nature biotechnology

SUPPLEMENTARY INFORMATION

https://doi.org/10.1038/s41587-020-0584-2

In the format provided by the authors and unedited.

Automated design of thousands of nonrepetitive parts for engineering stable genetic systems

Ayaan Hossain[1], Eriberto Lopez[2], Sean M. Halper[3], Daniel P. Cetnar[3], Alexander C. Reis[3], Devin Strickland[2], Eric Klavins[2] and Howard M. Salis[1,3,4,5] ✉

[1]Bioinformatics and Genomics, Pennsylvania State University, University Park, PA, USA. [2]Department of Electrical & Computer Engineering, University of Washington, Seattle, WA, USA. [3]Department of Chemical Engineering, Pennsylvania State University, University Park, PA, USA. [4]Department of Biological Engineering, Pennsylvania State University, University Park, PA, USA. [5]Department of Biomedical Engineering, Pennsylvania State University, University Park, PA, USA. ✉e-mail: salis@psu.edu

Supplementary Information: Automated Design of Thousands of Highly Non-Repetitive Genetic Parts for Engineering Evolutionary Robust Genetic Systems

Ayaan Hossain[1], Eriberto Lopez[5], Sean M. Halper[2], Daniel P. Cetnar[2], Alexander C. Reis[2], Devin Strickland[5], Eric Klavins[5], and Howard M. Salis[1-4 *]

[1]Bioinformatics and Genomics, [2]Department of Chemical Engineering, [3]Department of Biological Engineering, [4]Department of Biomedical Engineering, Pennsylvania State University, University Park, PA 16802, USA.

[5]Department of Electrical & Computer Engineering, University of Washington, Seattle, WA 98195, USA.

* Corresponding author: Howard M. Salis, salis@psu.edu

Table of Contents

Supplementary Note 1

We formally show that the problem of deciding the existence of a NON-REPETITIVE SET of strings of a desired integer cardinality in a given set of strings is equivalent to the problem of deciding if there exists an INDEPENDENT SET of desired integer size in an undirected graph, and therefore NP-COMPLETE.

Definition 1: A string T is NON-REPETITIVE to itself or another string U with respect to an integer L_{MAX}, where $0 \leq L_{MAX} < \min(|T|, |U|) \Leftrightarrow$ there does not exist any substring $T[m, m + L_{MAX} + 1) = T[n, n + L_{MAX} + 1)$ or $T[m, m + L_{MAX} + 1) = U[r, r + L_{MAX} + 1)$, where $0 \leq m, n \leq |T| - L_{MAX} - 1, 0 \leq r \leq |U| - L_{MAX} - 1, m \neq n$ but possibly $m = r$.

Definition 2: A set of strings forms a NON-REPETITIVE SET when all strings and pairs of strings in the set are NON-REPETITIVE with respect to an integer L_{MAX}.

Definition 3: The problem of deciding if there exists a NON-REPETITIVE SET of integer size k with respect to an integer L_{MAX} inside a given set of strings is defined as the NON-REPETITIVE SET PROBLEM.

Definition 4: A set of vertices I forms an INDEPENDENT SET in a simple graph $G = (V, E)$, where V is the set of all vertices in G and E is the set of all undirected edges in $G \Leftrightarrow I \subseteq V$ and for all pair of vertices $\{I_i, I_j\} \in I$ the edge $(I_i, I_j) \notin E$.

Definition 5: The problem of deciding if there exists an INDEPENDENT SET of integer size k in a given simple graph is defined as the INDEPENDENT SET PROBLEM[1], and is known to be NP-COMPLETE.

Lemma 1: NON-REPETITIVE SET PROBLEM $\leq_p$ INDEPENDENT SET PROBLEM.

Proof: For any given set of strings and an integer L_{MAX} consider a simple graph R built such that for every string in the given set a single vertex labelled by the string is added to R. Next, all vertices in R whose labelled string is not NON-REPETITIVE to itself are removed from R. Finally, for every pair of vertices in R if their labelled strings are not NON-REPETITIVE, an edge between the vertices is added. Naturally, there is a NON-REPETITIVE SET of size k in the given string set if R has an INDEPENDENT SET of size k. ■

Lemma 2: INDEPENDENT SET PROBLEM $\leq_p$ NON-REPETITIVE SET PROBLEM.

