Platforms for Biological Control

by

Dana Gretton

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author . . . . . .
Program in Media Arts and Sciences,
School of Architecture and Planning
August 19, 2022

Certified by . . . . . .
Kevin M. Esvelt
Associate Professor
Thesis Supervisor

Accepted by . . . . . .
Tod Machover
Academic Head, Program in Media Arts and Sciences

Platforms for Biological Control

by

Dana Gretton

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on August 19, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

Innovation at the level of platforms provides the greatest leverage for shaping science and technology. Here, two key weaknesses in platforms for biology are addressed: the lack of accessible, extensible open-source software for the exploding field of robotic bioautomation, and the lack of a standardized, consistent screening platform for the manufacture of dangerous DNA. Pyhamilton and SecureDNA are introduced. Pyhamilton is the first open-source Python package for controlling Hamilton liquid-handling robots for biology. SecureDNA is the only DNA screening concept to prioritize anti-proliferation of pathogen genomes, and the first to employ modern cryptography to secure a global screening system that can keep up with anticipated exponential growth of the DNA synthesis market. For each platform developed, a software implementation is provided and exercised in a range of applications, and hardware demonstrators have been produced.

Thesis Supervisor: Kevin M. Esvelt
Title: Associate Professor

This thesis has been reviewed and approved by the following committee members:

Professor Ron Rivest . . . . . .
Chairman, Thesis Committee
Institute Professor of the Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Computer Science and Artificial Intelligence Laboratory

Professor Kevin Esvelt . . . . . .
Thesis Supervisor
Associate Professor of Media Arts and Sciences
Leader, Sculpting Evolution Group
MIT Media Lab

Professor Joseph Jacobson . . . . . .
Member, Thesis Committee
Associate Professor of Media Arts and Sciences
Leader, Molecular Machines Group
MIT Media Lab

Acknowledgments

I would like to acknowledge my academic advisor, Prof. Kevin Esvelt, for boundless expertise and advice, for reassuring me and celebrating successes with me, and for offering me grace when it was needed, over the past 9 years.

I owe a debt of gratitude to Turing Award winners Prof. Andrew Yao and Prof. Ron Rivest, who have graciously donated their time in numerous meetings related to the SecureDNA project over the past several years.

I would like to thank Professor Mingyu Gao of Tsinghua University for outstanding contributions to all aspects of the SecureDNA project, and for helping me feel at ease whenever I was uncertain. I also thank Professor Yu Yu of Tsinghua for his consistent and necessary technical advice.

Dr. Emma J. Chory and Dr. Erika A. DeBenedictis exploded my small project into a world of possibilities, applications and potential that I never could have seen. It's been an absolute joy.

Dr. Leonard Foner brought much needed grounding and practical advice throughout development of the SecureDNA project, and to him I owe my thanks.

A large list of references for me to look over in my research was graciously provided by CERI fellows Oscar Delaney and Hanna Pálya.

I want to thank my parents, Dorothy Hoskins and Geoffrey Gretton, for making all of this possible, and reliably nudging me in the right direction.

And to my partner Simon, for going out of his way to care for me, always, over all these 5 years, and for his creative out-of-the-box suggestions that have significantly influenced the course of my work.

Table of Contents

[1] Adapted from Erika A. DeBenedictis, *et. al.* Systematic molecular evolution enables robust biomolecule discovery[36], on which Gretton, D. W. is a co-author.

[2] Adapted from Emma J. Chory, *et. al.* Enabling high-throughput biology with flexible open-source automation[28], on which Gretton, D. W. is a co-author.

[3] Adapted from Carsten Baum, *et. al.* Cryptographic Aspects of DNA Screening[14] on which Gretton, D. W. is a co-author.

[4] Adapted from Gretton, *et. al.* Random Adversarial threshold search enables specific, secure, and fully automated DNA synthesis screening[58]

Chapter 1

Introduction

> A biological system can be exceedingly small. [. . .] Consider the possibility that we too can make a thing very small which does what we want — that we can manufacture an object that maneuvers at that level!
>
> *Richard P. Feynman*
> There's Plenty of Room at the Bottom*, 1959*

The velocity of research progress in molecular biology is unprecedented, and it is only expected to rise in coming decades. It is the opinion of the author that the magnetic draw to biology as a tool for technology is due to its proximity to "the bottom," as Feynman[49] put it: the very smallest scale we know of that can support meaningful objects and mechanisms. This is the scale running from handfuls to hundreds of thousands of atoms. Biology is at present our most promising window into the nanoscale. Indeed, biology is the only true nanotechnology that is real and pervasive today.

The extreme velocity of research can be attributed to the rapid development of *platforms*, contexts with which to frame research problems, along with context-specific tools

to manipulate subject matter. The clearest example is the CRISPR/Cas9 system, which effectively enabled arbitrary "cut-paste" genome editing for the first time in 2012[72]. Cas9 is a protein that cleaves DNA, also known as an endonuclease, which typically seeks out a 20-nucleotide guide RNA sequence that matches a target site in a genome to make its "cut." "Paste" is accomplished by providing extra DNA bookended with sequences that are similar, or homologous, to the sequences before and after the cut, which can cause a cell to incorporate the new strand with the insertion when it attempts to reassemble its genome (this process is called *homologous recombination*). The experimenter or engineer can choose any target sequence (subject to some limitations), and the Cas9 gene itself can be encoded in DNA, opening the possibility of fully autonomous, "programmable" *in vivo* genome editing. The publication[72] introducing the optimized CRISPR/Cas9 system has been cited about 15,000 times at time of writing, and the work earned Jennifer A. Doudna and Emmanuelle Charpentier the 2020 Nobel Prize in Chemistry under the title, A TOOL FOR GENOME EDITING. Notably, the Nobel was not awarded under a title recognizing its connection to adaptive viral immunity in bacteria, from which CRISPR gets its cryptic name. CRISPR/Cas9 receives such intense attention and recognition, not primarily for being a landmark discovery, but for its boundless potential as a platform that established a new context (positioning by binding to short target sequences in the DNA) in which the broader concept of genome editing could be framed, and tools (the family of homing endonucleases, including Cas9, Cas12a, others, and their many functional variants) that operate in that context. CRISPR has so suffused the field of gene editing that it is now difficult to imagine genome manipulation in any context other than short target sequence binding[1]; such is the influence of platforms.

Other platforms that have accelerated microbiology include convolutional, recurrent, and transformer neural nets of billions of parameters that have been applied to DNA and

[1]One example of a framing for genome editing that is not based on homing to unique target sequences in DNA might be a method for indexing into the genome, like accessing elements in an array in computer science, from a known starting point. CRISPR systems will be refined continuously to the exclusion of any such framing. Other framings are now exceedingly unlikely to reach a useful level of advancement, unless CRISPR-like homing techniques should run into a fundamental barrier.

protein sequences, to predict e.g. protein folding and function; Cre-Lox recombination sites, which execute deletions, insertions, translocations and inversions in DNA; AddGene, an ever-growing repository of useful DNA sequences, or constructs, paired with online tools such as Benchling that facilitate creating and sharing designs; among many others.

This thesis regards two domains in which platform innovation is poised to be particularly influential: laboratory automation and DNA synthesis.

Laboratory automation multiplies the productivity of traditional hand-pipetted molecular biology techniques by factors of one hundred to one thousand, and, in the near future, much more. This observation alone justifies its claim to transformative impact in: CRISPR editing, the products of deep learning, Cre-Lox, all of the constructs designed and shared, and all the rest. Beyond accelerating and parallelizing the work performed by today's microbiologists, we already see hints of the impact robotic capabilities beyond those of humans will have on the life sciences, a few of which will be featured in later chapters, e.g. feedback control algorithms. Meanwhile, DNA synthesis provides a crucial raw material that brings, again, CRISPR editing, the products of deep learning, Cre-Lox, all of the constructs designed and shared, and all modern methods in molecular biology in part or whole, from the province of information into the physical world[2]. Frighteningly, the exact choice of platform that becomes the new norm for faster, more cost-effective DNA synthesis, and the degree to which it is controlled with appropriate rules and regulations, may determine the fate of most people in the world and their descendants. Observing that the COVID-19 virion is only a short polynucleotide wrapped in a protein envelope, one need only refer to the global pandemic[3] still evolving worldwide to recognize the power that even a single strand of DNA might hold.

In initial chapters, a new platform for laboratory automation specific to the popular liquid handling robots made by the Hamilton Company is described. These robots, starting about the size of a Mini Cooper, find frequent use in drug discovery, protein engineering,

[2]Directed evolution could be viewed as a platform that does not require DNA synthesis, though in practice all modern methods utilize synthetic DNA.

[3]Since everyone is tired of it, it will be attempted herein to reference the global pandemic as little as possible; please excuse any lapses.

DNA preparation and analysis, sequencing, and many other routine laboratory tasks. They often replace or complement manual labor that would otherwise be performed by laboratory workers. It is not uncommon for more than 100 Hamilton robots to be in use at a single large pharmaceutical or biotechnology company, e.g. Ginkgo Bioworks, and many smaller research laboratories use Hamiltons worldwide. The work described here brings an open-source programming interface in a popular programming language to Hamilton robots for the first time, which, we demonstrate, enables the use of modern programming principles to unlock the potential for laboratory robots to do more than those that are set up in a more standard way, or their human counterparts, ever could.

Later chapters describe another new platform, this one to be applied to the problem of keeping DNA synthesis safe. DNA synthesizers are mechanical, electrical, and fluidic devices that convert DNA precursors into precise long-chain molecules, or strands, of DNA. The DNA is made according to a sequence specification, usually a file or stream of data, perhaps from a customer's order on the Internet. Sequence data is defined in terms of the four canonical DNA bases[4] Adenine, Thymine, Cytosine and Guanine, abbreviated A, T, C and G, and sometimes some other special codes indicating where to start and stop, or where multiple bases may be admissible. The DNA strands that synthesizers produce range in size from tiny segments of no more than five or six bases, to massive strands of several thousand base pairs. Much longer DNA strands up to whole prokaryotic genomes or larger can be synthesized by assembling smaller pieces. To a synthesizer, no DNA sequence is much different from any other, but different DNA sequences have wildly varying functions in biology. Some of these functions may be pathogenic. For example, the DNA might code for a deadly toxin protein. Other DNA might spread exponentially through populations. It is a problem of paramount importance to grant DNA synthesizers the capacity to perceive

[4]These four molecular building blocks of DNA will be variously referred to as bases, nucleotides, and base pairs, depending on context. "Base" is usually used when referring to the single small molecules A, T, C, G, e.g. when discussing adding a new one to a growing strand of DNA. "Nucleotide" is frequently used when counting bases, e.g. a sequence of 4 nucleotides, or 20 nucleotides. (This may also be referred to as a "tetramer" and a "20-mer," respectively.) "Base pair" is most often used like a unit, and has the abbreviation "bp." For example, the genome of the SARS-CoV-2 virus is about 30,000bp long.

whether the products they are asked to create might be extremely dangerous, and to opt out of synthesizing these products if necessary. The platform presented here is designed to be compatible with every future synthesizer, and to provide this DNA screening functionality against a live-updating global database.

By applying leverage where it will have the greatest long-running impact, at the level of platforms, this work contributes in a small way to the ongoing program to shape molecular biology into a field that makes intelligent use of open-source software to bring about powerful positive change in medicine, materials science, chemistry, energy, and every other application to which biology may be relevant, while avoiding and discouraging the powerful negative effects that might come about if the tools of molecular biology are misused.

Chapter 2

A Platform for High-Throughput Robotic Bioautomation

2.1. Motivation

A comparison could be drawn between the pre-industrial-revolution norms for production and the typical way biological research is performed today. Methods are executed by hand, heavily dependent on specialized knowledge hidden to the outside world, sometimes only accessible to those who train under established molecular biologists in a relationship dynamic reminiscent of that of masters to journeymen. Furthermore, this working knowledge is not captured in publications. Anecdotally, only a few researchers who did not directly train under the inventors of the Phage Assisted Continuous Evolution (PACE) directed evolution technique[46], using apparatus familiar to the author and many collaborators on this thesis, have ever succeeded in running a PACE evolution; the vast majority of publications on PACE are from the same few labs, because hands-on knowledge is primarily passed from senior to junior individuals. Progress in molecular biology is often limited by the all-too-common issue of recreating what another scientist has done before in order to build on it, the infamous "reproducibility crisis"[10]. The rise in use of the phrase “in our hands” added before stating the results of attempts to use others’ work in experiments helps under-

score the normalcy of irreproducibility in this field. (Contrast, say, referring to the behavior of a software library in computer science, or a 2D airfoil in aerospace engineering, with the qualifier "in our hands," as if the results should vary.) Platform innovation may be able to accelerate molecular biology toward a transformation similar to the industrial revolution. Afterward, we might have more interchangeable biological parts, fully automated try-retry experimental policies defined on an abstract level like quality-control procedures in factories, and so on. However, despite its promise, no one can be criticized at present for continuing to wait a while longer before embracing lab automation.

New biology labs contemplating an infrastructure upgrade to achieve partial or full automation of their operations face some critical barriers today. Of course, there is the matter of cost; liquid-handling robots like Hamiltons typically run from hundreds of thousands to millions of dollars each. Still, labs often make such large purchases for other crucial infrastructure, such as upgraded HVAC and air filtration systems, autoclaves, cryogenic refrigerators, sequencers, and multimodal microscopes. One difference between laboratory automation tools and these other items is that, while the standard items have well defined uses and tend to be very reliable, investing in a new automation tool represents a great risk. Consider the problem of finding personnel to operate the robots. While any junior researcher may be shown how to use the standard lab infrastructure by almost anyone else on staff or in a later stage of their academic program, robotic operation is currently a highly specialized skill usually known only to a small fraction of a lab's occupants. To use a familiar example, Hamilton robots are programmed either directly, in the esoteric Hamilton Scripting Language (HSL), or using Venus, a drag-and-drop method composer software, which compiles to HSL as a target. Weeks of training are required for most people to grasp Venus fully, regardless of whether they are wet lab biologists or experienced programmers, and even after years of working with Hamiltons, according to experience, almost no one at all outside of Hamilton Company is fluent in HSL. A similar situation exists for other liquid handling robot companies, which are protective of the intellectual property intrinsic to their individual and separate solutions to the problem of control. New adopters of

lab automation can expect to run into difficulties when hiring technicians with the right experience.

The problem of finding programmers for liquid handling robots can be resolved by making robot control accessible via a well-known programming language that hundreds of thousands or millions of people already know, of which there are several: Java, Javascript, Python, C/C++/C#, Ruby, R, Swift, MatLab, Go, and others. Python was chosen for its well-known suitability as "glue code" to combine disparate systems, and because it is the most popular language in the world by most metrics. The majority of people with basic programming experience will be partly literate in Python, and know how to read Python module documentation (see App. E) to find out how to use it, whereas they certainly will not know any HSL or how to use Venus.

Beside its inscrutability, HSL/Venus suffers other shortcomings that limit the utility of a lab robot. The most basic is probably its cosmological model, i.e. what is assumed to be the center of its "universe," the center of control and execution, and what is considered external or peripheral. The Venus software assumes it is the main process executing on the host computer to which the robot is attached via a serial connection. In order to integrate other peripheral devices with Hamiltons, Venus must initialize the devices and issue all commands. For example, our ClarioSTAR multispectral microplate reader, which we use for feedback control of cultures and real-time data monitoring, is controlled via programming blocks in Venus. In Venus, plate reading is a sequential, or "blocking," operation: the plate reader is instructed to open its tray; the robot moves a microplate into the reader with a robotic arm; the plate reader closes its tray and performs the measurement, usually for several minutes; the tray opens; and then the arm returns the tray to the robot deck. Suppose the programmer would prefer that the robot take some actions on deck during the minutes of wait time for the reader measurement, a parallelization optimization that turned out to be critical to our success in certain scenarios, discussed later. Because Venus is the cosmological center, we are subject to the constraints of how the plate reader interface code was written, and so (without significant effort to write raw HSL, and with no certainty of

success) we cannot perform tasks while the reader is running.

A more general model imagines the robot as a large peripheral, very complex, but in essence no different from a printer or external drive. The plate reader is another such independent peripheral device. It is not generally expected that a printer driver will issue commands to the host operating system and direct connections to and from unrelated peripheral devices, send emails, etc. By moving the focus away from Venus, and reorganizing our code to support asynchronous or semi-synchronous execution, often trivial in modern languages, we can interact with the robot freely while the plate reader is performing measurements. This is not to say that no software that can independently operate the robot with no external support, like Venus, should exist; it is only to suggest that there should be some simple and accessible means of controlling the robot that casts it as a peripheral, "exposing" its functionality to higher-level processes with arbitrary functionality, algorithms, data structures, etc. of their own.

Possibly the most general connectivity model for the laboratory robot is to pose it as a network device. This model imagines the robot as an entity that can support multiple simultaneous connections from other computers or devices, executing some in parallel when possible, queueing others, and returning information about its state and recent actions in real time when requested. The work presented here does not reach this level of sophistication, but rather occupies an in-between space. Pyhamilton will hopefully provide a solid foundation for the development of such a network model. Initial steps in this direction can be found in the PyLabRobot project, which builds on and supersedes this work.

To make a contribution to the transformation of microbiology from pre-industrial to post-industrial, and to address much frustration shared both between laboratory team members and corresponding researchers at the level of software sophistication available when attempting to construct relatively simple controllers, e.g. those that depend on data continuously fed back from their own earlier operations, the author has invested time into building new software tools that make better use of existing hardware for bioautomation. Specifically, this work regards the first open-source, real-time implementation of a control layer

over a Hamilton liquid handling robot in a popular programming language, which has been named **pyhamilton**.

2.2. Architecture

Pyhamilton is composed of an eponymous supermodule, and five submodules (App. E). Together, they constitute:

- an automatically managed interface to a Hamilton robot
- a selection of standard actions a liquid-handling robot can perform
- informative error definitions compatible with the Python exception system
- a framework for defining labware resources on the robot deck

"Automatically managed" means that the `interface` class offered in the pyhamilton package can be used as a context (App. B Figure S1), where entering the context launches Hamilton's Windows executables for driving the robot or displaying a simulation, a local HTTP server, and a message queue, and exiting the context gracefully closes these programs and connections with no action necessary from the programmer.

Regarding "labware," examples of labware resources that might be accessed or manipulated by the robot include pipette tips, liquid reservoirs and microplates of 12, 96, 384, or 1536 wells each. When errors inevitably occur during debugging or weeks-long experiments, a control flow can be established as desired (App. A Extended Data Fig. 4) for handling errors, notifying experimenters, or shutting down. Exiting the context due to an exception also triggers graceful cleanup of all pyhamilton connections. Pyhamilton's modular architecture enables more complex and integrated experiments (App. A Fig. 1, Fig. 4, App. B Fig. S4) than would be practical ordinarily.

In one example, discussed in more detail in Section 3.2 and [28], a simulation of bacteria growing exponentially in a plate was created and validated first. This was followed by a noise model of the robot's minor errors in liquid transfer volumes, along with a noise

model for measurement error in the plate reader. Combining the models with the simulation, an overall system model for bacterial growth on the deck of the robot as measured by the plate reader in terms of optical density was established and qualitatively verified. Next, a discrete-time nonlinear control loop was developed that models the growth constant of the bacteria with no initial knowledge the hidden ground-truth randomly distributed growth constants from the simulation. This composed closed-loop system of bacterial culture plus controller constitutes a turbidostat, or bacterial culture maintained in log-phase growth at a constant optical density. A collection of simulated turbidostats with realistic parameters were validated to correctly converge to arbitrary optical density setpoints, regardless of bacterial growth rate (App. B Fig. S3). Finally, the nonlinear feedback controller was transplanted directly on top of the physical robot, and it was then tuned up to converge to arbitrary optical density setpoints with only minor adjustments to controller constants. As a fortunate add-on, the growth constants (k-values) modeled by the controller turned out to be of scientific interest in their own right, when the environments in which the bacteria grew were modulated along various axes[28]. The simulation also helped bound the number of plates' worth of turbidostats could be maintained, as a function of k-value, given a limited number of robot operations per hour (App. B Fig. S6).

Needless to say, the successful adaptation of this nonlinear controller, developed using a custom simulation, into a real robot method, would not have been possible using Hamilton's Venus, nor practical in HSL. This is not a shortcoming in Venus or Hamilton Company, but rather of the current state of programming resources for bioautomation in general, for which open-source norms have not yet been established, hampering collective progress. The following applications show what is possible with pyhamilton. They also hint at what others may do with similar control possibilities when open-source automation is commonplace across all types of robotic systems for molecular biology.

Chapter 3

Applications in High-Throughput Automation for Molecular Biology

In the following adapted publications from Nature Methods and Molecular Systems Biology, all code for robot operations was written using pyhamilton.

3.1. Application to Biomolecular Evolution [1]

Evolution occurs when selective pressures from the environment shape inherited variation over time. Within the laboratory, evolution is commonly used to engineer proteins and RNA, but experimental constraints have limited the ability to reproducibly and reliably explore factors such as population diversity, the timing of environmental changes and chance on outcomes. We developed a robotic system termed phage- and robotics-assisted near-continuous evolution (PRANCE) to comprehensively explore biomolecular evolution by performing phage-assisted continuous evolution in high-throughput. PRANCE implements an automated feedback control system that adjusts the stringency of selection in response to real-time measurements of each molecular activity. In evolving three distinct types of biomolecule, we find that evolution is reproducibly altered by both random chance and

[1] Adapted from Erika A. DeBenedictis, *et. al.* Systematic molecular evolution enables robust biomolecule discovery[36], on which Gretton, D. W. is a co-author.

the historical pattern of environmental changes. This work improves the reliability of protein engineering and enables the systematic analysis of the historical, environmental and random factors governing biomolecular evolution.

3.1.1 Real-time monitored, flexible, automated environmental control

Biologists have long sought to understand the effects of history and the environment on evolutionary outcomes. The long-term evolution experiment set the gold standard for characterizing genomic evolution at scale: tracking 12 separate Escherichia coli populations over 65,000 generations[13, 56]. However, it has proven difficult to gather datasets of similar scope for the evolution of individual biomolecules. Laboratory protein evolution traditionally uses discrete rounds of targeted in vitro mutagenesis and selection[136], sacrificing throughput, replicates and trajectory length in favor of restricting mutations to a single gene or pathway within an otherwise fixed environment. High-throughput molecular evolution methods are needed to quantify the effects of history and environmental selection pressures on single-gene evolution[110] and enable more robust biomolecular engineering.

Continuous directed evolution methods accelerate the process of diversification and selection by coupling gene function to the fitness of a rapidly replicating organism, such as a bacteriophage[46], mammalian virus[17, 44] or a yeast plasmid[33, 108]. The power of these techniques comes from the ability to rapidly cycle through dozens of generations, and they have been used to quantify mutation rate[79], track evolution pathways[39, 108], and rapidly engineer diverse proteins including highly specific nanobodies[44] and improved CRISPR effectors[69]. While valuable additions to the biomolecule evolution toolkit[67, 131, 139], these experiments are laborious, limited by low throughput and difficult to execute with precise environmental control. Extracting fundamental insights about biomolecular evolution and systematically producing evolved biomolecules with desired activities requires comprehensively sampling the parameters known to influence the results of evolution—including the initial genotype[32, 74, 94] and the evolution environment[3]—in high replicate[142].

We present a new approach that combines the speed of continuous evolution, precise control of the environment and the throughput required to evolve many independent populations in parallel. Whereas classic evolution 'replay' experiments[19] and existing continuous evolution methods[46] evolve a single population contained within a continuous-flow bioreactor (App. A Fig. 1a), we developed a high-throughput plate-based method and applied it to a widely used evolution technique, phage-assisted continuous evolution (PACE)[46]. In our method, evolving bacteriophage populations are housed within 96-well plates on a high-throughput liquid-handling system, which is controlled using Python[28] to achieve the same host bacteria flow-through rates as continuous-flow bioreactors. This system enables continuous evolution in 96-plex using commonly available robotic equipment and open-source software[28]. It offers several distinct advantages over existing methods[67, 131, 139] including scalability to hundreds of parallel populations, historical sampling in 96-well plate format, real-time monitoring of any molecular fitness output rather than just growth effects[11, 139] and precise temporal and chemical control of the environment. We use this platform to identify improbable mutations in a single molecular evolution by performing the experiment in 96-plex, and next demonstrate that the inclusion of controls, replicates and many diverse initial genotypes increases the likelihood of successful evolution and improves interpretation of results. Further, miniaturization enables us to precisely fine-tune the chemical environment of each experiment using reagent-limiting compounds[11]. Using real-time activity monitoring, we implement a feedback control system in which the stringency of the bacterial host strain is autonomously adjusted in response to measurements of biomolecule activity, robustly producing evolved variants by eliminating failures associated with improper selection strength. When combined with high-throughput sequencing, we are also able to distinguish both contingent and deterministic evolution outcomes. Finally, we present an optimized method using limited consumables, capable of week-long multi-path experiments with only once-per-day researcher intervention. Thus, the high-throughput continuous evolution platform we describe enables systematic exploration of the factors responsible for evolutionary outcomes.

3.1.2 Development of a systematic evolution platform

We began by developing a 96-well plate-based method, wherein 500-500-µL cultures of evolving M13 bacteriophage (App. A Fig. 1a) are serially diluted with fresh host bacteria twice per hour using an automated liquid handler (App. A Fig. 1b). To enable the pipetting speeds required to approximate continuous flow, we developed a robotic Python interface[28] that precisely times the distribution of bacteria, the addition of chemical stimuli, the sampling of populations for real-time monitoring and historical sample preservation (App. A Extended Data Fig. 1). Integrated real-time measurement of luminescence, fluorescence and turbidity enables activity-dependent fitness tracking, which we show is more precise than monitoring turbidity alone[139] (App. A Extended Data Fig. 2), Thus, a fluorescent or luminescent reporter can be coupled to either the presence of phage or to the direct activity of the evolving biomolecule itself. We refer to this platform as PRANCE. Each evolution round is initiated by sterilizing a bacterial culture reservoir (App. A Extended Data Fig. 3) and adding culture to each population (App. A Fig. 1b and App. A Extended Data Fig. 1b). Bacterial cultures can be sourced from an active turbidostat or chemostat, enabling high-volume experiments, or from preprepared bacterial stock stored at 4 °C, enabling experiments that use many bacterial cultures (App. A Fig. 1b). Accessory molecules (for example, chemical mutagens, stimuli, small molecules) are then pinned to each population, which are monitored in real time by an integrated, automated plate reader that measures not only the population density, but also the fluorescence and luminescence of each population at discrete 30-min intervals. Samples are preserved in 96-well format and retained for downstream analyses such as sequencing of accumulated changes, or in vitro and in vivo activity measurements (App. A Fig. 1b). The system also incorporates error handling, failure-mode prevention and wireless experimenter communication, (App. A Extended Data Fig. 4), and is optimized for minimal human intervention (App. A Extended Data Fig. 1c,d). The fast iteration time of PRANCE enables dozens of rounds of evolution per day, comparable to traditional PACE, with the ease and throughput of a plate-based format. We first demonstrated the real-time activity-monitoring capability by propagating

M13 bacteriophage encoding T7 RNA polymerase (RNAP) in place of the pIII phage coat protein using host bacteria expressing pIII and a luminescence reporter (luxAB) under the control of a T7 promoter in 48 independent populations (App. A Fig. 1c). We observed T7 RNAP-dependent luminescence in all samples in less than 4 hours from both 37 and 4 °C culture (App. A Fig. 1d), showing that PRANCE can reproducibly monitor real-time reporters of fitness.

3.1.3 Multiplexing identifies previously inaccessible genotypes

While current directed evolution methods suffice when the primary goal is to engineer a single functional protein, they are limited in their ability to probe the randomness and reproducibility of any given biomolecule evolution and characterize the ensemble of possible outcomes. We wondered if a well-studied evolution[39, 46, 79] could provide new outcomes if sufficiently sampled. First, to measure the stochasticity of evolution, we evolved the T7 RNAP to initiate on the T3 promoter and performed the evolution in 90-plex. In this experiment, host bacteria contain an inducible mutagenesis plasmid[8] and an accessory plasmid-containing pIII and luxAB under the control of the T3 promoter (App. A Fig. 2a). With 500-µL populations typically harboring 108 infected cells per ml experiencing high mutagenesis, each population should traverse single-mutation fitness valleys to explore a large fraction of double mutants each day[8]. We inoculated 96 total populations with or without T7 RNAP-expressing ΔpIII phage (with six no-phage controls) and tracked their progress in real time with luminescence (App. A Fig. 2b). We found that bacteria sourced from 4°C and mutagenized on-deck perform similarly (App. A Extended Data Fig. 5a) and tracked absorbance depression (App. A Extended Data Fig. 5b). This high-throughput exploration of the evolution of the T7 RNAP, with >5× more parallel populations than the largest previously reported experiment[79], allowed us to measure the frequency and reproducibility of the emergence of different genotypes. Both novel and previously reported mutations were observed. In addition, we quantified the elapsed evolution times and found the distribution to be logistically distributed (goodness of fit, CvM stat 0.017,

Kolmogorov–Smirnov test 0.046) (App. A Fig. 2c and Extended Data Fig. 5c,d), consistent with only a single mutation (N748D/S or M219R) being required for improved activity (App. A Fig. 2d). The single M219R mutation, which exhibits substantially delayed emergence relative to N748D (chi-square P=0.0037)(App. A Fig. 2e), has not been previously reported despite the many previous iterations of the T7 RNAP evolution. This may be partly due to N748D/S resulting from a transition mutation (A $\rightarrow$ G), whereas M219R results from a transversion mutation (T$\rightarrow$G), which occurs less frequently[8]. Thus, systematic high-replicate evolution allows for the extensive profiling of evolutionary reproducibility[142] and enables deeper sampling of less accessible genotypes that cannot be readily identified from a single population.

3.1.4 Miniaturization enables reagent-limited evolution

As traditional PACE uses continuous flow to constantly refresh host bacteria and consumes large quantities of media, it has previously been infeasible to evolve biomolecules in environments that require small molecules that are difficult to synthesize or are new[106]. PRANCE reduces each bioreactor volume by 100-fold, thus making small molecule-dependent environments more feasible and controllable. To demonstrate these capabilities, we modified an established evolution of the pyrrolysine aminoacyl-synthetase (PylRS) to incorporate noncanonical amino acids (ncAAs)[22], using substantially lower quantities of ncAAs than previously reported. To enable multiplexing of diverse transfer RNA (tRNA)–PylRS pairs, we encoded a PylRS variant and a UAG-containing tRNAPyl within the M13 phage genome and inserted a UAG amber stop codon within the pIII phage coat protein along with a luciferase reporter expressed from host bacteria (App. A Fig. 3a). Thus, phage proliferation and luminescence are both directly coupled to suppression of the UAG amber codon via ncAA incorporation (App. A Fig. 3b). We used three pyrrolysine synthetases—chPylRS, an evolution intermediate (chPylRS-IP)[22] and an evolved variant (chPylRS-IPYE)—and quantified their ability to incorporate Boc-lysine as measured by ncAA-dependent kinetic luminescence activity (App. A Fig. 3c) and amber codon-dependent phage enrichment

(App. A Fig. 3d). We then used PRANCE to evolve the PylRSs to incorporate Boc-lysine by inoculating eight populations with phage encoding either chPylRS and chPylRS-IP in quadruplicate. Aminoacyl-synthetase (AARS) evolution is prone to the emergence of 'cheaters', that is promiscuous charging of canonical amino acids[125]. To monitor the emergence of nonspecific AARSs, we included eight populations with phage encoding each variant but in the absence of ncAAs; phage propagation under these conditions would indicate the evolution of nonspecific charging of canonical amino acids. Together, we monitored luminescence and absorbance across a total of 24 populations over 36h of evolution. We observed propagation of both chPylRS- and chPylRS-IP-encoding phage in the presence of Boc-lysine (App. A Fig. 3e), and identified novel genotypes with previously unidentified mutations (App. A Fig. 3f,g). The lack of luminescence in the absence of ncAA indicates that cheater AARSs are unlikely to emerge under the evolution conditions used (App. A Fig. 3e). Thus, the inclusion of control populations—typically neglected in directed evolution experiments due to throughput limitations—enabled the extraction of additional information previously unobtainable. This capability can be used to determine whether or not negative selection against promiscuous activity[22] is necessary. Additionally, the automated addition of Boc-lysine to 12 evolving populations over 36h of PRANCE consumed less than 100mg of total compound, nearly ten times less than what would have been required for a single population within a bioreactor. Given this substantial reduction in reagents, PRANCE enables multiplexed and well-controlled evolution experiments with fine control over the chemical environment using molecules that are too expensive (for example, 4-azido-Phe, $2,500 per gram[25]) or rare (for example, pyrrolysine[119]) to be used with traditional continuous-flow bioreactors.

3.1.5 Simultaneous evolution of dozens of biomolecules

Previously, we used PACE to evolve tRNAs[35] capable of decoding quadruplet codons[5, 90] toward the goal of engineering a four-base codon translation system[37, 130]. To evolve new quadruplet tRNAs (qtRNAs), we inserted a quadruplet codon (AGGG) into pIII and

generated a variety of qtRNA-encoding, pIII-deficient phage (App. A Fig. 4a). In the absence of a functional qtRNA, the quadruplet codon generates a frameshift, truncating pIII and precluding phage propagation (App. A Fig. 4b). The success of qtRNA evolution can depend on which tRNA paralog is used to initiate evolution, highlighting the importance of studying a variety of starting genotypes. Here, we used PRANCE to simultaneously identify many functional qtRNAs by subjecting a full set of 20 different paralogs to evolution within a single experiment. We first replicated the evolution of six TAGA-decoding qtRNAs, and observed similar genotypes as previously described[35] (App. A Extended Data Fig. 6). Next, we initiated PRANCE by seeding 48 populations in an optimized configuration (App. A Extended Data Fig. 7) with phage encoding 20 different qtRNA paralogs corresponding to every canonical amino acid containing a library of randomized anticodons (NNNN) (20 populations); phage encoding eight different qtRNA paralogs (Ala, Glu, Gly, His, Arg, Ser, Thr and Trp) each with directed AGGG frameshift anticodons (24 populations) or no-phage controls (four populations). We then tracked luminescence of all 48 populations over 36h and found that phage encoding Gly, His, Ser, Arg, Thr and Trp qtRNA paralogs successfully decoded quadruplet codons (App. A Extended Data Fig. 6e). Indeed, when subcloned, several of the isolated qtRNAs exhibited improved activity (App. A Fig. 4c,d), although further characterization would be required to determine the amino acid identity of the evolved qtRNAs[37]. These results are consistent with the observation that only some tRNAs are capable of improvements to this biomolecular activity, highlighting the importance of genotype diversification in the success of evolution. Notably, a single PRANCE experiment evolved multiple AGGG-decoding qtRNAs, which would have previously required dozens of individual PACE experiments.

3.1.6 Feedback evolution of activity-diverse biomolecules

Although we successfully identified AGGG-decoding qtRNAs with improved activity, we also observed a high failure rate. Many of the qtRNAs with low initial activity (Ala, Cys, Asp, Phe and so on) never evolved, but rather experienced an experimental failure mode

referred to as 'washout' in which the effective population size decreases to zero (App. A Extended Data Fig. 6e). Additionally, we observed that qtRNAs with high initial activity (Arg, Trp) maintained population size and triggered luminescence but did not acquire mutations. These results indicate that selection was too stringent in some cases and too lenient in others—both common directed evolution failure modes. We hypothesized that the ability to dynamically tune selection stringency in accordance with population fitness would improve the likelihood of successfully evolving biomolecules from diverse starting points, by both reducing the possibility of phage washout and maintaining selection pressure on high-activity variants. To improve likelihood of evolution success, we developed a feedback control system[100, 115] that adjusts the stringency of selection by modifying the host bacterial strain in response to a real-time analysis of molecular activity-dependent luminescence. As a model system, we selected four qtRNA paralogs (Phe, His, Ser and Arg) that decode the TAGA quadruplet codon, are known to have improved variants and differ greatly in initial activity. We sought to use feedback control to evolve all four qtRNAs in a single experiment. First, we characterized three bacterial APs that confer different levels of selection pressure. The most 'lenient' of these APs encodes T7 RNAP containing two quadruplet codons, with the T7 promoter driving production of pIII and luxAB (App. A Fig. 4e). We also characterized more stringent APs containing either one (moderate) or two (stringent) quadruplet codons directly in pIII (App. A Fig. 4e) and found that these APs adequately cover phage enrichment space (App. A Fig. 4f). We next evolved these qtRNAs on each of the three individual bacterial sources separately, under static selection. The variant with the highest initial activity, qtRNAArgTAGA, only evolves under high selection pressure. Conversely, under lenient pressure all phage propagate, but only the qtRNAs with the lowest initial activity experience selection and acquire mutations (App. A Fig. 4g). None of the three levels of stringencies could evolve all four qtRNAs. To customize selection pressure, we implemented automated feedback control in which the bacterial source of each population is adjusted in response to real-time measurements of fitness as measured by luminescence (App. A Fig. 4h and Methods). This strategy success-

fully propagated phage populations encoding all four qtRNAs to the end of the 36-hour experiment (App. A Fig. 4i). Additionally, by measuring clonal variant activity from each of the eight feedback-controlled populations using a luciferase reporter, we found that all eight populations had evolved qtRNA variants with improved translation efficiency (App. A Fig. 5a). Thus, feedback control avoided phage washout in all cases while simultaneously exposing high-activity qtRNAs to more challenging evolution environments, in a short time window, without researcher intervention. These results demonstrate that feedback control is more robust and less failure prone, enabling the evolution of biomolecules with diverse activities.

