	607053	*IS186*
	571	76.0	2145	R1b	457835	*lon*
adk.d6-Esc1	8	38.8	1784	L1b	608399	*IS186*
	8	38.8	1784	L1a	457848	*lon*
	8	38.8	1784	R1a	607053	*IS186*
	8	38.8	1784	R1b	457836	*lon*
adk.d6/tyrS.d8-Esc3	76	9.2	608	L1	457841	*lon*
	76	9.2	608	R1	15385	*IS186*
	2	10.3	481	L2	2519646	*IS186*
	2	10.3	481	R2	457835	*lon*
adk.d6/tyrS.d8-Esc2	4	3.6	236	R1	2519128	*IS186*
	4	3.6	236	L1	457405	*clpX*
	4	3.6	236	L1	456777	*clpX*
	14	6.6	575	L2	16733	*IS186*
	14	6.6	575	R2	457848	*lon*

d

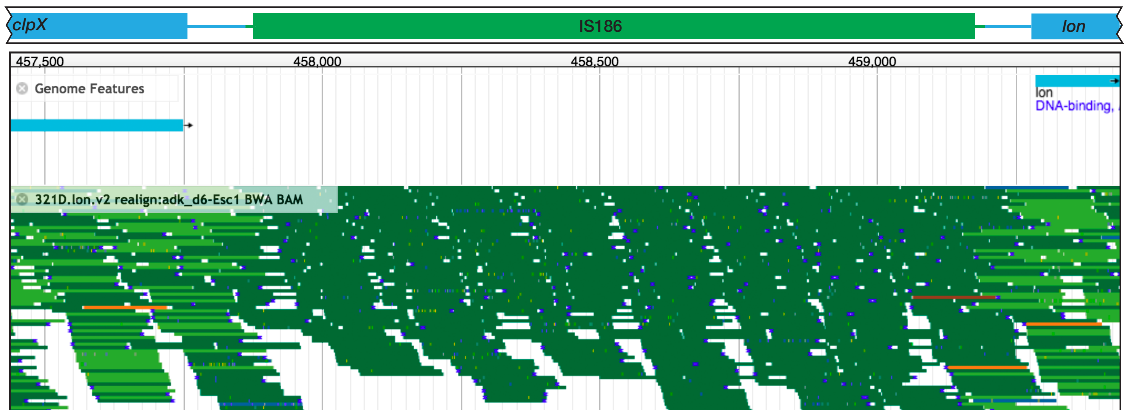


Figure C.3: Millstone uses *de novo* assembly to identify a mobile element insertion at the *lon* locus across 5 escapee clones from Mandell *et al.* **(a)** Millstone's integration with JBrowse shows evidence for a disruption upstream of the *lon* gene for 5 escapee strains. Each colored line represents a single read, and the reads are grouped vertically by strain. A wild-type dependent strain is shown at the bottom for comparison. Darker reads map to the forward strand, and lighter reads map to the reverse strand. Properly paired reads are shown in green, reads with an unmapped mate are shown in blue, and reads with improperly paired mates are shown in orange. The dark blue vertical lines at the center of this locus denote split reads, indicating a disruption in alignment. **(b)** Millstone performs *de novo* assembly followed by alignment of assembled contigs back to the reference, and then uses a graph traversal algorithm to identify large insertions. Millstone identifies either one or two contigs for the insertion of the IS186 element into the *lon* locus. Finally, Millstone generates an updated reference genome which reflects the insertion. **(c)** A table of contigs, their sizes, and multiple reference alignments for each of the 5 samples with an IS186 insertion. **(d)** A new JBrowse view with the the updated reference genome. The split and mismapping reads are gone, revealing a clean alignment in the region with the inserted IS element. The dark region indicates reads which map to multiple IS186 paralogs across the genome.

Feature	Millstone	BreSeq	SPANDx	Galaxy
Free & Open Source	•	•	•	•
Effect Prediction	•	•	•	•
Variant Visualization	•	•		
Multiple Sample Comparison	•		•	
Interactive Querying	•			
Structural Variant Detection	•	•	•	
Genome Versioning	•			
Easy Deployment / Install	•	•		•
Genome Editing	•			
Sharing / Collaboration	•	•		•
Supports Paired-End Data	•		•	•
Modular Pipeline				•

Table C.1: Comparison between Millstone and Other Tools. A tabular comparison of features between different whole-genome sequencing tools, with a focus on those that are meant for use with haploid microbial genomes and are scalable to large datasets. All tools listed here are free and open-source. A more detailed description of the differences between the features and limitations of each is provided in Supplementary Note 1. *Effect Prediction:* Prediction of variant effects based on genome annotation. *Variant visualization:* can visualize read alignments for individual variants built into the tool. *Multiple Sample Comparison:* can compare the evidence for and presence/absence of a variant across multiple samples within the tool. *Interactive Querying:* can interactively search and subset variant list based on genomic position, gene, quality, mutation type, etc. within the tool. *Structural Variant Detection:* Supports detection of longer variants not normally detected by SNV callers like Freebayes and GATK Unified Genotyper, such as insertions, deletions, and translocations longer than 50-100 bp in length. *Genome Versioning:* Capable of generating an updated reference genome based on a subset of variants found in an initial reference genome. *Easy Deployment / Install:* Can be used without command-line compilation or scripting. *Genome Editing:* Generates designs for iterative editing/reversion of selected variants. *Sharing / Collaboration:* Built-in data-sharing via the web among teams of multiple users. *Uses Paired-End Data:* Utilizes paired-end read information to generate alignments and identify structural variants. *Modular Pipeline:* Capable of custom pipelines using different user-specified modules.