Proof: For any given simple graph G, consider the set of strings W of the same size as V and defined over the alphabet Σ, where $|\Sigma| = |E|$, and the availability of a bijective function $f: E \longrightarrow \Sigma$, such that $W_i = f\big((V_i, V_j)\big)f((V_i, V_k)) \dots ((V_i, V_l))$ for every V_i and all its neighbors in G. Any NON-REPETITIVE SET in W with respect to $L_{MAX} = 0$ then encodes an INDEPENDENT SET in G because all repeated substrings of length 1 between a pair of strings W_m and W_n is of the form $f((V_m, V_n))$ denoting the edge between V_m and V_n, and by definition either W_m or W_n can exist in the NON-REPETITIVE SET but never both. If there exists a NON-REPETITIVE SET of size k in W, then there must also exist an INDEPENDENT SET of size k in G. ■

Theorem 1: NON-REPETITIVE SET PROBLEM is equivalent to INDEPENDENT SET PROBLEM.

Proof: From Lemma 1 and 2. ■

Theorem 2: NON-REPETITIVE SET PROBLEM is NP-COMPLETE.

Proof: From Theorem 1 and Definition 5. ■

Supplementary Note 2

We prove formally that the problem of deciding the existence of a single binary NON-REPETITIVE string of a desired integer length that also satisfies a MODEL FUNCTION is NP-HARD by reducing in polynomial time the RESTRICTED SUPERSTRING PROBLEM to the former, and show that it is in fact NP-COMPLETE.

Definition 6: For a given set of strings S over an alphabet Σ, a string T over Σ is defined as the COMMON SUPERSEQUENCE of $S \Leftrightarrow \forall\, s \in S\ \exists\, T[m: m + |s|) = s$, where $|s| \leq |T|$ and $0 \leq m \leq |T| - |s|$.

Definition 7: The problem of deciding if there is a COMMON SUPERSEQUENCE of an integer length $\leq k$ for a given set of strings S over Σ is defined as SHORTEST COMMON SUPERSEQUENCE PROBLEM[2], and is known to be NP-COMPLETE whenever $|\Sigma| \geq 2$. A related problem in which we want to decide the existence of a COMMON SUPERSEQUENCE T of length $\leq k$ for a given set of strings S over $\Sigma = \{0,1\}$ such that $\forall\, s \in S$ the string s is a substring of T exactly once is defined as the RESTRICTED SUPERSTRING PROBLEM.

Lemma 3: RESTRICTED SUPERSTRING PROBLEM is in NP.

Proof: If every binary string in a given set S occurs exactly once in a candidate binary string T and $|T| \leq k \Leftrightarrow T$ is a valid solution to the problem instance. These checks can be done in polynomial time. ■

Lemma 4: SHORTEST COMMON SUPERSEQUENCE PROBLEM $\leq_p$ RESTRICTED SUPERSTRING PROBLEM.

Proof: For a given set of binary strings S and an integer k, if we decide the existence of a COMMON SUPERSEQUENCE of length $\leq k$ in which every $s \in S$ occurs exactly once then there is a valid solution to the related instance of SHORTEST COMMON SUPERSEQUENCE PROBLEM as well. Otherwise, if we decide the non-existence of such a COMMON SUPERSEQUENCE, then any other COMMON SUPERSEQUENCE of length $\leq k$ where any $s \in S$ is a substring more than once is clearly impossible, and therefore a solution to the related instance of SHORTEST COMMON SUPERSEQUENCE PROBLEM is also non-existent. ■

Lemma 5: RESTRICTED SUPERSTRING PROBLEM is NP-HARD.

Proof: From Lemma 4 and Definition 7. ■

Theorem 3: RESTRICTED SUPERSTRING PROBLEM is NP-COMPLETE.

Proof: From Lemma 3 and 5. ■

Definition 8: A MODEL FUNCTION h is a selection function on a binary string B using a tuple (c, P, Q), where $c: \mathbb{Z}_2^* \longrightarrow \{True, False\}$ and both P and $Q \in \mathbb{R}$, that returns P if $c(B) = True$, or Q otherwise.