3.1.7 Evolution outcomes are determined by temporal dynamics

It is well established that selection stringency can affect the trajectory of evolution[3, 79], but the importance of the timing of those changes has remained largely unexplored. In nature, changes in the physical environment are complex, and can be random or even exhibit mutual dependence with population fitness (for example, in predator–prey dynamics[63]). Thus, we wondered how static versus dynamic selective environments would affect the trajectory of single-biomolecule evolution within the laboratory. Due to the small size of qtRNAs, we used next-generation sequencing to characterize the evolutionary history of 32 populations as they were subjected to different temporal perturbations to their selective environment. We tracked the genotypic abundance relative to population size of four qtRNA paralogs over 36 hours, under four selection schedules in duplicate as they underwent evolution under static (lenient or stringent), dynamic yet unresponsive (discrete) or feedback-controlled (responsive) stringency modulation. We found that the selective environment affects whether and how quickly evolved variants reach fixation in a population. For example, the variant qtRNAArgTAGAU43 can only be evolved using stringent selection because the tRNA genotype initiating that evolution has high initial fitness (App. A Fig. 5a,b). Accordingly, all qtRNAArgTAGA evolution experiments arrive at a convergent solution, but the speed of evolution depends on how quickly the stringent selective envi-

ronment is introduced (App. A Fig. 5b). Further, qtRNAPheTAGA, qtRNAHisTAGA, and qtRNASerTAGA are each vulnerable to washout at high stringency due to lower overall fitness (App. A Fig. 5c–e); thus, evolution only occurs for these qtRNAs in environments that are initially lenient. During qtRNAPheTAGA evolution, an accessible and convergent single-point mutant (32 A) arises that is highly fit in lenient and moderate selective environments, resulting in purification and population size growth in those environments (App. A Fig. 5d). The qtRNAHisTAGA evolution is more complex, containing several variants with elevated fitness composed of mutations to base 32 together with modifications in the variable loop (App. A Fig. 5e,g). Finally, the synergistic epistasis between mutations in qtRNASerTAGA (App. A Fig. 5a) makes purification of the highly improved qtRNASerTAGAA32-C38 mutant[90] less robust: only one responsive environment completely purified this variant (App. A Fig. 5c). Together, these data show how the unique fitness landscape of each biomolecule determines the dynamics of its evolution in different selective environments. We also observed that the dynamics of environmental changes can affect the phylogenetics of evolution. Unlike Arg and Phe-qtRNA evolution, which each appear to deterministically converge on particular high-activity variants irrespective of changes in stringency timing (App. A Fig. 5b,d), we found that the genotypes resulting from qtRNAHisTAGA evolution are particularly sensitive to historical changes in the environment. The discrete selection schedule resulted in wide genotypic variety, with seven unique genotypes each reaching >10% of phage population share at some point during evolution (App. A Fig. 5f). In this schedule, the arbitrary introduction of moderate stringency (t=12h) reproducibly enriches intermediately active variants (C32 or G32) and their phylogenetic descendants with variable loop mutations (G32-Δ48, C32-A48, C32-Δ48, C32-Δ47) (App. A Fig. 5g), before converging on a globally optimal variant. Conversely, we see that during responsive evolution, where moderate stringency is delayed until the population is sufficiently fit (t = 18 h), a single active point mutant (Δ45) emerges as the predominant variant without widely exploring other genotypes at high population abundance (App. A Fig. 5e,f). These data show that seemingly small perturbations to the

historical selective environment, whether arbitrary or in response to a changing ecosystem, can drive purification of distinct genetic variants that are either moderately or highly fit[15]. Collectively, these results demonstrate that although single-biomolecule evolution may appear deterministic on simple fitness landscapes with a sharp peak, more complex landscapes may produce outcomes contingent on seemingly inconsequential events[19].

3.1.8 Long-running evolution with PRANCE

The longevity of most PACE experiments (>100 hours) requires the platform to be capable of performing long-running, multi-trajectory evolution experiments. Due to the large quantities of consumables used by liquid-handling robots, the need for frequent researcher intervention (for tip-replenishing) and the ongoing tip shortage resulting from the COVID-19 pandemic[128], we developed an optimized method capable of sterilizing and reusing tips on the robot deck (Supplemental Video 1). This optimized method uses approximately five boxes of tips per day, and can be run for over a week at a time with user intervention only once every 24 hours. During method optimization, we observed that different robotic configurations introduce varying amounts of cross-contamination when propagating highly active phage (App. A Extended Data Fig. 8a,b). To demonstrate the capabilities of this method, we first validated that tip reuse and sterilization introduced no quantifiable cross-contamination within 12 hours (App. A Extended Data Fig. 8c), indicating that tips could be replenished once or twice per day. We then used this technique to enable a 10-day evolution in which we evolved T7 RNAP to bind eight new promoters (App. A Fig. 6a,b). During this experiment, we tested three techniques to maintain large population sizes during long-running experiments: allowing the evolution to proceed without intervention (no pulse); spiking the population with bacteria expressing pIII under the phage shock promoter (psp) that enable activity-independent phage propagation periodically every 12h (12h pulse) or spiking only before transitions to new evolution stringency (pretransition pulse). We evolved 32 populations for 240 hours (10 days) in a single, uninterrupted experiment (App. A Fig. 6c). During this time, no cross-contamination in the eight no-phage control

wells was detected. We found that the phage titer maintenance schemes affected the genotypes that evolved (App. A Fig. 6d). To quantify activity, individual variants containing all of the dominant mutations from each replicate (App. A Fig. 6d and Supplementary Table 3) were subcloned into plasmid reporter constructs in which LuxAB was driven by the TP6, -3 variant or -5 variant promoter. The activities of 24 total subclones were then quantified by luminescence and compared to wild-type (WT) T7 RNAP on each respective promoter (App. A Fig. 6e and App. A Extended Data Fig. 9). Most variants obtained in the T7→T3→-3/-5 trajectories were found to exhibit between 10 and 20-fold higher activities than WT T7 RNAP on the same promoter. In addition, we found that the stringency conditions interacted with the evolution goals; for example, the reduced population size of 'No pulse' in the SP6 trajectory was the only condition that converged on mutations to E222, a residue associated with nonspecific promoter binding[39] (App. A Fig. 6e). The highest activity variant obtained from -3 variant evolution (prepulse, population no. 2, App. A Fig. 6e) was also the only population to reach saturation with a novel E218A mutation even though many populations obtained the M219R/K solutions described above (App. A Fig. 6f). Thus, PRANCE enables the seamless exploration of complex, multi-mutational pathways that evolve over the course of many days and require several intermediate evolution goals.

3.1.9 Advances

PRANCE provides a number of advances over existing methods. The platform has higher throughput, supporting hundreds of diverse evolution conditions simultaneously. The 100× volume reduction and robotic compound pinning capabilities support evolution in chemical environments that were not previously practical due to reagent cost[25] or synthesis limitations[119], e.g., those involving the use of small-molecule fluorescent reporters of metabolism[137], pH[134] or CO2[140], or supplementation with complex intermediates[26]. Further, the system can be used with either continuously grown or chilled host cells to enable evolution at diverse temperatures. The universal 96-well format enables sample

preservation for immediate or downstream analysis of historical genotypic and phenotypic changes. Additionally, the system is capable of performing feedback-controlled evolution: we demonstrated feedback control triggered by luminescence, and the system is extensible to many activity-dependent fluorescent reporters (transcription, quorum sensing, solubility, protease activity, splicing and so on) as well as any measurement that can be integrated robotically, such as PCR, binding measurements or orthogonal in vitro assays. Similarly, the system is amenable to experimentation with other feedback control algorithms[100] or machine learning-guided adaptive control. With the improved availability and decreasing cost of high-throughput equipment, this approach is an increasingly accessible option for many laboratories. Finally, although we demonstrate the use of this platform with bacteriophage, we believe that it is practicable to accommodate continuous evolution platforms in eukaryotes such as yeast[108, 139] or possibly mammalian–virus systems[17, 44]. From a strictly engineering perspective, our results highlight how multiplexed evolution enables engineering that is comprehensive, robust and systematic. With PRANCE, we sampled all parameters known to affect evolution outcomes: the initial genotype, the evolution environment and events that happen by chance. Evolution in high throughput thus improves our ability to sample the ensemble of possible outcomes. For example, even though the evolution of T7 RNAP has been extensively reported[39, 46, 79], we continued to identify novel genotypes with systematic parallel sampling. We observed that initiating experiments with a diverse set of tRNA paralogs was a robust strategy for evolving highly active variants and further identified that feedback control enables side-by-side evolution of biomolecules with diverse initial activity, in the absence of preexisting knowledge. Finally, we found that populations evolving with differentially timed stringency perturbations can yield distinct solutions, highlighting the use of examining many evolution schedules. By improving the reliability of laboratory evolution, PRANCE enables the reinvention of directed evolution as a more robust and reproducible engineering discipline. Systematic evolution could be of great use to disciplines ranging from natural evolution to epidemiology. Indeed, this platform is capable of collecting datasets about molecular evolution that approach the size and

scope of the long-term evolution experiment dataset of whole-genome evolution[13, 56] by examining numerous individual populations and environments over many generations. By performing a single evolution with many replicates, we can characterize the reproducibility and stochasticity of single events, a longstanding challenge for evolutionary biologists. As selection schedules reconstitute the natural process of evolving related biomolecules in different selective environments, PRANCE also offers the opportunity to better understand the relationship between time, stringency, the environment and the traversal of a fitness landscape. More generally, the ability to systematically explore evolutionary outcomes across many populations may help resolve controversial questions in evolutionary biology, such as the friction between determinism and contingency. For example, we find that the timing as well as the nature of environmental changes can reproducibly affect the genotypic outcomes of individual biomolecules (that is, contingency), but outcomes vary according to the particular selective landscape (that is, determinism). Thus, PRANCE provides the opportunity to answer fundamental questions in evolutionary biology, recapitulate naturally occurring environmental changes and simulate perturbations to these environments—all within the laboratory.

3.2. Application to Sampling the Bacterial Fitness Landscape [2]

In the following adapted publication from Molecular Systems Biology, all code for robot operations was written using pyhamilton.

Our understanding of complex living systems is limited by our capacity to perform experiments in high throughput. While robotic systems have automated many traditional hand-pipetting protocols, software limitations have precluded more advanced maneuvers required to manipulate, maintain, and monitor hundreds of experiments in parallel. Here, we present Pyhamilton, an open-source Python platform that can execute complex pipetting patterns required for custom high-throughput experiments such as the simulation of metapopulation dynamics. With an integrated plate reader, we maintain nearly 500 remotely monitored bacterial cultures in log-phase growth for days without user intervention by taking regular density measurements to adjust the robotic method in real-time. Using these capabilities, we systematically optimize bioreactor protein production by monitoring the fluorescent protein expression and growth rates of a hundred different continuous culture conditions in triplicate to comprehensively sample the carbon, nitrogen, and phosphorus fitness landscape. Our results demonstrate that flexible software can empower existing hardware to enable new types and scales of experiments, empowering areas from biomanufacturing to fundamental biology.

3.2.1 Application Overview

Comprehensive, well-replicated experiments are foundational to rigorous science, but humans can only perform so many actions simultaneously. One possible solution is automation, which has been widely implemented in biotechnology[6, 50, 118] to facilitate routine tasks involved in DNA sequencing[93], chemical synthesis[82], drug discovery[112], and molecular biology[117]. In principle, flexibly programmable robots could enable diverse experiments requiring conditions and replicate numbers beyond the capabilities of

[2] Adapted from Emma J. Chory, *et. al.* Enabling high-throughput biology with flexible open-source automation[28], on which Gretton, D. W. is a co-author.

human researchers across a range of disciplines[62, 75, 126]. However, existing software for liquid-handling robots focuses narrowly on automating protocols designed for hand pipettes, while foundry languages such as Antha and remote labs such as Emerald Cloud focus on automating workflows rather than expanding experimental limits. As such, even labs with well-established high-throughput infrastructures struggle to utilize the full potential of their robots, precluding many complex experiments that require flexible programming[6].

Bioautomation lags behind the advancing field of manufacturing, where robots are expected to be task-flexible, responsive to new situations, and interactive with humans or remote management systems when ambiguous situations or errors arise[6]. A key limitation is the lack of a comprehensive, suitably abstract, and accessible software ecosystem[9, 85, 129]. Though bioinformatics is increasingly open-sourced[29, 51], bioautomation has been slow to adopt key practices such as modularity, version control, and asynchronous programming. To enable flexible high-throughput experimentation, we developed Pyhamilton, a Python package that facilitates high-throughput operations within the laboratory, with protocols that can be easily shared and modified. Further, Pyhamilton allows liquid-handling robots to execute previously unimaginable and increasingly impressive methods. With this package, users can run robot simulations to troubleshoot and plan experiments, schedule experimental processes, implement error handling for quick troubleshooting, and easily integrate robots with external equipment.

Pyhamilton enables Hamilton STAR, STARlet, and VANTAGE liquid-handling robots to be programmed using Python. This allows for robotic method development to benefit from standard software paradigms, including exception handling, version control, object-oriented programming, and other cornerstone computer science principles (App. B Table EV1, Movie EV1). Pyhamilton seamlessly connects with Hamilton robots (App. B Fig S1), can interface with custom peripherals (App. B Fig 1A), and contains unique Python classes corresponding to robotic actions (i.e. aspirate and dispense) and consumables (i.e. plates and pipette tips) (See App. B Dataset EV1). To enable method troubleshooting, Pyhamilton can also simulate methods through Hamilton run control software (App. B

Movies EV2–EV4) and incorporate any Python package (i.e. enabling error notifications via push, text message, or Slack). Finally, in addition to the functionalities we present, researchers can now also develop their own flexible code that may be useful for increasingly specialized applications.

Enabling improved throughput of basic robotic tasks

Complex procedures are built from simple tasks, but the capabilities of a pipetting robot are limited by standard liquid-handling software. For example, an 8-channel head cannot be readily programmed to pipette into two 24-well plates simultaneously, although doing so is physically possible (App. B Fig 1B). This limits many high-throughput assays: automation of methods involving 24-well plates is no faster than hand-pipetting, since robots and researchers pipette one plate at a time. Thus, we first demonstrate that Pyhamilton easily enables pipetting of liquids over two 24-well plates simultaneously (App. B Fig 1B and C, Table EV2), thereby doubling the speed (App. B Movie EV5). This can be critical for bacterial assays involving heated liquid agar which solidifies quickly. This simple example demonstrates the advantages of making full use of the robot's mechanical capabilities, freed from software constraints.

Enabling liquid transfers requiring complex calculations

Despite having far greater physical capabilities than a fixed-volume multichannel pipette, it is difficult to implement complex liquid transfer patterns involving different volumes on a robot because programming using standard software is prohibitively monotonous (App. B Movie EV1). The ability to faithfully execute experiments involving hundreds of different pipetting volumes could enable new types of applications such as evolutionary dynamics experiments examining gene flow[116], population symbiosis[73], sources and sinks[38], genetic drift[54, 78], and the spread of gene drive systems[47, 99] (App. B Fig 1D). We accordingly used Pyhamilton to enable the flexible transfer of organisms between populations in a 96-well plate, using pre-programmed migration rates to simulate geographic

barriers (App. B Fig 1E). A human would have great difficulty performing or programming hundreds of variable pipetting actions in many directions, in any reasonable time frame, without errors. With Pyhamilton, simple abstractions and data structures make this task straightforward. Instead of exhaustively specifying each pipetting step, we specified liquid transfer patterns as matrices and allowed Python to compile the requisite steps. We demonstrate liquid transfer to nearby plates and between adjacent wells to model "flow" or "diffusion" across the miniaturized landscape of a 96-well plate. We then simulate genetic flow by visualizing the point spread of a drop of dye near the center of a plate (App. B Fig 1F and G, Table EV2). The amount of liquid exchanged and the number of wells is arbitrary, defined as a sparse matrix where the rows are source wells, the columns are destination wells, and the values are the fraction of liquid transferred (App. B Fig S2). Each iteration, the robot performs several hundred bi-directional liquid transfers to apply the matrix operations (App. B Movie EV6). Succinct code (App. B Fig 1G) can generate both symmetric and asymmetric diffusion patterns, which could be combined with a phenotypic reporter to experimentally simulate arbitrarily directionally bounded or unbounded migration (App. B Fig 1D) with many model organisms such as E. coli, yeast, or even nematodes.

3.2.2 Enabling feedback control to maintain culture conditions

Though most liquid-handling robots are used to execute a list of precompiled instructions (e.g., assembling reagents for many PCRs), many potential applications require making real-time modifications in response to changing data. For example, a turbidostat is a culture of cells that is maintained at a constant density by making real-time adjustments to the flow rate of media based on turbidity sensing. In practice, this is accomplished with process controls which measure the optical density (OD) of a culture in situ[61, 67]. However, turbidity probes are both costly and not amenable to very high-throughput experiments[64, 120, 131]. Thus, we sought to leverage the flexibility of Pyhamilton to multiplex the maintenance of many bacterial turbidostats by adjusting the vol-

ume of liquid transfers in response to real-time density measurements obtained using an integrated plate reader (App. B Fig 1H–J). The method equilibrates each culture, growing in a multiwell microplate, to a setpoint (App. B Fig 1 I) in response to these measurements by applying a transfer function to calculate the growth rate (k-value) and adjustment volume for each individual well over time (App. B Fig 1J).

Feedback controller algorithm

Bacteria optical density (OD) was modeled to evolve as: $x = x_0 e^{kt}$, where x is the culture OD, x_0 is the initial OD, k is the bacteria exponential growth constant (k-value) in reciprocal hours, and t is elapsed time in hours.

A media replacement cycle is modeled as dilution of a culture by instant uniform mixing with transparent media of a fraction y of its initial volume, which linearly scales its OD x to a new OD x' (e.g. if a 100µL culture is at OD 0.3 and $y = \frac{1}{2}$, then the replacement is modeled as diluting with 50µL transparent media, and the final OD x' is 0.2), summarized as:

$$x' = \frac{1}{1+y}x$$

The culture OD is to be maintained at a constant setpoint, x^{set}. In each cycle $i = 0, 1, 2, \ldots$, each representing a time interval Δt, the turbidostat controller is responsible for producing an output command and state update according to a transfer function:

$$(y_i, \Phi_i) = f(x_i, \Phi_{i-1})$$

where y_i is the new controller output command as a fraction of the turbidostat volume, Φ_i is the new controller internal state, x_i is the present OD measurement, and $f(x_i, \Phi_{i-1})$ is the controller transfer function based on the OD measurement and the previous controller state Φ_{i-1}). The controller state may depend on the history of prior

OD measurements $x0, \ldots, x_{i-1}$ and prior controller commands $y_0, \ldots, y_{i-1}$.

Controller state

A feedback controller with a distinct state was created for each culture. The controller state is a triple $\Phi_i = (x_i, k_i^e, y_i)$: the present OD measurement, x_i; the current estimate of the culture's growth k-value, k_i^e ; and the output command, y_i.

Transfer function

The transfer function updates the three state variables and computes an output. From the model equations, the current k-value, given a new measurement x_i taken an interval Δt after the previous replacement executed, is

$$k_i = \frac{\ln(\frac{x_i}{x_{i-1}}(y_{i-1}+1))}{\Delta t}$$

This k_i contributes to the state k-value estimate k_i^e through a first-order linear filter to dampen the effect of measurement noise. The output to restore the turbidostat OD to the setpoint is

$$y_i = \max(0, \min(y^{max}, \frac{x_i e^{k_i^e \Delta t}}{x^{set}-1}))$$

where the final output y_i is subject to physical limits, being both nonnegative and not greater than the largest volume the robot can move with a pipette tip as a fraction of the turbidostat volume, appearing as y_{max}. After output limiting, y_i is saved in the controller state.

Controller was developed as an abstract Python class and tested in simulation with mechanical and measurement noise models before application in experiments (App. B Figs S3 and S6). Filtered k-value estimates were used to draw conclusions about bacterial growth rates. Source code for implementation can be found at: https://github.com/dgretton/many_basic_turbidostats/blob/master/turb_control.py.

3.2.3 Asynchrony enables high-throughput turbidostats

To maximize the number of turbidostats that can be maintained, we next developed a more complex method which uses asynchronous programming to execute multiple robotic steps simultaneously— in this case plate reading and pipetting (App. B Fig S4). This allows for up to 480 cultures to be maintained with real-time fluorescent reporter monitoring on a single small robot, nearly 20× more than can be readily achieved with multiplexed mini-bioreactor setups[61, 62]. In this method, bacterial cultures are inoculated into 96-well clear-bottom plates and their ODs and fluorescence levels are measured with an integrated plate reader (App. B Fig 2A, Movie EV7). To minimize waste, consumables, and prevent media contamination, we also implemented a cleaning process (App. B Fig 2A): after each media transfer, each tip is sterilized with 1% bleach, rinsed in water, and returned to its housing unit (App. B Fig 2A, Table EV2). To further minimize the possibility of cross-contamination between wells, each culture is assigned its own tip and media reservoir by housing replenishing media within high-volume 96-well plates. We confirmed that this method introduces no measurable cross-contamination by inoculating 96 turbidostats with four different bacterial cultures expressing RFP, YFP, CFP, or no fluorescent protein in a grid-like pattern with no-bacteria controls (App. B Fig 2C).

We then monitored the absorbance and fluorescence levels in real time and maintained the cultures at OD 0.8 for 24 h. We observed no cross-contamination and no growth in the no-bacteria controls during this time (App. B Fig 2 C). We also inoculated the same bacterial strains at 6 different starting densities (OD = 0.0–0.8) and demonstrated that irrespective of initial conditions, the feedback control algorithm equilibrates each culture to its set point within 12 h (App. B Fig 2D). Finally, we confirmed that the method could support culture maintenance of bacteria with varying growth rates (App. B Fig S5A–C), with no measurable back-contamination of media (App. B Fig S5D), for up to 2 days without experimenter intervention (App. B Figs S5E and F, and S6).

3.2.4 High-throughput perturbation analysis of metabolites

We next sought to use high-throughput turbidostat tracking to address an outstanding question in metabolic engineering by systematically mapping the chemical landscape that supports bacterial growth and protein expression. To do this, we surveyed the contributions of carbon, nitrogen, and phosphorus on growth and recombinant protein production by permuting chemical gradients for these metabolites in high-throughput using our multiplexed turbidostat maintenance protocol. These dependencies, while seemingly well-studied, have not been explored in depth. Truly comprehensive mapping requires sufficient conditions, replicates, and controls, long-term maintenance of log-phase growth, and real-time monitoring, each of which is trivial to implement with Pyhamilton.

It has traditionally been thought that cells regulate protein production by allocating their resources to optimize for both expression and growth[83, 97]. However, it has recently been shown that in either carbon-, nitrogen- or phosphorus-limiting conditions, cells are able to fine-tune their ribosomal usage to maintain equal levels of protein[84]. Thus, we wondered whether exploration of the entire metabolite landscape (Fig 3A) could more rigorously identify bacterial growth conditions optimized for recombinant protein production. To do this, we inoculated cultures with E. coli BL21, a strain commonly used for recombinant protein production in metabolic engineering or biomanufacturing, engineered for high constitutive expression of a fluorescent protein (CFP) [111].

In a single experiment spanning 36 h with no user intervention, we simultaneously quantified the equilibrium log-phase growth rates and respective fluorescence levels of 300 individual turbidostats, representing 100 different media compositions in triplicate (App. B Fig 3B). Cells were grown in modified M9 media containing 100 different ratios of carbon, nitrogen, and phosphorus and the cultures were maintained in log-phase growth for 36 h with feedback control. All cultures grew within +/- 20% of the standard M9 media growth rate, with the exception of cultures that were starved of both carbon and phosphorus (App. B Fig 3C). We observed that increases in growth rate are primarily correlated with increases in phosphorus (independent of nitrogen or carbon levels), which is likely a result of

increased DNA synthesis. Further, in phosphorus-limiting conditions, we find that the depressed growth rate can be rescued by supplementing carbon, but not nitrogen, suggesting that carbon precursors are a more limiting reagent than amino acids in metabolism (App. B Fig 3C). Consistent with previously published results[84], we observe that the total amount of protein is generally not affected by limiting carbon or nitrogen, nor by supplementing the cells with excess of either nutrient. However, we additionally find that when phosphorus is limited (0.25X), excess carbon supplementation not only rescues the growth rate of the culture (App. B Fig 3C, Dataset EV2), but also results in an increase in total fluorescence (App. B Fig 3D, Dataset EV2). Since we observe negligible growth defects, this finding suggests that on a per-cell basis, supplementing carbon in phosphorus-limiting conditions (such as in the soil[102, 127] or P-limited lakes[65]) can shunt bacterial metabolism from DNA/mRNA synthesis to protein translation without sacrificing growth. Collectively, these findings demonstrate that Pyhamilton enables researchers to answer rigorous metabolic engineering questions by enabling facile, low-consumable, yet rich hypothesis-generating experiments.

3.2.5 Implications for Automation

Liquid-handling robots have traditionally automated workflows that were explicitly designed for human researchers rather than enabling new high-throughput experimental modalities. Pyhamilton is an open-source Python framework intended for experiments that could never be done by hand, such as protocols that must pipette continuously for multiple days, perform complex calculations about future steps based on real-time data, or make use of hardware that is more sophisticated than any hand-held multichannel pipette.

We showcase these improved capabilities by simultaneously quantifying the metabolic fitness landscape of 100 different bacterial growth conditions to identify ideal conditions for recombinant protein production. Though recent fluidic advances have enabled the maintenance of many continuous cultures[60, 61, 131], our liquid-handling platform can accommodate several times as many. Moreover, liquid-handling systems can easily incor-

porate a plate reader for real-time reporter monitoring, which vastly expands the types of questions that can be approached with facile, multiplex solutions. For example, one could maintain cultures of, and accurately quantify any reporter output for massively-parallel experiments including genetic knockout or CRISPR collections[7, 104], mutagenesis variants[96], or even small-molecule compound libraries[52]. With high accuracy, any suspension culture of mixed populations could be maintained in log-phase growth for days in order to study transient invaders into microbial communities[4] or even microbiome system dynamics[88]. The advent of small-molecule fluorescent reporters for metabolic fitness[138], pH[113, 135], and CO2[141], in addition to the hundreds of fluorescent protein sensors available to the synthetic biology community at large[68, 103], underscore the many potential applications of being able to multiplex and quantify changes in growth, gene expression, and the environment in real-time. Presently, Pyhamilton is only extensible to Hamilton robots. However, since it uses a platform-independent, web-based protocol (HTTP) and common readable data format (JSON) to bridge Python and the Hamilton Scripting Language (HSL) (App. B Fig S1), Pyhamilton could be ported to other biological automation systems that provide an API, such as Tecan or alternative platforms.

As such, Pyhamilton is a small part of an ongoing transition to a paradigm which leverages insights from computer science[9] and applies them to biology. Similar to how Bioconductor[51] and The Biopython project[29] have revolutionized computational biology, bioinformatics, and genomics, our hope is that by making this software open-source and freely available, a community of scientists and developers could begin to similarly transform bioautomation. The experiments we have described represent only a small sampling of many possible Pyhamilton applications. Collectively, they highlight the potential of high-throughput robotic systems to transcend the repetitive processes for which they were conceived and directly address broad questions in microbiology, genetics, and evolution that are beyond the physical capabilities of human researchers.

3.2.6 Methods, materials and models

Robotic equipment set-up and interfacing

A Hamilton Microlab STARlet 8-channel base model was augmented with a Hamilton CO-RE 96 Probe Head and a Hamilton iSWAP Robotic Transport Arm. Air filtration was provided by an overhead HEPA filter fan module integrated into the robot enclosure. A BMG CLARIOstar luminescence multimode microplate reader was positioned inside the enclosure, within reach of the transport arm.

Software

A general-purpose driver method was created using MicroLab STAR VENUS ONE software and compiled to Hamilton Scripting Language (HSL) format. Instantiation of this method and management of its local network connection was handled in Python. A new Pyhamilton-compatible supporting Python package provided an overlying control layer interface to the CLARIOstar plate reader. We used Git to develop and version control the packages and the specific Python methods used for each experiment; our software implementation can be found on github at: https://github.com/dgretton/pyhamilton.

Bacterial assays

For bacterial assay validation, bacterial plaque assays were used to confirm dilutions and agar solidification. Briefly, overnight cultures of S2060 cells (Addgene bacterial strain #105064) were grown in 2XYT media (Digest Peptone 16 g/l, Yeast extract: 10 g/l, Sodium Chloride: 5 g/l; Research Products International #X15600) supplemented with maintenance antibiotics were diluted 1,000-fold into fresh media with maintenance antibiotics and grown at 37°C with shaking at 230 rpm to OD600 0.6–0.8 before use. M13 bacteriophage were serially diluted 100-fold (4 dilutions total) in H2O. 20 µl of bacterial were added to 100 µl of each phage dilution, and to this 200 µl of liquid (70°C) “soft” agar (2XYT media + 0.6% agar) supplemented with 2% Bluo-Gal was added onto a well of

a 24-well plate already containing 235 µl of hard agar per well (2XYT media + 1.5% agar, no antibiotics). To prevent premature cooling of soft agar, the soft agar was placed on the robot deck in a 70°C heat block. After solidification of the top agar, plates were incubated at 37°C for 16–18 h. Source code from our implementation can be found at: https://github.com/dgretton/roboplaque

Population dynamics experiments

Briefly, 96-well clear-bottom plates were filled with 100 µl of water in each well. Point-spread analysis was initiated by adding colored dye to the first well, and liquid transfers were compiled and executed in real time using Pyhamilton on a Hamilton Microlab STARlet. Source code for our implementation can be found at: https://github.com/dgretton/pyhamilton_population_dynamics

3.2.7 On-deck high-throughput turbidostat cultures

Peristaltic pump array

To pump media onto the deck, up to seven miniature 12 volt, 60 ml/min peristaltic pumps ("fish tank pumps") were actuated by custom motor drivers. A Raspberry Pi mini single-board computer received instructions over local IP and commanded the motor drivers via I2C (extended pump configuration details, see[34]. Between each iteration, the reservoir was filled with fresh 2XYT media, and then media was added to each bacterial turbidostat growing in a 24-well plate, based on OD and parameter estimation. Each turbidostat was then sampled by aspirating culture into a 96-well plate reader plate, which was then read using an integrated ClarioStar plate reader. Excess media was then drained from the reservoir, and all other system components were rinsed 1× 5% bleach and 4× water between each iteration. S2060 bacterial strains were grown in 2XYT media supplemented with antibiotics. Source code for implementation can be found at: https://github.com/dgretton/many_basic_turbidostats.

Cell strains and growth conditions

To generate fluorescent reporter strains, plasmids pRSET-B YFP, pRSET-B mCherry, and pRSET-B mCherry were transformed into E. coli strain BL21(DE3) (New England Biolabs). Plasmids were a gift from Kalina Hristova (Addgene #108856, Addgene #108857, Addgene #108858). Bacteria cells were grown overnight in LB media, and then conditioned to grow in M9 Minimal Media: 33.7 mM Na_2HPO_4, 22.0 mM KH_3PO_4, 8.5 mM NaCl, 9.35 mM NH_4Cl, 0.4% Glucose, 1 mM $MgSO_4$, 0.3 mM $CaCl_2$, 1 µg biotin, 1 µg thiamin, 1× trace elements (100× stock solution, 100 mg/l $MnCl_2\,4\,H_2O$, 170 mg/l $ZnCl_2$, 43 mg/l $CuCl_2 \bullet 2\,H_2O$, 60 mg/l $CoCl_2 \bullet 6\,H_2O$, 60 mg/l $Na_2MoO_4 \bullet 2\,H_2O$). For modified M9 Media, Phosphorus, Carbon, and Nitrogen sources were increased or decreased by 2 or 4 fold. For turbidostat inoculations, starter cultures were grown overnight at 37°C for 16–18 h, and then diluted 1:100, and then grown for another 4–8 h until in log-phase growth. When each strain reached log-phase growth (OD 0.6–0.8), cultures were first diluted to an OD of 0.6 and then turbidostats were inoculated 1:100 into 175 µl in 96-well plate reader plates (Black/Clear flat bottom polystyrene plates, Corning #3631) prior to initiation of the robotic method. Unlike the first on-deck turbidostat culture method, in which media was pumped onto the robotic deck, for high-throughput tubidostats, 2 ml of media for each well was aliquoted into a 96-deep well plate (Thomas Scientific, Item #1149J23). Media is replenished daily, or when running low. The robot deck was organized as described in App. B Fig 2A.

Antibiotics

Antibiotics (Gold Biotechnology) were used at the following working concentrations: carbenicillin, 50 µg/ml; chloramphenicol, 40 µg/ml.

Source code

Source code for implementation can be found at: https://github.com/dgretton/many_asynchronous_turbidostats.

Data availability

Source code can be found at: https://github.com/dgretton/pyhamilton. All correspondence regarding Pyhamilton software development should be directed to DWG: (dgretton@mit.edu, https://github.com/dgretton/). Data from the metabolic landscape experiment can be found in App. B Dataset EV2.

3.3. Pyhamilton conclusion

With pyhamilton, scientists can approach laboratory evolution experiments at a scale much greater than typically possible. One application space where pyhamilton can be applied immediately is the present effort to establish novel metabolic pathways for biologically engineered organisms. Metabolic innovation is broadly useful to the burgeoning field of ecological engineering, because the spread of engineered organisms in natural environments is limited by the primary challenge of coming to occupy a natural physical or nutrient space. Such spaces are reliably occupied to carrying capacity by earlier organisms. A fitness advantage is conferred when an engineered microbe can consume what other microbes cannot. An evolved variant with a new metabolic pathway can create and occupy a new ecological niche, and therefore spread to have a concrete constructive outcome on a whole ecosystem. In one case, novel metabolic engineering was previously demonstrated by continuous passaging of 12 cultures of *Escherichia coli*[80] for 34 years and 75,000 generations[23], which resulted in the evolution of a metabolic path consuming citrate[18] as a carbon source in *E. coli*. Pyhamilton can speed up this 34-year timeline by maintaining log-phase growth in *E. coli* using a control algorithm and feedback loop as demonstrated in [28], potentially replicating historic results on a highly compressed timeline. This and several other experiments taking advantage of the unique capabilities of pyhamilton are currently in progress. Pyhamilton also enables new, not yet conceived evolution experiments on an unprecedented (100-1000× parallel) scale.

To date, in the handful of years since its first publication, pyhamilton has accelerated

the adoption of automation in biology in multiple laboratories outside of its lab of origin, with applications from directed evolution to drug discovery and basic science in oncogenesis. Platform-level tools like pyhamilton will continue to open new frontiers in molecular biology.

Chapter 4

A Platform for High-Throughput DNA Synthesis Screening

Our world is currently in an invisible balance between the wide adoption of biotechnologies, and the possible use or misuse of these technologies for destructive purposes. Powerful genetic tools have become available for research, medicine, and even in hobby makerspaces. Prediction experts warn[43] that medical devices and facilities are expected to be subject to cyberattacks, and that an intervention strategy is necessary. These conclusions were shared among experts, regardless of their backgrounds, whether specialists in the field of technology forecasting, or non-traditional experts e.g. biohackers from open bio-focused makerspaces[43]. Most genetic tools rely on synthetic DNA in one form or another. By introducing platform-level, pervasive controls for DNA synthesis, many possible instances of misuse may be circumvented. Here we introduce a counter-technology to screen all synthetic DNA sequence orders to prevent actors from obtaining dangerous genes (e.g., from a deadly virus).

For the purpose of this and following chapters, biological control refers to the need to ensure that the proper controls are placed on the manufacture of biological materials, so as to ensure safe operation of laboratories, and to reduce the probability of future pandemics.

4.1. Introduction to SecureDNA

To establish a precedent for a highly integrated screening technology capable of blocking synthesis of pathogenic DNA at the hardware level, the global networked cryptographically secure **SecureDNA** system is introduced.

4.1.1 Definitions and Problem Space

The database of hazards to be screened is a very dangerous collection of data, especially if it contains hazardous sequences not already recognized publicly as agents of concern. Therefore, storage of the hazard database either on-device, or within the confines of some controlled zone, like a company internal network or an independently hosted cloud platform, in any form that can be read, queried rapidly with guesses, or decrypted, is not acceptable.

Regarding reactivity to emerging situations, SecureDNA recognizes the need to instantly ban newly released or publicized pathogen sequences, within minutes, globally. This quick reaction might become necessary, for example, in a scenario where many people own synthesizers, and a particularly short sequence coding for a particularly dangerous, unforeseen pathogen might be released publicly on the internet.

While no sequence screening system should report excessively many harmless sequences as potential hazards as a general guideline, extreme minimization of false alarm rates comes as a natural constraint when the near future of DNA synthesis is considered. No matter the technique, screening requires much more computing per base than is required to synthesize the DNA (indeed, often, no computing other than communicating the sequence data and actuating internal synthesizer effectors must be performed by the synthesizer or provider). Screening costs, in both time and money, are already a driving factor in DNA providers' decisions about how to screen and what orders to approve[41]. Moreover, as synthesis costs per nucleotide drop by orders of magnitude with advancements in synthesis technologies, the cost of screening stays relatively fixed[41], meaning that, over time, screening will

come to dominate the cost of synthesis. Accordingly, since lists of many uncertain matches take much time and effort to disambiguate, the false alarm rate must be driven as close to zero as possible to make screening a practical possibility in any real market.

With these considerations in mind, exact- or nearly-exact-match screening emerges as an attractive solution, if it can be made sensitive enough despite natural biological variability. Ideally, to achieve excellent sensitivity, DNA screening would match sequences down to 50bp or less, but without additional strategies, the false alarm rate for such small sequences rises too steeply for practical applications in screening[92].

General and Precision Screening

SecureDNA differs from historical and contemporary DNA screening concepts in its objectives. To understand the similarities and differences between ways in which the screening problem is posed, the dichotomy of general versus precision screening is introduced:

General Screening	Precision Screening
Should report all plausible matches	Only cares about a minimal set of matches
Prioritizes sensitivity	Prioritizes specificity
False alarms ok	False alarms not ok
For human interpretation	For machine interpretation
May have many independent services	Must be one global system
Should report metadata for all matches	Reports no information about some matches

- General screening asks the question, “could this sequence be pathogenic?”
- Precision screening asks the question, “is this sequence crucial to the synthesis of a pathogen, and unique to that pathogen?”

SecureDNA seeks to provide *both* general and precision screening. As a general design guideline, where compromises must be made between opposing objectives, SecureDNA favors precision screening.

4.1.2 Comparison to other techniques

SecureDNA	NTI Common Mechanism[70]	ThreatSeq[123]	SeqScreen[12]
Bliss[114]	FAST-NA[48]	Aclid[2]	BLAST[30]

SecureDNA occupies a unique design and constraint space that is not fully comparable to other DNA screening concepts. The most salient are its prioritization of hazard database protection, and of achieving a vanishingly small false alarm rate. In other words, it must unfailingly protect its hazard database, while providing global precision screening, and it must also perform competitively on general screening to achieve any meaningful rate of adoption.

Other systems do not attempt to meet the hazard database protection design constraint. Across all screening techniques surveyed [2, 12, 48, 70, 114, 123], none[1] mention that the contents of the biohazard databases might be sensitive or should be protected. All assume that the sequences (or, in [12], pathogenic functionalities) of concern are given, and the screening problem is a matter of matching input sequences to these known features. Only [123] mentions restrictive access protections to their screening service, and these are intended to protect client data, not hazard identities. [70] emphasizes free and open global access. Only [12, 70] mention the concept of novel or emerging pathogens on short time scales. As a result, much of the technical innovation that sets SecureDNA apart, which is focused on protecting *both* client data and the hazard identifiers themselves, has no analog in other screening concepts.

For completeness, SecureDNA should be compared directly against Basic Local Alignment Search Tool (BLAST) as a candidate tool for screening. Historically considered a golden standard for detecting sequence similarity to biohazards[16, 30], BLAST-based "fuzzy matching" creates a large number of matches per query of the relevant size for screening, i.e. gene-scale, or a few hundred to a few thousand bases. Human screening experts must comb through the resulting long lists to determine whether there are any concerning matches among many false alarms. Additionally, over time, BLAST is becoming

[1] Not applicable to BLAST because it was not designed to be a screening tool.

less useful as a sensitive hazard detection tool[16]. Common genetic constructs inserted into viral genomes, e.g. for the purpose of studying virulence, which are committed to NCBI databases, cause some BLAST searches for ordinary molecular biology tools to have >99% sequence identity with hazards[16]. BLAST simply was not designed to screen DNA for hazards, and its use for such should be deprecated immediately.