	Line	Position	Ref	Alt	Millstone	Tenaillon
Large Deletions	LIne142	547700	-	71438 bp	•	
	LIne17	3512040	-	415 bp	•	
	Line106	4499185	-	15462 bp	•	
	Line4	664688	-	1443 bp	•	
Deletions	Line74	474295	`GATGGTTAATGCC`	`GC`	•	
	Line74	474295	`GATGGTTAATGCC`	`GC`	•	
Insertions	Line14	2236102	-	9_bp insertion		•
	Line79	474706	-	15_bp insertion		•
	Line106	4090527	-	25_bp insertion		•
	Line18	4089885	`TCAGTTTGC`	`TCAGTTTGCAGTTTGC`	•	
	Line58	3571137	`TC`	`TCAGTAC`	•	
Mobile Elements	Line13	3652063	-	IS150 insertion		•
	Line92	4202813	-	IS186 insertion		•
	Line132	2651161	-	IS150 insertion	•	
	Line112	4376973	-	IS1 insertion	•	
Point Mutations	Line7	2031537	`G`	`A`	•	
	Line7	2031545	`A`	`T`	•	
	Line27	798255	`T`	`C`	•	
	Line67	4195430	`G`	`A`	•	

Table C.2: Variant differences between Millstone and the original analysis performed by Tenaillon et al. We compared variants found by the Tenaillon et al. to those found automatically Millstone, focusing on Type II errors. Here we split discrepancies into 5 classes, including 3 short nucleotide variant (SNV) classes - short Deletions, Insertions, and Point Mutations, and 2 structural variant classes - Large Deletions and Mobile Element insertions. The final two columns describe 'True Positive' variants which were found by only one of the two pipelines. To identify these, we examined the read evidence using Millstone's JBrowse visualization feature and determined whether the variant was correct as called by either pipeline.

	Biocontainment	GRO	Improving GRO Fitness	Tenaillon
Number of Genome Samples	24	68	97	115
Reference Genome Size	4.6E6 (+6124 plasmid)	4.6E+06	4.6E+06	4.6E+06
Mean Aligned Reads per Sample	2.2E+06	1.2E+07	2.4E+06	5.4E+06
Alignment + SNV-calling Time	1hr 12min	5hr 10min	2hr 12min	4hr 35min
SV calling time	37min	1hr 30min	50min	30min
Variants called	1533	3127	2284	4171
Average Query Time	0.5sec	1.5sec	1.4sec	3sec

Table C.3: Benchmarking. Millstone's analysis pipeline was executed on datasets of various size from four different projects: Biocontainment[7], GRO[4], Improving GRO Fitness (*Kuznetsov et al., submitted*), and Tenaillon[62]. Average query time was calculated using no filter and a simple filter for strong alt calls: `GT_TYPE = 2`. All benchmarking was performed on Amazon Web Services (AWS) Elastic Cloud Compute (EC2) instances r3.8xlarge (32 cores, 244 Gb memory).

.

C.2 Supplementary Note 1: Cost of Multiplexed Genome Library Preparation and Sequencing

Sample Preparation

Low-cost high-throughput sample preparation workflows for Illumina sequencing based on transposon insertion and fluorescent dye-based sample quantification can reduce the cost of preparation to below $15 USD per sample and be performed in approximately 5 hours per 96-well plate[47].

Multiplexed Illumina Sequencing

For accurate discovery of structural variants, 20-30x coverage is ideal per sample. For the 4.6 megabase *Escherichia coli K12 MG1655* genome at 30x coverage, this is approximately 1 million 150 bp reads. The cost per read for Illumina sequencing can vary widely depending on the platform (NexSeq, MiSeq, HiSeq, etc). As a conservative estimate, a whole HiSeq 2500 lane in Rapid Run mode can produce 250 million paired end reads of 150 base pairs for a cost of approximately $2500 USD, or approximately $10 dollars per bacterial genome. Paired-end 150bp sequencing in Rapid Run mode takes approximately 40 hours. Larger-scale sequencing formats are generally cheaper, but longer to run, and smaller formats, like the MiSeq, are faster, but more expensive per genome.

C.3 Supplementary Note 2: Comparison of Millstone to Other Tools

While other packages exist to solve the integration and automation of whole genome resequencing and annotation, most of these tools are built for large diploid genomes, such as *Homo sapiens*. Here we compare features and performance between Millstone and a few other related automated genome re-sequencing tools (see also Table C.1).

Galaxy. Some tools, like Galaxy, allow users to create their own custom pipelines without bash scripting, and do support the creation of pipelines for microbial genomes. However, Galaxy requires that the user to understand and optimize settings for each individual tool. Galaxy also does not allow visualization or interactive querying of the output, and cannot generate new reference genomes or use the output of one round of sequencing to inform the next round. Finally, because of Galaxy's one-size-fits-all nature, optimizing pipeline performance (for example, via inline compression and piping of input and output streams) is not possible.