Definition 9: A binary string B is defined to be OBJECTIVE with respect to an integer H_{MIN} for a given set of MODEL FUNCTIONS $\{h_1, h_2, \dots, h_n\}$ using the set of tuples $\{(c_1, P_1, Q_1), (c_2 P_2, Q_2), \dots (c_n, P_n, Q_n)\}$, if $\sum_{i=1}^{n} h_i(B) \geq H_{MIN}$.

Definition 10: The problem of deciding if there is a single binary string of an integer length k that is NON-REPETITIVE to itself with respect to an integer L_{MAX} as well as OBJECTIVE for a set of MODEL FUNCTIONS with respect to another integer H_{MIN} is defined as the NON-REPETITIVE STRING PROBLEM.

Lemma 6: NON-REPETITIVE STRING PROBLEM is in NP.

Proof: If a candidate string B is composed of binary alphabet, is of length k, is NON-REPETITIVE with respect to L_{MAX} and OBJECTIVE with respect to H_{MIN} for all defined MODEL FUNCTIONS, then it is a valid solution to the problem instance. All individual conditions are easily verifiable in polynomial time. ■

Lemma 7: RESTRICTED SUPERSTRING PROBLEM $\leq_p$ NON-REPETITIVE STRING PROBLEM.

Proof: Given any set of binary strings S, where the longest string in S of length t_{MAX} and the shortest string in S is of length t_{MIN}, the least possible COMMON SUPERSEQUENCE length may be $k_{MIN} = t_{MAX}$ (when the longest string in S has every string in S as a substring) while the longest non-trivial COMMON SUPERSEQUENCE may be of length $k_{MAX} = \sum_{i=1}^{|S|} |S_i|$ (a concatenation of all strings in S). For a set of binary strings S and an integer k given as an instance of RESTRICTED SUPERSTRING PROBLEM, where $k_{MIN} \leq k \leq k_{MAX}$ (COMMON SUPERSEQUENCES shorter than k_{MIN} cannot exist but longer than k_{MAX} may exist trivially), consider the generated set of MODEL FUNCTIONS M, defined over the set of tuples N, where $|M| = |N| = |S|$, such that $\forall\, S_i \in S$, M_i for an input binary string B uses $N_i = (c_i, 1, 0)$, where c_i returns $True$ if S_i is a substring of B, else returns $False$. We then decide in polynomial time if a binary string B of length k exists that is OBJECTIVE for the generated set of MODEL FUNCTIONS with respect to $H_{MIN} = |S|$, and NON-REPETITIVE with respect to $L_{MAX} = k_{MIN} - 1$ (every string in S may then occur only once in B). On non-existence of B, we iterate again to decide the existence of an OBJECTIVE NON-REPETITIVE string B' with same L_{MAX} but of length $k - 1$, and so on until our last iteration where the length requirement is k_{MIN}. At any iteration if we decide the existence of an OBJECTIVE NON-REPETITIVE string B'' satisfying the instanced NON-REPETITIVE STRING PROBLEM, then there exists a COMMON SUPERSEQUENCE of length $\leq k$ for the given instance of RESTRICTED SUPERSTRING PROBLEM – in B'' every string $s \in S$ is a substring exactly once. On non-existence of B'', it follows that a COMMON SUPERSEQUENCE for the instance of RESTRICTED SUPERSTRING PROBLEM of the same length as B'' does not exist. ■

Lemma 8: NON-REPETITIVE STRING PROBLEM is NP-HARD.

Proof: From Lemma 7 and Theorem 3. ■

Theorem 4: NON-REPETITIVE STRING PROBLEM is NP-COMPLETE.