SecureDNA differs from existing concepts in other ways as well. It is not known whether most privately developed screening concepts have real-time update capability due to their proprietary nature[41]. Among those that do share some information, some (e.g. [70, 123]) rely on static databases that update only at some long time interval, weekly or monthly for example, and [48] emphasizes that it runs fully locally, implying that it never strictly requires updates at all. Some ([2, 114]) have not addressed the need to update, deeming it sufficient to detect toxins, the organisms on the Select Agent List[107], and/or Australia List[133], or meeting some other slow-moving definition of pathogenicity like "state-of-the-art" [12, 114, 123]. It seems a fair conclusion that none of these existing tools can update to screen new hazards globally, within minutes.

4.1.3 Outline of solutions

SecureDNA assumes that no compromise is possible on updatability. The approach SecureDNA takes to the instant update constraint is to require a live network connection for all synthesizers on earth at all times. While this may seem extreme, the devices on which we are placing controls have pandemic-level destructive capacity; in any case, today, constant network connections are assumed without question for modern hardware devices like printers, home security, and even refrigerators. We think this requirement is not too stringent.

Along with the instant global update design constraint comes the need to continuously communicate data, in some way related to the input sequences, outside the premises of the company or other party owning the synthesizer. If these sequences appear in "plaintext," meaning that they could be legible to someone eavesdropping on the network connection,

companies and parties concerned with preserving their elements of DNA sequence intellectual property, which are extremely valuable today, are very unlikely to consent to its use on their sequences. We address this valid objection by using so-called "oblivious" cryptographic constructs, which provably provide no information about their inputs to any parties involved in their instantiation or use, regardless of whether network connections are compromised. An eavesdropper would only see meaningless random letters and numbers.

Finally, the SecureDNA project achieves resistance to guessing database content by rate-limiting the evaluation of the private protocol. Even an attacker who steals the whole database cannot extract information about its contents in any feasible span of time, because their rate of guessing is limited by the continued participation of multiple authorized parties. As one final measure, the SecureDNA database contains an undisclosed fraction of plausibly concerning non-hazard sequences, so that even correct guesses are potentially worthless to attackers.

Chapter 5

Framing and Analysis of Global Synthesis Screening

Preliminary author's note

SecureDNA is a highly interdisciplinary collaborative project. While the author is an originator of the project as a whole, and a core member of its team, the author does not have all of the skills and experience necessary to bring it to fruition. Accordingly, there are some research products that, despite containing some significant contributions from the author, stray sufficiently deeply into specialized fields unfamiliar to the author, that it neither makes sense to exclude them from this thesis, nor to include them in their entirety. Most of the publication *Cryptographic Aspects of DNA Screening*[14] is a highly specialized technical note in the field of cryptography. Much of its content was not created by the author. The document as a whole was partly edited, and some sections were written, by the author, and some important aspects of SecureDNA proposed by the author are discussed here. The following adaptation of *Cryptographic Aspects of DNA Screening* is heavily abridged to exclude long passages to which the author made no contribution. An attempt was made to keep what remains coherent, but, as some references within the document point to removed sections, some concessions have been made. To resolve any confusion that might arise, the interested reader is referred to [14].

5.1. Formal Statement of the SecureDNA Problem and Proposed Solution[1]

5.1.1 Summary

This technical note provides a detailed but non-specialist-friendly presentation of the DNA screening protocol, including a brief note on its motivations, a formal description of the protocol, and some experimental results on a preliminary implementation. We will begin by presenting the cryptographic formulation of DNA screening, followed by a high-level idea of the current version of screening protocol. We also include some preliminary comparisons between different cryptographic techniques and explain why the project chooses the current version as its privacy-enforcing means.

5.1.2 Motivation

While cheap, accessible DNA synthesis has revolutionized molecular biology since its first practical availability in the early 2000s, it also presents a serious risk to the health and safety of people everywhere. Accidental or malicious use of DNA synthesis to create self-replicating pathogens could lead to global pandemics. It is imperative to introduce an effective screening system for all DNA synthesizers that checks sequence orders against a database of known hazardous sequences to prevent users from "printing out" destructive biohazards. The global hazard database should be up-to-date, secure against attackers seeking to learn its contents, and capable of keeping screened sequences confidential.

Present DNA screening efforts are slow, expensive, and suffer incomplete adoption among DNA providers. Much of the time and money consumed by today's DNA screening protocols is spent on "fuzzy matching" between orders and hazard sequences. In contrast with exact-match screening, which concerns whether a sequence appears in its entirety in a database, fuzzy matching assesses the degree of similarity between a query sequence and a database. In screening protocols that use fuzzy matching, a threshold of similarity is established for DNA sequences that are deemed too close to hazards. Due to the high degree

[1] Adapted from Carsten Baum, *et. al.* Cryptographic Aspects of DNA Screening[14] on which Gretton, D. W. is a co-author.

of variability among DNA sequences, even those that have the same biological function, it is widely assumed that this inexact matching is critical to catch subtle variants on known hazards.

However, fuzzy matching comes with costs that make it untenable as a candidate for global screening. Many innocuous sequences look similar to parts of hazards, leading them to be identified incorrectly as malicious, an event we refer to as a "false alarm." All false alarms produced by modern screening systems must be forwarded to a human expert who can examine the sequences to determine whether they are dangerous. Since nearly all synthesis orders are not biohazards, this expert intervention is almost completely wasted effort. Furthermore, screening protocols with a human component cannot scale to meet the demands of an ever growing global DNA synthesis market.

Instead, we phrase DNA synthesis screening as an exact match problem between short (e.g. 42 base pair) windows selected from the orders and the hazard sequences. These windows are constant-length strings of the DNA bases A, T, C, and G. If a small sequence window from an order exactly matches some window from a hazard, the order is not synthesized. To contend with variability in genetic sequences, we include a selection of likely variations on these windows in the database. The effectiveness of this approach is supported by the observation that all pathogens have many critical regions in their genomes, and that extensive changes in all such regions simultaneously will almost certainly yield an inactive agent. As we will demonstrate, an exact-match screening system can be fast, fully automated, and secure. Exact-match screening can also keep the rate of false alarms low enough that global adoption is feasible.

In this document, we examine the DNA screening problem in the exact-match setting. Details on the sequence, window, and variation selection strategies used to construct the database are not discussed, as the cryptographic treatment is independent of any such choices.

5.1.3 Problem-Specific Challenges

As a query of the form, "Does any sequence window in the order appear in the hazard database?", exact-match DNA screening amounts to set membership testing, where both the query and the database are private. Private set membership testing is an extensively studied topic in cryptographic literature that already has solutions under a variety of privacy constraints. Nevertheless, we found that no single existing technique will overcome the unique challenges posed by the DNA screening problem due to the precise details of its intended application. These challenges are:

- **Small maximum element size.** Short sequence windows, which are the elements of the database set, are limited in length. This constraint makes it extremely difficult to maintain database privacy, given arbitrary set membership queries, because it may be possible to guess short sequences.

  > The window size limitation comes from the physical characteristics of DNA. Long strands of DNA are routinely assembled from short overlapping pieces by ligation. Since exact-match screening cannot screen sequences up to, but short of, the window size in length, the window size must be short enough that unscreened pieces cannot be assembled. For DNA, this limit is around 50-60 base pairs due to thermodynamic properties of DNA binding. Sequence windows must not exceed this limit.

  The challenge could be restated as: *The input space has low entropy*.

- **High query rate.** The present global synthesis market is estimated at around a trillion bases per year. By 2029, queries are expected to exceed 10^{15} queries per year, or over 10 million per second. Many orders consist of quasi-random sequences for DNA libraries.

> This challenge has two main repercussions:
>
> 1. Much of the input space of short windows is effectively searched each year, raising the minimum false alarm rate.
> 2. The performance demands on such a global networked screening system are immense, requiring many servers and robustness to server failure.

To ensure that the servers hosting the hazard database cannot be hacked to obtain its dangerous contents, we store the database as hashes, or results of a pseudorandom function (PRF) applied to the hazard sequences.

The only way to address the low entropy of the input space to the PRF is to limit evaluation of the PRF itself. We accomplish rate limiting by requiring the participation of multiple trusted third parties each time the PRF is evaluated.

To enable the required level of distribution and robustness to server downtime, we require some, but not all, of the trusted "key-holder" third parties to participate, a property known as "threshold evaluation." We settled on the *distributed pseudorandom function* as the core component.

Our method is private in that no party, whether the querier, database holder, or key-holder, ever learns the input sequence strings for the query *or* the database elements, a very strong property. It has *imperfect privacy* in a strict cryptographic sense, in that it is possible for a server to deduce that the same query was executed twice by a particular client, a relatively weak concession that we make for performance reasons.

In the rest of this section, we first fix the notations to be used in this document. Then, we briefly summarize the formal security requirements for DNA screening, which frames further specification of the ideal functionalities to be realized with provable security. Finally, we provide rationales for our choices of cryptographic building blocks wherever these diverge from established norms, mainly due to scalability issues.

Formulation

At the core of the DNA screening problem is the private membership query, in which a server P^{s} maintains a database D and a client P^{c} makes queries of the form "Is x in D?", with the privacy of both server and client respected. In addition to P^{s} and P^{c}, our solution additionally includes some key-holders P^{kh} to facilitate an efficient solution.

In this framework, a solution can be specified by:

1. An *oblivious distributed pseudorandom function* F (DPRF) where the key-holders P^{kh} first generate a shared key $k = (k_1, k_2, \ldots, k_n)$, so that $F_k(x)$ can be computed by the client P^{c} when at least t key holders (t being a security parameter) supply certain information $a(k_i, x)$ to it.

2. A two-party protocol P between client P^{c} (also known as the receiver) and server P^{s} (the sender) for membership query.

Initially, P^{kh} generates a shared secret key k, and then sets up a database D to contain all $F_k(y)$ for the known harmful strings y (in encrypted form). For P^{c} to ask a query x, it first interacts with t keyholders P^{kh} to obtain $F_k(x)$, and then uses the two-party protocol P to obtain the answer to whether $F_k(x)$ is in D.

5.1.4 Technical Details of the Implementation

We begin with a high level view of our initial implementation, which uses a highly efficient protocol with imperfect privacy of client queries. We then describe why the approach is superior for the purpose of DNA synthesis screening to alternatives based on different paradigms.

A basic model:

1. For the DPRF, we use the Naor-Pinkas-Reingold [98] scheme which is based on the hardness assumption of DDH (Decisional Diffie-Hellman). The DDH assumption is well-studied and used in many deployed cryptographic systems.

2. For security and privacy, we represent each string x by a one-way hashed image $H(x)$. The membership query simply does a binary search for $F_k(H(x))$ against the database containing $F_k(H(y))$ of all known harmful strings y.

Two remarks:

1. It is assumed that $H(x)$ is a random oracle. Even then, the client's privacy is not perfect, as the server can detect when two queries x, y are the same (as $H(x) = H(y)$).

2. This DPRF is not quantum resistant. There exist DPRFs designed for post-quantum security, but they are presently too inefficient for practical use. We discuss our approach to making the system quantum-resistant in Sec. 5.1.9.

We refer to Sec. 5.1.5 through Sec. 5.1.8 of this report for the details about the preliminary design of the screening protocol.

5.1.5 The Screening Functionality

In this section we introduce the mathematical formulation of the efficient (but with imperfect privacy) screening protocol, which is presented in the form of ideal functionalities. These functionalities capture the actions of a virtual trusted third party that a cryptographic protocol seeks to simulate. We first list all the notations to be used in this report in Sec. 5.1.5, and then present the functionalities in Sec. 5.1.6.

Notations

This section summarizes all the symbols we use in the white paper.

Mathematical Symbols

We use $[n]$ to denote the set of integers $\{1,\ldots,n\}$. $S = \langle x_1,\ldots,x_n\rangle$ denotes an ordered list, $S[i]$ refers to the i^{th} entry of S, and $|S|$ denotes the number of entries in S. In particular, we use `DB` to denote the set of all database entries.

We use $\mathbb{Z}_q$ to represent the set $\{0,1,\ldots,q-1\}$, and we denote by $x \in_R S$ sampling an element from the finite set S uniformly at random. We use $poly(\cdot)$ to refer to some polynomial function.

We use κ to denote the security parameter. Let p and q be two prime numbers such that the length of q is at least κ bits[2] and $q|p-1$. Let $\mathbb{G}$ denote a multiplicative subgroup of $\mathbb{Z}_p^*$ of order q, and we use g to denote a generator in that subgroup. The *Decisional Diffie-Hellman* assumption for group $\mathbb{G}$ holds if for all probabilistic polynomial-time algorithms D, the distinguishing advantage

$$\left|\Pr[a,b \in_R \mathbb{Z}_q : \mathsf{D}(g,g^a,g^b,g^{ab}) = 1] - \Pr[a,b,c \in_R \mathbb{Z}_q : \mathsf{D}(g,g^a,g^b,g^c) = 1]\right|$$

is negligible in the security parameter κ.[3]

We use "sequence" to denote a string of base pairs which make up a biological DNA sequence. We denote by Q the genetic alphabet $Q = \{\mathrm{A},\mathrm{T},\mathrm{C},\mathrm{G}\}$, and we use $|x|$ to denote the length of a sequence x. Our protocol mainly considers a sequence of fixed length w, and we use Q^w to denote the set that consists of all sequences of length w. Considering the low-entropy property of the distribution over Q^w in the DNA screening application, we further add *salt* to the input of PRF[4]. We denote S as the (sufficient large) space of salts and $\mathsf{salt} \leftarrow S$ an efficient sampling procedure to generate a random element $\mathsf{salt} \in S$. Additionally, we include a special symbol $\perp$ in the set S indicating a constant placeholder

[2]DWG note: Meaning that κ is of order $\log(q)$, a condition for DDH.

[3]DWG note: The Decisional Diffie-Hellman (DDH) assumption essentially means that no realistic algorithm could ever tell whether a group element is the result of combining two other group elements, any better than by randomly guessing. This expression's important features are g^{ab} (two group elements combined) on one side, and g^c (any random group element) on the other. It should be noted that DDH is a less general assumption than the discrete log assumption, which is more like that which safeguards public-private key encryption in everyday use. DDH expresses the hope that our group $\mathbb{G}$ is not one of the unlucky ones where combinations can be distinguished easily, even if separating them directly (discrete log) is hard. In practice, this assumption is commonly made and seems a trustworthy assumption upon which to build a secure system.

[4]DWG note: Colloquially in cryptography, a salt is some piece of data, which does not even need to be secret, that is added in when computing functions of secret inputs like passwords (or valuable DNA sequences). The salt scrambles the function so that its outputs will never be familiar to anyone, even if they have seen many outputs from this function before.

used as salt.

A random oracle (RO) is a popular cryptographic heuristic that emulates a truly random function and is typically instantiated by a cryptographic hash function[5]. We use three random oracles in our protocol: H is a RO that maps any sequence of length w to a group element in $\mathbb{G}$ (i.e. $H : Q^w \to \mathbb{G}$), H' one that maps a GenBank ID (essentially a bit string) to a group element (i.e. $H' : \{0,1\}^* \to \mathbb{G}$), and H'' one that maps a group element plus a salt to a group element (i.e. $H'' : \mathbb{G} \times S \to \mathbb{G}$).

Parties

Since screening is essentially a multi-party computation task, we need to model different roles in screening as separate parties. There are four types of parties in our protocol.

- **Clients.** Synthesizers who query the database with their input to find out whether it matches any of the entries in the database or not.
- **Server/DB.** Cloud servers that hold a database of encrypted DNA sequences.
- **Key holders.** Parties that hold the key shares of the main key.
- **DB Maintainer.** The trustworthy party who determines entries in the database. To add an entry to the database, he first interacts with key holders to get the entry encrypted and then stores the encrypted value in the database.

We can further divide clients into two groups: 1) large commercial DNA synthesis providers (e.g. BGI) who typically share dedicated and fast network connection with other parties (i.e. the server and key holders); 2) desktop clients who make queries less frequently with restricted bandwidth connections and are more likely to be compromised. We model client, server, and key holders as semi-honest for now, and extend our protocol to handle malicious clients (drawing from existing work on maliciously secure oblivious DPRFs) in the next version.

[5]DWG note: One example of a cryptographic hash function is SHA-256, which is one way Bitcoin stays secure. SHA-256 can be used like a random oracle if set up properly.

We use superscript to distinguish the roles of the parties, and subscript to index different parties of the same class. We use P^{c} to denote client, P^{s} to denote the server, $\mathsf{P}_i^{\mathsf{kh}}$ to denote key holders and $\mathsf{P}^{\mathsf{dbm}}$ to denote the database manager. Throughout this note, we restrict our discussion to the single DB setting since it can be extended in a manner oblivious to the screening protocol. We discuss the extension to distributed database in [14].

The server holds a database where all entries are encrypted under the main secret key. As a slight abuse of terminology, encryption refers to the evaluation of a pseudorandom function (under the main secret key) on the cleartext to produce the corresponding output. Strictly speaking, it is not an encryption as its inverse operation, decryption, may not exist and neither is it needed in our protocol. In order to protect the main key, we run Shamir's (t,n)-secret sharing among the n key holders, where any t (or more) parties can efficiently reconstruct the secret key from their shares, and any less than t parties cannot. Further, the database manager (or client) runs a protocol with any t key holders in an oblivious manner such that he gets his input encrypted under the main key without revealing any information about his query to the key holders. The database manager (resp., client) then adds his encrypted entry to the database (resp., runs an exact matching with the database).

Exemption list

A further optimization to further decrease false alarm/rejection rate in the large provider setting (assumed trustworthy) is to include respective exemption lists for each client/provider in the database. This is achieved by the following method.

The database in this case consists of two parts. The first part (the hazardous sequence section) is hashed sequences associated with their hashed GenBank ID (gbid). Both hashing requires no salt. The second part (the exemption list section) consists of a list of all registered providers, associated with their salt and a list of their registered gbid *doubly* hashed with this salt[6].

[6]DWG Note: double salted hashing is used to decorrelate gbids, so that someone with direct access to the database would not be able to see which hashes or how many correspond to each hashed value of any particular gbid.

During the setup phase each client (e.g. a biology lab frequently conducting experiments on hazardous sequences) would submit a list of sequences associated with their respective gbid's, which are considered to be hazardous, but which would serve a non-hazardous use according to a biosafety authority. Once approved, the client would further doubly hash gbid using a freshly sampled salt $\mathsf{salt} \leftarrow S$ and submit the list and salt to the database.

In more detail, the database would contain a *hazardous sequence section* of the hashes of hazardous sequences and GenBank ID's (both without salt):

$$\{\langle f_\alpha(\mathsf{seq}), f_\alpha(\mathsf{gbid})\rangle\} \ ,$$

and a *exemption list section* of each client's identifier and their salted exemption list:

$$\{\langle \mathsf{P^c}, (f'_\alpha(\mathsf{gbid}_1, \mathsf{salt}), \ldots, f'_\alpha(\mathsf{gbid}_n, \mathsf{salt}))\rangle\} \ ,$$

where $f'_\alpha(\cdot)$ is the double hash function.

Provable Security

Following the conventions and notations of multiparty computation, we use $\mathscr{F}$ to denote an ideal functionality to be realized and use Π to denote a multiparty protocol. We use subscript to further specify the functionality or protocol.

We consider security against a static, semi-honest adversary. More specifically, the adversary chooses the parties to corrupt prior to the protocol execution and the corruption status does not change throughout the course of protocol execution. The corrupted parties follow the protocol specification exactly. However, the adversary tries to learn more information than allowed by looking at the transcript of messages that it received and the internal state of corrupted parties.

A protocol that is secure in the presence of semi-honest adversaries guarantees that there is no inadvertent leakage of information; when the parties involved essentially trust

each other but want to make sure that no record of their input is found elsewhere, then this can suffice. Beyond this, protocols that are secure for semi-honest adversaries are often designed as the first step towards achieving stronger notions of security.

5.1.6 Security Requirement and Ideal Functionality

To support efficient and secure screening, we need four ideal functionalities.

- **Initialization.** Register and assign key shares to all the key holders and initialize the database;
- **Adding a new sequence.** Add a new sequence to the database;
- **Querying a sequence.** Decide whether a sequence being queried exists in the database or not;
- **Refreshing key shares.** Update the key shares of the key holders.

Notice that these ideal functionalities do not correspond to the actual operations executed by the protocol, but instead they specify the functionalities to be realized. As new sequences will be added to the database and possibly queried subsequently, the functionality is by nature reactive, which means that a trusted third party must maintain state in order to realize this functionality. Therefore, we consider the security of a session where the initialization functionality is first invoked followed by addition, querying, and key share refreshing operations. Indeed, the sid field in all the messages presented below is used to distinguish different sessions.

In the following, the functionality keeps three lists S_1, S_2, and S_3 where S_1 and S_2 are of the same size. S_3 is a list of GenBank ID's. S_1 keeps a record of the sequences that have appeared (either queried by client or input by the data owner) in the view of the server *together* with a list of associated GenBank ID in the form of indices in S_3, while S_2 indicates whether such sequence has been actually added or not.

More specifically, let i be an index in S_1 and S_2 and $S_1[i].\mathsf{Val}$ be the sequence of the i^{th} entry in S_1, and $S_1[i].\mathsf{List}$ be its associated list of GenBank ID's, then ($S_1[i] = (x, \emptyset)$, $S_2[i] = 0$) indicates that seq has already been queried, but it does not exist in the database, whereas ($S_1[i] = (\mathsf{seq}, L)$, $S_2[i] = 1$) indicates that x has been added, with a list of associated GenBank ID's $L \subseteq 2^{[|S_3|]}$ (i.e. a list of indices of S_3).

While seemingly unnecessary, the formulation is useful in the simulation-based proof (to simulate queries in a consistent manner). We use "append" to denote the operation of adding an element (a sequence for S_1 or a bit for S_2) to the end of a list.

As for the exemption list (denoted as WL), the functionality will maintain two lists $\mathsf{WL}[\mathsf{P}^c].\mathsf{ID}$ and $\mathsf{WL}[\mathsf{P}^c].\mathsf{Val}$ associated with each registered client P^c, as well as an individual salt for each client. The first list $\mathsf{WL}[\mathsf{P}^c].\mathsf{ID}$ keeps track of all the GenBank ID's that have been queried with the client's salt while the second list $\mathsf{WL}[\mathsf{P}^c].\mathsf{Val}$ indicates whether each entry in the previous list in on the exemption list or not.

5.1.7 The Screening Protocol

In this section we present the screening protocol. We first introduce a helper functionality $\mathscr{F}_{\text{prf}}$ for oblivious threshold evaluation of a pseudorandom function and a protocol that implements this functionality in the semi-honest model. The screening protocol is then introduced in the $\mathscr{F}_{\text{prf}}$-hybrid model, which facilitates the modular presentation of the protocol and as well as security analysis.

The Oblivious Threshold PRF Protocol

First we present an ideal functionality that defines the oblivious threshold evaluation of a pseudorandom function. The pseudorandom function we consider here is a variant of (with minor adaption to) the Naor-Pinkas-Reingold PRF introduced in [98]. The PRF $F : \mathbb{Z}_q \times \mathscr{X} \to \mathbb{G}$ is defined as

$$F(\alpha, x) = f_\alpha(x) = H(x)^\alpha,$$

where $\mathbb{G}$ is a multiplicative group of order q used in the DDH assumption, and H is a random oracle that maps an element in the set $\mathscr{X}$ to a group element. The property of F is agnostic to the input space of H, conditioned on it being modelled as a random oracle. And thus in our application we effectively acquire three PRF's by instantiating $\mathscr{F}_{\text{prf}}$ with H, H', and H''. We distinguish different PRF's by their respective RO's (e.g. $\mathscr{F}_{\text{prf}}^{H}$) in the following presentation.

Two types of parties participate in this protocol, namely the key holders $\mathsf{P}_i^{\mathsf{kh}}$ and a receiver P^{r}. For convenience, we assume that the key holders in the protocol are the same as those in the screening protocol and thus inherit the notations for the key holders. Each key holder holds his respective (t,n)-Shamir secret share of the PRF master key, and the receiver evaluates the PRF on his input under the master secret key by interacting with t key holders.

The functionality consists of three functions, the PRF secret key generation, oblivious evaluation of the PRF under the secret key and refreshing the key shares of key holders.

Functionality $\mathscr{F}_{\text{prf}}$

The functionality keeps a mapping $F : \mathscr{X} \to \mathbb{G}$ that maps elements in $\mathscr{X}$ to group elements. Initially, it sets F to an empty mapping.

- Upon receiving (gen, sid, $\mathsf{P}_i^{\mathsf{kh}}$) from all key holders, the functionality chooses a new key identifier kid and sends (gen.recv, sid, $\mathsf{P}_i^{\mathsf{kh}}$, kid) to $\mathsf{P}_i^{\mathsf{kh}}$. It ignores any subsequent calls under the same sid.

- Upon receiving (eval, sid, $\mathsf{P}_{i_j}^{\mathsf{kh}}$, kid) from t distinct key holders $\mathsf{P}_{i_1}^{\mathsf{kh}}, \ldots, \mathsf{P}_{i_t}^{\mathsf{kh}}$ and (eval, sid, P^{r}, x) from the receiver, the functionality checks if x appears in F. If x has already been added to F, let y be the corresponding value in F (i.e., $y = F(x)$). Otherwise, it samples $y \in_R \mathbb{G}$ and adds $\langle x, y \rangle$ to F. The functionality sends (eval.recv, sid, P^{c}, y) to the receiver, and (eval.recv, sid, $\mathsf{P}_{i_j}^{\mathsf{kh}}$) to the t key holders.

- Upon receiving (refresh, sid, $\mathsf{P}_i^{\mathsf{kh}}$, kid) from all key holders, the functionality samples an unused key identifier kid′ and sends (refresh.recv, sid, $\mathsf{P}_i^{\mathsf{kh}}$, kid′) to the respective key holders.

Fig. 5-1. The oblivious threshold PRF functionality.

Querying Protocol

The querying protocol involves a client, t key holders and the server. The protocol is similar to the two-step new sequence adding protocol except that this time the client acts as the receiver as in the ideal functionality $\mathscr{F}_{\text{prf}}^{H}$ in the first step. It then gets and forwards the result from $\mathscr{F}_{\text{prf}}^{H}$ to the server, who checks whether the received value is in the database or not.

Due to the exemption list mechanism, a hit in the sequence database does not necessarily mean rejection. Instead, the server should retrieve the hashed GenBank ID from the sequence database and then by calling $\mathscr{F}_{\text{prf}}^{H'}$ together with the client's individual salt value, determine whether the GenBank ID associated with this query appears in the client's exemption list. Only when 1) the sequence is hazardous *and* 2) its GenBank ID is missing in the exemption list will the server reject such query. Otherwise this query is accepted.

Key Share Refreshing Protocol

In order to achieve proactive security, the key shares need to be "refreshed" periodically, which is achieved by the key share refreshing protocol (see Fig. 5-1, bullet 3). Now that the ideal functionality $\mathscr{F}_{\text{prf}}$ already supports key share refreshing operation, this protocol simply invokes it.

5.1.8 Performance Evaluation

Testing Setup

There are many elliptic curves recommended by NIST [76] and more implemented by OpenSSL [121]. Our performance evaluation and comparison are based on the OpenSSL 1.1.1 implementation on a laptop equipped with Intel® Core™ i5-7400 CPU @ 3.00GHz x4 and the Ubuntu 18.04.2 LTS 64bit OS.

Table 5.1. The performance of various curves in terms of the number of base and random multiplications (i.e., the operations of $s \cdot P$ and $t \cdot Q$ respectively) per second and the number of $s \cdot P + t \cdot Q$ operations per second, where $s, t \in \mathbb{Z}_q$, q is the order of the curve, P is the generator, and Q is an arbitrary element.

curve name	bit sec.	#.base mul/s	#.random mul/s	#.s*P+t*Q/s
secp112r1	112	3305	6889	8982
wap-wsg-idm-ecid-wtls6	112	7692	7485	9175
sect113r1	113	12343	12290	6079
wap-wsg-idm-ecid-wtls1	113	13731	13142	6660
secp128r1	128	6572	6676	8953
sect131r1	131	7097	6998	3473
secp160k1	160	4063	4066	5340
wap-wsg-idm-ecid-wtls7	160	4257	4124	5683
brainpoolP160r1	160	4000	3854	5212
sect163k1	163	5339	5547	2687
wap-wsg-idm-ecid-wtls3	163	5569	5513	2783
c2pnb176v1	176	5429	5277	2621
c2tnb191v3	191	6019	5895	2917
prime192v1	192	3400	3347	4710
sect193r1	193	5154	5151	2565
secp224r1	224	26932	10703	7830
wap-wsg-idm-ecid-wtls12	224	2346	2332	3377
brainpoolP224r1	224	2147	2188	3064
secp256k1	256	1859	1859	2681
prime256v1	256	101937	20462	17406
brainpoolP256r1	256	1912	1920	2753
c2pnb272w1	272	2368	2324	1167
brainpoolP512r1	512	548	550	855
secp521r1	521	5322	2765	2032

Performance Evaluation

Let us mention that the PRF key generation $\Pi_{\text{prf.gen}}$ and key refreshing operations $\Pi_{\text{prf.ref}}$ (and the corresponding Initialization and Key Share Refreshing Protocols that invoke them) involve only a few additions and multiplications, and they do not constitute the bottleneck of performance.

Obliviously, the main computationally intensive operation is the Oblivious PRF Evaluation $\Pi_{\text{prf.eval}}$, which is invoked by querying and adding new sequence protocols. It takes quite some group multiplications and exponentiation operations. An exponentiation operation can be computed by $\log q$ multiplications where q is the order of group. Thus, a client needs $2\log q + t$ multiplications and each key holder requires $\log q$ multiplications. As shown in Table 1, we compare the performance on various elliptic curves and highlight the (relatively) good ones in red. The evaluation of the PRF is interactive and thus it depends on the bandwidth too. Concretely, to handle 10^5 queries in the 256-bit security setting, the communication cost between client and server is approximately 14MB, which should not affect the performance too much.

In summary, the performance bottleneck is the basic operations over the elliptic curves despite many other less dominant factors. As illustrated in Table 3, our testing runs on an off-the-shelf (actually out-of-date) personal laptop, which takes about 27 minutes to answer 10^5 queries. We estimate that a high performance server should easily accomplish the task in less than a minute.

Table 5.2. The estimated numbers of basic protocol operations per second on our laptop implementing those curves highlighted in red in Table 1. Note that "bit Sec." uniquely identifies the corresponding red curve from Tab. 5.1.

bit Sec.	operation	#.operation/s
113	init	200
113	add	60
113	query	60
113	refresh	200
256	init	200
256	add	60
256	query	60
256	refresh	200
521	init	200
521	add	15
521	query	15
521	refresh	200

5.1.9 Quantum Resistance

In this section, we detail why the possibility of efficient quantum computation is relevant to the long-term security of our proposal, and our approach to securing our established privacy properties, especially of the hazard database, against future quantum attacks.

Rationale for Exploring Quantum-Resistant Approaches

There is considerable debate as to whether quantum computation will become a serious threat to certain techniques in non-quantum, or "classical," cryptography in the foreseeable future. Regardless of the strength of the arguments on each side, we must accept that there is some chance that any cryptographic primitive in our proposal whose hardness assumptions rely on a classical computation setting could be broken by a quantum-capable

adversary. One example of a hardness assumption that is considered secure under classical computation, but not quantum computation, is DDH (Decisional Diffie-Hellman). Our oblivious DPRF construction relies on DDH for its security. Under quantum computation, an adversary could efficiently evaluate the oblivious DPRF on arbitrary inputs without involving the threshold number of participating servers. Free DPRF evaluation would permit oracle access to set membership in the database; due to the low entropy of the input sequence space, the database contents could be exfiltrated quickly, constituting a catastrophic failure of the screening system.

We have considered several possible extensions of our basic protocol to address such weaknesses with respect to future quantum attacks. For our purposes, it is sufficient to secure the database contents against a quantum adversary, for now setting aside query privacy, which is much less crucial. Our extensions fall into two categories: information-theoretic DPRF and MPC-embedded encryption.

PRF with Information-Theoretic Security.

Having obtained the result of the oblivious DPRF $F_k(\cdot)$, the client may participate in another round of PRF evaluation with multiple (possibly the same) parties to compute $G_{k^q}(F_k(\cdot))$ before participating in the two-party membership query protocol P. If this round is made quantum secure, e.g. by using fast block cipher based PRFs, query privacy is still preserved under classical computation and the database is secure against a quantum adversary.

The following approaches, with increasing sophistication, each achieve information-theoretic security (and thus quantum security) by requiring key information from extra sources:

1. In its simplest form, a single trusted server P^{Q} holds k^q, the key for the quantum-secure PRF. The client sends its computed result $F_k(x)$ to P^{Q} and receives $G_{k^q}(F_k(x))$.

2. Slightly more securely, several servers $\mathsf{P}_i^{\mathsf{Q}}$ each hold pieces of $k^q = (k_0^q, k_1^q, \ldots, k_n^q)$. The quantum-secure "distributed" PRF is computed as the XOR of the independent

keyed PRFs, $G_{k^q}(\cdot) = G'_{k_0^q}(\cdot) \oplus G'_{k_1^q}(\cdot) \oplus \cdots \oplus G'_{k_n^q}(\cdot)$. Evaluation of this $G_{k^q}(\cdot)$ requires all servers $\mathsf{P}_i^{\mathsf{Q}}$ to be online, a robustness concern.

3. In keeping with threshold, rather than obligatory, participation by trusted third parties being a desirable property, the keys for the second round can be distributed by replication. Using $\binom{n}{t}$ keys replicated across the servers $\mathsf{P}_i^{\mathsf{Q}}$ such that every subset of size $n-t$ servers is assigned one unique key, any $t+1$ servers are guaranteed to have at least one copy of all n key pieces between them. $G_{k^q}(\cdot)$ is then computed by XOR of the individually communicated $G'_{k_i^q}(\cdot)$ as above. This approach is only feasible for small t, and does not scale well for t as a constant fraction of n. Still, even with many keys, an implementation utilizing block ciphers could achieve sufficient throughput. One concern is that this approach also requires high communication overhead due to transmission of many redundant intermediate results.

4. Excessive communication overhead may be avoided by specifying some local computation on each server $\mathsf{P}_i^{\mathsf{Q}}$ as in [31] to combine the PRF evaluations over all of its $\binom{n-1}{t-1}$ keys into one value before sending, such that the information transmitted to the querier collectively forms a Shamir secret sharing of the result $G_{k^q}(F_k(x))$.

5.2. Efficacy of fully automated DNA synthesis screening[7]

The following adapted text regards the content of the known harmful strings discussed in the previous chapter, and what reason we have to trust that they can truly protect against the synthesis of pandemic-capable agents. The author was responsible for most code, algorithms, database generation pipelines, integration of third-party tools and cloud data management; some text, some experimental design, some analysis; and some figures.

Today, many individuals can assemble infectious viruses from synthetic DNA that is not screened for hazards[42, 77]. A major barrier to universal screening is the high rate of false alarms, which requires expensive human curation. Here we develop, test, and implement "random adversarial threshold" (RAT) search, a highly specific approach that looks for exact matches to short sequence windows and predicted functional equivalents found in hazards but not in any unrelated genes. To determine whether bad actors could obtain replication-competent viruses by incorporating mutations to evade screening, we built databases to protect nine peptide windows found in M13 bacteriophage virus and launched 21,000 attacks at each window by experimentally building and measuring the fitness of variants with up to six amino acid changes. Finding that defensible windows capable of reliably blocking attacks shared certain predictable features, we identified similar windows from the Australia Group list of pathogens, constructed databases of variants, and wrote software enabling the cryptographically secure screening of synthesis orders. RAT search offers a way to safeguard biotechnology by fully automating DNA synthesis screening.

5.2.1 Importance of measuring efficacy

The ongoing COVID-19 pandemic has underscored the danger posed by exponentially spreading biological agents. Virus assembly protocols and inexpensive commercial de novo DNA synthesis services have made many hazardous agents accessible to a large and grow-

[7] Adapted from Gretton, *et. al.* Random Adversarial threshold search enables specific, secure, and fully automated DNA synthesis screening[58]

ing number of individuals with relevant technical skills[101]. Future advances may one day lead to publicly available genetic blueprints for pandemic agents considerably more destructive than SARS-CoV-2[45]. Fortunately, most individuals with the skills required to create potential pandemic agents cannot synthesize DNA on their own. Members of the International Gene Synthesis Consortium (IGSC), a trade industry group committed to biosecurity, screen commercial DNA synthesis orders above a certain length for sequences that match the Regulated Pathogen Database[71]. The IGSC firms deserve praise for voluntarily prioritizing safety because doing so is costly: current screening methods based on BLAST generate many false alarms from unrelated sequences that require evaluation by human experts[40, 42, 91]. As the price of synthetic DNA continues to fall, the effective cost of screening grows[42]. Unfortunately, 20% of commercially synthesized DNA is generated by non-members who do not screen. Since the list of IGSC members is public, the extent to which current screening meaningfully restricts access to biohazards is questionable. Even if all current providers did screen, the anticipated arrival of benchtop machines enabling immediate on-site synthesis[45] may open another window of vulnerability[42]. Creating a system to screen all commercial and academic DNA synthesis for current and emerging hazards[40, 45] demands a method triggering negligibly few false alarms. We hypothesized that automated DNA synthesis screening could be achieved by "random adversarial threshold" (RAT) search, a strategy that relies on exact-match screening against a database comprising unique signatures of hazards (App. C Fig. C-1a). Here we sought to determine whether our approach is specific and secure enough to fully automate DNA synthesis screening.

5.2.2 Experimentally verifying efficacy

Random adversarial threshold search

To screen DNA via random adversarial threshold search, a database of hazard signatures is created for comparison to orders (App. C Fig. C-1b). We randomly choose 19-amino-acid peptide windows from the protein-coding gene(s) of a hazard for inclusion in the database

using a distribution function biased towards defensible windows (see "security analysis" below). Reliably assembling hazards from smaller DNA pieces would be challenging[53, 105]. To further increase the difficulty, we include all 42-mer DNA sequences and some 30-mers. Next, a list of peptide variants predicted to be functional is computed for each window using one or more variant effect predictors[21, 27, 57, 66, 87, 95, 109]. The list of variants is curated to remove peptides matching unrelated sequences in repositories (App. C Suppl. Fig. C-1). Curation of database entries ensures the system will never wrongly flag unrelated sequences known to science, mostly eliminating false alarms. Finally, DNA synthesis orders are searched for exact matches to database entries, and any matched orders are rejected. Given sufficiently high specificity (see below), RAT search can in principle be fully automated, avoiding the requirement for human curation characteristic of current screening procedures.