SPANDx. Another recent tool, SPANDx, can also perform genome resequencing for multiple strains simultaneously, but its widespread use is limited because it can only be run on UNIX computing clusters using the venerable and closed-source commercial PBS job scheduling system. SPANDx has no user interface or interactive components, and so users are required to gather the data manually and run the pipeline using a command line interface. Because we could not readily locate a PBS system to test the pipeline on, we were not able to compare the output between SPANDx and Millstone.

breseq. *breseq* is purpose-built to perform haploid genome resequencing, and has become a standard tool for Adaptive Laboratory Evolution experiments.[106]. *breseq* reports SNV and SV events for single genomes and provides a visualization of raw read evidence for the event. An advantage of Millstone is its ability to perform variant calling on hundreds of genomes in parallel, facilitating analysis of mutation frequency across a population of clones. Millstone's JBrowse integration allows interactive visualization feature allows researchers to zoom, scroll, and rapidly compare read alignments among genomic samples on the fly (as in Figure C.3).

Both *breseq* and Millstone use a default variant detection threshold that works well in most cases, and Millstone complements this with an interactive search feature that allows researchers to filter variants after the variant calling pipeline has been run according to characteristics including read depth, number of samples with the variant, or predicted variant effect.

Millstone further supports paired-end reads, allowing for enhanced detection of structural variation, whereas *breseq* treats paired-end reads as single reads. Millstone can further assemble and place *de novo* contigs onto existing reference genomes. Millstone can be used pre-configured on Amazon Web Services (AWS) and so does not require proficiency with UNIX nor the manual installation of dependencies. Millstone's user interface also automates the process of copying data to the remote server.

References

[1] Michael G Napolitano, Matthieu Landon, Christopher J Gregg, Marc J Lajoie, Lakshmi Govindarajan, Joshua A Mosberg, Gleb Kuznetsov, Daniel B Goodman, Oscar Vargas-Rodriguez, Farren J Isaacs, Dieter Söll, and George M Church. Emergent rules for codon choice elucidated by editing rare arginine codons in escherichia coli. *Proc. Natl. Acad. Sci. U. S. A.*, September 2016.

[2] Joshua B Plotkin and Grzegorz Kudla. Synonymous but not the same: the causes and consequences of codon bias. *Nat. Rev. Genet.*, 12(1):32–42, January 2011.

[3] Daniel G Gibson, John I Glass, Carole Lartigue, Vladimir N Noskov, Ray-Yuan Chuang, Mikkel A Algire, Gwynedd A Benders, Michael G Montague, Li Ma, Monzia M Moodie, Chuck Merryman, Sanjay Vashee, Radha Krishnakumar, Nacyra Assad-Garcia, Cynthia Andrews-Pfannkoch, Evgeniya A Denisova, Lei Young, Zhi-Qing Qi, Thomas H Segall-Shapiro, Christopher H Calvey, Prashanth P Parmar, Clyde A Hutchison, 3rd, Hamilton O Smith, and J Craig Venter. Creation of a bacterial cell controlled by a chemically synthesized genome. *Science*, 329(5987):52–56, July 2010.

[4] Marc J Lajoie, Alexis J Rovner, Daniel B Goodman, Hans-Rudolf Aerni, Adrian D Haimovich, Gleb Kuznetsov, Jaron A Mercer, Harris H Wang, Peter A Carr, Joshua A Mosberg, and others. Genomically recoded organisms expand biological functions. *Science (New York, NY)*, 342(6156):357–360, 2013.

[5] M J Lajoie, S Kosuri, J A Mosberg, C J Gregg, D Zhang, and G M Church. Probing the limits of genetic recoding in essential genes. *Science*, 342(6156):361–363, October 2013.

[6] Philippe Marliere. The farther, the safer: a manifesto for securely navigating synthetic species away from the old living world. *Syst. Synth. Biol.*, 3(1-4):77–84, December 2009.

[7] Daniel J Mandell, Marc J Lajoie, Michael T Mee, Ryo Takeuchi, Gleb Kuznetsov, Julie E Norville, Christopher J Gregg, Barry L Stoddard, and George M Church. Biocontainment of genetically modified organisms by synthetic protein design. *Nature*, 518(7537):55–60, 2015.

[8] Alexis J Rovner, Adrian D Haimovich, Spencer R Katz, Zhe Li, Michael W Grome, Brandon M Gassaway, Miriam Amiram, Jaymin R Patel, Ryan R Gallagher, Jesse Rinehart, and Farren J Isaacs. Recoded organisms engineered to depend on synthetic amino acids. *Nature*, 518(7537):89–93, February 2015.

[9] Nicholas R Markham and Michael Zuker. UNAFold: software for nucleic acid folding and hybridization. *Methods Mol. Biol.*, 453:3–31, 2008.

[10] Joseph N Zadeh, Conrad D Steenberg, Justin S Bois, Brian R Wolfe, Marshall B Pierce, Asif R Khan, Robert M Dirks, and Niles A Pierce. Nupack: analysis and design of nucleic acid systems. *Journal of computational chemistry*, 32(1):170–173, 2011.

[11] Howard M Salis. The ribosome binding site calculator. *Methods Enzymol.*, 498:19–42, 2011.

[12] K J Blight, A A Kolykhalov, and C M Rice. Efficient initiation of HCV RNA replication in cell culture. *Science*, 290(5498):1972–1974, December 2000.

[13] Jeronimo Cello, Aniko V Paul, and Eckard Wimmer. Chemical synthesis of poliovirus cDNA: generation of infectious virus in the absence of natural template. *Science*, 297(5583):1016–1018, August 2002.