Proof: From Lemma 6 and 8. ■

Supplementary Note References

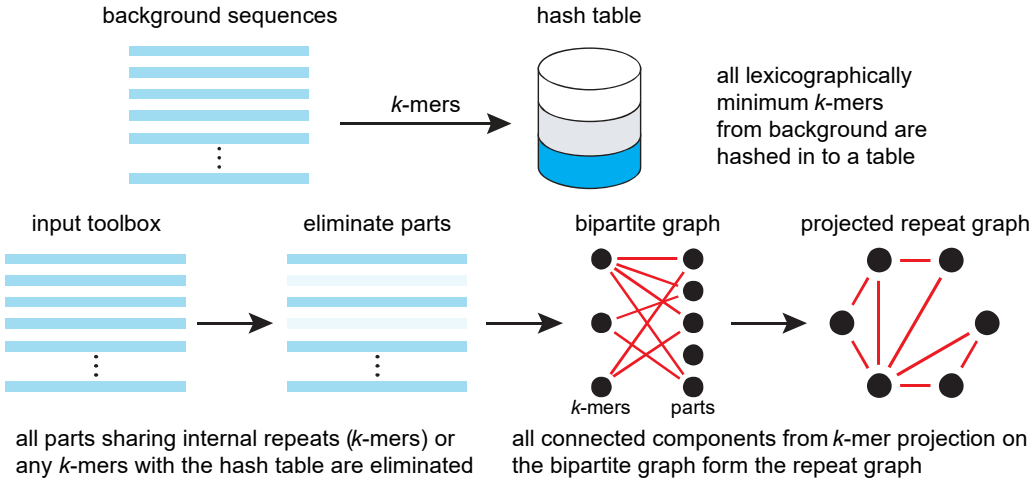
1. Tarjan, Robert Endre, and Anthony E. Trojanowski. "Finding a maximum independent set." *SIAM Journal on Computing* 6.3 (1977): 537-546.

2. Räihä, Kari-Jouko, and Esko Ukkonen. "The shortest common supersequence problem over binary alphabet is NP-complete." *Theoretical Computer Science* 16.2 (1981): 187-198.

Finder Mode Algorithm

Input to the algorithm are two sets of strings in alphabet $\Sigma = \{\mathcal{A}, \mathcal{T}, \mathcal{G}, \mathcal{C}\}$ called background (B) and parts (P) respectively, and the maximum allowed repeat length L_{MAX}. Note than B doesn't have to be non-repetitive with respect to L_{MAX}.

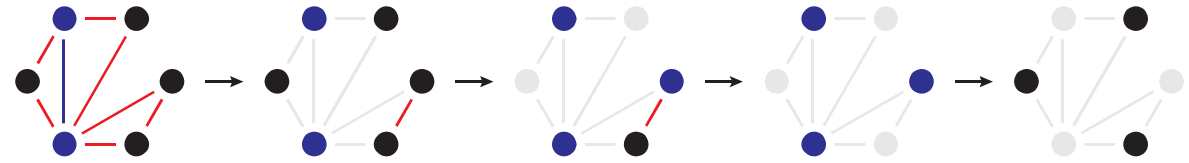
NRP Calculator Finder Mode algorithm first hashes all lexicographic minima of k-mers (k = L_{MAX} + 1) from each $b \in B$ in to a hash table (F). Next, every $p \in P$ that contains a lexicographic minimum of a k-mer that is already present in F or contains the same k-mer twice inside it is eliminated from P. For all remaining parts, a bipartite graph $C = (U, V, E)$ is initialized where vertices in U are lexicographic minima of k-mers from all $p \in P$ while vertices in V are the set of integers $\{0,1, \ldots, |P| - 1\}$ | $(U_i, V_i) \in E \Leftrightarrow U_i$ or its reverse complement is a substring of P_i. All connected components from the unweighted V-projection of C then form a simple graph $R = (V, E')$ referred to as a repeat graph.



Two steps are then repeated while $|E'| > 0$.

Step 1: all vertices u with degree ≤ 1 are deleted from R while their neighbors u', if any, are first included in the vertex cover V' and then deleted from R.