The efficiency of exact-match screening permits the use of cryptography to protect the privacy of DNA synthesis orders sent to be screened and prevent attacks aiming to determine which sequences are protected, which could enable screening for emerging hazards without disclosing their identities[122].

5.2.3 Theoretical specificity analysis

Database curation ensures that RAT search will never flag an unrelated sequence from any known repository. Other sources of false alarms include purely random matches, legitimate research on related pathogens, and oligo libraries encoding variants of known sequences.

The frequency of random matches depends on the quantity of DNA synthesized and the number of sequences in the database. The probability that the forward or reverse translation of a random 57-mer will match any given database entry is $4.5{\times}10^{-25}$. Assuming a hazard database of 10^{10} entries and that 10^{15} unique (not total) oligonucleotides will be synthesized in 2030[1], we expect approximately one random false positive for the entire world's DNA synthesis in that year[86].

To ensure that laboratories with permission to study hazards can obtain them, GenBank

accession numbers from biosafety registration documents listing genes and organisms can be used to create an exemption list. Each entry in the hazard database is associated with an accession number and taxon ID. When an order matches a hazard database entry, the associated values can be compared to the customer's exemption list and the order automatically approved (App. C Suppl. Fig. C-2). Orders seeking to generate variants of known sequences for experiments such as deep mutational scanning are also algorithmically recognizable by their similarity to wild-type exemption list entries.

Together, these methods suggest that searching for exact matches to functional signatures of hazards is specific enough to achieve automated screening.

5.2.4 Theoretical security analysis

RAT search is only useful insofar as it can reliably detect any given hazard. Including many wild-type sequence windows across a gene or genome will unfailingly detect any order seeking to synthesize that gene or genome[132]. However, a human adversary might incorporate at least one change every 19 amino acids and 30 base pairs and still generate a functional sequence. Detecting orders generated by intelligent adversaries requires a careful consideration of how best to include functional peptide variants in the hazard database.

Suppose an adversary seeks hazard W. For each window w_i, there are three possible outcomes:

1. w_i is present in the database, causing the synthesis order to be rejected
2. w_i escapes detection, but imposes a fitness cost c_i that reduces the functionality of W
3. w_i escapes detection without cost

Success requires the adversary to achieve the third outcome for most w_i to preserve function. We define the random adversarial threshold R as the probability that an adversary with perfect knowledge of the fitness of each variant – but ignorant of which windows and variants are defended – will be detected on attempting to synthesize functional W.

In theory, defending all variants that do not completely abolish the function of W at a single essential window w_i can perfectly thwart the adversary, achieving $R = 1$. In practice, variant fitness prediction is imperfect.

We can maximize R while minimizing false alarms for a given set of windows by selecting the window predicted to be least tolerant of variation and adding variants with the highest predicted function scores (App. C Fig. C-2). At some point, our predictive accuracy will decline until it is better to start defending the window with the second-fewest predicted functional variants. As accuracy declines at the second window, we will either return to the first window or move on to the third.

Since real adversaries do not have perfect knowledge of fitness, we will deviate from the deterministically optimal strategy by stochastically screening for randomly chosen variants from quasi-randomly chosen windows. This forces the adversary to modify all windows throughout the hazard to have a chance at avoiding detection, reducing the odds of functional W.

5.2.5 Experimentally testing sensitivity

To experimentally measure the likelihood that an adversary could evade screening for diverse library sizes and window characteristics, we chose to treat the harmless M13 virus that infects E. coli as a "hazard". We began by analyzing M13 peptide windows using funtrp, a computational method that categorizes residues within proteins as "neutral" in tolerating most any mutation, "rheostat" in suffering reduced fitness from many but not all mutations, or "toggle" in losing all function when mutated[95] (App. C Suppl. Fig. C-3). From four different M13 proteins, we selected nine total windows with fairly low to very low neutral values and a range of rheostat and toggle scores.

Next, two "blue team" members constructed databases of 10^6 predicted functional variants for each window using a Metropolis-Hastings algorithm that combined the funtrp scores of each residue with the BLOSUM62 matrix of observed substitutions across proteins (App. C Extended Data Fig. 4). While many variant effector predictors are markedly

superior to BLOSUM62[87], our method serves as a baseline for the efficacy of defense that can certainly be improved upon.

"Red team" members experimentally tested the security of RAT search by launching up to 21,000 attacks at each of the nine windows (App. C Fig. C-3a). We ordered oligonucleotide pools with all possible combinations of the four most common substitutions at the six positions with the highest neutral scores, pairwise substitutions of all amino acids at those six positions, and all possible single substitutions, plus some predicted deleterious mutations as controls, generated libraries of variants, and measured their individual effects on phagemid replicative fitness (App. C Supplementary Figure C-4). Together, our libraries were equivalent to 10^{36} combinatorial attacks on the databases.

We defined "functional" variants as those with a measured fitness of at least 0.05 relative to wild-type, which is the level at which the most infectious virus known can just barely spread in an unprotected population[59]. Of the functional attacks on the most defensible window, fully 92% were blocked (App. C Fig. C-3b). That is, even an adversary with perfect knowledge of fitness who already possesses the other 99% of the wild-type M13 genome sequence was likely to be detected and thwarted just at this one window. While the other windows were less easily defended (App. C Fig. C-3c), most windows could still block 40-50% of attacks (App. C Fig. C-3 d-e).

Importantly, we observed that the average "toggle" score generated by funtrp for each of the nine windows was predictive of R at that window (App. C Fig. C-3e). Analyzing the false positive and false negative rates for prediction at each window allowed us to estimate the optimal number of database entries to devote to each window (App. C Extended Data Fig. ROC predictions). Together, these tools allow us to adopt deterministically optimal strategies for defense for adversaries lacking knowledge of window selection: we can pick a desired number of windows to defend, allocate a total number of database entries, set a target value of R, then distribute entries so as to meet R and leave the remainder to be allocated per game theoretical considerations.

Evenly distributing our $9{\times}10^6$ database entries blocked 99.96% of functional attacks on

this virus by an adversary with perfect knowledge of fitness. Since real adversaries lacking perfect knowledge of fitness are forced to include at least one nonsynonymous mutation for every nineteen amino acids throughout the genome, and we can afford to allocate 10^9 total entries per hazard, these results underscore the security of RAT search.

5.2.6 Defending against known hazards

Having established a reasonably efficient strategy for randomized defense, we next applied it to the set of well-known hazards considered most important for screening. We analyzed the genomes of all viruses, microbial pathogens, and known genes encoding toxins or pathogenicity islands from the Australia Group list for toggle scores using funtrp. Notably, all of the viruses save for swine vesicular disease virus harbored multiple windows with scores above 0.5, for which allocating 10^6 entries blocked over half of attacks (App. C Fig. C-3e). Since there are fewer than 100 Australia Group pathogens with publicly available genes or genomes, we can afford to compute and include variants for over a hundred windows per hazard. This can also effectively block the synthesis of closely related agents, which we can distinguish by virtue of software that assigns a taxon ID to most[8] GenBank entries.

After optimizing our Metropolis-Hastings algorithm, we chose to create an initial public hazard database comprising 10^7 variants for each of 10 different windows, 10^6 variants for another 40 windows, and $4{\times}10^5$ for a further 150 windows. We subsequently added 42-mer DNA sequences for all wild-type sequence windows, all 1-base mutants of those sequences, and a small number of 30-mers for each hazard.

5.2.7 Experimentally testing specificity

Now that we have a functional RAT search database capable of screening against all Australia Group pathogens, we are currently working with DNA synthesis providers to empirically measure the false positive rate using real customer order data.

[8] According to our observations and experience. Quantitative measure forthcoming in subsequent publications after Alpha version of SecureDNA screening software is complete.

5.2.8 Discussion

Protocols enabling the generation of functional viruses from synthetic DNA have given an increasing number of actors the capacity to build potential pandemic pathogens. As our knowledge of how to build such agents improves, so will the risk of misuse.

Well-resourced state actors cannot be prevented from making DNA, but they are vastly outnumbered by individuals and small groups who possess the necessary technical skills in biology. Screening all commercial DNA synthesis would substantially increase the difficulty of obtaining exponential agents for these non-state actors. A future in which fewer than a hundred groups can build pandemic-class agents is considerably safer than one in which such constructs are accessible to tens of thousands. Our results suggest that predicting functional variants of randomly chosen windows from hazards, curating them to remove any that match unrelated sequences from repositories, and searching DNA synthesis orders for exact matches would be superior to current methods and could automate screening for public and emerging hazards (App. C Table 5.3). Implementation could eventually become universal given appropriate incentives favoring incorporation into benchtop DNA synthesizers and assemblers, and eventually into next-generation enzymatic DNA synthesis machines intended for large-scale providers. Once the implementation has proven secure, cryptographic methods similar to those we employ to protect customer data might be used to screen for emerging hazards without inadvertently increasing the likelihood of misuse[122]. By offering a way to fully automate and secure DNA synthesis screening, random adversarial threshold search could substantially mitigate the global catastrophic risks posed by increasingly widespread access to pandemic-class biological agents.

5.2.9 Methods

Window selection: peptides

The proteins of bacteriophage M13 (Accession NC_003287.2) were analyzed using funtrp[95] to identify peptide windows with few predicted neutral positions and varying numbers of

	RAT search	Current similarity search screening
Speed	O(1); very fast	O(database size); slow
Minimum window size	19 amino acids	>= 200 base pairs
False alarm rate	Checking database vs repositories implies no nonrandom false alarms	Many matches to unrelated genes; needs human curation
Fully automatable	Yes	No, requires human curation
Compatible with benchtop synthesizers/assemblers	Yes, given Internet connection	No, requires human curation
Can screen for emerging hazards without disclosure	Yes[122]	No, requires disclosure; too inefficient to encrypt at scale[124]

Table 5.3. Characteristics of RAT search versus similarity search for DNA synthesis screening

toggle and rheostat positions across and within proteins (App. C Supplementary Table 1, Extended Data Figure 4).

From PI:

- High toggle: MAVYFVTGKLGSGKTLVSV (PI:1)
- High rheostat: YSYLTPYLSHGRYFKPLNL (PI:219)

From PII:

- High toggle: VEIKASPAKVLQGHNVFGT (PII:89)
- Higher neutral, mostly toggle: NFYPCVEIKASPAKVLQGH (PII:84)
- Low neutral overall with several mid-neutral: LLDVNATTISRIDAAFSAR (PII:131)

From PIII:

- High toggle: PQSVECRPFVFGAGKPYEF (PIII:367)
- High toggle: FRGVFAFLLYVATFMYVFS (PIII: 395)
- High rheostat: YANYEGCLWNATGVVVCTG (PIII:48)

From PIV:

- High rheostat: IATTVNLRDGQTLLLGGLT (PVI:360)

Specifically, windows were chosen to enable the following comparisons of defensibility[9]:

Given few neutral positions, how important is the number of toggle vs rheostat positions? (higher=important)

Sequence (protein:amino acid start)	Neutral	Rheostat	Toggle
VEIKASPAKVLQGHNVFGT (PII:89)	1.12	3.48	14.4
MAVYFVTGKLGSGKTLVSV (PI:1)	1.46	7.11	10.43
FRGVFAFLLYVATFMYVFS (PIII: 395)	1.6	8.37	9.03

Given few neutrals, how important is the number of toggle vs rheostat positions within proteins?

Sequence (protein:amino acid start)	Neutral	Rheostat	Toggle
MAVYFVTGKLGSGKTLVSV (PI:1)	1.46	7.11	10.43
YSYLTPYLSHGRYFKPLNL (PI:219)	1.23	13.62	4.15
PQSVECRPFVFGAGKPYEF (PIII:367)	2.6	5.76	10.64
YANYEGCLWNATGVVVCTG (PIII:48)	3.03	12.17	3.8

Given more neutral positions, is it better to choose windows with more toggle or rheostat positions?

[9]To interpret tables, compare relative values of Neutral, Toggle and Rheostat scores compared with the objectives named above each.

Sequence (protein:amino acid start)	Neutral	Rheostat	Toggle
NFYPCVEIKASPAKVLQGH (PII:84)	3.47	4.39	11.14
YANYEGCLWNATGVVVCTG (PIII:48)	3.03	12.17	3.8
IATTVNLRDGQTLLLGGLT (PIV:360)	3.08	10.42	5.5

For windows with comparable mean neutral scores, is it better if they are concentrated or spread out?

Sequence (protein:amino acid start)	Neutral	Rheostat	Toggle
NFYPCVEIKASPAKVLQGH (PII:84)	3.47	4.39	11.14
LLDVNATTISRIDAAFSAR (PII:131)	3.48	7.43	8.09

Procedurally generating variants for each 19aa peptide window

1. We included the wild-type sequence (1)

2. We included all one-mutants at each position (19 × 19 = 361)

3. At the six positions predicted to be most neutral, we added all combinations of the four predicted least pathological substitutions according to BLOSUM62 ($5^6 = 15,625$) (overlaps with one-muts and WT at $4 \times 6 + 1 = 25$)

4. As negative controls, we included up-to-six mutants of neutral positions using the two most pathological substitutions according to BLOSUM62 ($3^6 = 729$) (overlaps with one-muts and WT at $2 \times 6 + 1 = 13$)

5. We added all pairwise combinations of all possible substitutions at the six most neutral positions ($19^2 \times 15$ pairwise combinations $= 5415$) (overlaps with 4 most tolerated at $4^2 \times 15 = 240$) (overlaps with 2 most pathological at $2^2 \times 15 = 60$)

Total: $1 + 361 - 13 + 15625 - 25 + 5415 - 240 - 60 = \boxed{21,793}$ peptide variants at each window

Procedurally generating nucleic acid variants for each 42-mer window

1. We analyzed the region of DNA along with 60bp of flanking sequence in NUPACK in order to assess the likely secondary structure in a knowledge-agnostic manner.
2. For each probably-unpaired, we included all 1-mutants.
3. For each probably-paired, we included all 2-mutants.
4. We included all 2-combinations of the top 10 highest probability single and paired bases.
5. We included all up-to-three mutants of the three highest probability stem pairs and the three highest probability unpaired bases.

However, when we ordered oligos and constructed libraries of variants at these positions, we found that they could not be reliably sequenced, presumably due to secondary structures. We consequently chose to concentrate RAT screening evaluation on peptide windows, but will include 42-mers comprising many synonymous mutants of the wild-type sequence and many predicted functional variants to block the assembly of hazards from short fragments by the small number of non-state actors who are capable of doing so.

Construction of phagemid libraries

Oligo libraries comprising variants for each 19aa peptide window were synthesized as a pool by Twist Bioscience. Individual libraries were amplified by PCR and ligated into a phagemid backbone—encoding an ampicillin resistance gene, containing an M13 phage origin of replication, and designed for library variant expression upon induction by IPTG—using NEBuilder Hifi DNA Assembly Master Mix (NEB, E2621L). Nucleic acid libraries within the origin of replication were ligated into phagemid backbones expressing wild-type pIII.

All libraries were then precipitated with isopropanol, transformed into electrocompetent DH5α cells (NEB, C2989K), and plated on 2XYT-carbenicillin-1% glucose; after overnight growth at 37°C, colonies were counted to ensure >50-fold library coverage. Colonies were scraped with 2XYT and plasmid DNA extracted with the ZymoPURE II Plasmid Maxiprep Kit (Zymo Research, D4203); the extracted plasmid DNA was then precipitated with isopropanol. These plasmid libraries constitute the "pre-selection libraries."

Construction of helper cells

M13cp[24], a plasmid containing all M13 phage genes but with a p15a origin and a chloramphenicol resistance gene replacing the phage origin of replication, was used to construct helper plasmids. Primer pairs were designed for the precise deletion of genes I, II, III, and IV from M13cp following PCR amplification and ligation using the In-Fusion Snap Assembly Master Mix (Takara Bio, 638944). The resulting helper plasmids were transformed into DH5α competent cells (NEB, C2987H), yielding four individual helper cell lines (M13cp-dg1, M13cp-dg2, M13cp-dg3, and M13cp-dg4). The helper cells were made electrocompetent for subsequent same-day transformations. Helper cells are capable of extruding phagemid particles when transformed with a phagemid library variant with a functional gene (complementing the missing phage gene in the helper plasmid) and origin of replication (App. C Supplementary Figure C-1).

Phagemid growth

Phagemid libraries were transformed into their corresponding helper cells (nucleic acid variant libraries were transformed into M13cp-dg3) by electroporation and plated on 2XYT-carbenicillin-chloramphenicol-1% glucose. After overnight growth at 37°C, colonies were counted to ensure >15-fold library coverage. Colonies were scraped with 50 mL 2XYT, the bacterial pellet washed sequentially 3× with 50 mL 2XYT, then a 1:1000 dilution used to inoculate a 50 mL phagemid growth culture in 2XYT with maintenance antibiotics and 1% glucose. The culture was grown to OD600 = 0.5 with shaking at 37°C and 250 rpm,

at which point the culture was centrifuged and the media replaced with 2XYT containing maintenance antibiotics and 1 mM IPTG. The culture was grown for 16 h at 37°C and 250 rpm, after which phagemid-containing supernatants were collected by culture centrifugation and filtration through a 0.22 μm filter.

Phagemid infection

Phagemid-containing supernatants were added to 2.5 mL S2060 cells (streptomycin-resistant, Addgene 105064) grown to OD600 = 0.5 and allowed to infect at 37°C and 250 rpm for 1 h. The resulting infected cultures were plated on 2XYT-carbenicillin-streptomycin-1% glucose to select for phagemid-containing cells. After overnight growth at 37°C, colonies were scraped with 50 mL 2XYT and plasmid DNA extracted with the ZymoPURE II Plasmid Maxiprep Kit (Zymo Research, D4203). These plasmid libraries constitute the "post-selection libraries."

Illumina NGS sequencing

Pre- and post-selection libraries were prepared for illumina NGS sequencing by sequential PCR amplification. PCR amplification was first performed with PrimeSTAR GXL Premix (Takara Bio, R051A) to attach Nextera-style adapter sequences, followed by a second PCR amplification to attach library-specific barcodes and the p5 and p7 indices. Following PCR purification, library concentrations were quantified with qPCR using the NEBNext Library Quant Kit for Illumina (NEB, E7630S), and pre- and post-selection libraries were combined as two pools. Libraries were pooled such that libraries were present in equimolar quantities corrected for library size. Libraries were submitted to the MIT BioMicro Center for MiSeq Illumina sequencing (v3, 2 × 300 bp paired-end).

5.2.10 Implementation

Public hazards

We have predicted and curated variants of sequences from each gene or genome listed on the Australia Group List or the U.S. Department of Health and Human Services Select Agent List. Providers may use this database to test the efficacy and false alarm rate of random adversarial threshold screening. We hope to see the system in widespread use by 2022, including its incorporation into the first benchtop synthesis and assembler machines.

Emerging hazards

RAT search is compatible with cryptographic approaches capable of obscuring the identities of entries in an emerging hazard database. Such a system would enable a researcher concerned about an emerging potential biological agent to safely take action to restrict global access without creating information hazards[20, 81] by securely conveying their concern to one of the biologist curators responsible for emerging hazards. If a curator concurs that the threat is serious, they could use their unique key to add sequences from the hazard to the encrypted emerging hazard database without requiring further disclosure. Several times as many genes or genomes would be chosen as "decoys:" related genes or agents that might seem to pose a threat, but are not actually of concern. Decoys can ensure that anyone who finds a match to the database will learn only that it corresponds to a plausible-seeming threat, not that it is a credible destructive agent. This would ensure that adversaries cannot learn whether a virus can be used to build an "agent of mass destruction" by attempting to synthesize it. A detailed explanation of the cryptographic approaches required is available elsewhere[122]. Emerging hazards would only be added to the database after multiple years of testing to verify that the implementation is secure.

5.3. Additional discussion and conclusion

5.3.1 Performance on Bacterial Genomes

Because bacterial pathogenicity is much more convoluted and ambiguous than viral pathogenicity[12], bacterial hazard detection presents significant challenges to exact match screening. There are two major factors relating to SecureDNA's potentially reduced capability on bacteria.

1. While whole bacterial genomes have been assembled from synthetic DNA[89], such genomes are still so inconvenient and expensive to produce that whole genome synthesis is not the primary mode to protect, in contrast to the situation for most viruses. Instead, *in vivo* pathogenesis by gene insertion or knockout is the expected mode to make use of existing bacteria for destructive purposes[55][10]. To counter this threat, SecureDNA's hazard database may be augmented with any genes known to be useful for such purposes. To the extent that the addition or subtraction of one or a few genes can give rise to pathogenicity[55], SecureDNA experts will be aware of any public sequences of which an attacker might also be aware, and key subsequences may be protected. Regarding counterintuitive means of inducing pathogenicity in bacteria, if attackers employ methods not well-known in the literature, and without using any specific constructs we can guess, such attackers must embark on an ambitious research project. This level of required sophistication is sufficiently high for the defensive objectives of SecureDNA.

2. Taking a broader view, while SecureDNA must detect attempts to synthesize known pathogenic bacteria, e.g. on the US Select Agent list for regulatory purposes, in terms of pandemic potential, bacteria are regarded as a lower priority than viruses. Though the deadliest pandemic in history, the plague, is caused by a bacterium, *Yersinia Pestis*, bacteria are still less concerning as engineered biohazards in the short term because of the great difficulty of creating complete, novel pathogenic bacteria on

[10] Other bioengineering means may be employed to pathogenize bacteria, such as directed evolution, but those that do not make use of synthetic DNA are outside the scope of what can be protected by SecureDNA.

purpose using synthetic DNA. Other tools, such as SeqScreen[12], excel on the general screening task of estimating the pathogenic potential of DNA of bacterial origin, and flagging it for review. The defense niche that SecureDNA's precision screening features seek to fill is to unfailingly stop synthesizers with no human review, at the hardware level, whenever a sequence that has been marked as a hazard appears. Strictly at the hardware level, passing over a putatively concerning bacterial gene, for which pathogenicity is at best a strong guess, is actually the desired behavior. SecureDNA's general screening features conservatively flag bacteria at the species level.

In summary,

- If an order contains a flagrant characteristically pathogenic sequence from a listed bacterium, SecureDNA halts the synthesizer. This behavior is sufficient to comply with present regulation.
- If attackers are using public information to decide what sequences they will use to "pathogenize" a known bacterium, we can predict the sequences of concern. SecureDNA can include these telltale sequences and their variants as "hazardous sequences," which they are in this context, even if they do not appear on any pathogen lists.
- If it is necessary to launch ones own independent research effort to discover how to engineer bacterial pathogenicity, one likely has the capability to modify or fully manufacture ones own synthesizers, which voids the utility of SecureDNA (out of scope).
- Ultimately, bacteria hold a much lower priority for precision screening than viruses.

Chapter 6

Ongoing Work and Future Directions

6.1. Pyhamilton adoption and PyLabRobot

Since its development in 2017-2019 and initial publication in 2020-2021, pyhamilton has been applied in multiple laboratories possessing Hamilton robots. Pyhamilton has found a home within the emerging PyLabRobot project, whose ambition is to enable general-purpose, brand-agnostic, open-source libraries for interacting with automated laboratory devices, such as liquid handling robots, plate readers, pumps, and others. This group has already adapted pyhamilton to bypass Venus entirely, greatly expanding the list of supported robot commands and behaviors.

6.2. Preliminary SecureDNA hardware implementation

A tabletop hardware demonstrator called "SecureDNA Mini" has been constructed (App. C Fig. D-1). SecureDNA Mini is a collection of ARM-architecture minicomputers (Raspberry Pi and others) connected together via a local area network, each of which runs a distinct role in SecureDNA. SecureDNA's Alpha has reached the level of containerization during development, meaning that each client/server role in SecureDNA is implemented as a container that can issue and receive requests to other containers. However, these containers operate within a host computer's local loopback network, not between physically dis-

tinct computers. SecureDNA Mini is the first fully networked, functional implementation of SecureDNA. Its purpose is to concretize, or physicalize, SecureDNA's relatively complex architecture, in order to provide a straightforward means of explaining and demoing SecureDNA. It also serves to remove any doubt that SecureDNA, with all its cryptographic innovations described in the various manuscripts, is operational today.

An updated SecureDNA hardware demonstrator in currently in development, which leverages the strict session-based enforcement of the cryptographic functionalities discussed in Section 5.1.6. It remains an open research question whether the sequence manipulations and cryptographic functions necessary to operate under SecureDNA's strict protocols may be embedded in *purely combinational* logic suitable for integration on-chip on synthesizers' control electronics. Ideally, there will be no software that could be compromised. Most likely, if this path proves fruitful, a dedicated hardware integration effort will be required at each synthesizer manufacturer. Should SecureDNA become the global standard for hardware DNA synthesis security, SecureDNA hardware engineers will consult, ideally *gratis*, to provide this service.

6.3. Conclusion

Pyhamilton and SecureDNA represent innovations at the platform level, which help in a small way to further widen the problem space that is safely addressable by biology. Future work will primarily focus on bringing SecureDNA to fruition as a finalized and globally available software package, which is compatible with DNA synthesizers at a hardware level. The work that a successful SecureDNA deployment would serve, however, is better represented by pyhamilton and related systems, as more advanced, standardized, reliable, shareable methods emerge to explore the microscopic and nanoscopic possibilities in biology.

Appendix A

Graphical and Complementary Content

of Erika A. DeBenedictis, *et. al.*

Systematic molecular evolution enables robust biomolecule discovery

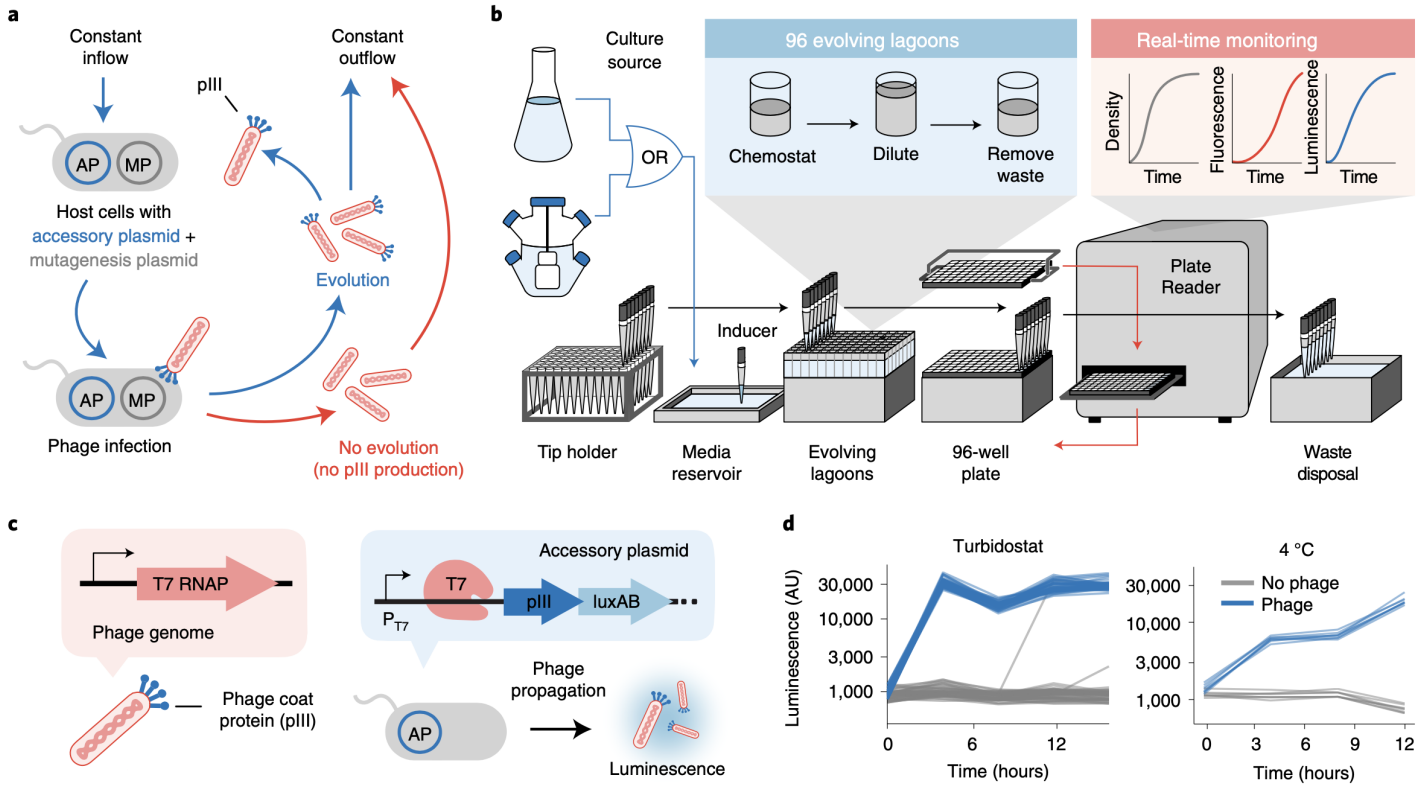


Fig. 1 | Design and validation of high-throughput evolution. a, Overview of continuous evolution. **b**, PRANCE method summary for up to 96 simultaneous experiments. Movements by robotic pipette (black arrow) and plate reader interactions (red arrow) are shown. **c**, Propagation of T7 RNAP-encoding phage on bacteria expressing pIII and luxAB driven by a T7 promoter. **d**, Real-time monitoring of T7 RNAP-expressing phage propagation by monitoring activity-dependent luminescence monitoring from cultures sourced either from a turbidostat (left) or from a culture stored at 4 °C (right). AU, arbitrary units.

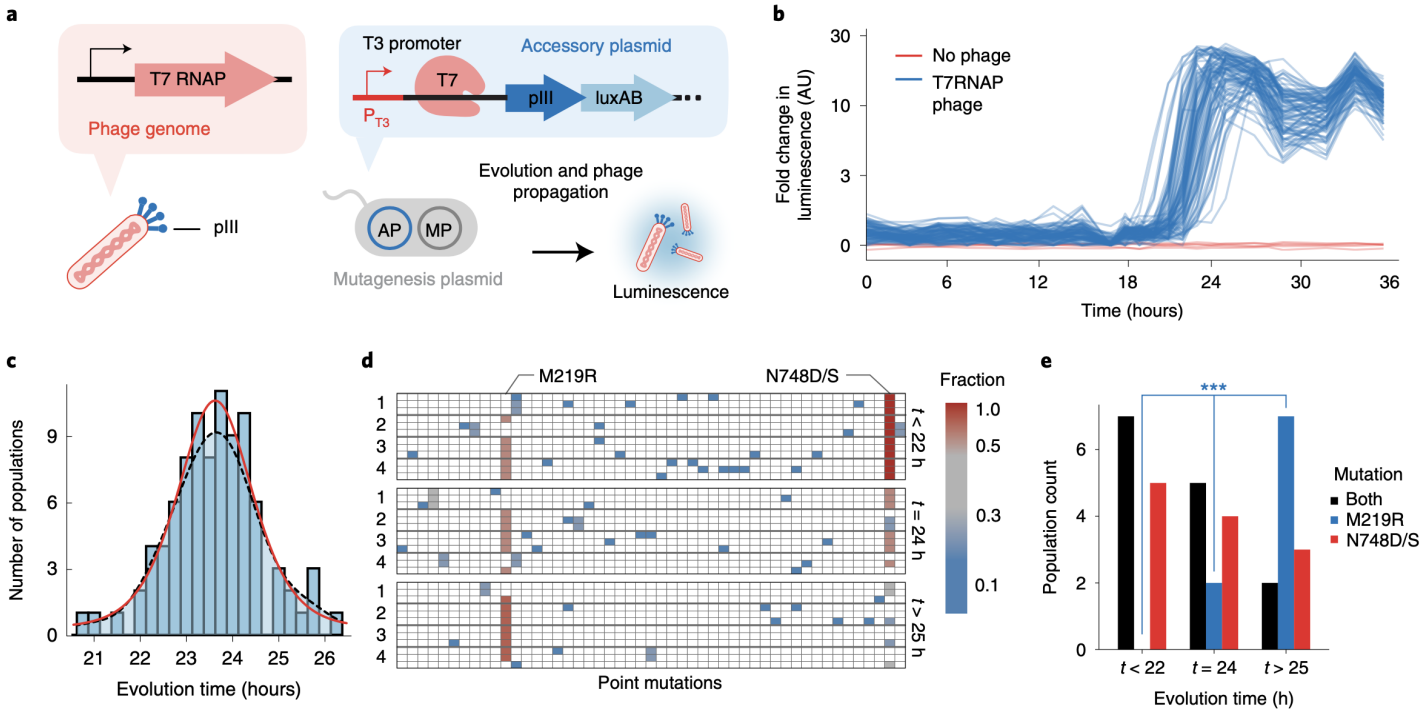


Fig. 2 | Quantifying the stochasticity of biomolecular evolution. a, Strategy for evolving T7 RNAP to recognize the T3 promoter, in which expression of pIII is driven by the T3 promoter on an accessory plasmid and the evolving T7 RNAP is encoded on the replicating phage genome. **b**, Real-time luminescence monitoring of 90 simultaneous populations with six no-phage controls. **c**, Histogram of times required to acquire mutations permitting the T7 RNAP to recognize the T3 promoter, obtained from the inflection point of logistic regressions of each population (Extended Data Fig. 5, Methods and Data analysis). Smoothed fit is calculated with a kernel density estimate (black dashed line) or logistical distribution fit (red). **d**, Mutations observed from 12 representative populations that exhibited evolution of early (-1σ, $t<22$ h), mid (mean, $t=24$ h) and late ($+1\sigma$, $t>25$ h) time points. Three clonal phage from each population are shown, filled-in boxes indicate a mutation at a given location and are colored by 'fraction' referring to the mutational frequency within the given time window ($t<22$, $t=24$, $t>25$), where red is 12/12 clones, blue is 1/12 of the clones. Genotypes of subclones are listed in Supplementary Table 2. **e**, Frequency of mutations arising that exhibited evolution of early (-1σ, $t<22$ h), mid (mean, $t=24$ h) and late ($+1\sigma$, $t>25$ h) time points. *** indicates time-dependent chi-square $P=0.0037$.

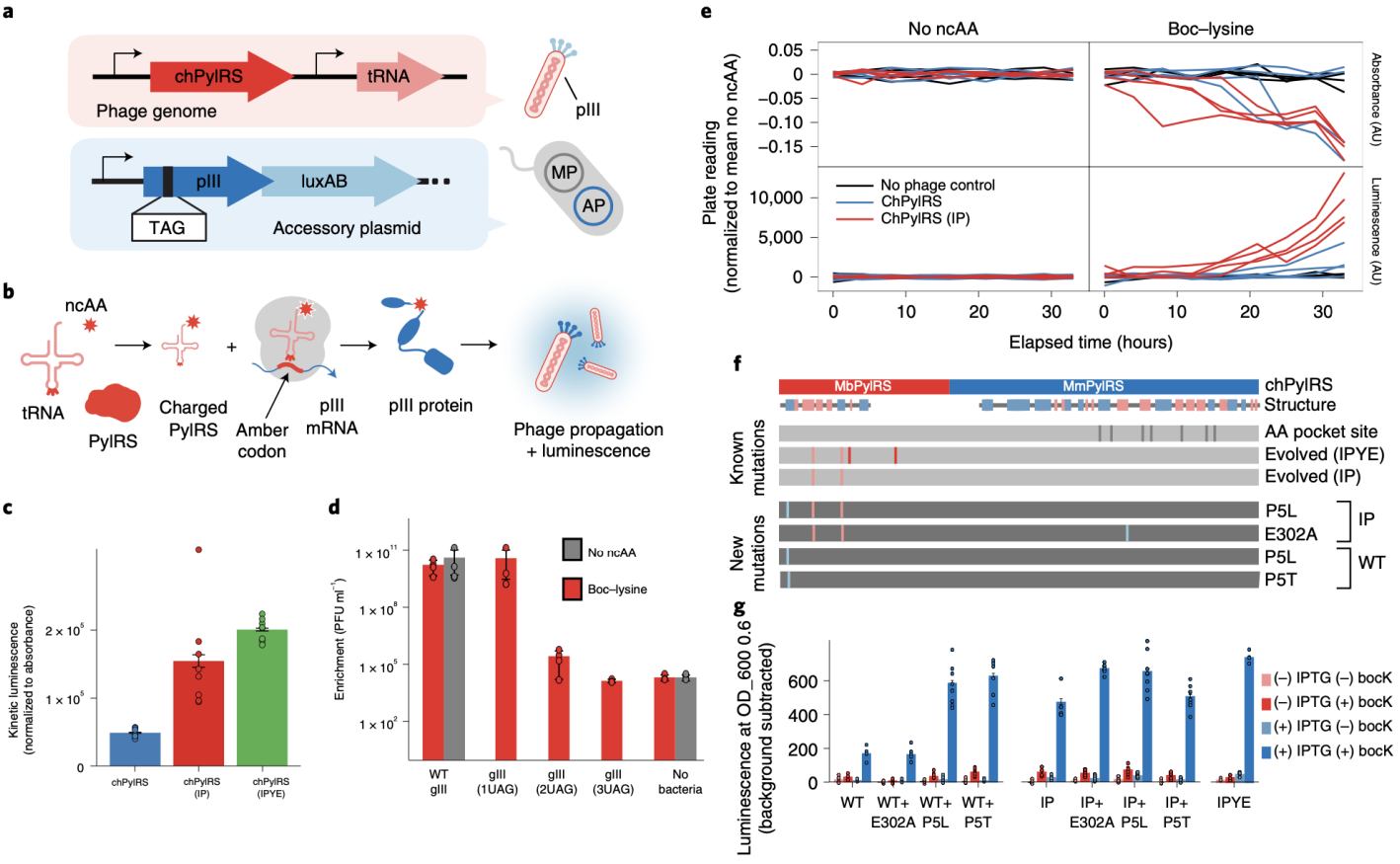


Fig. 3 | Controlling the chemical environment in high-throughput evolution. a, Strategy for evolving AARSs and tRNAs to incorporate noncanonical amino acids. TAG amber codons are inserted into the pIII protein, and the evolving $tRNA^{Pyl}$ and chPylRS are encoded in the evolving phage genome. **b**, Phage propagation and luminescence are contingent on the successful incorporation of ncAAs into pIII. **c**, Efficiency of unevolved versus evolved chPylRS variants (IP, IPYE mutations) at incorporating Boc-lysine into an inducible TAG-luxAB reporter, normalized to no-ncAA and no-IPTG controls. Data are presented as mean values ± s.e.m. for $n = 8$ biologically independent samples. **d**, Quantifying the selection stringency of incorporating one, two or three ncAAs into pIII, in either the presence or absence of Boc-lysine, using the evolved variant chPylRS-IPYE. Data are presented as mean values ± s.e.m. for $n = 4$ biologically independent samples. **e**, Real-time absorbance depression and luminescence monitoring beginning from either the unevolved state (chPylRS, blue) or an intermediate evolved state (chPylRS-IP, red). **f**, Genotypes of the evolved variants. Phage persisted in all of the chPylRS-IP populations, and clonal phage acquired novel mutations (P5L or E302A). Only half of the chPylRS populations resulted in persistent phage propagation at 36 h; each acquired a distinct mutation at the same N-terminal proline residue (P5L or P5T) within the conserved essential N-terminal domain of PylRS[44,45]. **g**, Efficiency of evolved variant mutations in both chPylRS and chPylRS-IP at incorporating Boc-lysine into an inducible TAG-luxAB reporter, compared to no-ncAA and no-IPTG controls. Data are presented as mean values ± s.e.m. for $n = 8$ biologically independent samples.