[14] Hamilton O Smith, Clyde A Hutchison, Cynthia Pfannkoch, and J Craig Venter. Generating a synthetic genome by whole genome assembly: ϕX174 bacteriophage from synthetic oligonucleotides. *Proceedings of the National Academy of Sciences*, 100(26):15440–15445, December 2003.

[15] Leon Y Chan, Sriram Kosuri, and Drew Endy. Refactoring bacteriophage T7. *Mol. Syst. Biol.*, 1:2005.0018, September 2005.

[16] Daniel G Gibson, Gwynedd A Benders, Cynthia Andrews-Pfannkoch, Evgeniya A Denisova, Holly Baden-Tillson, Jayshree Zaveri, Timothy B Stockwell, Anushka Brownley, David W Thomas, Mikkel A Algire, Chuck Merryman, Lei Young, Vladimir N Noskov, John I Glass, J Craig Venter, Clyde A Hutchison, 3rd, and Hamilton O Smith. Complete chemical synthesis, assembly, and cloning of a mycoplasma genitalium genome. *Science*, 319(5867):1215–1220, February 2008.

[17] Narayana Annaluru, Héloïse Muller, Leslie A Mitchell, Sivaprakash Ramalingam, Giovanni Stracquadanio, Sarah M Richardson, Jessica S Dymond, Zheng Kuang, Lisa Z Scheifele,

Eric M Cooper, Yizhi Cai, Karen Zeller, Neta Agmon, Jeffrey S Han, Michalis Hadjithomas, Jennifer Tullman, Katrina Caravelli, Kimberly Cirelli, Zheyuan Guo, Viktoriya London, Apurva Yeluru, Sindurathy Murugan, Karthikeyan Kandavelou, Nicolas Agier, Gilles Fischer, Kun Yang, J Andrew Martin, Murat Bilgel, Pavlo Bohutski, Kristin M Boulier, Brian J Capaldo, Joy Chang, Kristie Charoen, Woo Jin Choi, Peter Deng, James E DiCarlo, Judy Doong, Jessilyn Dunn, Jason I Feinberg, Christopher Fernandez, Charlotte E Floria, David Gladowski, Pasha Hadidi, Isabel Ishizuka, Javaneh Jabbari, Calvin Y L Lau, Pablo A Lee, Sean Li, Denise Lin, Matthias E Linder, Jonathan Ling, Jaime Liu, Jonathan Liu, Mariya London, Henry Ma, Jessica Mao, Jessica E McDade, Alexandra McMillan, Aaron M Moore, Won Chan Oh, Yu Ouyang, Ruchi Patel, Marina Paul, Laura C Paulsen, Judy Qiu, Alex Rhee, Matthew G Rubashkin, Ina Y Soh, Nathaniel E Sotuyo, Venkatesh Srinivas, Allison Suarez, Andy Wong, Remus Wong, Wei Rose Xie, Yijie Xu, Allen T Yu, Romain Koszul, Joel S Bader, Jef D Boeke, and Srinivasan Chandrasegaran. Total synthesis of a functional designer eukaryotic chromosome. *Science*, 344(6179):55–58, April 2014.

[18] Clyde A Hutchison, 3rd, Ray-Yuan Chuang, Vladimir N Noskov, Nacyra Assad-Garcia, Thomas J Deerinck, Mark H Ellisman, John Gill, Krishna Kannan, Bogumil J Karas, Li Ma, James F Pelletier, Zhi-Qing Qi, R Alexander Richter, Elizabeth A Strychalski, Lijie Sun, Yo Suzuki, Billyana Tsvetanova, Kim S Wise, Hamilton O Smith, John I Glass, Chuck Merryman, Daniel G Gibson, and J Craig Venter. Design and synthesis of a minimal bacterial genome. *Science*, 351(6280):aad6253, March 2016.

[19] Grzegorz Kudla, Andrew W Murray, David Tollervey, and Joshua B Plotkin. Coding-sequence determinants of gene expression in escherichia coli. *Science*, 324(5924):255–258, April 2009.

[20] Tamir Tuller, Yedael Y Waldman, Martin Kupiec, and Eytan Ruppin. Translation efficiency is determined by both codon bias and folding energy. *Proc. Natl. Acad. Sci. U. S. A.*, 107(8):3645–3650, February 2010.

[21] Daniel B Goodman, George M Church, and Sriram Kosuri. Causes and effects of n-terminal codon bias in bacterial genes. *Science*, 342(6157):475–479, October 2013.

[22] Mian Zhou, Jinhu Guo, Joonseok Cha, Michael Chae, She Chen, Jose M Barral, Matthew S Sachs, and Yi Liu. Non-optimal codon usage affects expression, structure and function of clock protein FRQ. *Nature*, 495(7439):111–115, March 2013.

[23] Tessa E F Quax, Nico J Claassens, Dieter Söll, and John van der Oost. Codon bias as a means to Fine-Tune gene expression. *Mol. Cell*, 59(2):149–161, July 2015.

[24] Grégory Boël, Reka Letso, Helen Neely, W Nicholson Price, Kam-Ho Wong, Min Su, Jon D Luff, Mayank Valecha, John K Everett, Thomas B Acton, Rong Xiao, Gaetano T Montelione, Daniel P Aalberts, and John F Hunt. Codon influence on protein expression in e. coli correlates with mRNA levels. *Nature*, 529(7586):358–363, January 2016.