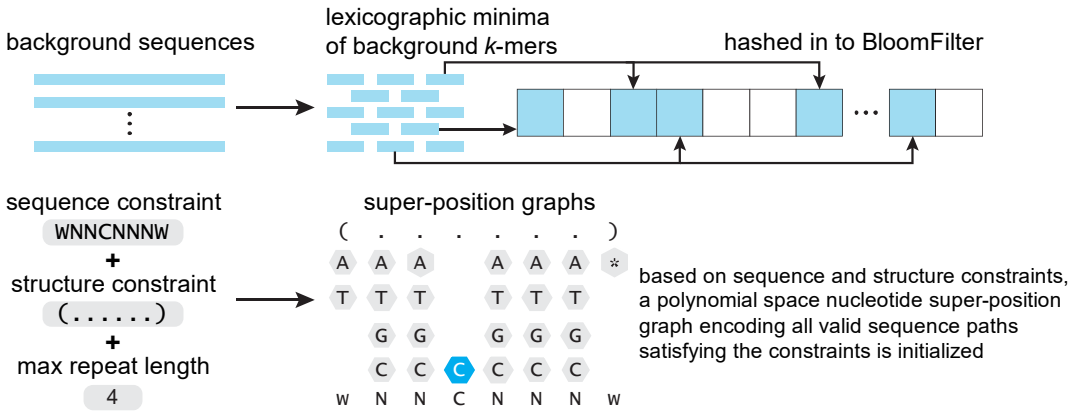
Step 2: all vertices w with maximum degree in R and their maximum degree neighbor w' are selected for inclusion in the vertex cover V' and then both w and w' are deleted from R.



The algorithm then computes $P' = \{P_v \mid v \in V \setminus V'\}$ as the non-repetitive subset in P with respect to L_{MAX}. If any of the vertices in V' are non-adjacent to vertices in P', denoted as Q, then the non-repetitive subset recovered from the subgraph of R, induced by vertices in Q (using the two steps above) are added to P'. This recovery of vertices is repeated iteratively and minimizes the size of vertex covers computed previously. When vertices can no longer be recovered, the algorithm returns (an enlarged) P' to the user.

Maker Mode Algorithm. Input to the algorithm are a set of background sequences (B) over the alphabet $\Sigma = \{\mathcal{A}, \mathcal{T}, \mathcal{G}, \mathcal{C}\}$, a degenerate IUPAC sequence constraint string (P) over $\Sigma' = \mathbb{P}(\Sigma) \setminus \{\}$ (e.g. 'N' for $\{\mathcal{A}, \mathcal{T}, \mathcal{G}, \mathcal{C}\}$ or 'K' for $\{\mathcal{G}, \mathcal{T}\}$ and so on), a balanced nucleic acid secondary structure constraint string (Q) over $\Sigma'' = \{(,), x, .\}$, where $|Q| = |P|$, a set of user defined model functions (M) that must be satisfied for part inclusion in the toolbox, a maximum toolbox size S_{MAX}, an initial failure limit X_{MAX}, and the maximum allowed repeat length L_{MAX}.

All lexicographic minima of k-mers (k = L_{MAX} + 1) from $b \in B$ are hashed in to a bloom filter (F), for memory efficiency, sized using the number of unique k-mers estimated in B (via HyperLogLog algorithm) and the total number of unique k-mers to be added based on S_{MAX}. The algorithm then initializes a multi-partite digraph $H = (V^0, V^1, \ldots, V^{n-1}, A)$ referred to as a nucleotide superposition graph, where $n = |P| = |Q|$ and the arcs $A: V^{i-1} \longrightarrow V^i$ for all $1 \leq i \leq n-1$. Note that super-position graphs are also directed acyclic graphs due to the topological ordering induced by arcs. Every V^i of H is then either a set of vertices labelled by alphabets associated with P_i, or a special vertex automatically labelled as the complement of an upstream index as constrained by Q (following canonical Watson-Crick base pairing).



After initializing index variable $i = 0$, and traceback pointers $t_1 = t_2 = \Phi$, each genetic part string in the toolbox set S satisfying given constraints is then generated by tracing a path through H from V^0 to V^{n-1} using six steps repeated until $|S| = S_{MAX}$.

Step 1: a vertex in V^i is selected at random if $i \neq \Phi$, else path finding halts.