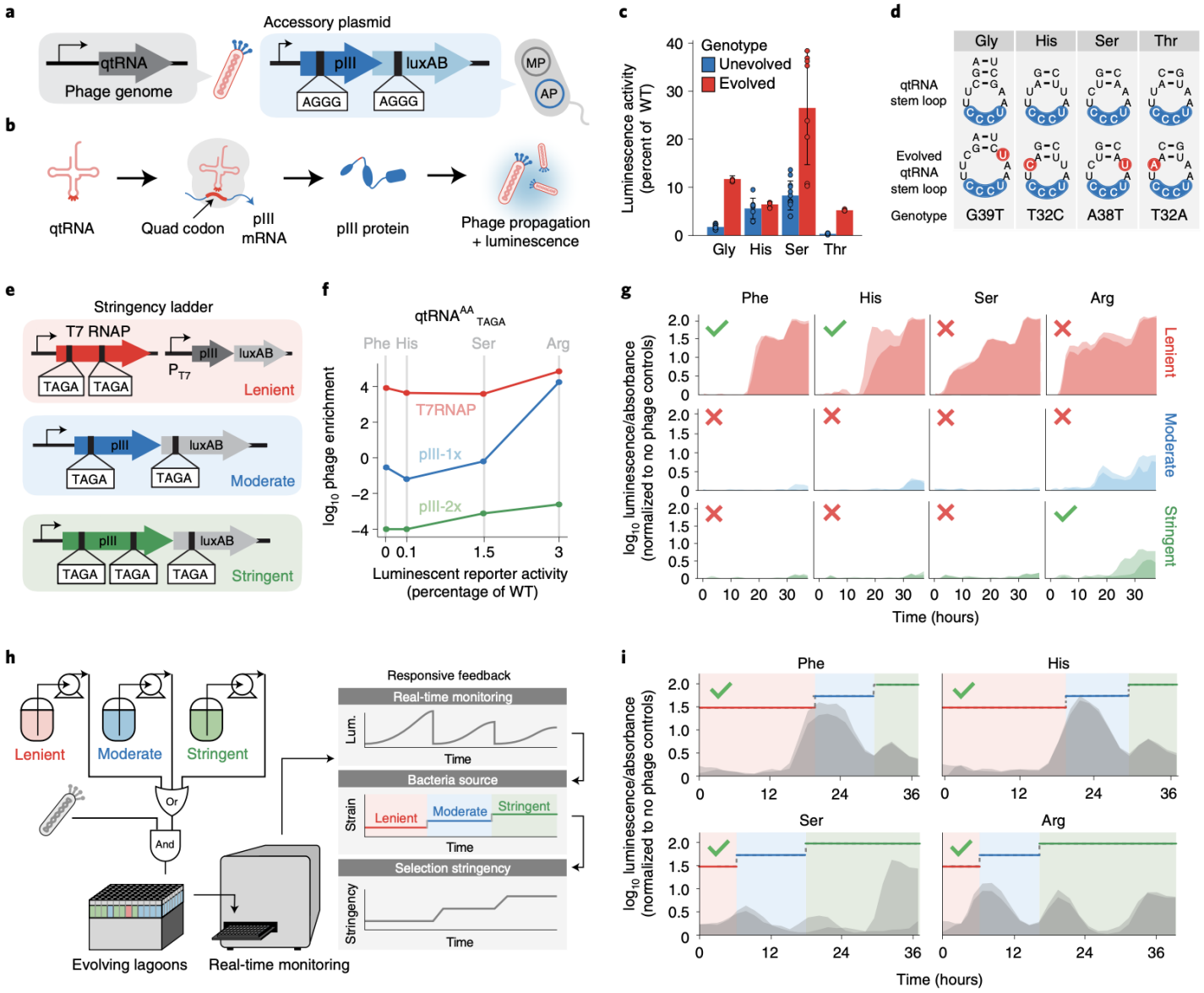


Fig. 4 | Feedback-controlled evolution of diverse starting genotypes. a, Strategy for evolving qtRNA to recognize quadruplet codons. AGGG quadruplet codons are inserted into the pIII protein and luxAB, and the evolving qtRNA are encoded in the evolving phage genome. AP, accessory plasmid. **b**, Phage propagation and luminescence are contingent on the successful decoding of the quadruplet codons in the pIII and luxAB proteins. Failure to decode a quadruplet codon results in premature termination and truncated protein. **c**, Efficiency of evolved versus unevolved qtRNA at incorporating amino acids compared to a triplet codon. **d**, Initial and evolved qtRNA genotypes. Data are presented as mean values ± s.e.m. for $n = 3$–8 biologically independent samples. **e**, Strategy for evolving qtRNAs with increasing stringency of selection (red/lenient, T7 RNAP; moderate/blue, pIII-1x; stringent/green, pIII-2x). **f**, The transfer function that determines the efficiency of phage propagation as a function of biomolecule activity for each AP, measured using phage-bearing qtRNAs at different starting activities (Phe, His, Ser, Arg). Starting activity is quantified as percentage of WT, or the luminescence generated by coexpressing the qtRNA and luxAB-357-TAGA, relative to an all-triplet luxAB. **g**, Evolution of four qtRNAs on APs with varying stringencies (successful evolution indicated by green checkmarks). **h**, In a feedback experiment, three bacterial strains with increasing stringency are added to phage populations and monitored in real time. The bacteria source for each well is adjusted in response to real-time luminescence measurements to automatically increase stringency of the environment. **i**, Real-time luminescence measurements of evolving qtRNAs with feedback-controlled selection stringency intervals.

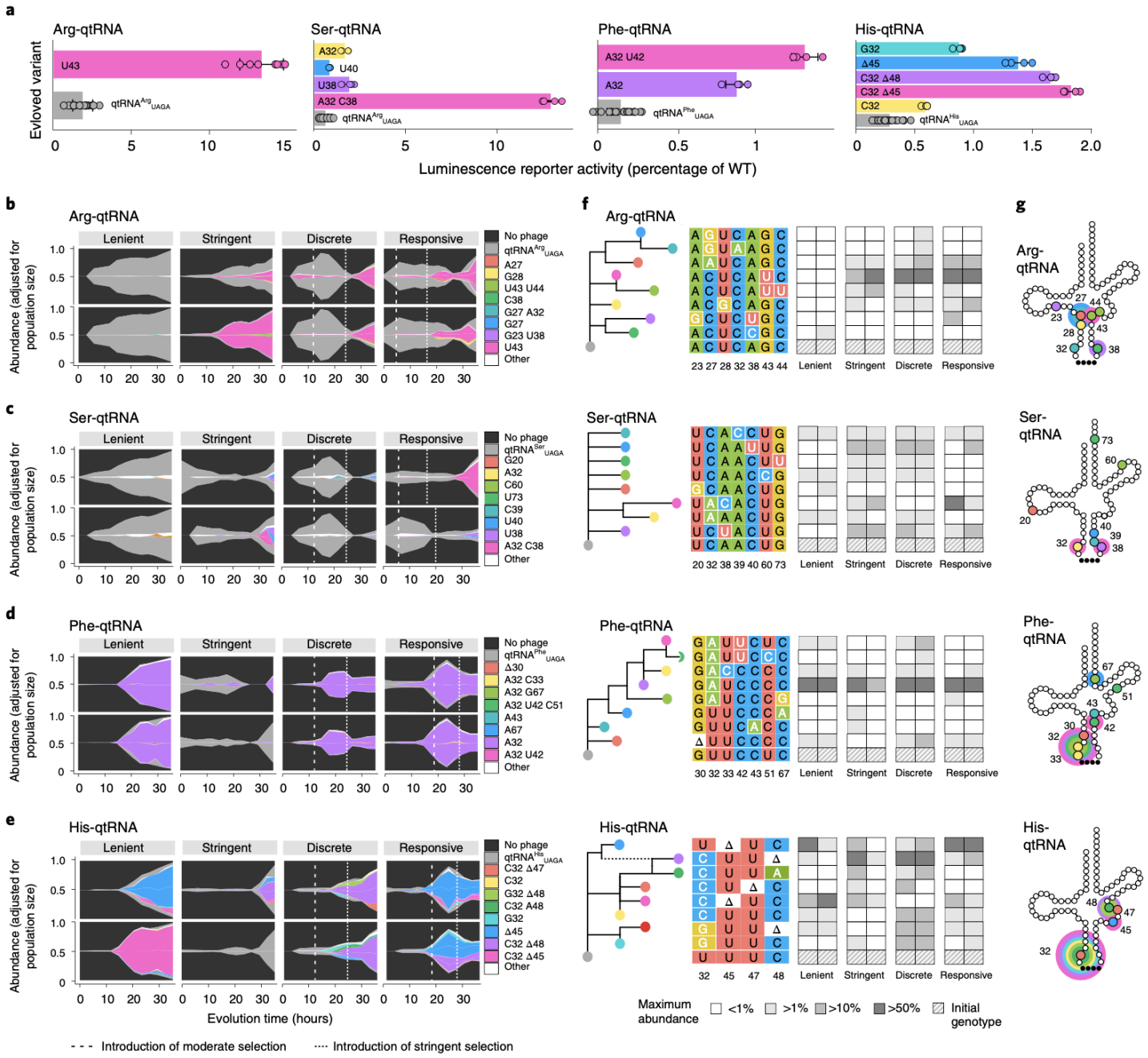


Fig. 5 | Varying the timing of environmental changes yields diverse evolution trajectories. a, Relative activity of the initial and evolved variants as measured by a kinetic luminescence assay. Data are presented as mean values ± s.e.m. for $n = 3$–8 biologically independent samples. **b**–**e**, Muller plots as abundance adjusted for population size for Arg-qtRNA (**b**), Phe-qtRNA (**c**), Ser-qtRNA (**d**) and His-qtRNA (**e**). The dashed line shows the introduction of moderate selection and the dotted line shows the introduction of stringent selection. **f**, Predicted phylogenetic relationship of the variants arising from evolution of each qtRNA, and their respective maximum population abundance between replicates. **g**, Location of mutations within the standard qtRNA secondary structure. The colors of variants arising in each evolution are consistent with the legends in Fig. 6b–e.

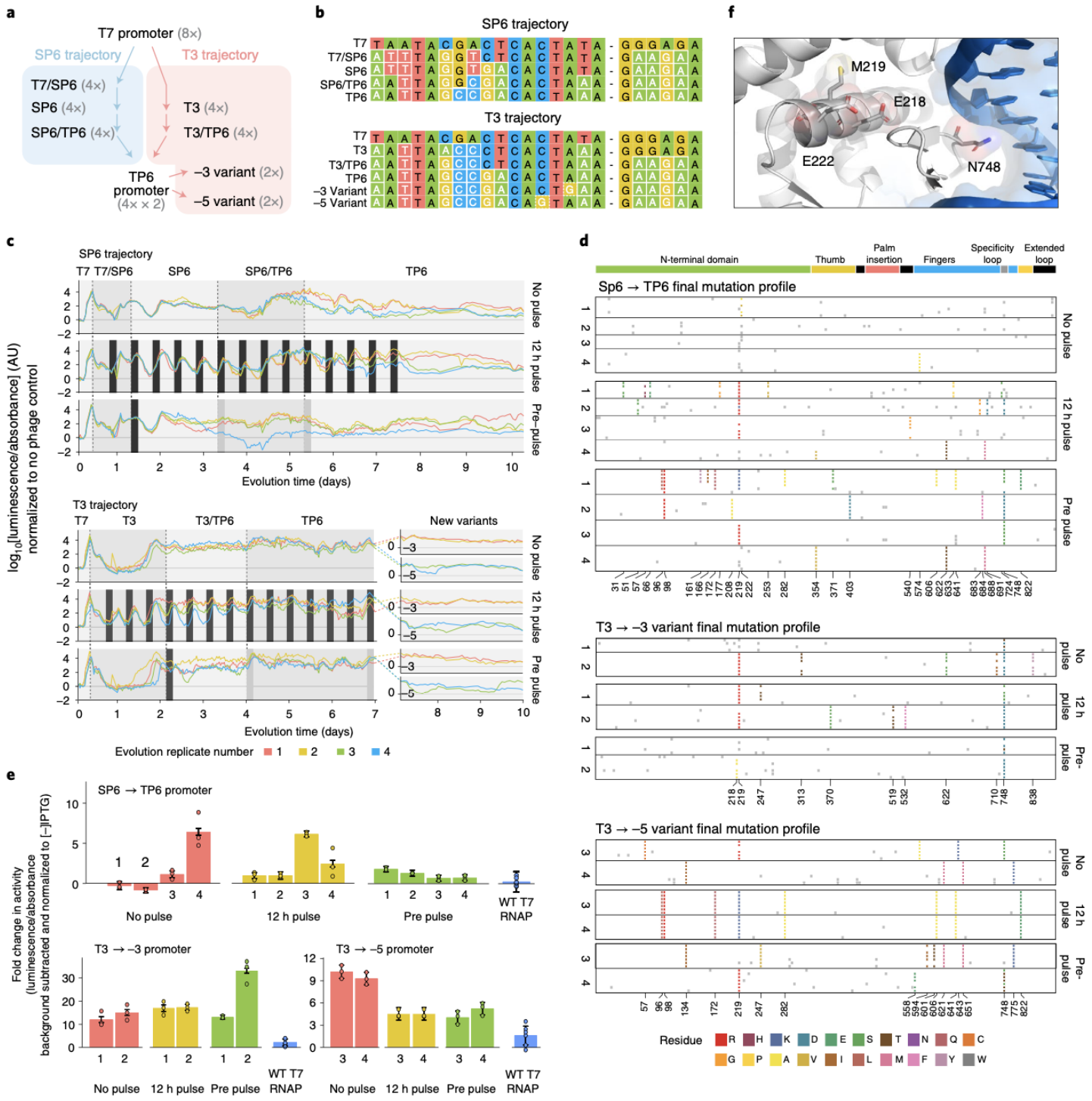
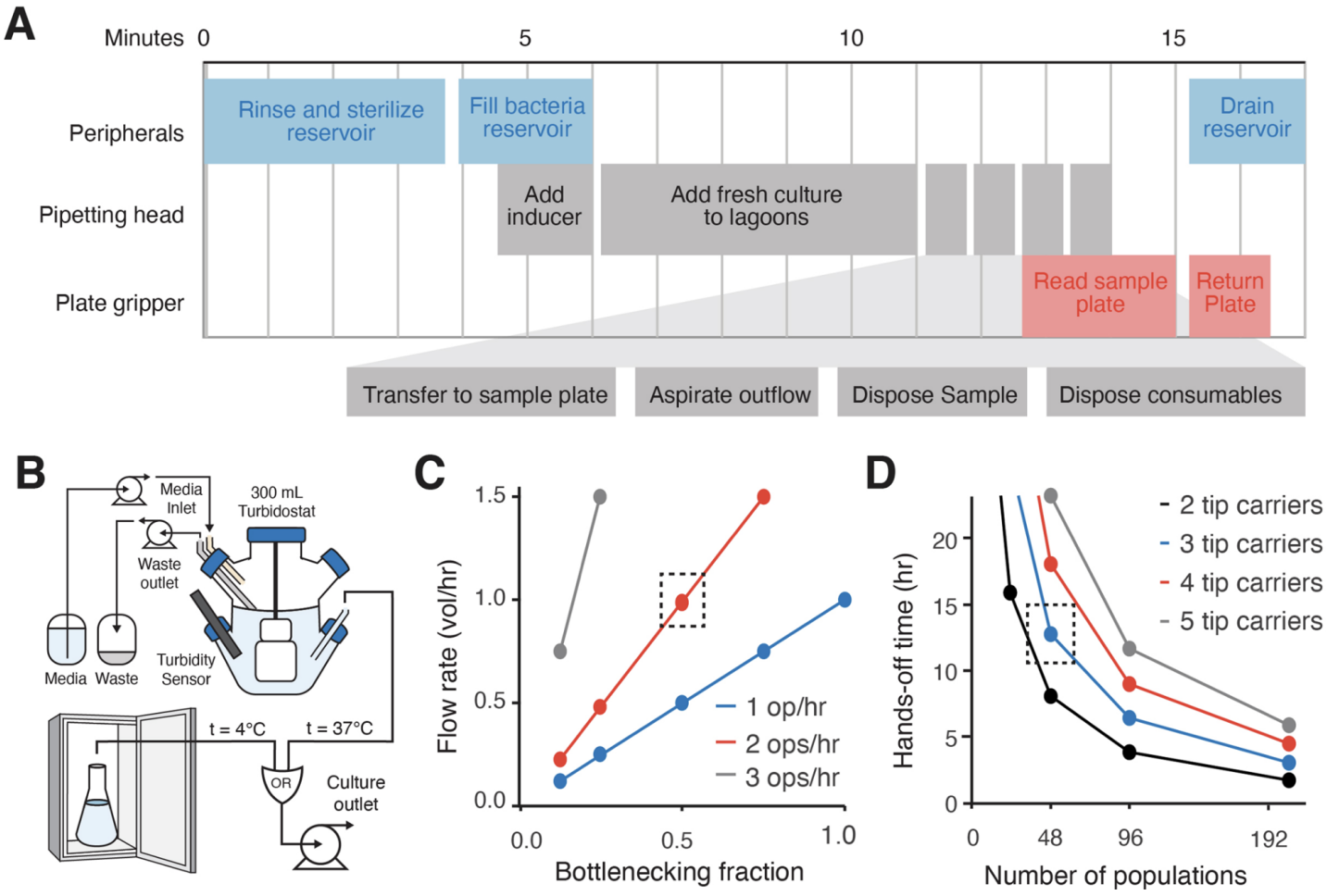
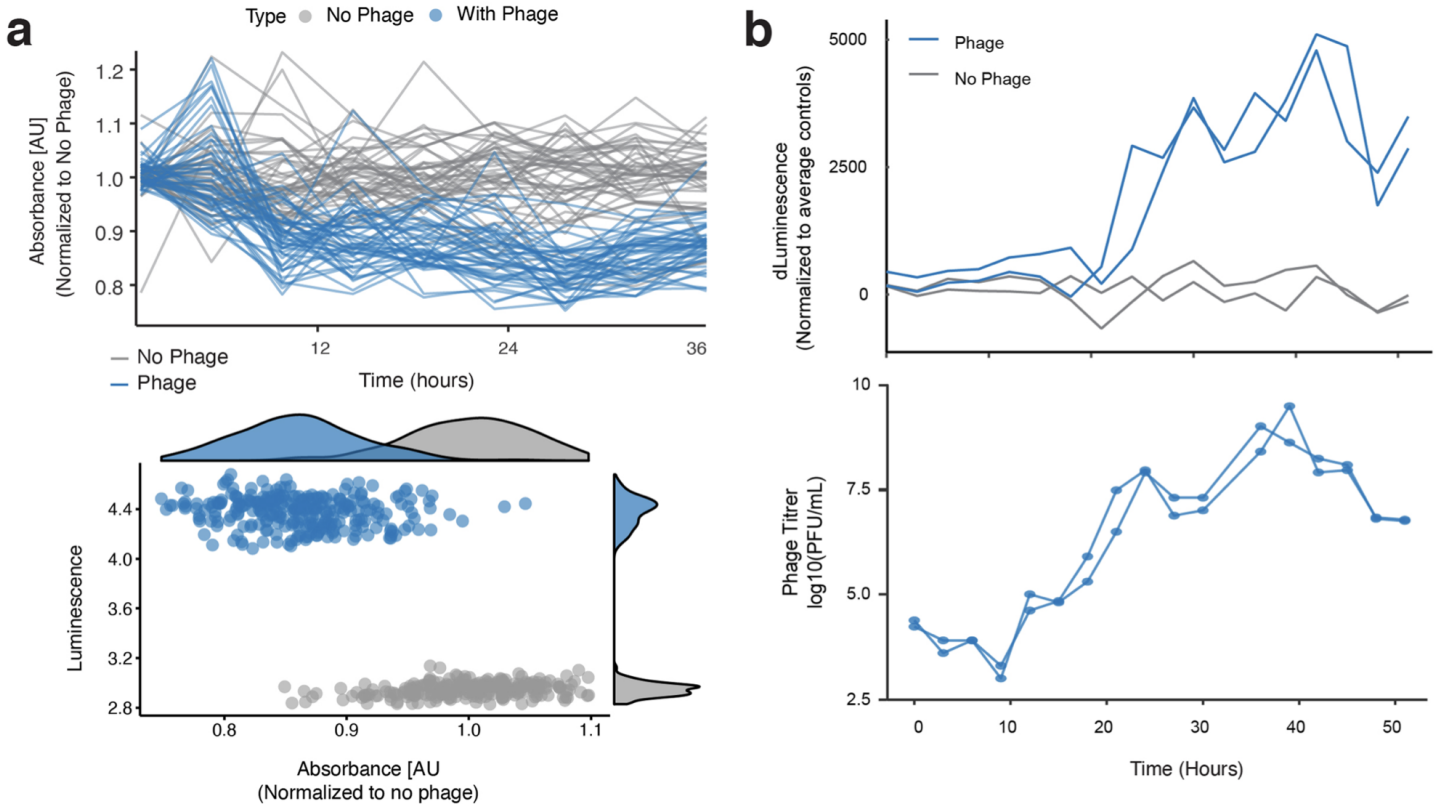


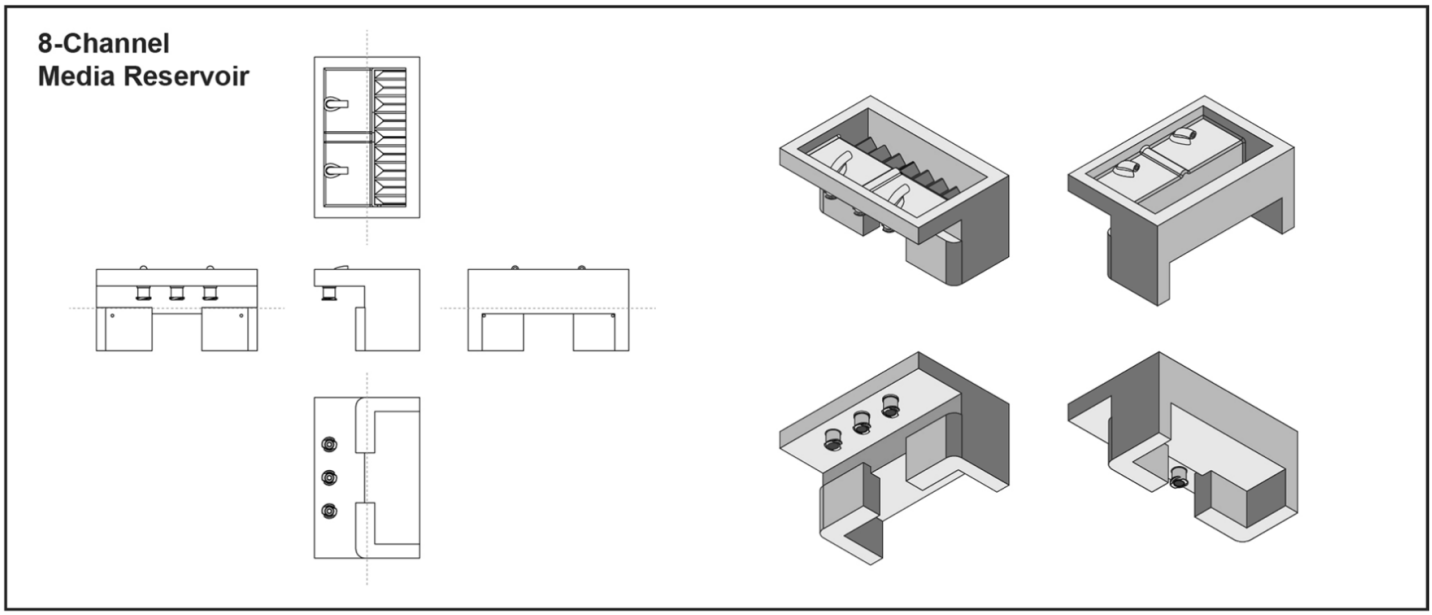
Fig. 6 | Long-term evolution. a, Thirty-two populations of phage encoding T7 RNAP were evolved to bind new promoters. Evolution proceeded first along two different paths, being challenged to bind either SP6 or T3-derived promoters. All populations were then evolved to bind the TP6 promoter. The 16 populations that traversed the T3 path were then split in half and challenged to bind novel promoters with mutations at highly conserved −3 or −5 locations. **b**, Genotypes of the promoters used in this evolution. Changes from the WT T7 promoter are highlighted. **c**, Real-time monitoring of the 48 populations undergoing three different stringency management schemes in which psp-pIII bacteria are periodically spiked into populations for 3-h intervals (dark gray bars) to increase population size. **d**, Genotypes of subclones obtained from each population (replicate populations labeled 1–4) from each evolution trajectory with each of the three stringency management schemes. Dominant (>50% of population) variants are highlighted and colored by their AA mutation. Nondominant (that is background mutations) are shown in light gray. The *x* axis is annotated by dominant mutations that appear within a given trajectory. **e**, Normalized luciferase activity of individual subclones from populations (1–4) from each trajectory, compared to WT T7 RNAP (blue). Luciferase reporter vectors are driven by the TP6 promoter, −3 variant promoter and −5 variant promoters. Data are presented as mean values ± s.e.m. for $n = 3$ biologically independent samples. Genotypes of subclones are listed in Supplementary Table 3. **f**, E218, M219 and E222 in the WT T7 RNAP structure (PDB 1CEZ, ref. [46]), near the promoter specificity loop.

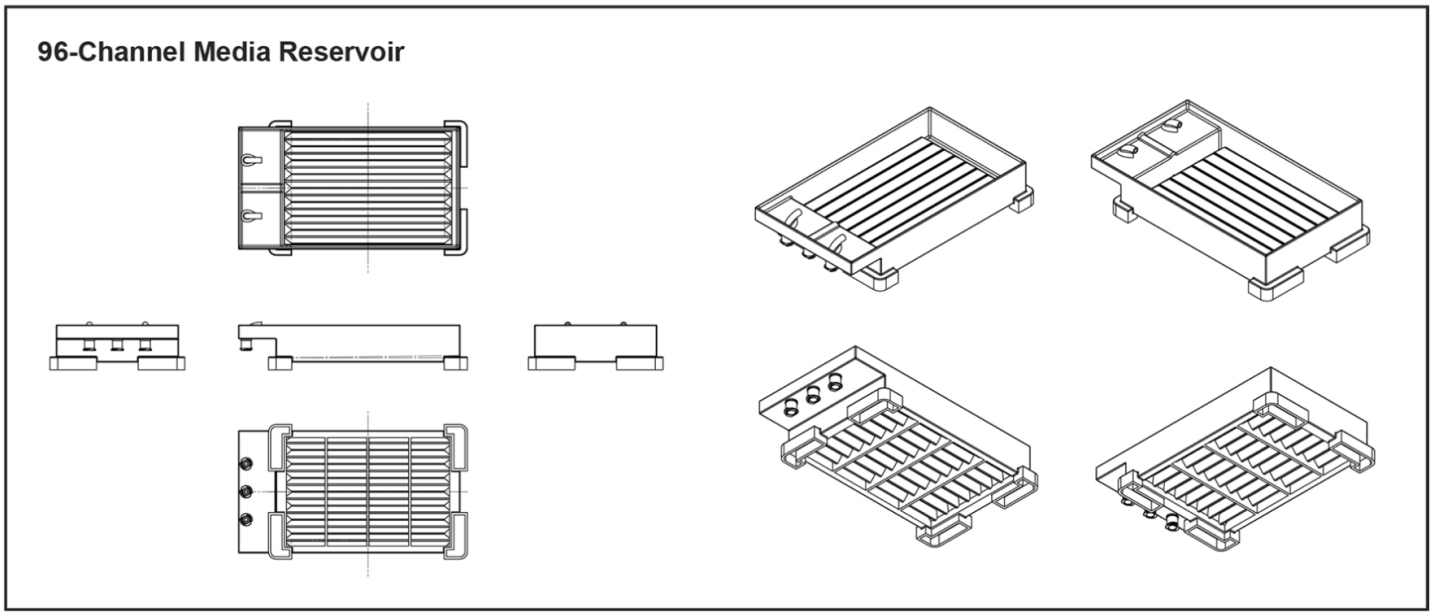


Extended Data Fig. 1 | PRANCE optimization. (**a**) Robotic manipulations operate in a loop, which repeats every 30 minutes. **(b)** Culture source fluidics (media, turbidostat/static culture, waste) are peripherally separated from the robot. The maximum flow-through rate is determined by the frequency with which the robot exchanges liquid (operations per hour), as well as the fraction of the standing volume of the population that is exchanged during each operation (the bottlenecking fraction). There is a trade-off between the maximum flow rate and the extent of bottlenecking. **(c)** The number of populations that can be serviced assuming 2 robot operations per hour (ops/hr) impacts the experimenter-free/hands-off operation time of the robot. **(d)** Larger robot decks can fit more tip carriers, more tips, and therefore require less frequent servicing.

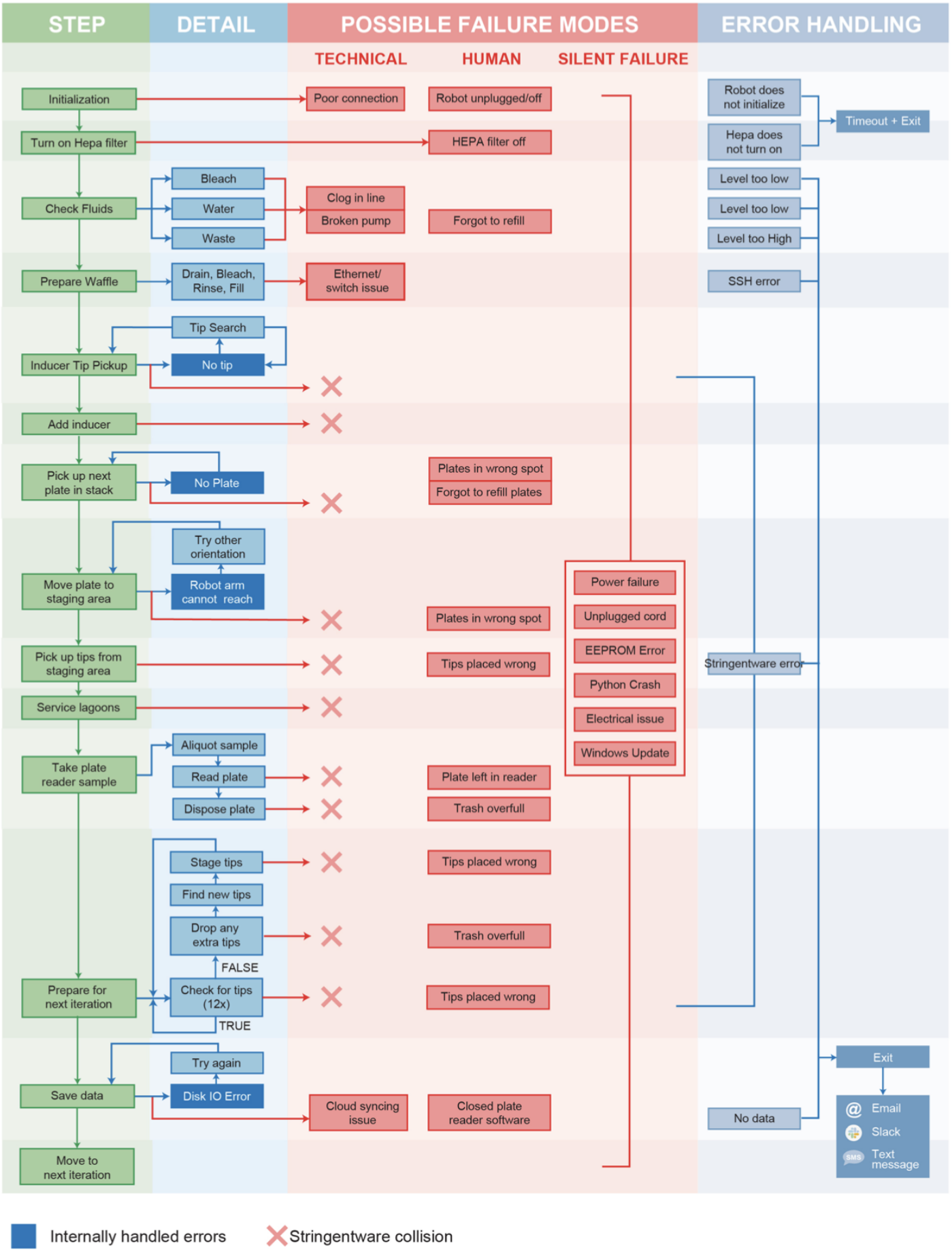


Extended Data Fig. 2 | Relationship between real-time monitoring data and phage titer. (**a)** Correlation between absorbance depression and luminescence for each evolving replicate. Kernel density estimates of the absorbance and luminescence data for the population are plotted on x and y axis, respectively. Luminescence data from Fig. 2b. (**b)** Comparison of real-time luminescence tracking (top) and corresponding phage titer as measured by traditional plaque assays (bottom). See Supplementary Table 1 for evolution construct details.

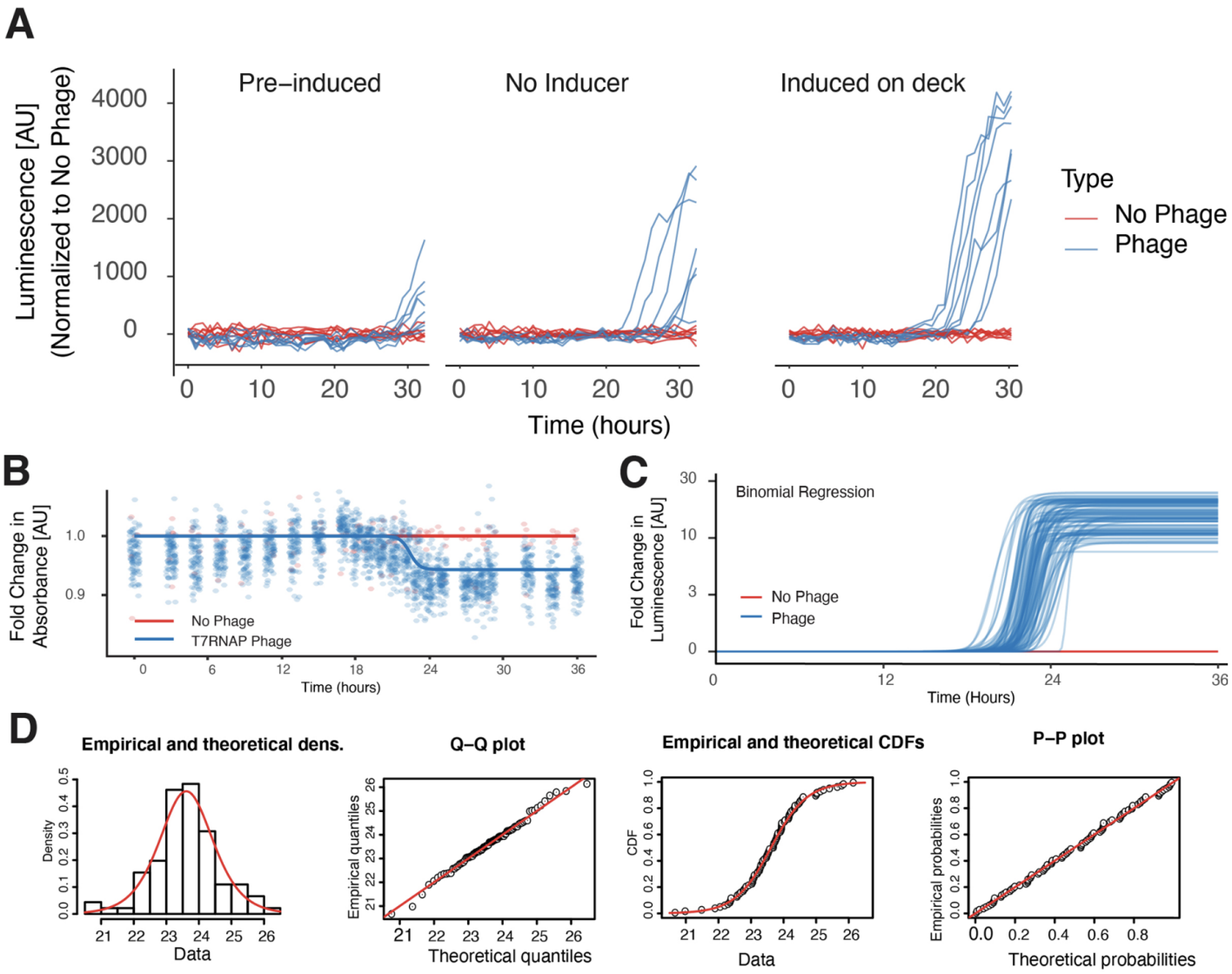




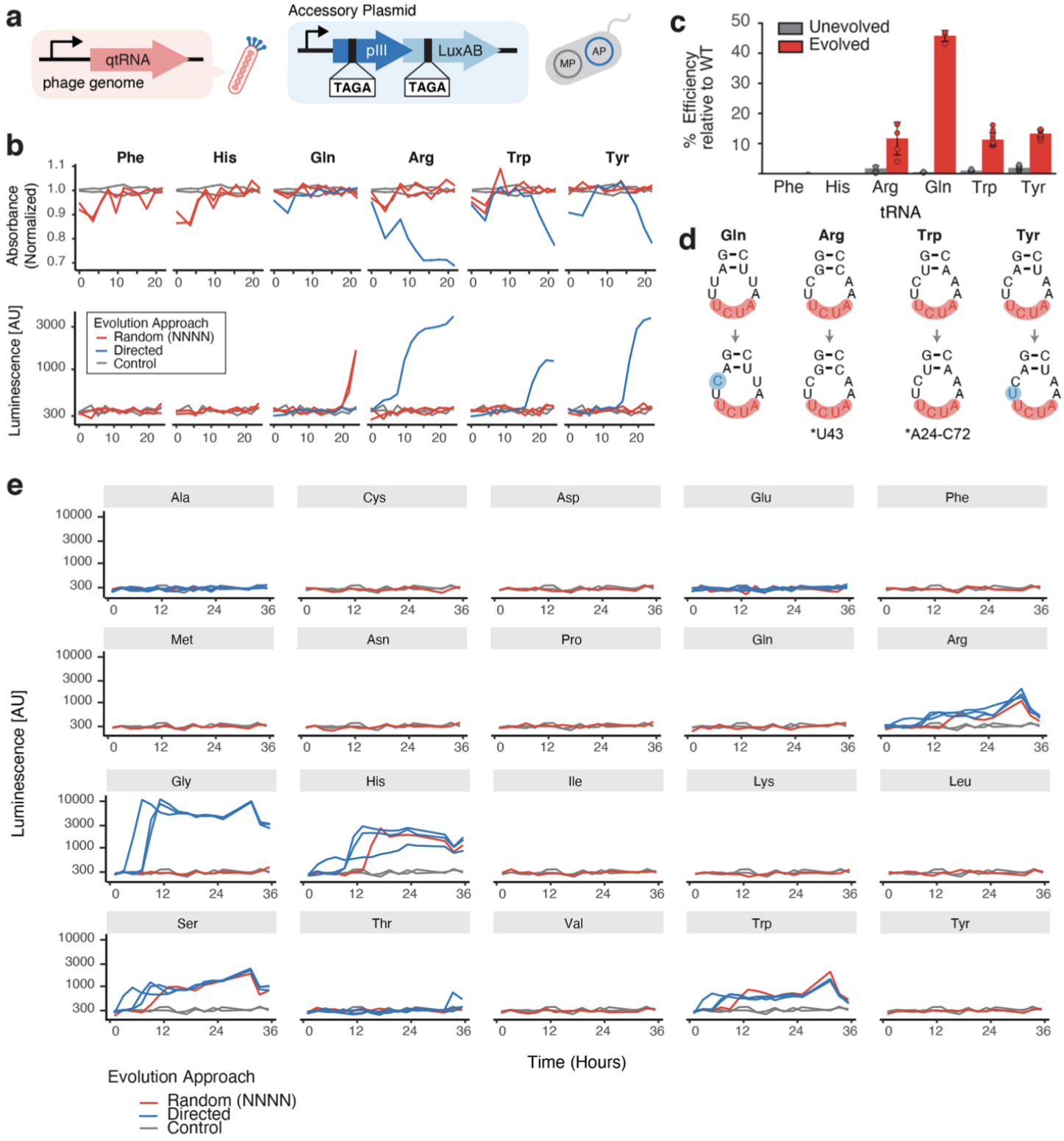
Extended Data Fig. 3 | Reservoir diagrams. Schematics of the 8-channel and 96-channel media reservoirs. These were printed on a Form 3 resin 3D printer. See the Supplementary Table 4 for .stl files for each.