[25] Farren J Isaacs, Peter A Carr, Harris H Wang, Marc J Lajoie, Bram Sterling, Laurens Kraal, Andrew C Tolonen, Tara A Gianoulis, Daniel B Goodman, Nikos B Reppas, Christopher J Emig, Duhee Bang, Samuel J Hwang, Michael C Jewett, Joseph M Jacobson, and George M Church. Precise manipulation of chromosomes in vivo enables genome-wide codon replacement. *Science (New York, NY)*, 333(6040):348–353, 2011.

[26] Harris H Wang, Farren J Isaacs, Peter A Carr, Zachary Z Sun, George Xu, Craig R Forest, and George M Church. Programming cells by multiplex genome engineering and accelerated evolution. *Nature*, 460(7257):894–898, 2009.

[27] Kevin M Esvelt, Prashant Mali, Jonathan L Braff, Mark Moosburner, Stephanie J Yaung, and George M Church. Orthogonal cas9 proteins for RNA-guided gene regulation and editing. *Nat. Methods*, 10(11):1116–1121, November 2013.

[28] György Pósfai, Guy Plunkett, 3rd, Tamás Fehér, David Frisch, Günther M Keil, Kinga Umenhoffer, Vitaliy Kolisnychenko, Buffy Stahl, Shamik S Sharma, Monika de Arruda, Valerie Burland, Sarah W Harcum, and Frederick R Blattner. Emergent properties of reduced-genome escherichia coli. *Science*, 312(5776):1044–1046, May 2006.

[29] Karsten Temme, Dehua Zhao, and Christopher A Voigt. Refactoring the nitrogen fixation gene cluster from klebsiella oxytoca. *Proc. Natl. Acad. Sci. U. S. A.*, 109(18):7085–7090, May 2012.

[30] Avihu H Yona, Zohar Bloom-Ackermann, Idan Frumkin, Victor Hanson-Smith, Yoav Charpak-Amikam, Qinghua Feng, Jef D Boeke, Orna Dahan, and Yitzhak Pilpel. tRNA genes rapidly change in evolution to meet novel translational demands. *Elife*, 2:e01339, December 2013.

[31] Yukiko Yamazaki, Hironori Niki, and Jun-Ichi Kato. Profiling of escherichia coli chromosome database. In Andrei L Osterman and Svetlana Y Gerdes, editors, *Microbial Gene Essen-*

tiality: Protocols and Bioinformatics, volume 416 of *Methods in Molecular Biology™*, pages 385–389. Humana Press, Totowa, NJ, 2008.

[32] Kajetan Bentele, Paul Saffert, Robert Rauscher, Zoya Ignatova, and Nils Blüthgen. Efficient translation initiation dictates codon usage at gene start. *Molecular systems biology*, 9(1):675, 2013.

[33] Gene-Wei Li, Eugene Oh, and Jonathan S Weissman. The anti-shine–dalgarno sequence drives translational pausing and codon choice in bacteria. *Nature*, 484(7395):538, 2012.

[34] Tao Wei, Bi-Yan Cheng, and Jian-Zhong Liu. Genome engineering escherichia coli for L-DOPA overproduction from glucose. *Sci. Rep.*, 6:30080, July 2016.

[35] Jessica S Dymond, Sarah M Richardson, Candice E Coombes, Timothy Babatz, Héloïse Muller, Narayana Annaluru, William J Blake, Joy W Schwerzmann, Junbiao Dai, Derek L Lindstrom, Annabel C Boeke, Daniel E Gottschling, Srinivasan Chandrasegaran, Joel S Bader, and Jef D Boeke. Synthetic chromosome arms function in yeast and generate phenotypic diversity by design. *Nature*, 477(7365):471–476, September 2011.

[36] Nili Ostrov, Matthieu Landon, Marc Guell, Gleb Kuznetsov, Jun Teramoto, Natalie Cervantes, Minerva Zhou, Kerry Singh, Michael G Napolitano, Mark Moosburner, Ellen Shrock, Benjamin W Pruitt, Nicholas Conway, Daniel B Goodman, Cameron L Gardner, Gary Tyree, Alexandra Gonzales, Barry L Wanner, Julie E Norville, Marc J Lajoie, and George M Church. Design, synthesis, and testing toward a 57-codon genome. *Science*, 353(6301):819–822, August 2016.

[37] Natalie Jing Ma and Farren J Isaacs. Genomic recoding broadly obstructs the propagation of horizontally transferred genetic elements. *Cell Syst*, July 2016.

[38] Daniel B Goodman, Gleb Kuznetsov, Marc J Lajoie, Brian W Ahern, Michael G Napolitano, Kevin Y Chen, Changping Chen, and George M Church. Millstone: software for multiplex microbial genome analysis and engineering. *Genome biology*, 18(1):101, 2017.

[39] Pablo Cingolani, Adrian Platts, Le Lily Wang, Melissa Coon, Tung Nguyen, Luan Wang, Susan J Land, Xiangyi Lu, and Douglas M Ruden. A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of drosophila melanogaster strain w1118; iso-2; iso-3. *Fly*, 6(2):80–92, April 2012.