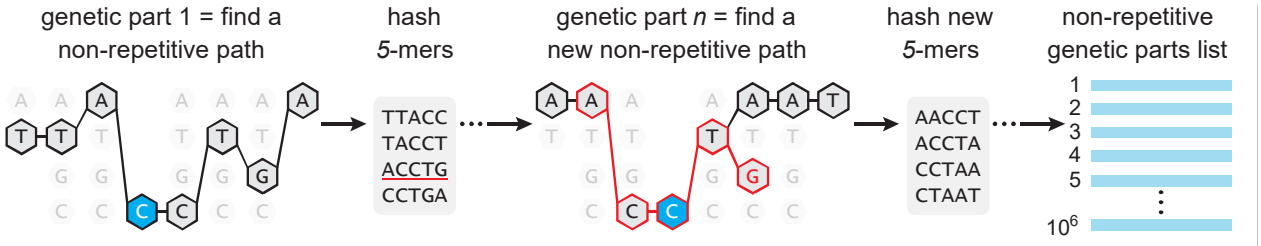
Step 2: if any model function $m \in M$ returns the tuple $(False, j)$, where $0 \leq j \leq i$ is the ending index of failure, for the partial path through $V^0, V^1, \ldots, V^i$, then $t_1 = j$ and step 4 is executed next. In case of multiple failures $t_1 = j$ farthest from i.

Step 3: if for any $i \geq k$ the sub-path r through $V^{i-k}, V^{i-k+1}, \ldots, V^i$ or reverse-complement of r occurs in F, then $t_1 = i$. Else if the sub-path r' through $V^{j-k}, V^{j-k+1}, \ldots, V^j = r$ or reverse-complement of r, for some $j \geq k$ and $j < i$ then $t_1 = i$ and $t_2 = j$.

Step 4: if $t_1 \neq \Phi$, then $i = t_1$ and $t_1 = \Phi$, else step 5 is executed. If $|V^i| \geq 2$, then the vertex selected in V^i is deleted and step 5 is executed next. Else, while $|V^i| = 1$, all vertices deleted from V^i are restored and $i = i - 1$. This is repeated at most k times, after which $i = \Phi$.

Step 5: same as step 4 but uses t_2, and executed only if $i = \Phi$ from step 4.

Step 6: if $i \neq \Phi$, the variable i is incremented by 1 and $t_1 = t_2 = \Phi$.



based on L_{MAX}, whenever path generation identifies the existence of repeats (shared or internal), in this case the *5*-mer, ACCTG at position 2 in the path for part*n* which occurred previously at position 3 in genetic part 1, a traceback is initiated to find a suitable alternative *k*-mer at the offending positions; traceback is also used to design sequences that satisfy local model functions applied concurrently with path generation. note: lexicographic minima of the*k*-mers are hashed in to the BloomFilter in practice

If a path is successfully generated and a structure constraint was provided, then the minimum free energy (mfe) secondary structure of the path Q' as predicted by the RNAfold algorithm in ViennaRNA is compared to the given Q. If Q' violates Q, then the ViennaRNA inverse-fold algorithm (RNAinverse) is used on the generated path to initialize a candidate path which is then corrected to be non-repetitive and conform to the model functions using the previous six steps. This structure reinforcement step is done only once per *de novo* path. If a part generation attempt halts, then the algorithm is restarted, initially up to X_{MAX} times, which is gradually lowered to $\lceil 1/Y_{EST} \rceil$ times, where Y_{EST} is the estimated Bernoulli probability of success. On successfully generating a path, all lexicographic minimum of k-mers from the string spelled by the path are hashed into F.

Maximum Number of Vertex Covers (using 11049 *E. coli* promoter toolbox as an example).