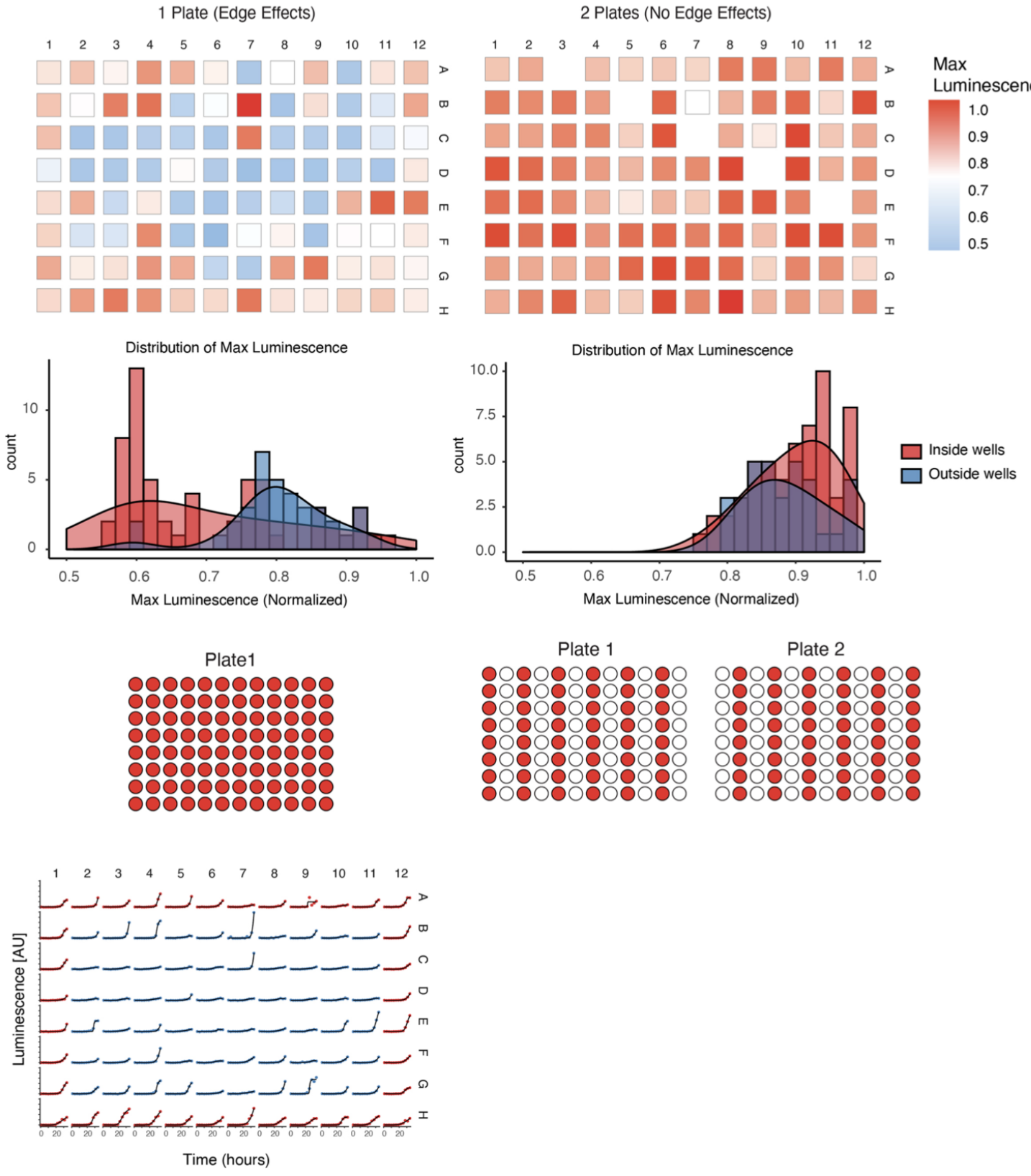
Extended Data Fig. 4 | Failure mode analysis. Analysis of failure modes used to improve reliability and error handling.



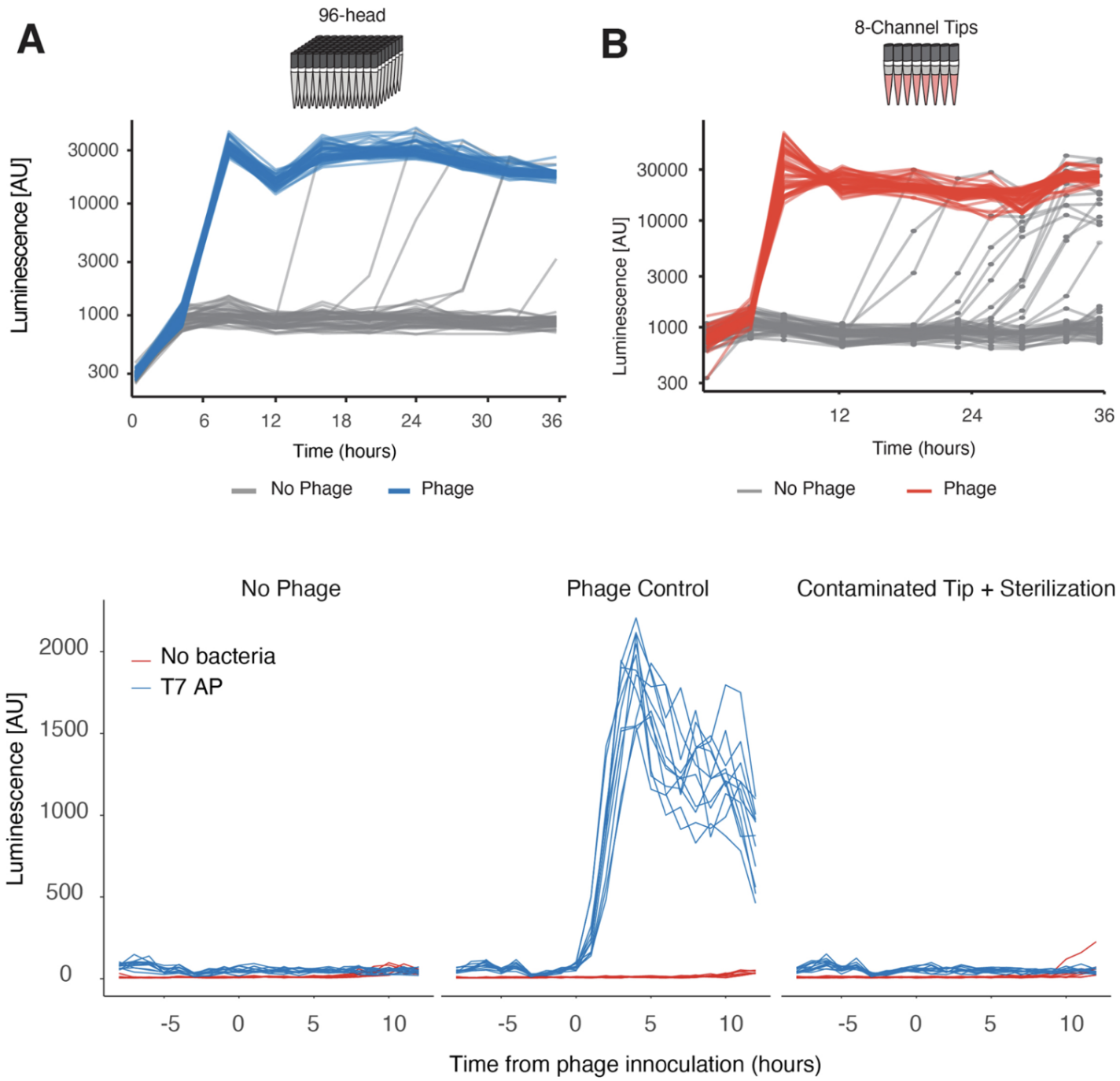
Extended Data Fig. 5 | Stochasticity of T7 RNAP evolution. Validating T7 mutagenesis with cool PRANCE. MP-containing bacteria were provided with either 1) induction prior to cooling to 4 C, 2) given no inducer, or 3) induced on the robotic deck at 37 C and their luminescence was tracked for 30 hours to validate that mutagenesis behaved similarly to induction of cultures directly from a turbidostat (see Fig. 1d). **(b)** Real-time absorbance depression monitoring of 90 simultaneous directed evolution experiments with 6 no-phage controls, fit with a binomial regression of the total data. **(c)** Logistic regression of each luminescence trace during T7 RNAP evolution to bind the T3 promoter, used to calculate the average evolution times (Supplemental Methods). **(d)** Goodness-of-fit estimates of a logistic distribution of the total T7 evolution time data.



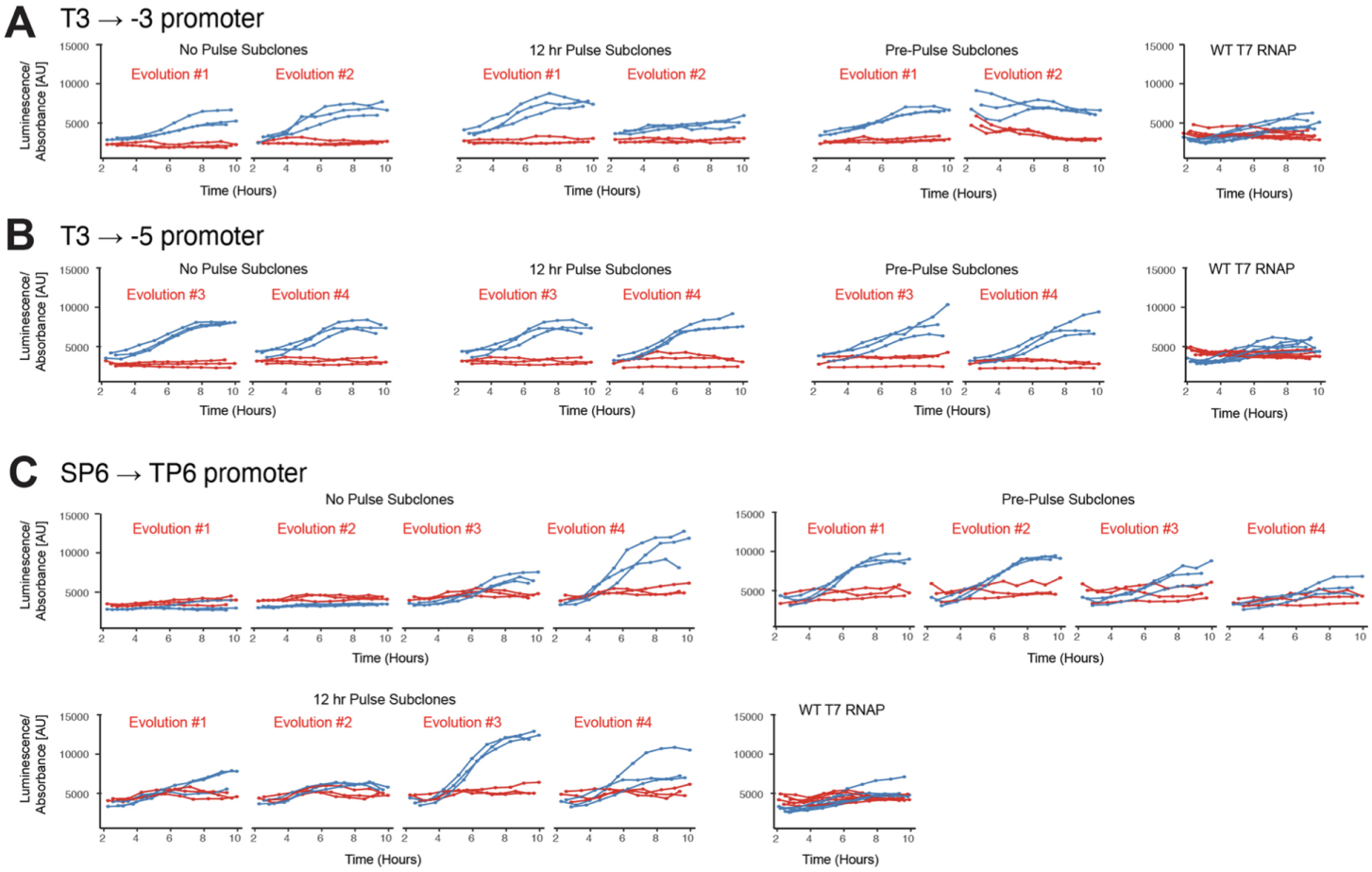
Extended Data Fig. 6 | TAGA-qtRNA and AGGG-qtRNA PRANCE. (a) Constructs for evolving TAGA-decoding qtRNAs. **(b)** Representative results for evolving TAGA-decoding qtRNAs. **(c)** Evolved qtRNAs exhibit increased ability to decode a TAGA quadruplet codon. Units are the percent luminescence when translating luxAB-357-TAGA in the presence of the qtRNA relative to expression of all-triplet-luxAB. (**d)** evolved genotypes **E)** Real-time absorbance and luminescence monitoring of qtRNA-encoding phage where either randomized or directed anticodons were used to evolve AGGG containing codons.



Extended Data Fig. 7 | Edge effects. Comparison between 96 replicates implemented in a densely packed 96-well plate (left) and 96 replicates split over two plates to reduce edge effects (right). Data plots the minimum time to phage detection via luminescence monitoring (below). Both plates above are normalized to their internal max value.



Extended Data Fig. 8 | Tip contamination, sterilization and reuse. To assess the maximum amount of possible cross-contamination, T7 RNAP-containing phage were inoculated into cultures containing pT7-psp-LuxAB bacteria in a grid-like pattern with 48 phage-containing wells and 48 no-phage-containing wells. PRANCE steps were performed with either **(a)** the 96-head channel or **(b)** the 8-channel pipettor. Use of the 96-well head gave less cross-contamination events, which we attributed to lower fly-over events. **(c)** To assess the impact of tip-sterilization and reuse in the optimized robotic method and configuration (Supplemental Video 1), robotic tips were submerged in either water or T7 RNAP-containing phage and then sterilized prior to being used to maintain high-throughput bacterial cultures containing pT7-psp-LuxAB[22]. Sterilized tips were also used to propagate bacteria inoculated with T7 RNAP phage as a positive control to ensure that bleach carryover did not affect phage propagation. No cross contamination was observed in the serialized tip condition over 12 hours, indicating that tips could be reused for a minimum of 12 hours without being replaced.



Extended Data Fig. 9 | Kinetic Luminescence Activity Assays of −3, −5, and TP6 Promoter Variants. To quantify the relative activities of each variant (independent of possible phage backbone mutations), phage from each replicate at time point t = 10 days were isolated and each T7 RNAP variant was cloned into an IPTG-inducible reporter construct. Subcloned variants were transformed into S2060 cells containing LuxAB driven by either the **(a)** −3 variant, **(b)** −5 variant, or **(c)** TP6 promoter and grown overnight to an OD > 1. Bacteria were then diluted 1:100 and grown to an OD of exactly 1.2 in DRM using a high-throughput robotic turbidostat method as described previously[22]. Once the bacteria reached an OD of 1.2 (approximately 2 hours), cells were induced with either 1 mM IPTG or [-] IPTG controls (n = 3 for each condition). Bacteria were autonomously maintained at an OD of 1.2 for the duration of the experiment and luminescence readings were taken once every 45 minutes for 10 hours. Fold change in luminescence (as shown in Fig. 6E), was calculated by averaging the luminescence in each turbidostat once the luminescence reached equilibrium (t > 8 hours) and then normalized to the average luminescence of the [-] IPTG controls within the same time window. WT T7 RNAP was used as a control for each respective promoter reporter construct (n = 6 for each WT control).

Appendix B

Graphical and Complementary Content of Emma J. Chory, *et. al.*

Enabling high-throughput biology with flexible open-source automation

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0001-Appendix.docx Appendix (Word document, 2.9 MB) (included below)

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0002-TableEV1.xlsx Table EV1 (application/excel, 7.3 KB)

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0003-TableEV2.xlsx Table EV2 (application/excel, 9.4 KB)

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0004-DatasetEV1.pdf Dataset EV1 (PDF document, 140.5 KB)

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0005-DatasetEV2.csv Dataset EV2 (application/csv, 5.7 KB)

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0006-MovieEV1.zip Movie EV1 (Zip archive, 79.3 MB)

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0007-MovieEV2.zip Movie EV2 (Zip archive, 29.3 MB)

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0008-MovieEV3.zip> Movie EV3 (Zip archive, 185.6 MB)

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0009-MovieEV4.zip> Movie EV4 (Zip archive, 147.2 MB)

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0010-MovieEV5.zip Movie EV5 (Zip archive, 44.1 MB)

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0011-MovieEV6.zip Movie EV6 (Zip archive, 69.3 MB)

- https://www.embopress.org/action/downloadSupplement?doi=10.15252%2Fmsb.20209942&file=msb20209942-sup-0012-MovieEV7.zip Movie EV7 (Zip archive, 70.4 MB)

Appendix

Enabling high-throughput biology with flexible open-source automation

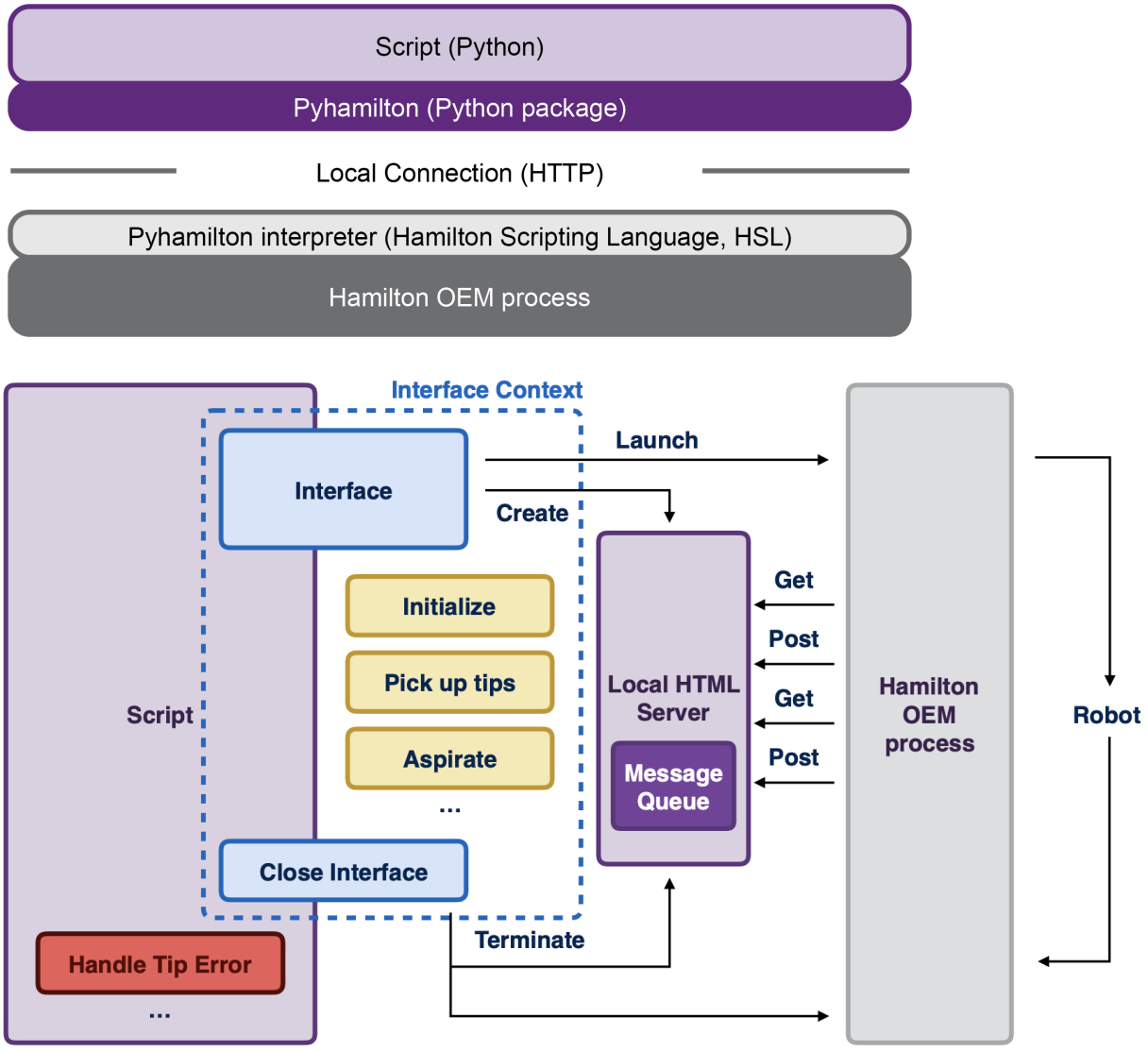
Emma J Chory[1,2,3] *, Dana W Gretton[1,*,ŧ] , Erika A DeBenedictis[1,4], Kevin M Esvelt[1,#]

[1]Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[2]Institute for Medical Engineering and Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[3]Broad Institute of MIT and Harvard, Cambridge, MA 02142, USA
[4]Department of Biological Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
* Designates equal contribution

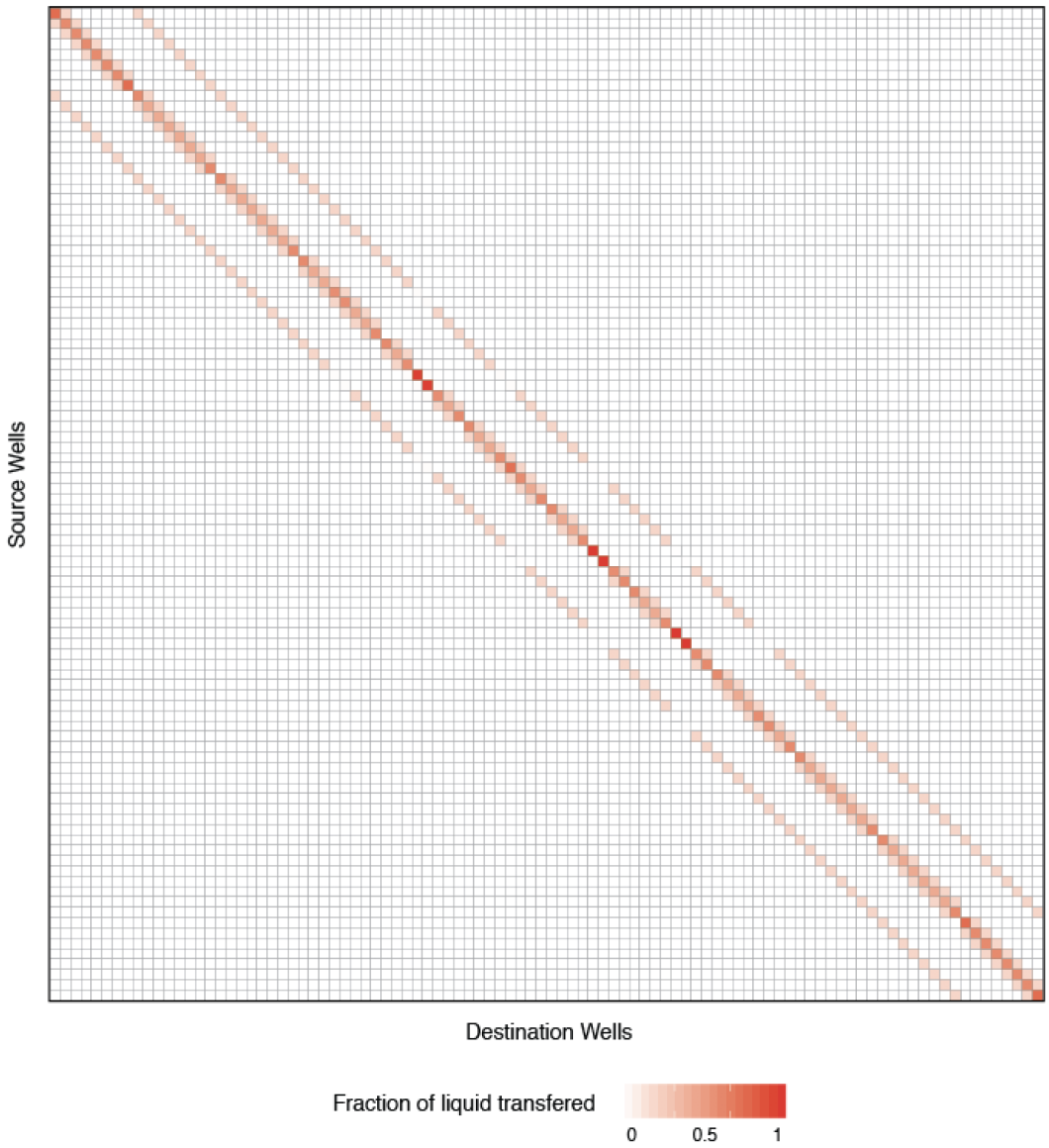
ŧ Correspondence for software development

For all other correspondence: esvelt@mit.edu

Appendix Figures

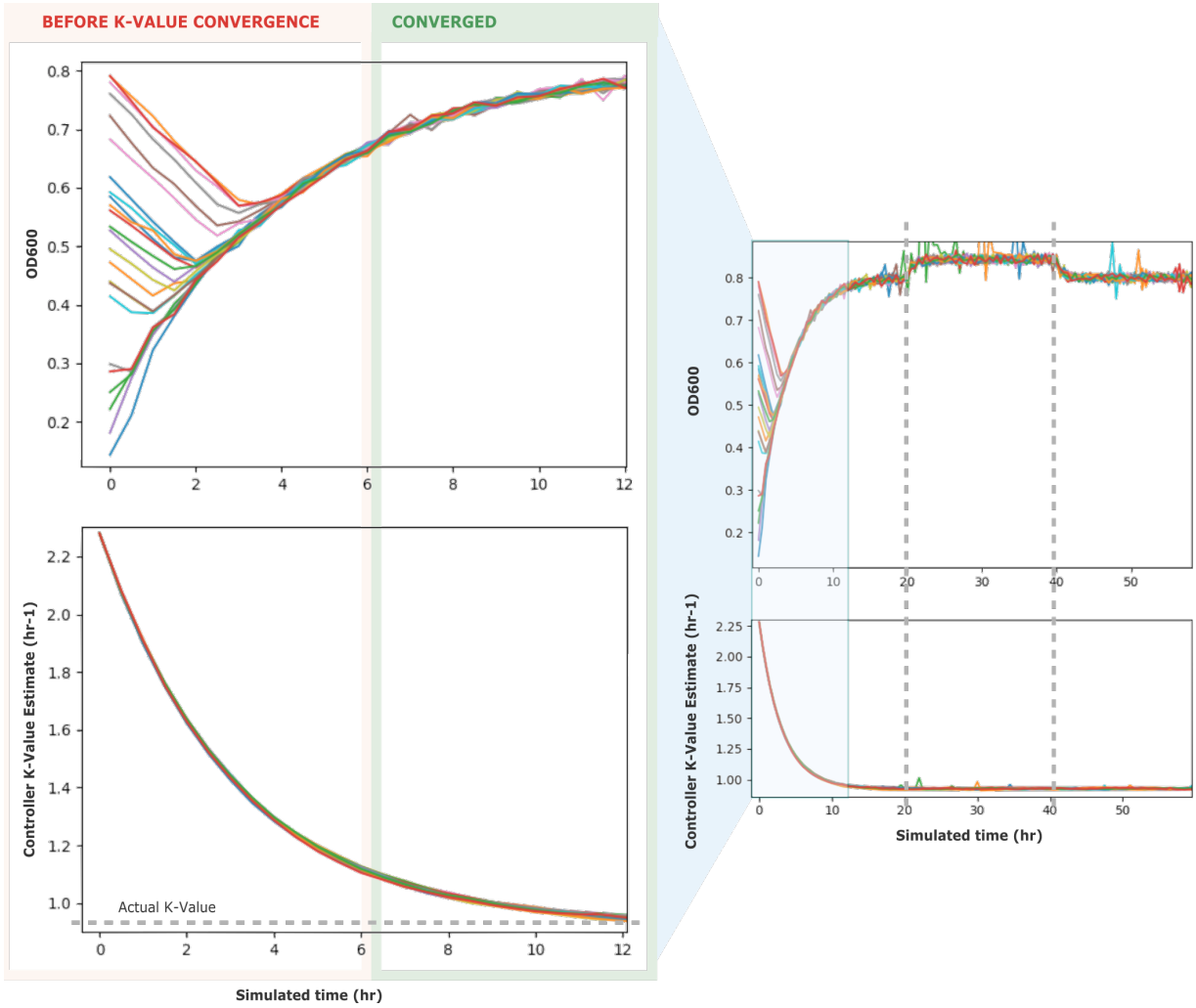


Appendix Figure S1: Layers of Abstraction in Pyhamilton
Pyhamilton it uses a platform-independent, web-based protocol (HTTP) and common readable data format (JSON) to bridge Python and the Hamilton Scripting Language (HSL).



Appendix Figure S2: Population Dynamics Diffusion Matrix.

Pyhamilton can be used in concert with all typical Python modules. The population dynamics application (Fig. 1D) makes use of matrix multiplication implemented by rectangular arrays from the scientific computing Python package NumPy. The diffusion matrix models arbitrary rates of flow from any of 96 wells in a microplate to all other wells. Gradient color scale represents fractional magnitudes of liquid transfers. Rows of the diffusion matrix sum to 1. Successive steps are computed by repeated multiplication of an initial 96-entry concentration vector by the 96x96 diffusion matrix. In this matrix, the dark main diagonal indicates that most of each population stays in its source well. Entries above and below the main diagonal by 1 and 8 rows represent transfers to vertical and horizontal neighbors respectively. All other entries are zero because population flow is not modeled to occur diagonally or beyond immediate neighbors. Obstacles to gene flow are captured by lighter colored areas in off-diagonal entries. Though this matrix is symmetric in that flows in both directions between each pair of wells are the same, matrix symmetry is not a requirement in general. The matrix construction facilitates offline analysis and visualization prior to robot execution. This construction would be difficult to implement in existing programming applications for Hamilton robots. Available online at: https://github.com/dgretton/pyhamilton_population_dynamics/blob/master/flow_matrix.csv.

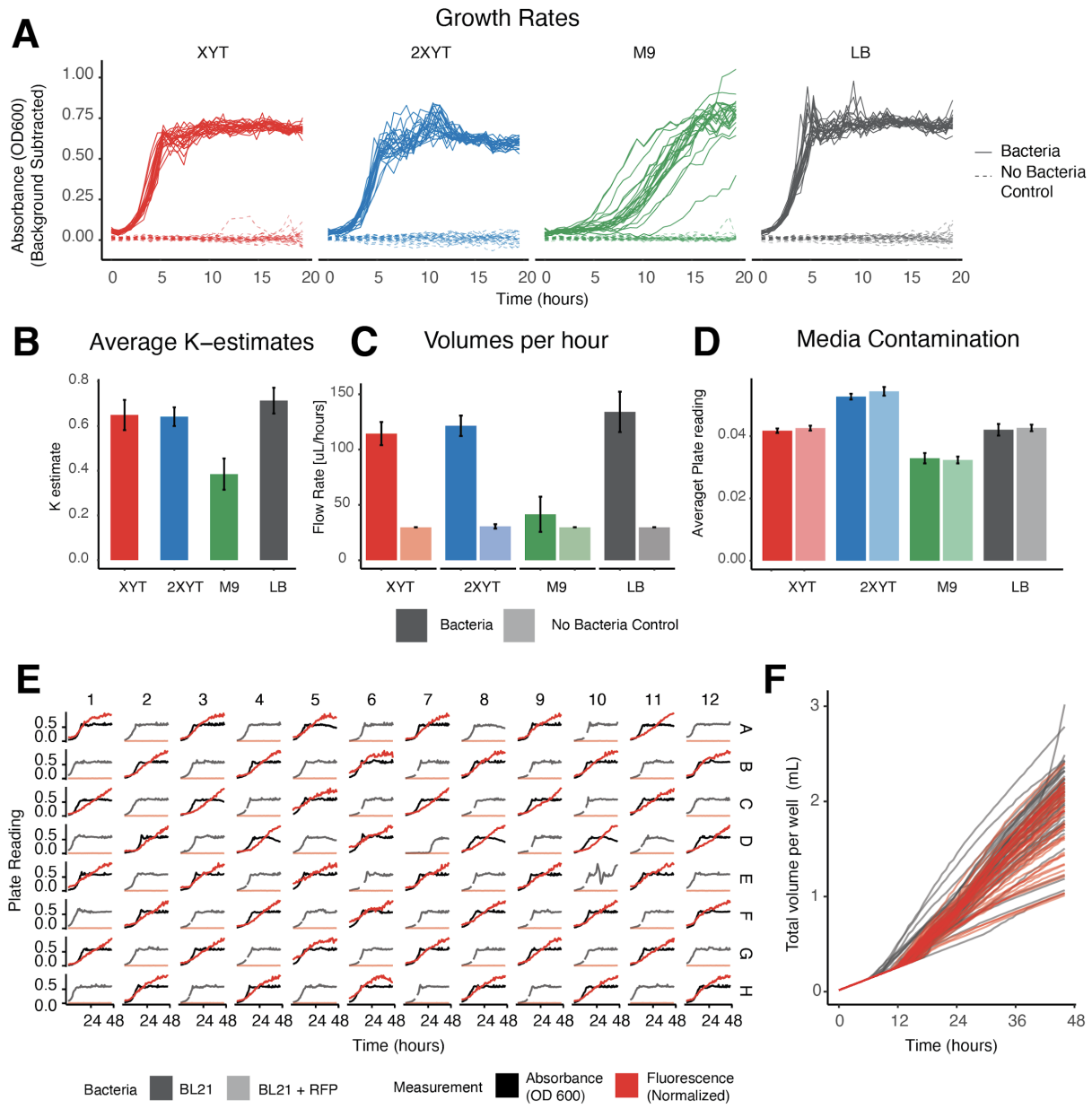


Appendix Figure S3: Turbidostat Controller Simulation

24 simulated turbidostats using the same controllers as experiments with initial k-value estimate of 2.3 hr^{-1} (20-minute doubling time typical of E. coli in LB Media) and actual k-value 0.93 hr^{-1} (typical of metabolite-poor media) converge in 12 hours. Start conditions vary between OD 0.1 and OD 0.8. Simulation includes uniform pipetting volume noise model and power law measurement noise model (spurious peaks). Controllers initially over-replace media in higher density cultures due to growth rate overestimate, causing OD to drop temporarily, before recovering as the k-value estimate converges more closely to the actual k-value. This behavior is exactly recapitulated in the turbidostat convergence study (Fig. 2D), indicating good correspondence between model and system. The simulated turbidostat OD setpoint was adjusted ±0.1 at 20 hours and 40 hours. Though the OD measurements and the controller's transfer volume commands both change at these times, the inferred k-value stays constant.