[40] Tomoya Baba, Takeshi Ara, Miki Hasegawa, Yuki Takai, Yoshiko Okumura, Miki Baba, Kirill A Datsenko, Masaru Tomita, Barry L Wanner, and Hirotada Mori. Construction of escherichia coli K-12 in-frame, single-gene knockout mutants: the keio collection. *Mol. Syst. Biol.*, 2:2006.0008, February 2006.

[41] Gleb Kuznetsov, Daniel B Goodman, Gabriel T Filsinger, Matthieu Landon, Nadin Rohland, John Aach, Marc J Lajoie, and George M Church. Optimizing complex phenotypes through model-guided multiplex genome engineering. *Genome biology*, 18(1):100, 2017.

[42] Hui Zou, Zou Hui, and Hastie Trevor. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Series B Stat. Methodol.*, 67(2):301–320, 2005.

[43] Aisha I Khan, Duy M Dinh, Dominique Schneider, Richard E Lenski, and Tim F Cooper. Negative epistasis between beneficial mutations in an evolving bacterial population. *Science*, 332(6034):1193–1196, June 2011.

[44] James E DiCarlo, Andrew J Conley, Merja Penttilä, Jussi Jäntti, Harris H Wang, and George M Church. Yeast oligo-mediated genome engineering (YOGE). *ACS Synth. Biol.*, 2(12):741–749, December 2013.

[45] Ami M Kabadi, David G Ousterout, Isaac B Hilton, and Charles A Gersbach. Multiplex CRISPR/Cas9-based genome engineering from a single lentiviral vector. *Nucleic Acids Res.*, 42(19):e147, October 2014.

[46] Srivatsan Raman, Jameson K Rogers, Noah D Taylor, and George M Church. Evolution-guided optimization of biosynthetic pathways. *Proc. Natl. Acad. Sci. U. S. A.*, 111(50):17803–17808, December 2014.

[47] Michael Baym, Sergey Kryazhimskiy, Tami D Lieberman, Hattie Chung, Michael M Desai, and Roy Kishony. Inexpensive Multiplexed Library Preparation for Megabase-Sized Genomes. *bioRxiv*, page 013771, 2015.

[48] Mads T Bonde, Sriram Kosuri, Hans J Genee, Kira Sarup-Lytzen, George M Church, Morten O A Sommer, and Harris H Wang. Direct mutagenesis of thousands of genomic targets using microarray-derived oligonucleotides. *ACS Synth. Biol.*, 4(1):17–22, January 2015.

[49] Ophir Shalem, Neville E Sanjana, Ella Hartenian, Xi Shi, David A Scott, Tarjei S Mikkelsen, Dirk Heckl, Benjamin L Ebert, David E Root, John G Doench, and Feng Zhang. Genome-

scale CRISPR-Cas9 knockout screening in human cells. *Science*, 343(6166):84–87, January 2014.

[50] Tim Wang, Jenny J Wei, David M Sabatini, and Eric S Lander. Genetic screens in human cells using the CRISPR-Cas9 system. *Science*, 343(6166):80–84, January 2014.

[51] Thomas J Mansell, Joseph R Warner, and Ryan T Gill. Trackable multiplex recombineering for gene-trait mapping in e. coli. *Methods Mol. Biol.*, 985:223–246, 2013.

[52] Ramsey I Zeitoun, Andrew D Garst, George D Degen, Gur Pines, Thomas J Mansell, Tirzah Y Glebes, Nanette R Boyle, and Ryan T Gill. Multiplexed tracking of combinatorial genomic mutations in engineered cell populations. *Nat. Biotechnol.*, 33(6):631–637, June 2015.

[53] Socorro Gama-Castro, Gama-Castro Socorro, Salgado Heladia, Santos-Zavaleta Alberto, Ledezma-Tejeida Daniela, Muñiz-Rascado Luis, Jair Santiago García-Sotelo, Alquicira-Hernández Kevin, Martínez-Flores Irma, Pannier Lucia, Jaime Abraham Castro-Mondragón, Medina-Rivera Alejandra, Solano-Lira Hilda, Bonavides-Martínez César, Pérez-Rueda Ernesto, Alquicira-Hernández Shirley, Porrón-Sotelo Liliana, López-Fuentes Alejandra, Hernández-Koutoucheva Anastasia, Víctor Del Moral-Chávez, Rinaldi Fabio, and Collado-Vides Julio. RegulonDB version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond. *Nucleic Acids Res.*, 44(D1):D133–D143, 2015.

[54] Christopher J Gregg, Marc J Lajoie, Michael G Napolitano, Joshua A Mosberg, Daniel B Goodman, John Aach, Farren J Isaacs, and George M Church. Rational optimization of tolc as a powerful dual selectable marker for genome engineering. *Nucleic Acids Res.*, 42(7):4779–4790, April 2014.

[55] Nadin Rohland and David Reich. Cost-effective, high-throughput DNA sequencing libraries for multiplexed target capture. *Genome Res.*, 22(5):939–946, May 2012.

[56] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[57] R Lutz and H Bujard. Independent and tight regulation of transcriptional units in escherichia coli via the LacR/O, the TetR/O and AraC/I1-I2 regulatory elements. *Nucleic Acids Res.*, 25(6):1203–1210, March 1997.

[58] Mitchell E Skinner, Andrew V Uzilov, Lincoln D Stein, Christopher J Mungall, and Ian H Holmes. JBrowse: a next-generation genome browser. *Genome Research*, 19(9):1630–1638, September 2009.