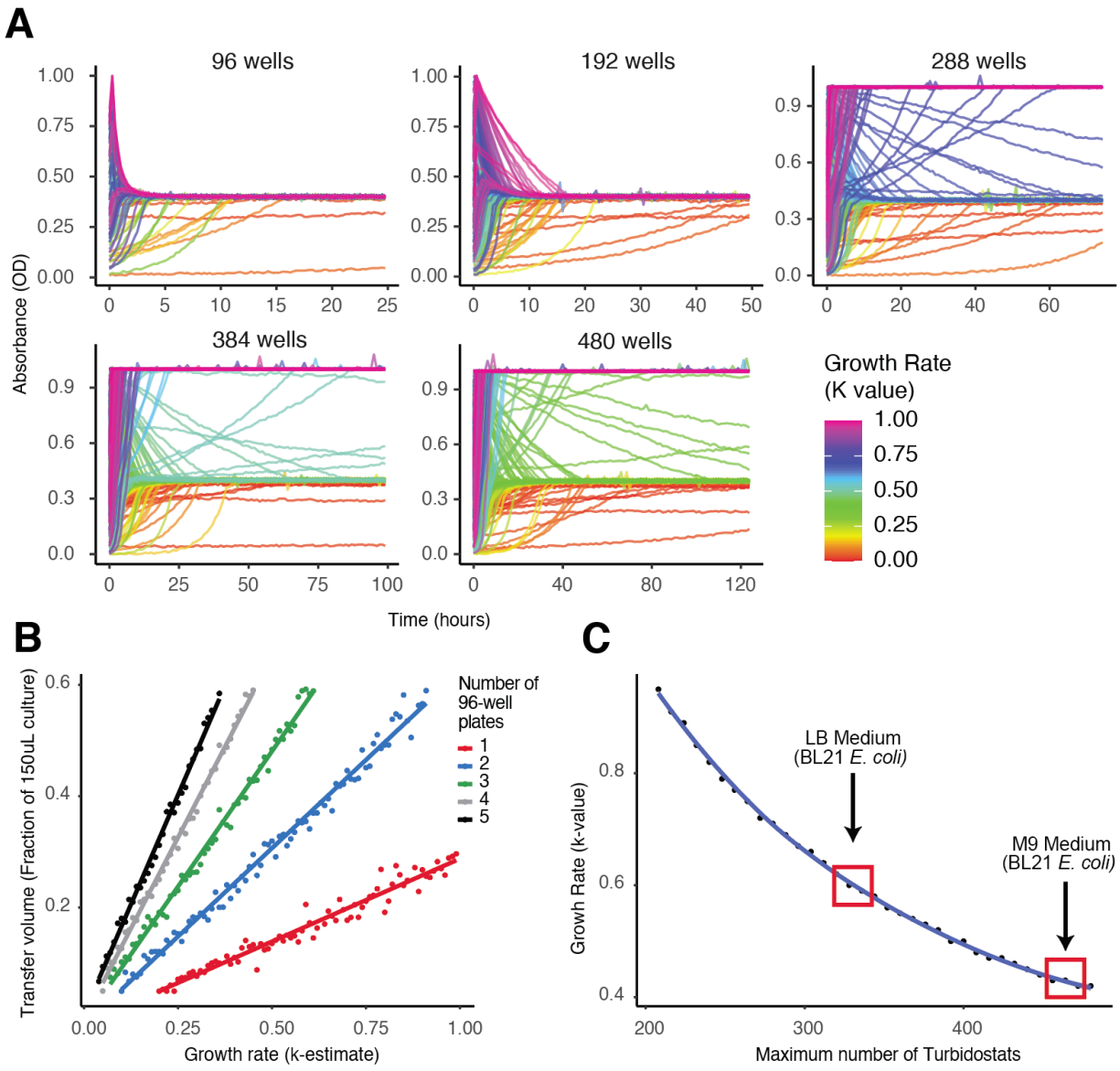
```
# define number of turbidostats (4 plates)
num_turbs = 384

def main():
    ## define required labware
    labware = plates, tip_boxes, media_sources
    ## define plate reader protocols
    reader_protocols = ['absorbance', 'mCherry', 'YFP', 'CFP']
    while True: ## Maintain turbidostats indefinitely
        ## Service each plate of turbidostats (in loop)
        for turbs_for_plate, controllers_for_plate, labware_for_plate in turbidostat_details:
            ## identify this plate's materials
            plate, tips, media_supply = labware_for_plate
            ## Simultaneously read plate, and perform pipetting steps
            platedatas = measure_plate(plate, reader_protocols,
simultaneously_execute=service_prev_plate)
            ## save plate reader data
            record_readings(plate, turbs_for_plate, platedatas)
            ## Calculate bacterial OD from optical density calibration curve
            od_readings = convert_to_ods(platedatas)
            ## calculate replacement volumes from transfer function
            remember.replace_vols = transfer_function(controllers_for_plate, od_readings)
            ## set up pipetting steps for next plate
            def service_prev_plate(args=(plate, tips, media_supply)):
                service(remember.replace_vols, *args)

## load Hamilton and plate reader
with HamiltonInterface() as ham_int, ClarioStar() as reader_int:
     ## Define instruments used in experiment
     sys_state.instruments = ham_int, reader_int
     ## initialize robot
     system_initialize()
     ## Run method
     main()
```

Appendix Figure S4: High-throughput turbidostats code



Appendix Figure S5: Turbidostat Controller Limitations in varying media.

(a) Real-time absorbance measurements of 192 cultures (2x 96 well plates) grown in either M9, XYT, 2XY, or LB media, with no bacteria controls **(b)** Average K-estimate of each bacteria growth condition demonstrates that fast and slow growing strains can both be supported. **(c)** Average volumes per hour of media consumed by each growth condition demonstrates that a 2mL deep well plate can support growth of a high growing strain for >16 hours without user intervention (100 uL/hour per hour in exponential growth phase), while a slow growing strain can be maintained for over 30 hours without user intervention (~ 50uL per hour in exponential growth phase). **(d)** Media sources show no detectable back contamination after 24 hours without replenished media. **(e)** Absorbance (black) and Normalized fluorescence (red) BL21 Cultures (50% opacity) and BL21 cultures expressing RFP (100% opacity) grown over 48 hours. **(f)** Total amount of media consumed over 48 hours for cultures shown in (e).



Appendix Figure S6: Simulations of Turbidostat Controller Limitations

(a) Simulations of growth equilibration of bacteria with varying exponential growth constants (k-values, hr^{-1}) when maintained at an OD setpoint of 0.4 in between 1-5 96-well plates. **(b)** Range of feasible transfer volumes as a function of growth rate when maintained in between 1-5 96-well plates. **(b)** Maximum number of turbidostats that can be maintained at a given growth rate. Average BL21 *E. coli* k-values when grown in either LB Medium or M9 medium are highlighted for reference.

Appendix C

Graphical and Complementary Content of Dana W. Gretton, *et. al.* Random adversarial threshold search enables specific, secure, and automated DNA synthesis screening

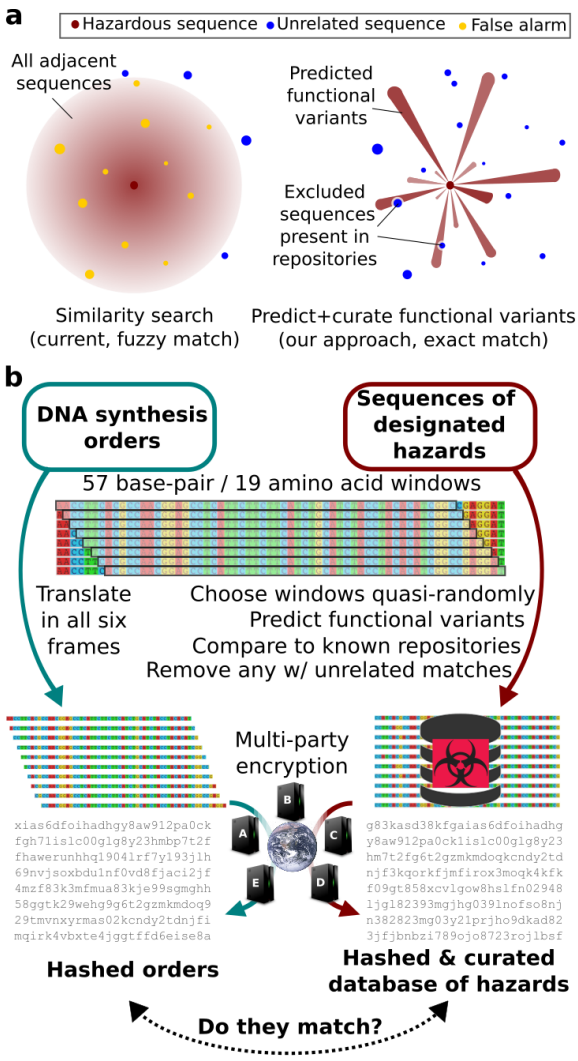


Fig. C-1. Improving DNA synthesis screening. a) Current screening methods yield false alarms from similar but unrelated sequences, requiring human curation (left). Predicting functional variants of k-mers, curating them to avoid false alarms, and searching orders for exact matches could automate screening (right). b) In random adversarial threshold (RAT) search, predicted functional variants of randomly chosen k-mer windows are computed, curated to remove unrelated sequences in repositories, and randomly added to a database. Since adversaries don't know which windows or how many variants are included, evasion requires them to include many mutations throughout the gene or genome and gamble that the result will still be functional. Window sizes of 19 amino acids are large enough to avoid random false alarms. Efficient exact match search allows cryptographic methods to protect the privacy of DNA synthesis orders and the entries in the hazard database[122].

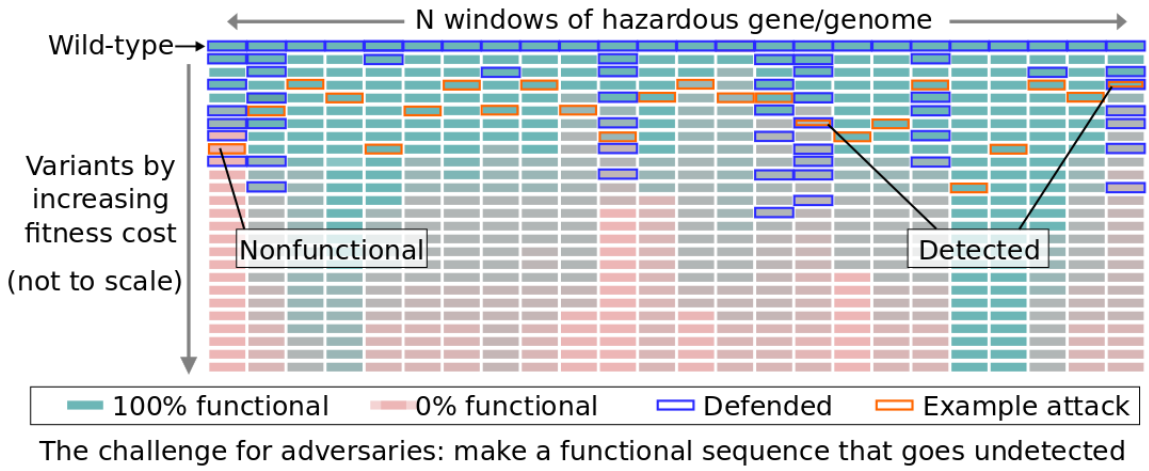


Fig. C-2. Predicted functional variants are included in the database to maximize the adversarial threshold: the probability that an adversary with perfect knowledge of fitness will be detected on choosing a variant for each window. The likelihood of detection rises with the defender's predictive capacity.

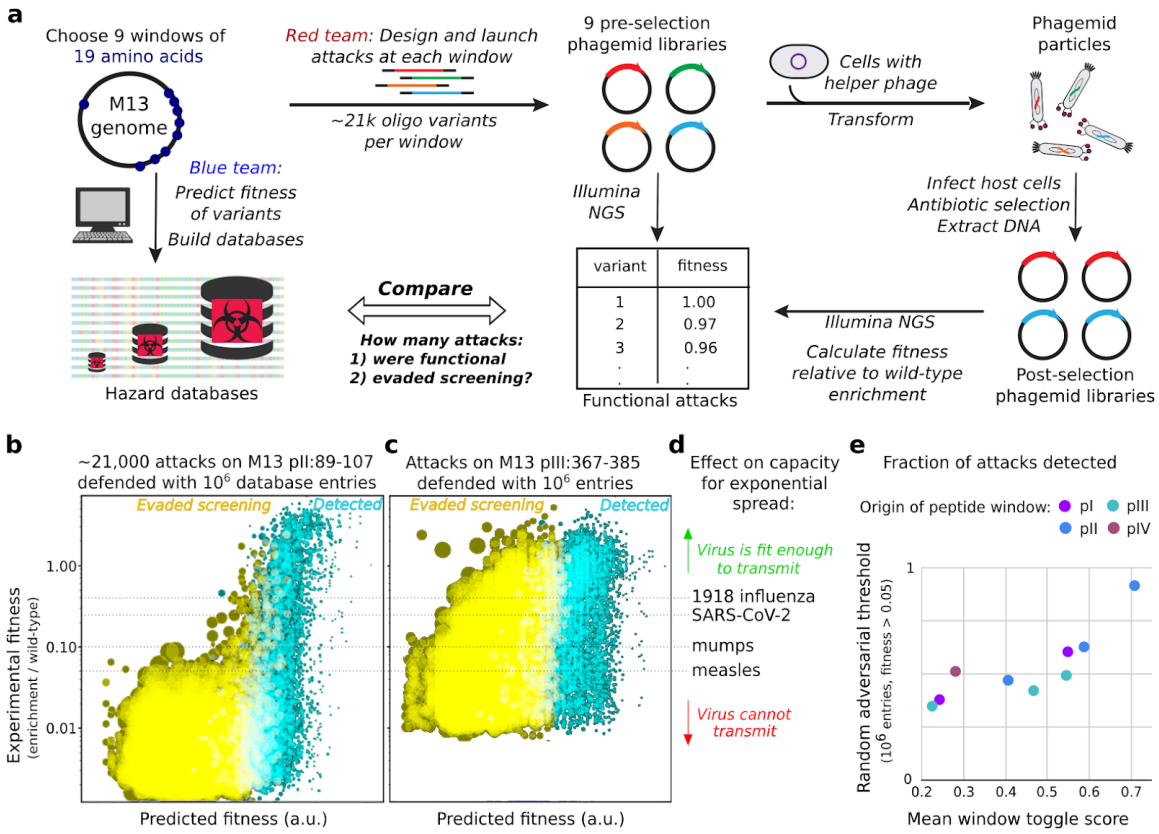
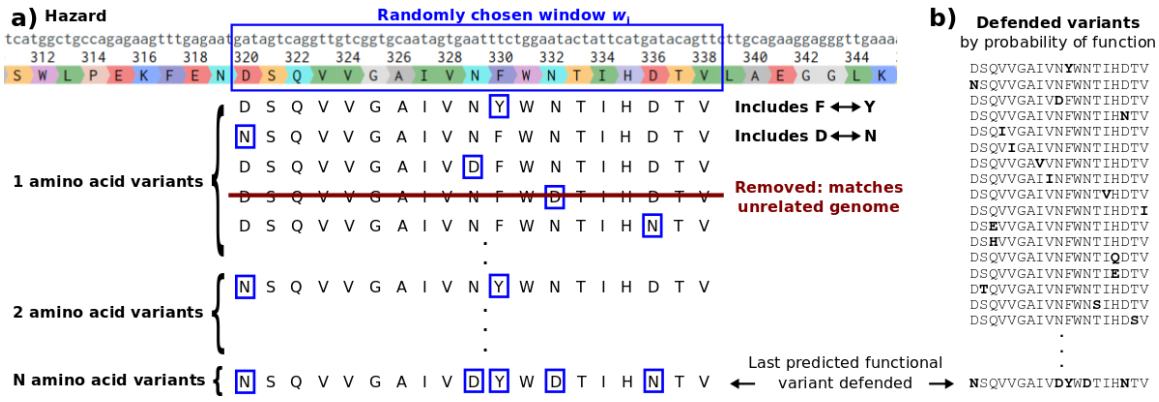
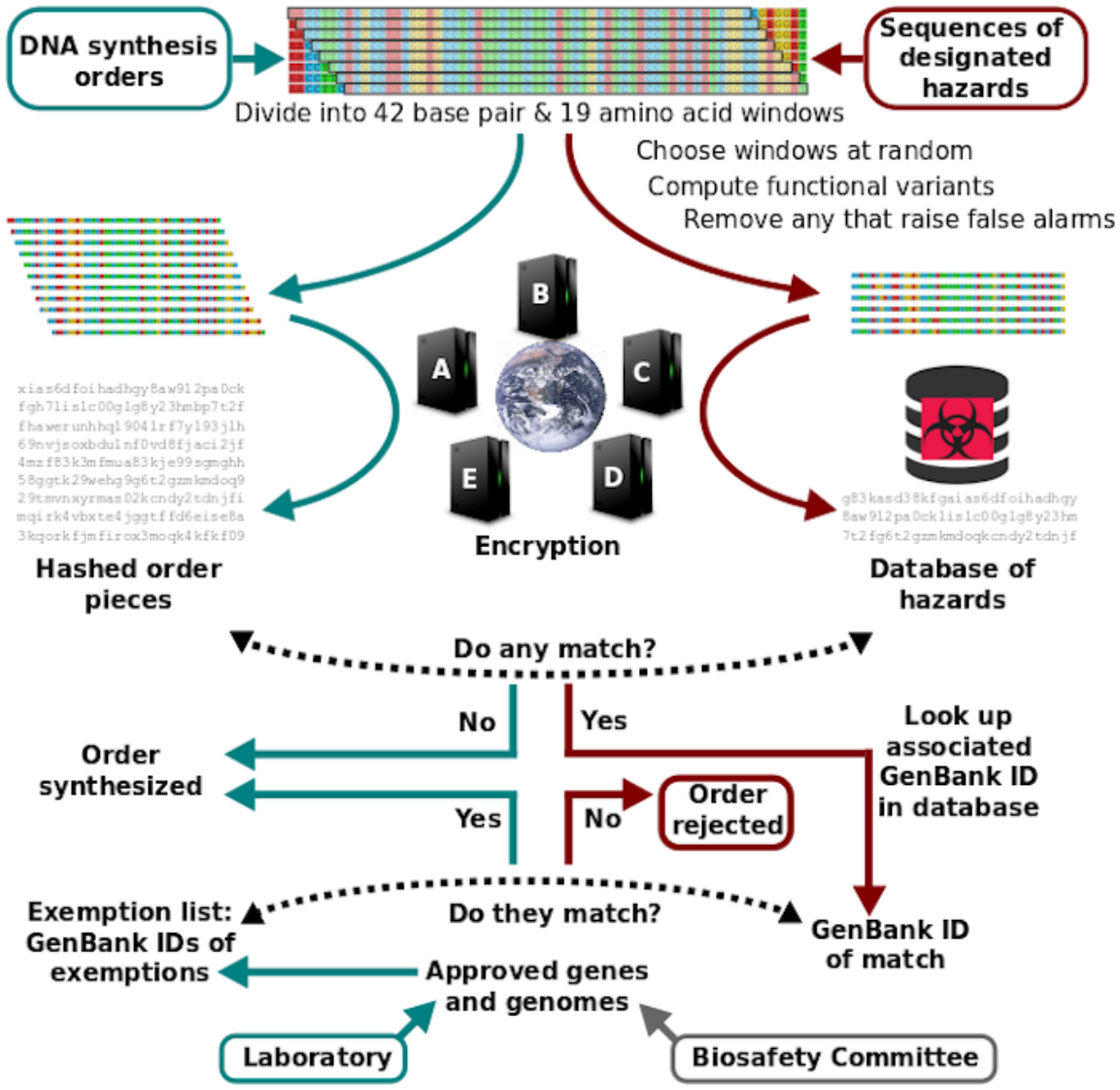


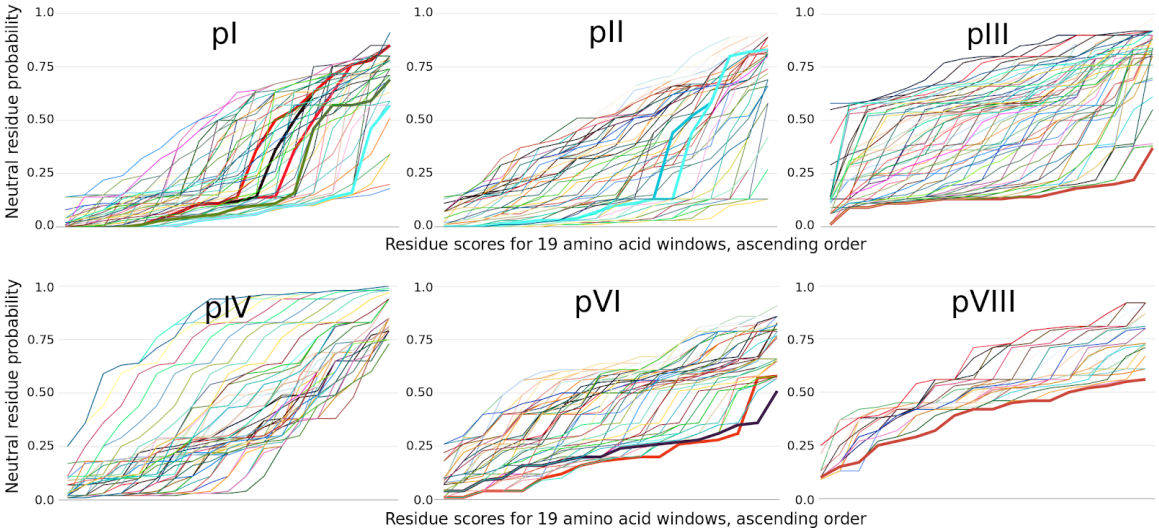
Fig. C-3. Incorporating mutations into the genomic blueprint of a virus cannot escape screening. a) Two team members built defensive databases by predicting functional variants for nine different windows in the genome of M13 bacteriophage. Two others launched 21,000 attacks at each window by synthesizing variants and using a phagemid assay to measure the fitness of each variant, which we defined as enrichment relative to the wild-type sequence. b) At the most defensible window, located within the M13 pII endonuclease, 92% of attacks yielding variants with fitness above 0.05 were thwarted by screening. c) At a moderately defensible window located within the M13 pIII receptor-binding protein, 49% of such attacks were thwarted, underscoring the importance of window choice. d) Potential pandemic pathogens can tolerate only so many mutations impairing fitness before they are no longer capable of sustained transmission. The corresponding fitness lines depict these fitness values for 1918 influenza (R0 2.5), SARS-CoV-2 (R0 4), mumps (R0 10), and measles (R0 18), which is the most infectious virus known. e) The fraction of attacks detected, which corresponds to the random adversarial threshold, as a function of the average funtrp toggle score for each of the nine windows given a database with 10^6 entries and a fitness cutoff of 0.05 (sufficient to prevent the sustained spread of measles).



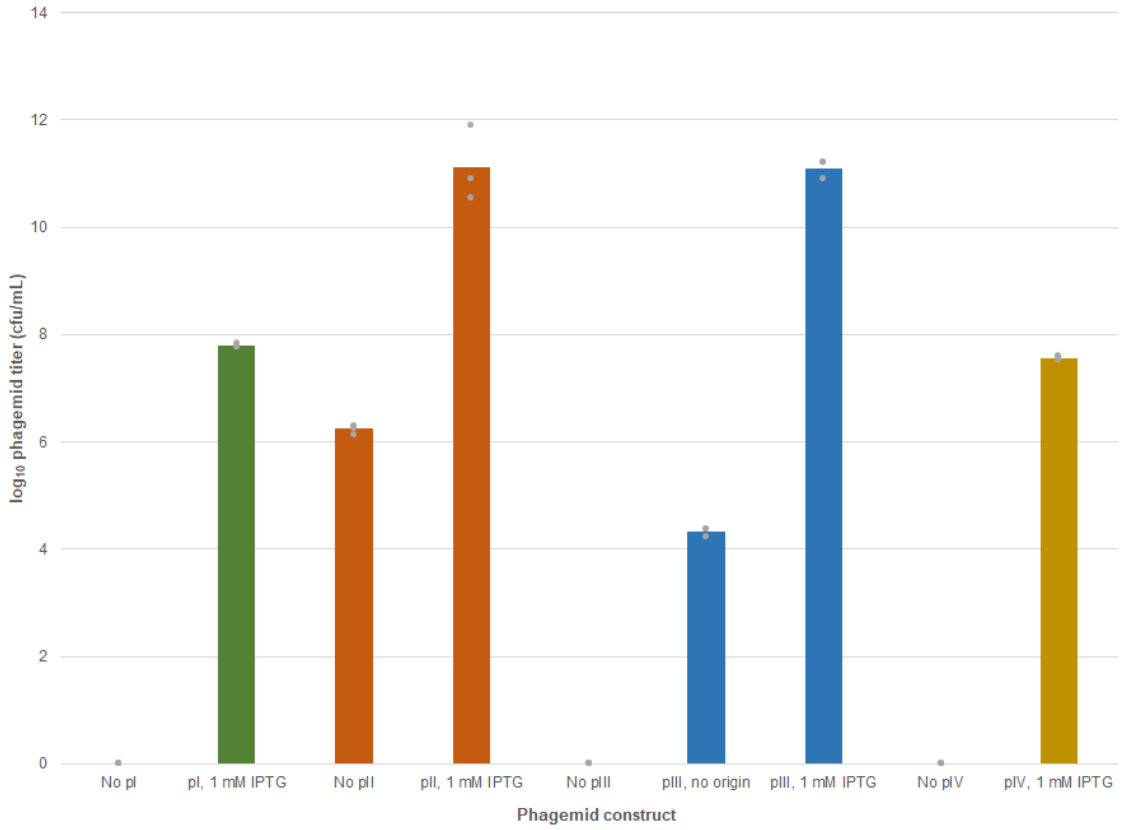
Supplementary Figure C-1. The hazard database includes wild-type sequences and predicted functional variants of randomly chosen windows comprising 19-amino acid peptides from hazardous proteins or 42-base pair DNA/RNA sequences from the noncoding regions of hazardous genomes. a) Examples of variant peptide sequences. Variants matching unrelated sequences in GenBank are removed. b) The random adversarial threshold increases as variants are added to the database. The exact method used to predict the function of variants in order to generate the list will be randomized across several prediction methods to prevent adversaries from predicting the contents of the hazard database.



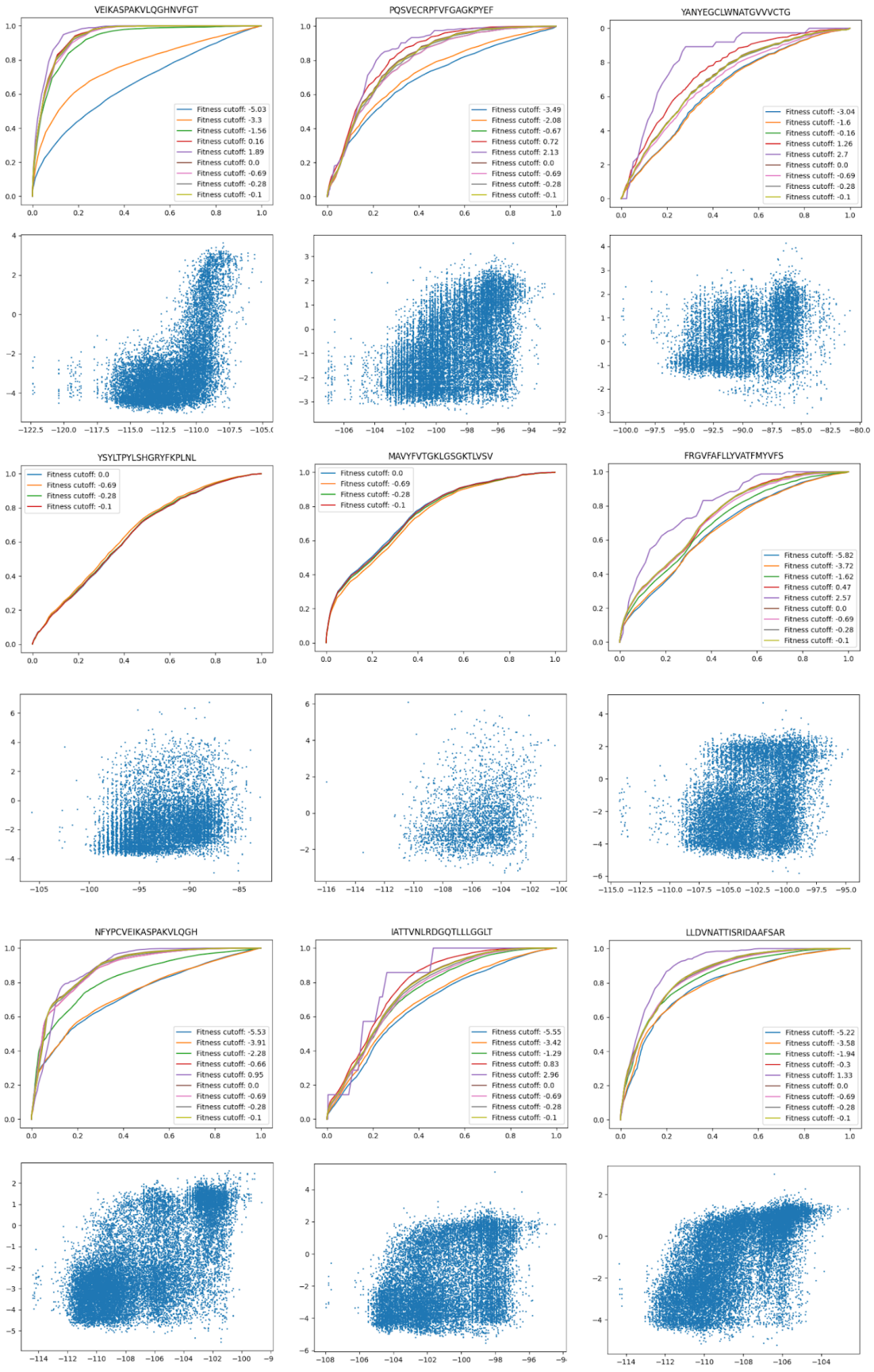
Supplementary Figure C-2. Exemption lists permit automated screening without impeding research. Each laboratory's approved biosafety committee registration is processed to identify gene and organism names, which are used to identify and construct an exemption list of corresponding GenBank accession numbers. Each sequence in the database is associated with one or more GenBank accession numbers to which it corresponds. When an order from a customer associated with an exemption list matches a database entry, the system checks to see whether any of the associated accession numbers match an entry on the exemption list. If so, that particular match is ignored.



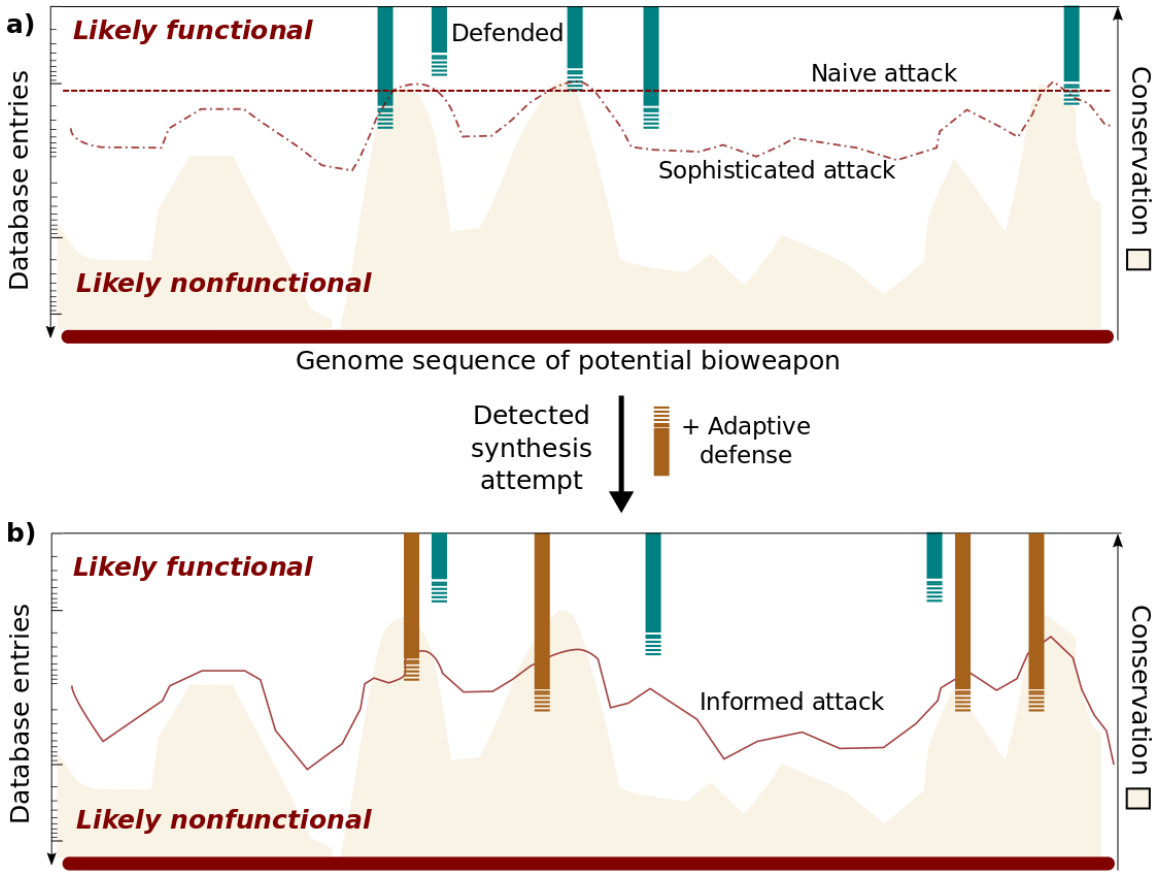
Supplementary Figure C-3. Graphical representations of funtrp window analyses for six M13 bacteriophage proteins. The probability of each residue being neutral for all 19 positions is plotted in ascending order. Windows represented by lines that follow the x-axis for most of their length should be less tolerant of mutations and therefore easier to defend according to funtrp. Larger proteins with more windows appear easier to defend, but there are noteworthy differences between the comparably sized proteins pI-pIV, with the lone enzyme pII predicted to harbor the most windows that are intolerant of mutation.



Supplementary Figure C-4. Evaluation of the fitness of phagemids encoding filamentous phage genes I-IV when generated from cells carrying helper plasmids deleted for the gene in question. In all cases, cells extruded phagemid particles when induced with 1 mM IPTG, as measured by infection of recipient cells with 3 independent biological replicates. Data from helper cells transformed with phagemids lacking the wild-type phage genes/origin of replication are provided for comparison. Phagemid titers below the limit of detection (100 cfu/mL) are plotted as zero. The difference in maximum titers suggests that the optimal level of each protein differs from the level produced upon induction, which may affect the relative fitness of variants. For example, overproduction may artificially increase the measured fitness of variants with reduced activity.



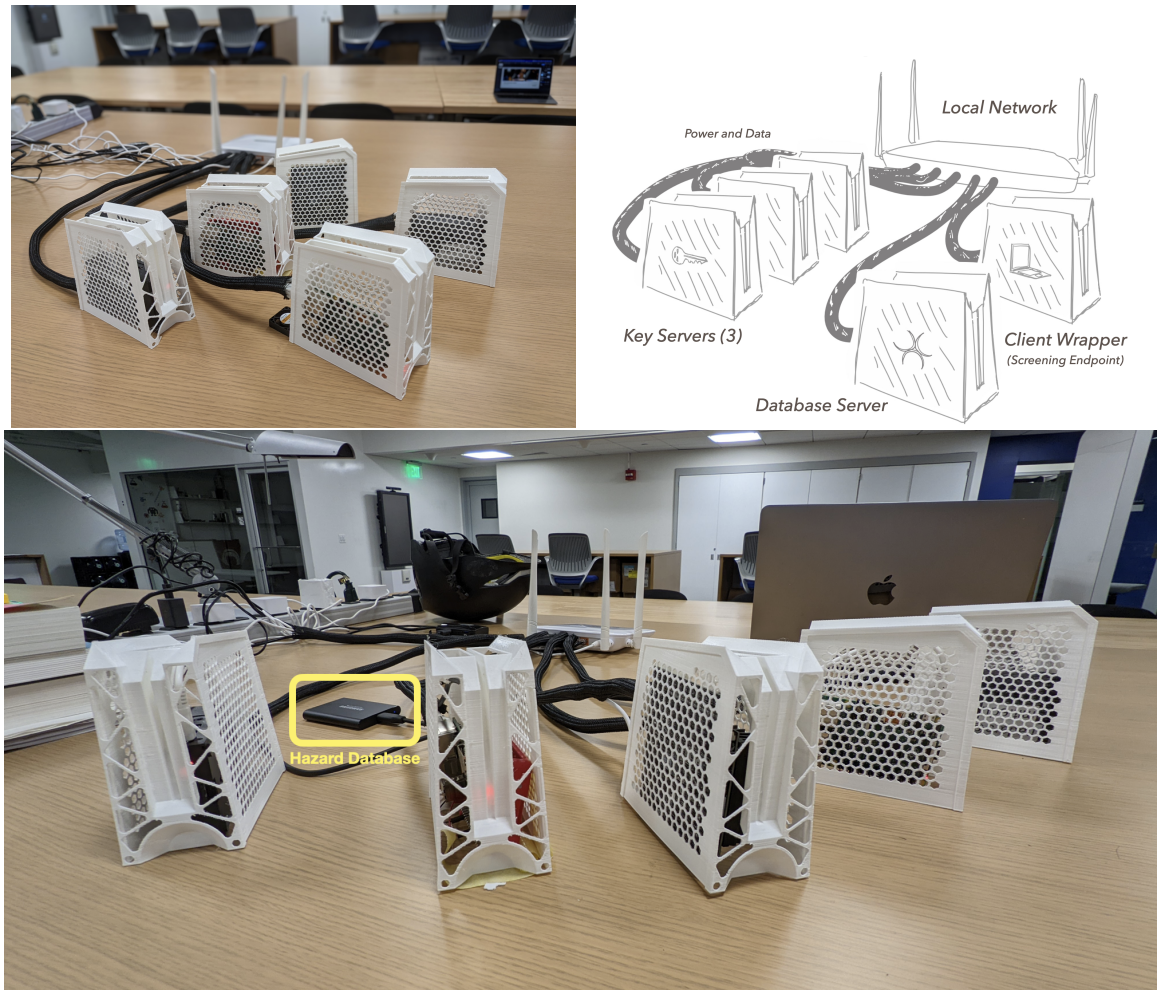
Supplementary Figure C-5. Left: Receiver-classifier-operator curves for funtrp+BLOSUM62 prediction for nine 19-amino acid windows across the genome of M13 bacteriophage. Curves measure distinct fitness cutoffs for prediction. Right: Plots of measured sequence function vs predicted function for each window.



Supplementary Figure C-6. Graphical representation of adaptive defense against a hazard. a) In this example, variants from five windows are included in the hazard database, mostly from relatively conserved regions of the gene. Most of the variants predicted to be highly functional by a variety of different algorithms at each window are included in the database, so a naive attacker who simply introduces a moderate number of mutations at a constant rate is both highly likely to be detected and risks generating a nonfunctional hazard due to the accumulated fitness cost. A sophisticated attacker may try to tune the number of mutations to the likelihood of obtaining a functional sequence across regions, thereby maximizing the chance of evading screening while preserving function. Their chances improve with the superiority of their variant prediction capabilities relative to the defender, but they still must trade off the risk of generating a nonfunctional hazard against being randomly detected upon picking a database variant at one of the five protected windows. b) If multiple attacks on a particular hazard are detected, the system can adaptively add more fragments and variants, precluding informed attacks based on probing or database interrogation. Windows may also rotate in and out periodically.

Appendix D

Additional Figures



Supplementary Figure D-1. SecureDNA Mini, a complete and functional tabletop demonstrator for SecureDNA. Queries may be issued to the client wrapper server via HTTP. The client wrapper first contacts the three key servers, transmitting only unreadable encrypted information about the input DNA sequences, and receiving equally indecipherable responses entangled with the secret key fragments held by the key servers. These responses are combined to construct the final query, which is then issued to the hazard database server. The answer as to whether the hazard appears in the database is returned to the client wrapper, at which time the client wrapper responds to the HTTP request with the final screening decision. Typical run times are a few seconds to a few minutes, depending on the size of the query sequence screened.

Appendix E

Pyhamilton Specification

Package **pyhamilton**

▸ EXPAND SOURCE CODE

Sub-modules

pyhamilton.deckresource

Couplings to Hamilton deck layouts …

pyhamilton.defaultcmds

Built-in commands, definitions of their parameters, and defaults.

pyhamilton.interface

Classes and utilities for automatic connection to a Hamilton robot.

pyhamilton.oemerr

`pyhamilton` -specific exception definitions.

pyhamilton.util

Index

Sub-modules

pyhamilton.deckresource
pyhamilton.defaultcmds
pyhamilton.interface
pyhamilton.oemerr
pyhamilton.util

One of the set of string literals recognized as command names by the `pyhamilton` interpreter, e.g. `'mph96Dispense'`. See `pyhamilton.defaultcmds` for examples.

`params_list` : `list`
exact list of string parameters that must have associated values for the command to be valid, other than those that are always present (`'command'` and `'id'`)

Static methods

`def unique_id()`

Return a "uniqe" hexadecimal string (`'0x...'`) based on time of call.

Methods

`def assemble_cmd(self, *args, **kwargs)`

Use keyword args to assemble this command. Default values auto-filled.

Args

`kwargs` : `dict`
map of any parameters (str) to values that should be different from the defaults supplied for this command in `pyhamilton.defaultcmds`

`def assert_valid_cmd(self, cmd_dict)`

Validate a finished command. Do nothing if it is valid.

`ValueError` will be raised if the supplied command did not have all required parameters for this command, as well as values for keys `'id'` and `'command'`, which are always required.

Args

`cmd_dict` : `dict`
A fully assembled `pyhamilton` command

Raises

`ValueError`
: The command dict is not ready to send. Specifics of mismatch summarized in exception description.

```
class HamiltonInterface (address=None, port=None, simulate=False)
```

Main class to automatically set up and tear down an interface to a Hamilton robot.

HamiltonInterface is the primary class offered by this module. It creates a Hamilton HSL background process running the `pyhamilton` interpreter, along with a `localhost` connection to act as a bridge. It is recommended to create a `HamiltonInterface` using a `with:` block to ensure proper startup and shutdown of its async components, even if exceptions are raised. It may be used with explicit `start()` and `stop()` calls.

Typical usage:

```
with HamiltonInterface() as ham_int: cmd_id =
ham_int.send_command(INITIALIZE) ...
ham_int.wait_on_response(cmd_id) ...
```

Class variables

```
var HamiltonServerThread
```

Private threaded local HTTP server with graceful shutdown flag.

```
var default_address
```

```
var default_port
```

```
var known_templates
```

Methods

```
def is_open(self)
```

Return `True` if the HamiltonInterface has been started and not stopped.

```
def log(self, msg, msg_type='info')
```

```
def log_and_raise(self, err)
```

```
def parse_hamilton_return(self, return_str)
```

Return a 2-tuple:

- [0] errflag: any error code present in response
- [1] Block map: dict mapping int keys to dicts with str keys (MainErr, SlaveErr, RecoveryBtnId, StepData, LabwareName, LabwarePos)

Result value 3 is the field that is returned by the OEM interface.

"Result value 3 contains one error flag (ErrFlag) and the block data package."

Data Block Format Rules

- The error flag is set once only at the beginning of result value 3. The error flag does not belong to the block data but may be used for a simpler error recovery. If this flag is set, an error code has been set in any of the block data entries.
- Each block data package starts with the opening square bracket character '['
- The information within the block data package is separated by the comma delimiter ','
- Block data information may be empty; anyway a comma delimiter is set.
- The result value may contain more than one block data package.
- Block data packages are returned independent of Num value (unsorted).

Block data information

- Num
  - Step depended information (e.g. the channel number, a loading position etc.).
  - Note: The meaning and data type for this information is described in the corresponding help of single step.
- MainErr

    - Main error code which occurred on instrument.
- SlaveErr
    - Detailed error code of depended slave (e.g. auto load, washer etc.).
- RecoveryBtnId
    - Recovery which has been used to handle this error.
- StepData
    - Step depended information, e.g. the barcode read, the volume aspirated etc.
    - Note: The meaning and data type for this information is described in the corresponding help of single step.
- LabwareName
    - Labware name of used labware.
- LabwarePos
    - Used labware position.

```
def pop_response(self, id, raise_first_exception=False)
```

Remove and return the response with the specified id from the response queue.

If there is a response, remove it and return the Hamilton-formatted response dict, like that returned from `HamiltonInterface.parse_hamilton_return()`. Otherwise, raise `KeyError`.

Args

`id` : `str`
Unique id of the command that initiated the response

`raise_first_exception` : `bool`
Optional; forwarded to `wait_on_response`. Default is `False`.

Returns

A 2-tuple:

1. parsed response block dict from Hamilton as in

parse_hamilton_return

2. Error map, a dict mapping int keys (data block Num field) that had exceptions, if any, to an exception that was coded in block; None to any error not associated with a block. {} if no error

Raises

KeyError
: if id has no matching response in the queue.

```
def send_command(self, template=None, block_until_sent=False, *args,
                 **cmd_dict)
```

Add a command templated after HamiltonCmdTemplate to the server send queue.

Args

template : HamiltonCmdTemplate
: Optional; a template to provide default arguments not specified in cmd_dict .

block_until_sent : bool
: Optional; if True , wait for all queued messages, including this one, to get picked up by the local server and sent across the HTTP connection, before returning. Default is False.

cmd_dict : dict
: keyword arguments to be forwarded to template when building the command, overriding its defaults. If template not given, cmd_dict must either have a 'command' key with value matching one of the command names in defaultcmds and might be missing an 'id' key, or itself be a fully formed and correct pyhamilton command with its own 'id' key.

Returns

unique id (str) of the command that can be used to index it later, either newly generated or same as originally present in cmd_dict.

```
def set_log_dir(self, log_dir)
```

```
def start(self)
```

Starts the extra processes, threads, and servers for the Hamilton connection.

Launches: 1) the pyhamilton interpreter using the Hamilton Run Control executable, either in the background for normal use, or in the foreground with a GUI for simulation; 2) a local HTTP server to ferry messages between the python module and the interpreter.

When used with a `with:` block, called automatically upon entering the block.

```
def stop(self)
```

Stop this HamiltonInterface and clean up associated async processes.

Kills the pyhamilton interpreter subprocess and executable and stops the local web server thread.

When used with a `with` block, called automatically on exiting the block.

```
def wait_on_response(self, id, timeout=0, raise_first_exception=False)
```

Wait and do not return until the response for the specified id comes back.

When the command corresponding to `id` regards multiple distinct pipette channels or devices, responses may contain encoded errors that might be different for different channels or devices. For this reason, the default behavior of `wait_on_response` is to not raise exceptions, but to delegate handling exceptions to the caller. For convenience, this method can optionally raise the first exception it encounters, often a useful behavior for succinct scripted commands that regard only one device, when raise_first_exception is `True` .

Args

`id` : `str`
The unique id of a previously sent command

`timeout` : `float`
Optional; maximum time in seconds to wait before raising `HamiltonTimeoutError` . Default is no timeout (forever).

`raise_first_exception`
Optional; if True, may raise if there is an error encoded in the response. Default is False.

[1] Guesstimating the size of the global array synthesis market. https://synbiobeta.com/guesstimating-size-global-array-synthesis-market/, September 2017. Accessed: 2021-3-28.

[2] Aclid, Inc. https://aclid.bio/ accessed August 2022.

[3] Zhaleh N Amini and Ulrich F Müller. Low selection pressure aids the evolution of cooperative ribozyme mutations in cells. *Journal of Biological Chemistry*, 288(46):33096–33106, 2013.

[4] Daniel R Amor, Christoph Ratzke, and Jeff Gore. Transient invaders can induce shifts between alternative stable states of microbial communities. *Sci Adv*, 6(8):eaay8676, February 2020.

[5] J Christopher Anderson, Thomas J Magliery, and Peter G Schultz. Exploring the limits of codon and anticodon size. *Chemistry & biology*, 9(2):237–244, 2002.

[6] Evan Appleton, Douglas Densmore, Curtis Madsen, and Nicholas Roehner. Needs and opportunities in bio-design automation: four areas for focus. *Curr. Opin. Chem. Biol.*, 40:111–118, October 2017.

[7] 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(1), 2006.

[8] Ahmed H Badran and David R Liu. Development of potent in vivo mutagenesis plasmids with broad mutational spectra. *Nature communications*, 6(1):1–10, 2015.

[9] Bryce T Bajar, Amy J Lam, Ryan K Badiee, Young-Hee Oh, Jun Chu, Xin X Zhou, Namdoo Kim, Benjamin B Kim, Mingyu Chung, Arielle L Yablonovitch, Barney F Cruz, Kanokwan Kulalert, Jacqueline J Tao, Tobias Meyer, Xiao-Dong Su, and Michael Z Lin. Fluorescent indicators for simultaneous reporting of all four cell cycle phases. *Nat. Methods*, 13(12):993–996, December 2016.

[10] Monya Baker. Reproducibility crisis. *Nature*, 533(26):353–66, 2016.

[11] Frederick K Balagaddé, Lingchong You, Carl L Hansen, Frances H Arnold, and Stephen R Quake. Long-term monitoring of bacteria undergoing programmed population control in a microchemostat. *Science*, 309(5731):137–140, 2005.

[12] Advait Balaji, Bryce Kille, Anthony D Kappell, Gene D Godbold, Madeline Diep, RA Elworth, Zhiqin Qian, Dreycey Albin, Daniel J Nasko, Nidhi Shah, et al. Seqscreen: accurate and sensitive functional screening of pathogenic sequences via ensemble learning. *Genome biology*, 23(1):1–29, 2022.

[13] Jeffrey E Barrick, Dong Su Yu, Sung Ho Yoon, Haeyoung Jeong, Tae Kwang Oh, Dominique Schneider, Richard E Lenski, and Jihyun F Kim. Genome evolution and adaptation in a long-term experiment with escherichia coli. *Nature*, 461(7268):1243–1247, 2009.

[14] Carsten Baum, Hongrui Cui, Ivan Damgård, Kevin Esvelt, Mingyu Gao, Dana Gretton, Omer Paneth, Ron Rivest, Vinod Vaikuntanathan, Daniel Wichs, et al. Cryptographic aspects of dna screening. 2020.

[15] Michael Baym, Tami D Lieberman, Eric D Kelsic, Remy Chait, Rotem Gross, Idan

Yelin, and Roy Kishony. Spatiotemporal microbial evolution on antibiotic landscapes. *Science*, 353(6304):1147–1151, 2016.

[16] Jacob Beal, Adam Clore, and Jeff Manthey. Studying pathogens degrades blast-based pathogen identification. *bioRxiv*, 2022.

[17] Chet M Berman, Louis J Papa III, Samuel J Hendel, Christopher L Moore, Patreece H Suen, Alexander F Weickhardt, Ngoc-Duc Doan, Caiden M Kumar, Taco G Uil, Vincent L Butty, et al. An adaptable platform for directed evolution in human cells. *Journal of the American Chemical Society*, 140(51):18093–18103, 2018.

[18] Zachary D Blount, Christina Z Borland, and Richard E Lenski. Historical contingency and the evolution of a key innovation in an experimental population of escherichia coli. *Proceedings of the National Academy of Sciences*, 105(23):7899–7906, 2008.

[19] Zachary D Blount, Richard E Lenski, and Jonathan B Losos. Contingency and determinism in evolution: Replaying life's tape. *Science*, 362(6415):eaam5979, 2018.

[20] Nick Bostrom. Information hazards: A typology of potential harms from knowledge. *Review of Contemporary Philosophy*, 10(2011):44–79, March 2012.

[21] Yana Bromberg and Burkhard Rost. SNAP: predict effect of non-synonymous polymorphisms on function. *Nucleic Acids Res.*, 35(11):3823–3835, May 2007.

[22] David I Bryson, Chenguang Fan, Li-Tao Guo, Corwin Miller, Dieter Söll, and David R Liu. Continuous directed evolution of aminoacyl-trna synthetases. *Nature chemical biology*, 13(12):1253–1260, 2017.

[23] Ewen Callaway et al. Legendary bacterial evolution experiment enters new era. *Nature*, 606(7915):634–635, 2022.

[24] L Chasteen, J Ayriss, P Pavlik, and A R M Bradbury. Eliminating helper phage from phage display. *Nucleic Acids Res.*, 34(21):e145, November 2006.

[25] Jason W Chin, Stephen W Santoro, Andrew B Martin, David S King, Lei Wang, and Peter G Schultz. Addition of p-azido-l-phenylalanine to the genetic code of escherichia c oli. *Journal of the American Chemical Society*, 124(31):9026–9027, 2002.

[26] Inha Cho, Zhi-Jun Jia, and Frances H Arnold. Site-selective enzymatic c–h amidation for synthesis of diverse lactams. *Science*, 364(6440):575–578, 2019.

[27] Yongwook Choi, Gregory E Sims, Sean Murphy, Jason R Miller, and Agnes P Chan. Predicting the functional effect of amino acid substitutions and indels. *PLoS One*, 7(10):e46688, October 2012.

[28] Emma J Chory, Dana W Gretton, Erika A DeBenedictis, and Kevin M Esvelt. Enabling high-throughput biology with flexible open-source automation. *Molecular systems biology*, 17(3):e9942, 2021.

[29] Peter J A Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J L de Hoon. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, June 2009.

[30] National Research Council et al. Sequence-based classification of select agents: a brighter line. 2010.

[31] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *Theory of Cryptography Conference*, pages 342–362. Springer, 2005.

[32] Andreas Crameri, Sun-Ai Raillard, Ericka Bermudez, and Willem PC Stemmer. Dna shuffling of a family of genes from diverse species accelerates directed evolution. *Nature*, 391(6664):288–291, 1998.

[33] Nathan Crook, Joseph Abatemarco, Jie Sun, James M Wagner, Alexander Schmitz, and Hal S Alper. In vivo continuous evolution of genes and pathways in yeast. *Nature communications*, 7(1):1–14, 2016.

[34] E A DeBenedictis, E J Chory, D Gretton, B Wang, and others. A high-throughput platform for feedback-controlled directed evolution. *bioRxiv*, 2020.

[35] Erika A DeBenedictis, Gavriela D Carver, Christina Z Chung, Dieter Söll, and Ahmed H Badran. Multiplex suppression of four quadruplet codons via trna directed evolution. *Nature communications*, 12(1):1–13, 2021.

[36] Erika A DeBenedictis, Emma J Chory, Dana W Gretton, Brian Wang, Stefan Golas, and Kevin M Esvelt. Systematic molecular evolution enables robust biomolecule discovery. *Nature Methods*, 19(1):55–64, 2022.

[37] Erika Alden DeBenedictis, Dieter Söll, and Kevin M Esvelt. Measuring the tolerance of the genetic code to altered codon size. *Elife*, 11:e76941, 2022.

[38] P C Dias. Sources and sinks in population biology. *Trends Ecol. Evol.*, 11(8):326–330, August 1996.

[39] Bryan C Dickinson, Aaron M Leconte, Benjamin Allen, Kevin M Esvelt, and David R Liu. Experimental interrogation of the path dependence and stochasticity of protein evolution using phage-assisted continuous evolution. *Proceedings of the National Academy of Sciences*, 110(22):9007–9012, 2013.

[40] Diane DiEuliis, Sarah R Carter, and Gigi Kwik Gronvall. Options for synthetic DNA order screening, revisited. *mSphere*, 2(4), July 2017.

[41] James Diggans and Emily Leproust. Next steps for access to safe, secure dna synthesis. *Frontiers in bioengineering and biotechnology*, 7:86, 2019.

[42] James Diggans and Emily Leproust. Next steps for access to safe, secure DNA synthesis. *Front Bioeng Biotechnol*, 7:86, April 2019.

[43] Mariam Elgabry, Darren Nesbeth, and Shane Johnson. The future of biotechnology crime: A parallel delphi study with non-traditional experts. *Futures*, page 102970, 2022.

[44] Justin G English, Reid HJ Olsen, Katherine Lansu, Michael Patel, Karoline White, Adam S Cockrell, Darshan Singh, Ryan T Strachan, Daniel Wacker, and Bryan L Roth. Vegas as a platform for facile directed evolution in mammalian cells. *Cell*, 178(3):748–761, 2019.

[45] Kevin M Esvelt. Inoculating science against potential pandemics and information hazards. *PLoS Pathog.*, 14(10):e1007286, October 2018.

[46] Kevin M Esvelt, Jacob C Carlson, and David R Liu. A system for the continuous directed evolution of biomolecules. *Nature*, 472(7344):499–503, 2011.

[47] Kevin M Esvelt, Andrea L Smidler, Flaminia Catteruccia, and George M Church. Emerging technology: concerning RNA-guided gene drives for the alteration of wild populations. *Elife*, 3:e03401, 2014.

[48] Raytheon Intelligence and Space. https://www.raytheonintelligenceandspace.com/what-we-do/advanced-tech/fast-na accessed August 2022.

[49] Richard P Feynman. Plenty of room at the bottom. In *APS annual meeting*, 1959.

[50] Paul S Freemont. Synthetic biology industry: data-driven design is creating new opportunities in biotechnology. *Emerging Topics in Life Sciences*, 3(5):651–657, 2019.

[51] Robert C Gentleman, Vincent J Carey, Douglas M Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus, Rafael Irizarry, Friedrich Leisch, Cheng Li, Martin Maechler, Anthony J Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney, Jean Y H Yang, and Jianhua Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.*, 5(10):R80, September 2004.

[52] H Mario Geysen, Frank Schoenen, David Wagner, and Richard Wagner. Combinatorial compound libraries for drug discovery: an ongoing challenge. *Nat. Rev. Drug Discov.*, 2(3):222–230, March 2003.

[53] Daniel G Gibson. Synthesis of DNA fragments in yeast by one-step assembly of overlapping oligonucleotides. *Nucleic Acids Res.*, 37(20):6984–6990, November 2009.

[54] J H Gillespie. Genetic drift in an infinite population. the pseudohitchhiking model. *Genetics*, 155(2):909–919, June 2000.

[55] Gene D Godbold, Anthony D Kappell, Danielle S LeSassier, Todd J Treangen, and Krista L Ternus. Categorizing sequences of concern by function to better assess mechanisms of microbial pathogenesis. *Infection and Immunity*, 90(5):e00334–21, 2022.

[56] Benjamin H Good, Michael J McDonald, Jeffrey E Barrick, Richard E Lenski, and Michael M Desai. The dynamics of molecular evolution over 60,000 generations. *Nature*, 551(7678):45–50, 2017.

[57] Vanessa E Gray, Ronald J Hause, Jens Luebeck, Jay Shendure, and Douglas M Fowler. Quantitative missense variant effect prediction using Large-Scale mutagenesis data. *Cell Syst*, 6(1):116–124.e3, January 2018.

[58] Dana Gretton, Brian Wang, Leonard Foner, Jens Berlips, Theia Vogel, Benjamin Weinstein-Raun, Martin Kysel, Walther Chen, Erika A. DeBenedictis, Andrew B. Liu, Emma Chory, Hongrui Cui, Xiang Li, Jiangbin Dong, Andres Fabrega, Christianne Dennison, Otilia Don, Tong Ye, Kaveri Uberoy, Ron Rivest, Mingyu Gao, Yu Yu, Carsten Baum, Ivan Damgard, Andrew C. Yao, and Kevin M. Esvelt. Random adversarial threshold search enables specific, secure, and automated dna synthesis

screening. https://www.securedna.org/download/Random_Adversarial_Threshold_Screening.pdf, 2020.

[59] Fiona M Guerra, Shelly Bolotin, Gillian Lim, Jane Heffernan, Shelley L Deeks, Ye Li, and Natasha S Crowcroft. The basic reproduction number (r0) of measles: a systematic review. *Lancet Infect. Dis.*, 17(12):e420–e428, December 2017.

[60] Vishal Gupta, Jesús Irimia, Iván Pau, and Alfonso Rodríguez-Patón. BioBlocks: Programming protocols in biology made easier. *ACS Synth. Biol.*, 6(7):1230–1232, July 2017.

[61] Benjamin Haby, Sebastian Hans, Emmanuel Anane, Annina Sawatzki, Niels Krausch, Peter Neubauer, and Mariano Nicolas Cruz Bournazou. Integrated robotic mini bioreactor platform for automated, parallel microbial cultivation with online data handling and process control. *SLAS Technol*, 24(6):569–582, December 2019.

[62] Sebastian Hans, Matthias Gimpel, Florian Glauche, Peter Neubauer, and Mariano Nicolas Cruz-Bournazou. Automated cell treatment for competence and transformation of escherichia coli in a High-Throughput Quasi-Turbidostat using microtiter plates. *Microorganisms*, 6(3), June 2018.

[63] Richard F Hartl, Alexander Mehlmann, and Andreas Novak. Cycles of fear: periodic bloodsucking rates for vampires. *Journal of Optimization Theory and Applications*, 75(3):559–568, 1992.

[64] Johannes Hemmerich, Stephan Noack, Wolfgang Wiechert, and Marco Oldiges. Microbioreactor systems for accelerated bioprocess development. *Biotechnol. J.*, 13(4):e1700141, April 2018.

[65] Dag O Hessen. Dissolved organic carbon in a humic lake: effects on bacterial production and respiration. *Hydrobiologia*, 229(1):115–123, 1992.

[66] Thomas A Hopf, John B Ingraham, Frank J Poelwijk, Charlotta P I Schärfe, Michael Springer, Chris Sander, and Debora S Marks. Mutation effects predicted from sequence co-variation. *Nat. Biotechnol.*, 35(2):128–135, February 2017.

[67] Takaaki Horinouchi, Teruaki Minamoto, Shingo Suzuki, Hiroshi Shimizu, and Chikara Furusawa. Development of an automated culture system for laboratory evolution. *J. Lab. Autom.*, 19(5):478–482, October 2014.

[68] Hanyang Hu, Yufeng Wei, Daocheng Wang, Ni Su, Xianjun Chen, Yuzheng Zhao, Guixia Liu, and Yi Yang. Glucose monitoring in living cells with single fluorescent protein-based sensors. *RSC Adv.*, 8(5):2485–2489, 2018.

[69] Johnny H Hu, Shannon M Miller, Maarten H Geurts, Weixin Tang, Liwei Chen, Ning Sun, Christina M Zeina, Xue Gao, Holly A Rees, Zhi Lin, et al. Evolved cas9 variants with broad pam compatibility and high dna specificity. *Nature*, 556(7699):57–63, 2018.

[70] Nuclear Threat Initiative et al. Common mechanism to prevent illicit gene synthesis. *Published March*, 22, 2019.

[71] International Gene Synthesis Consortium. Harmonized screening protocol V2. https://genesynthesisconsortium.org/wp-content/uploads/IGSCHarmonizedProtocol11-21-17.pdf, 2017.

[72] Martin Jinek, Krzysztof Chylinski, Ines Fonfara, Michael Hauer, Jennifer A Doudna, and Emmanuelle Charpentier. A programmable dual-rna–guided dna endonuclease in adaptive bacterial immunity. *science*, 337(6096):816–821, 2012.

[73] Kunihiko Kaneko and Takashi Ikegami. Homeochaos: dynamics stability of a symbiotic network with population dynamics and evolving mutation rates. *Physica D*, 56(4):406–429, June 1992.

[74] John Karanicolas, Jacob E Corn, Irwin Chen, Lukasz A Joachimiak, Orly Dym, Sun H Peck, Shira Albeck, Tamar Unger, Wenxin Hu, Gaohua Liu, et al. A de novo protein binding pair by computational design and directed evolution. *Molecular cell*, 42(2):250–260, 2011.

[75] B Keller, J Vrana, A Miller, G Newman, and E Klavins. Aquarium: the laboratory operating system (version v2. 5.0). *Zenodo*, 2019.

[76] Cameron F. Kerry, Acting Secretary, and Charles Romine Director. Fips pub 186-4 federal information processing standards publication digital signature standard (dss), 2013.

[77] Sriram Kosuri and George M Church. Large-scale de novo DNA synthesis: technologies and applications. *Nat. Methods*, 11(5):499–507, May 2014.

[78] Russell Lande. NATURAL SELECTION AND RANDOM GENETIC DRIFT IN PHENOTYPIC EVOLUTION. *Evolution*, 30(2):314–334, June 1976.

[79] Aaron M Leconte, Bryan C Dickinson, David D Yang, Irene A Chen, Benjamin Allen, and David R Liu. A population-based experimental model for protein evolution: effects of mutation rate and selection stringency on evolutionary outcomes. *Biochemistry*, 52(8):1490–1499, 2013.

[80] Richard E Lenski. Phenotypic and genomic evolution during a 20,000-generation experiment with the bacterium escherichia coli. *Plant breeding reviews*, 24(2):225–265, 2010.

[81] Gregory Lewis, Piers Millett, Anders Sandberg, Andrew Snyder-Beattie, and Gigi Gronvall. Information hazards in biotechnology. *Risk Anal.*, 39(5):975–981, May 2019.

[82] Steven V Ley, Daniel E Fitzpatrick, Richard J Ingham, and Rebecca M Myers. Organic synthesis: march of the machines. *Angew. Chem. Int. Ed Engl.*, 54(11):3449–3464, March 2015.

[83] Gene-Wei Li, David Burkhardt, Carol Gross, and Jonathan S Weissman. Quantifying absolute protein synthesis rates reveals principles underlying allocation of cellular resources, 2014.

[84] Sophia Hsin-Jung Li, Zhiyuan Li, Junyoung O Park, Christopher G King, Joshua D Rabinowitz, Ned S Wingreen, and Zemer Gitai. Escherichia coli translation strategies differ across carbon, nitrogen and phosphorus limitation conditions. *Nat Microbiol*, 3(8):939–947, August 2018.

[85] Gregory Linshiz, Nina Stawski, Garima Goyal, Changhao Bi, Sean Poust, Monica Sharma, Vivek Mutalik, Jay D Keasling, and Nathan J Hillson. PR-PR: cross-platform laboratory automation system. *ACS Synth. Biol.*, 3(8):515–524, August

2014.

[86] Zhandong Liu, Santosh S Venkatesh, and Carlo C Maley. Sequence space coverage, entropy of genomes and the potential to detect non-human DNA in human samples. *BMC Genomics*, 9:509, October 2008.

[87] Benjamin J Livesey and Joseph A Marsh. Using deep mutational scanning to benchmark variant effect predictors and identify disease mutations. *Mol. Syst. Biol.*, 16(7):e9380, July 2020.

[88] Jason Lloyd-Price, Anup Mahurkar, Gholamali Rahnavard, Jonathan Crabtree, Joshua Orvis, A Brantley Hall, Arthur Brady, Heather H Creasy, Carrie McCracken, Michelle G Giglio, Daniel McDonald, Eric A Franzosa, Rob Knight, Owen White, and Curtis Huttenhower. Erratum: Strains, functions and dynamics in the expanded human microbiome project. *Nature*, 551(7679):256, November 2017.

[89] Zhouqing Luo, Qing Yang, Binan Geng, Shuangying Jiang, Shihui Yang, Xiaozheng Li, Yizhi Cai, and Junbiao Dai. Whole genome engineering by synthesis. *Science China Life Sciences*, 61(12):1515–1527, 2018.

[90] Thomas J Magliery, J Christopher Anderson, and Peter G Schultz. Expanding the genetic code: selection of efficient suppressors of four-base codons and identification of "shifty" four-base codons with a library approach in escherichia coli. *Journal of molecular biology*, 307(3):755–769, 2001.

[91] Stephen M Maurer, Markus Fischer, Heinz Schwer, Cord Stähler, and Hubert S Bernauer. Making commercial biology safer: What the gene synthesis industry has learned about screening customers and orders. September 2009.

[92] Stephen M Maurer, Markus Fischer, Heinz Schwer, Cord Stähler, Peer Stähler, and Hubert S Bernauer. Making commercial biology safer: What the gene synthesis industry has learned about screening customers and orders. *Available at: https://gspp.dreamhosters.com/iths/Maurer_IASB_Screening.pdf*, 2009.

[93] D Meldrum. Automation for genomics, part two: sequencers, microarrays, and future trends. *Genome Res.*, 10(9):1288–1303, September 2000.

[94] Michelle M Meyer, Jonathan J Silberg, Christopher A Voigt, Jeffrey B Endelman, Stephen L Mayo, Zhen-Gang Wang, and Frances H Arnold. Library analysis of schema-guided protein recombination. *Protein Science*, 12(8):1686–1693, 2003.

[95] Maximilian Miller, Daniel Vitale, Peter C Kahn, Burkhard Rost, and Yana Bromberg. funtrp: identifying protein positions for variation driven functional tuning. *Nucleic Acids Res.*, 47(21):e142, December 2019.

[96] Kentaro Miyazaki and Misa Takenouchi. Creating random mutagenesis libraries using megaprimer PCR of whole plasmid. *Biotechniques*, 33(5):1033–4, 1036–8, November 2002.

[97] Matteo Mori, Severin Schink, David W Erickson, Ulrich Gerland, and Terence Hwa. Quantifying the benefit of a proteome reserve in fluctuating environments, 2017.

[98] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In *International conference on the theory and applications of cryptographic techniques*, pages 327–346. Springer, 1999.

[99] Charleston Noble, Jason Olejarz, Kevin M Esvelt, George M Church, and Martin A Nowak. Evolutionary dynamics of CRISPR gene drives. *Sci Adv*, 3(4):e1601964, April 2017.

[100] Armita Nourmohammad and Ceyhun Eksin. Optimal evolutionary control for artificial selection on molecular phenotypes. *Physical Review X*, 11(1):011044, 2021.

[101] OECD. OECD: Graduates by field. https://stats.oecd.org/Index.aspx?DataSetCode=EDU_GRAD_FIELD. Accessed: 2021-1-19.

[102] Rebecca Ostertag. Mechanisms to overcome ecosystem nitrogen and phosphorus limitation, 2008.

[103] Amy E Palmer, Yan Qin, Jungwon Genevieve Park, and Janet E McCombs. Design and application of genetically encoded biosensors. *Trends Biotechnol.*, 29(3):144–152, March 2011.

[104] Jason M Peters, Alexandre Colavin, Handuo Shi, Tomasz L Czarny, Matthew H Larson, Spencer Wong, John S Hawkins, Candy H S Lu, Byoung-Mo Koo, Elizabeth Marta, Anthony L Shiver, Evan H Whitehead, Jonathan S Weissman, Eric D Brown, Lei S Qi, Kerwyn Casey Huang, and Carol A Gross. A comprehensive, CRISPR-based functional analysis of essential genes in bacteria. *Cell*, 165(6):1493–1506, June 2016.

[105] Calin Plesa, Angus M Sidore, Nathan B Lubock, Di Zhang, and Sriram Kosuri. Multiplexed gene synthesis in emulsions for exploring protein functional landscapes. *Science*, 359(6373):343–347, January 2018.

[106] Carla Polycarpo, Alexandre Ambrogelly, Amélie Bérubé, SusAnn M Winbush, James A McCloskey, Pamela F Crain, John L Wood, and Dieter Söll. An aminoacyl-trna synthetase that specifically activates pyrrolysine. *Proceedings of the National Academy of Sciences*, 101(34):12450–12454, 2004.

[107] Kunal J Rambhia, Abigail S Ribner, and Gigi Kwik Gronvall. Everywhere you look: select agent pathogens, 2011.

[108] Arjun Ravikumar, Garri A Arzumanyan, Muaeen KA Obadi, Alex A Javanpour, and Chang C Liu. Scalable, continuous evolution of genes at mutation rates above genomic error thresholds. *Cell*, 175(7):1946–1957, 2018.

[109] Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nat. Methods*, 15(10):816–822, October 2018.

[110] Philip A Romero and Frances H Arnold. Exploring protein fitness landscapes by directed evolution. *Nature reviews Molecular cell biology*, 10(12):866–876, 2009.

[111] Sarvenaz Sarabipour, Christopher King, and Kalina Hristova. Uninduced high-yield bacterial expression of fluorescent proteins. *Anal. Biochem.*, 449:155–157, March 2014.

[112] Gisbert Schneider. Automating drug discovery. *Nat. Rev. Drug Discov.*, 17(2):97–113, February 2018.

[113] Yang Si, Chloé Grazon, Gilles Clavier, Jutta Rieger, Jean-Frédéric Audibert, Bianca Sclavi, and Rachel Méallet-Renault. Rapid and accurate detection of escherichia

coli growth by fluorescent ph-sensitive organic nanoparticles for high-throughput screening applications. *Biosens. Bioelectron.*, 75:320–327, January 2016.

[114] L. Simirenko, N. J. Hillson, S. Deutsch, and J. F. Cheng. Bliss: The black list sequence screening pipeline. In *2016 Synthetic Biology: Engineering, Evolution Design*, 2016.

[115] Rimvydas Simutis and Andreas Lübbert. Bioreactor control improves bioprocess performance. *Biotechnology journal*, 10(8):1115–1130, 2015.

[116] M Slatkin. Gene flow and the geographic structure of natural populations. *Science*, 236(4803):787–792, May 1987.

[117] Michael J Smanski, Swapnil Bhatia, Dehua Zhao, Yongjin Park, Lauren B A Woodruff, Georgia Giannoukos, Dawn Ciulla, Michele Busby, Johnathan Calderon, Robert Nicol, D Benjamin Gordon, Douglas Densmore, and Christopher A Voigt. Functional optimization of gene clusters by combinatorial design and assembly. *Nat. Biotechnol.*, 32(12):1241–1249, December 2014.

[118] Andrew Sparkes, Wayne Aubrey, Emma Byrne, Amanda Clare, Muhammed N Khan, Maria Liakata, Magdalena Markham, Jem Rowland, Larisa N Soldatova, Kenneth E Whelan, Michael Young, and Ross D King. Towards robot scientists for autonomous scientific discovery. *Autom. Exp.*, 2:1, January 2010.

[119] Gayathri Srinivasan, Carey M James, and Joseph A Krzycki. Pyrrolysine encoded by uag in archaea: charging of a uag-decoding specialized trna. *Science*, 296(5572):1459–1462, 2002.

[120] C N Takahashi, L Zaman, and E Klavins. Accelerating evolutionary hill climbs in parallel turbidostats. *bioRxiv*, 2017.

[121] The OpenSSL Project. OpenSSL: The open source toolkit for SSL/TLS. www.openssl.org, April 2003.

[122] The SecureDNA cryptography team. Hiding dangerous DNA in plain sight. *submitted*.

[123] Battelle. https://www.battelle.org/commercial-offerings/industry-solutions accessed August 2022.

[124] Alexander J Titus, Audrey Flower, Patrick Hagerty, Paul Gamble, Charlie Lewis, Todd Stavish, Kevin P O'Connell, Greg Shipley, and Stephanie M Rogers. SIG-DB: Leveraging homomorphic encryption to securely interrogate privately held genomic databases. *PLoS Comput. Biol.*, 14(9):e1006454, September 2018.

[125] Takuya Umehara, Jihyo Kim, Sangsik Lee, Li-Tao Guo, Dieter Söll, and Hee-Sung Park. N-acetyl lysyl-trna synthetases evolved by a ccdb-based selection possess n-acetyl lysine specificity in vitro and in vivo. *FEBS letters*, 586(6):729–733, 2012.

[126] Viktor Vasilev, Chenkai Liu, Traci Haddock, Swapnil Bhatia, Aaron Adler, Fusun Yaman, Jacob Beal, Jonathan Babb, Ron Weiss, Douglas Densmore, and Others. A software stack for specification and robotic execution of protocols for synthetic biological engineering. In *3rd International Workshop on Bio-Design Automation*, 2011.

[127] Peter M Vitousek, Stephen Porder, Benjamin Z Houlton, and Oliver A Chad-

wick. Terrestrial phosphorus limitation: mechanisms, implications, and nitrogen–phosphorus interactions. *Ecol. Appl.*, 20(1):5–15, January 2010.

[128] Meredith Wadman. United states rushes to fill void in viral sequencing, 2021.

[129] David I Walsh, 3rd, Marilene Pavan, Luis Ortiz, Scott Wick, Johanna Bobrow, Nicholas J Guido, Sarah Leinicke, Dany Fu, Shreya Pandit, Lucy Qin, Peter A Carr, and Douglas Densmore. Standardizing automated DNA assembly: Best practices, metrics, and protocols using robots. *SLAS Technol*, 24(3):282–290, June 2019.

[130] Kaihang Wang, Heinz Neumann, Sew Y Peak-Chew, and Jason W Chin. Evolved orthogonal ribosomes enhance the efficiency of synthetic genetic code expansion. *Nature biotechnology*, 25(7):770–777, 2007.

[131] Brandon G Wong, Christopher P Mancuso, Szilvia Kiriakov, Caleb J Bashor, and Ahmad S Khalil. Precise, automated control of conditions for high-throughput growth of yeast and bacteria with eVOLVER. *Nat. Biotechnol.*, 36(7):614–623, August 2018.

[132] Derrick E Wood and Steven L Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.*, 15(3):R46, March 2014.

[133] Jonathan Yang. Australia group adds 14 pathogens to control list. *Arms Control Today*, 33(6):31–31, 2003.

[134] Liqiang Zhang, Fengyu Su, Xiangxing Kong, Fred Lee, Kevin Day, Weimin Gao, Mary E Vecera, Jeremy M Sohr, Sean Buizer, Yanqing Tian, et al. Ratiometric fluorescent ph-sensitive polymers for high-throughput monitoring of extracellular ph. *RSC advances*, 6(52):46134–46142, 2016.

[135] Liqiang Zhang, Fengyu Su, Xiangxing Kong, Fred Lee, Kevin Day, Weimin Gao, Mary E Vecera, Jeremy M Sohr, Sean Buizer, Yanqing Tian, and Deirdre R Meldrum. Ratiometric fluorescent ph-sensitive polymers for high-throughput monitoring of extracellular ph. *RSC Adv.*, 6:46134–46142, May 2016.

[136] Huimin Zhao, Lori Giver, Zhixin Shao, Joseph A Affholter, and Frances H Arnold. Molecular evolution by staggered extension process (step) in vitro recombination. *Nature biotechnology*, 16(3):258–261, 1998.

[137] Yuzheng Zhao and Yi Yang. Profiling metabolic states with genetically encoded fluorescent biosensors for nadh. *Current Opinion in Biotechnology*, 31:86–92, 2015.

[138] Yuzheng Zhao and Yi Yang. Profiling metabolic states with genetically encoded fluorescent biosensors for NADH. *Curr. Opin. Biotechnol.*, 31:86–92, February 2015.

[139] Ziwei Zhong, Brandon G Wong, Arjun Ravikumar, Garri A Arzumanyan, Ahmad S Khalil, and Chang C Liu. Automated continuous evolution of proteins in vivo. *ACS synthetic biology*, 9(6):1270–1276, 2020.

[140] Zhang Zhujun and W Rudolf Seitz. A carbon dioxide sensor based on fluorescence. *Analytica chimica acta*, 160:305–309, 1984.

[141] Zhang Zhujun and W Rudolf Seitz. A carbon dioxide sensor based on fluorescence. *Anal. Chim. Acta*, 160:305–309, January 1984.

[142] Julia Zinkus-Boltz, Craig DeValk, and Bryan C Dickinson. A phage-assisted continuous selection approach for deep mutational scanning of protein–protein interac-

tions. *ACS Chemical Biology*, 14(12):2757–2767, 2019.