[59] Martin Dragosits and Diethard Mattanovich. Adaptive laboratory evolution – principles and applications for biotechnology. *Microbial cell factories*, 12(1):64, 2013.

[60] Wenyan Jiang, David Bikard, David Cox, Feng Zhang, and Luciano A Marraffini. RNA-guided editing of bacterial genomes using CRISPR-Cas systems. *Nature Biotechnology*, 31(3):233–239, 2013.

[61] Elaine B Shapland, Victor Holmes, Christopher D Reeves, Elena Sorokin, Maxime Durot, Darren Platt, Christopher Allen, Jed Dean, Zach Serber, Jack Newman, and Sunil Chandran. Low-Cost, High-Throughput Sequencing of DNA Assemblies Using a Highly Multiplexed Nextera Process. *ACS Synthetic Biology*, 4(7):860–866, July 2015.

[62] O Tenaillon, A Rodriguez-Verdugo, R L Gaut, P Mcdonald, A F Bennett, A D Long, and B S Gaut. The Molecular Diversity of Adaptive Convergence. *Science (New York, NY)*, 335(6067):457–461, January 2012.

[63] Adrian D Haimovich, Paul Muir, and Farren J Isaacs. Genomes by design. *Nature Reviews Genetics*, 16(9):501–516, 2015.

[64] Michael S Packer and David R Liu. Methods for the directed evolution of proteins. *Nat. Rev. Genet.*, 16(7):379–394, July 2015.

[65] Stefan Lutz. Beyond directed evolution–semi-rational protein engineering and design. *Curr. Opin. Biotechnol.*, 21(6):734–743, December 2010.

[66] Po-Ssu Huang, Scott E Boyken, and David Baker. The coming of age of de novo protein design. *Nature*, 537(7620):320–327, 2016.

[67] Debora S Marks, Thomas A Hopf, and Chris Sander. Protein structure prediction from sequence variation. *Nat. Biotechnol.*, 30(11):1072–1080, 2012.

[68] Ken A Dill and Justin L MacCallum. The protein-folding problem, 50 years on. *Science*, 338(6110):1042–1046, November 2012.

[69] Charlotte M Miton and Nobuhiko Tokuriki. How mutational epistasis impairs predictability in protein evolution and design. *Protein Sci.*, 25(7):1260–1272, 2016.

[70] S Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proc. Sixth Int. Congr. Genet.*, 1932.

[71] J Arjan G M de Visser and Joachim Krug. Empirical fitness landscapes and the predictability of evolution. *Nat. Rev. Genet.*, 15(7):480–490, July 2014.

[72] John Maynard Smith. Natural selection and the concept of a protein space. *Nature*, 225(5232):563–564, 1970.

[73] S A Kauffman and E D Weinberger. The NK model of rugged fitness landscapes and its application to maturation of the immune response. *J. Theor. Biol.*, 141(2):211–245, November 1989.

[74] Daniel L Hartl. What can we learn from fitness landscapes? *Curr. Opin. Microbiol.*, 21:51–57, October 2014.

[75] Douglas M Fowler and Stanley Fields. Deep mutational scanning: a new style of protein science. *Nat. Methods*, 11(8):801–807, 2014.

[76] Lea M Starita, David L Young, Muhtadi Islam, Jacob O Kitzman, Justin Gullingsrud, Ronald J Hause, Douglas M Fowler, Jeffrey D Parvin, Jay Shendure, and Stanley Fields. Massively parallel functional analysis of BRCA1 RING domain variants. *Genetics*, 200(2):413–422, June 2015.

[77] Douglas M Fowler, Carlos L Araya, Sarel J Fleishman, Elizabeth H Kellogg, Jason J Stephany, David Baker, and Stanley Fields. High-resolution mapping of protein sequence-function relationships. *Nat. Methods*, 7(9):741–746, September 2010.

[78] Karen S Sarkisyan, Dmitry A Bolotin, Margarita V Meer, Dinara R Usmanova, Alexander S Mishin, George V Sharonov, Dmitry N Ivankov, Nina G Bozhanova, Mikhail S Baranov, Onuralp Soylemez, Natalya S Bogatyreva, Peter K Vlasov, Evgeny S Egorov, Maria D Logacheva, Alexey S Kondrashov, Dmitry M Chudakov, Ekaterina V Putintseva, Ilgar Z Mamedov, Dan S Tawfik, Konstantin A Lukyanov, and Fyodor A Kondrashov. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, May 2016.

[79] Chuan Li, Wenfeng Qian, Calum J Maclean, and Jianzhi Zhang. The fitness landscape of a tRNA gene. *Science*, 352(6287):837–840, May 2016.

[80] Timothy A Whitehead, Aaron Chevalier, Yifan Song, Cyrille Dreyfus, Sarel J Fleishman, Cecilia De Mattos, Chris A Myers, Hetunandan Kamisetty, Patrick Blair, Ian A Wilson, and David Baker. Optimization of affinity, specificity and function of designed influenza inhibitors using deep sequencing. *Nat. Biotechnol.*, 30(6):543–548, May 2012.

[81] Jun Liao, Manfred K Warmuth, Sridhar Govindarajan, Jon E Ness, Rebecca P Wang, Claes Gustafsson, and Jeremy Minshull. Engineering proteinase K using machine learning and synthetic genes. *BMC Biotechnol.*, 7:16, March 2007.

[82] Richard J Fox, S Christopher Davis, Emily C Mundorff, Lisa M Newman, Vesna Gavrilovic, Steven K Ma, Loleta M Chung, Charlene Ching, Sarena Tam, Sheela Muley, et al. Improving catalytic function by prosar-driven enzyme evolution. *Nature biotechnology*, 25(3):338, 2007.

[83] Philip A Romero, Andreas Krause, and Frances H Arnold. Navigating the protein fitness landscape with gaussian processes. *Proc. Natl. Acad. Sci. U. S. A.*, 110(3):E193–201, January 2013.

[84] Claire N Bedbrook, Kevin K Yang, Austin J Rice, Viviana Gradinaru, and Frances H Arnold. Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS computational biology*, 13(10):e1005786, 2017.

[85] Elad Gilboa, Yunus Saatçi, and John P Cunningham. Scaling multidimensional inference for structured gaussian processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, September 2013.

[86] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[87] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, 1989.

[88] Hui Y Xiong, Babak Alipanahi, Leo J Lee, Hannes Bretschneider, Daniele Merico, Ryan K C Yuen, Yimin Hua, Serge Gueroussov, Hamed S Najafabadi, Timothy R Hughes, Quaid Morris, Yoseph Barash, Adrian R Krainer, Nebojsa Jojic, Stephen W Scherer, Benjamin J Blencowe, and Brendan J Frey. RNA splicing. the human splicing code reveals new insights into the genetic determinants of disease. *Science*, 347(6218):1254806, January 2015.

[89] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat. Biotechnol.*, 33(8):831–838, August 2015.

[90] Vladimir Golkov, Marcin J Skwark, Antonij Golkov, Alexey Dosovitskiy, Thomas Brox, Jens Meiler, and Daniel Cremers. Protein contact prediction from amino acid co-evolution using convolutional networks for graph-valued images. In *Advances in Neural Information Processing Systems*, pages 4222–4230, 2016.

[91] Guillaume Urtecho, Arielle D Tripp, Kimberly Insigne, Hwangbeom Kim, and Sriram Kosuri. Systematic dissection of sequence elements controlling $\sigma70$ promoters using a genomically-encoded multiplexed reporter assay in e. coli. *bioRxiv*, page 207332, 2017.

[92] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, May 2015.

[93] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, pages 2171–2180, 2015.

[94] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[95] Erik A Rodriguez, Robert E Campbell, John Y Lin, Michael Z Lin, Atsushi Miyawaki, Amy E Palmer, Xiaokun Shu, Jin Zhang, and Roger Y Tsien. The growing and glowing toolbox of fluorescent and photoactive proteins. *Trends Biochem. Sci.*, 42(2):111–129, February 2017.

[96] Peter Dedecker, Frans C De Schryver, and Johan Hofkens. Fluorescent proteins: shine on, you crazy diamond. *J. Am. Chem. Soc.*, 135(7):2387–2402, February 2013.

[97] Nathan C Shaner, Gerard G Lambert, Andrew Chammas, Yuhui Ni, Paula J Cranfill, Michelle A Baird, Brittney R Sell, John R Allen, Richard N Day, Maria Israelsson, Michael W Davidson, and Jiwu Wang. A bright monomeric green fluorescent protein derived from branchiostoma lanceolatum. *Nat. Methods*, 10(5):407–409, May 2013.

[98] Daphne S Bindels, Lindsay Haarbosch, Laura van Weeren, Marten Postma, Katrin E Wiese, Marieke Mastop, Sylvain Aumonier, Guillaume Gotthard, Antoine Royant, Mark A Hink, and Theodorus W J Gadella, Jr. mscarlet: a bright monomeric red fluorescent protein for cellular imaging. *Nat. Methods*, 14(1):53–56, January 2017.

[99] Jean-Denis Pédelacq, Stéphanie Cabantous, Timothy Tran, Thomas C Terwilliger, and Geoffrey S Waldo. Engineering and characterization of a superfolder green fluorescent protein. *Nat. Biotechnol.*, 24(1):79–88, January 2006.

[100] Andreas Crameri, Erik A Whitehorn, Emily Tate, and Willem PC Stemmer. Improved green fluorescent protein by molecular evolution using dna shuffling. *Nature biotechnology*, 14(3):315, 1996.

[101] Geoffrey S Waldo, Blake M Standish, Joel Berendzen, and Thomas C Terwilliger. Rapid protein-folding assay using green fluorescent protein. *Nature biotechnology*, 17(7):691, 1999.

[102] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. July 2012.

[103] Tushar Jain, Tingwan Sun, Stéphanie Durand, Amy Hall, Nga Rewa Houston, Juergen H Nett, Beth Sharkey, Beata Bobrowicz, Isabelle Caffry, Yao Yu, et al. Biophysical properties of the clinical-stage antibody landscape. *Proceedings of the National Academy of Sciences*, 114(5):944–949, 2017.

[104] Heng Li and Richard Durbin. Fast and accurate long-read alignment with burrows–wheeler transform. *Bioinformatics*, 26(5):589–595, 2010.

[105] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.

[106] Daniel E Deatherage and Jeffrey E Barrick. Identification of mutations in laboratory-evolved microbes from next-generation sequencing data using breseq. *Engineering and Analyzing Multicellular Systems: Methods and Protocols*, pages 165–188, 2014.