Consider a toolbox with N genetic parts that share some level of repetitiveness, which corresponds to a graph with N nodes and some large number of edges. The number of possible vertex covers is the number of ways that one can remove a set of nodes from the graph to remove all the edges and thereby create a disconnected graph. The trivial vertex cover is all N nodes, as removing them creates a null graph without any edges. But to find the minimum vertex cover by enumeration, we would need to try removing all possible subsets of nodes and check whether this removal created a disconnected graph. The number of ways that k nodes could be removed from a graph with N total nodes is:

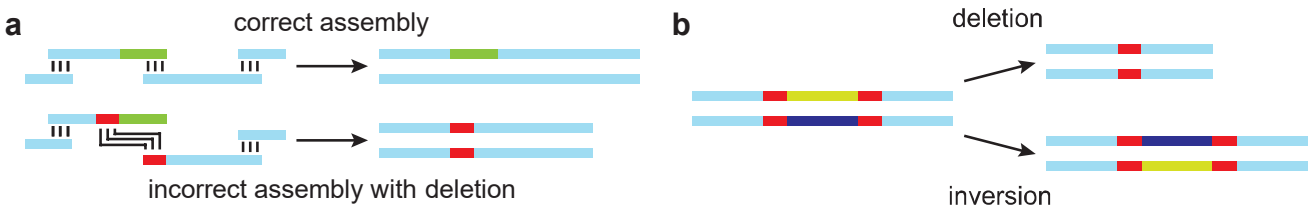
$$\frac{N!}{k!\,(N-k)!}$$

The number of ways that all node subsets could be removed is the sum of this expression where k is varied from 1 to N, which is:

$$\sum_{k=1}^{N} \frac{N!}{k!\,(N-k)!}$$

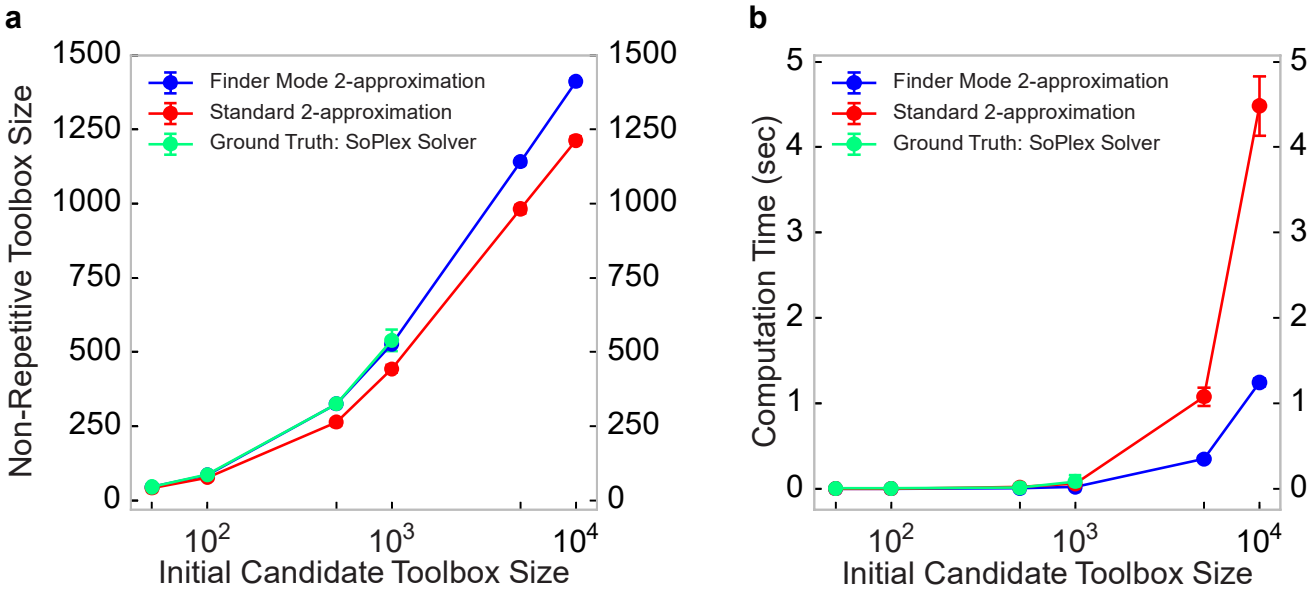
Now consider an *E. coli* promoter toolbox containing 11049 sequences. When $N = 11049$, the maximum number of possible vertex covers is:

$$\sum_{k=1}^{11049} \frac{11049!}{k!\,(11049-k)!} \approx 2^{11049} = 1.2 \times 10^{3326}$$



Supplementary Figure 1: Synthesis Difficulties and Genetic Instabilities due to Repeats

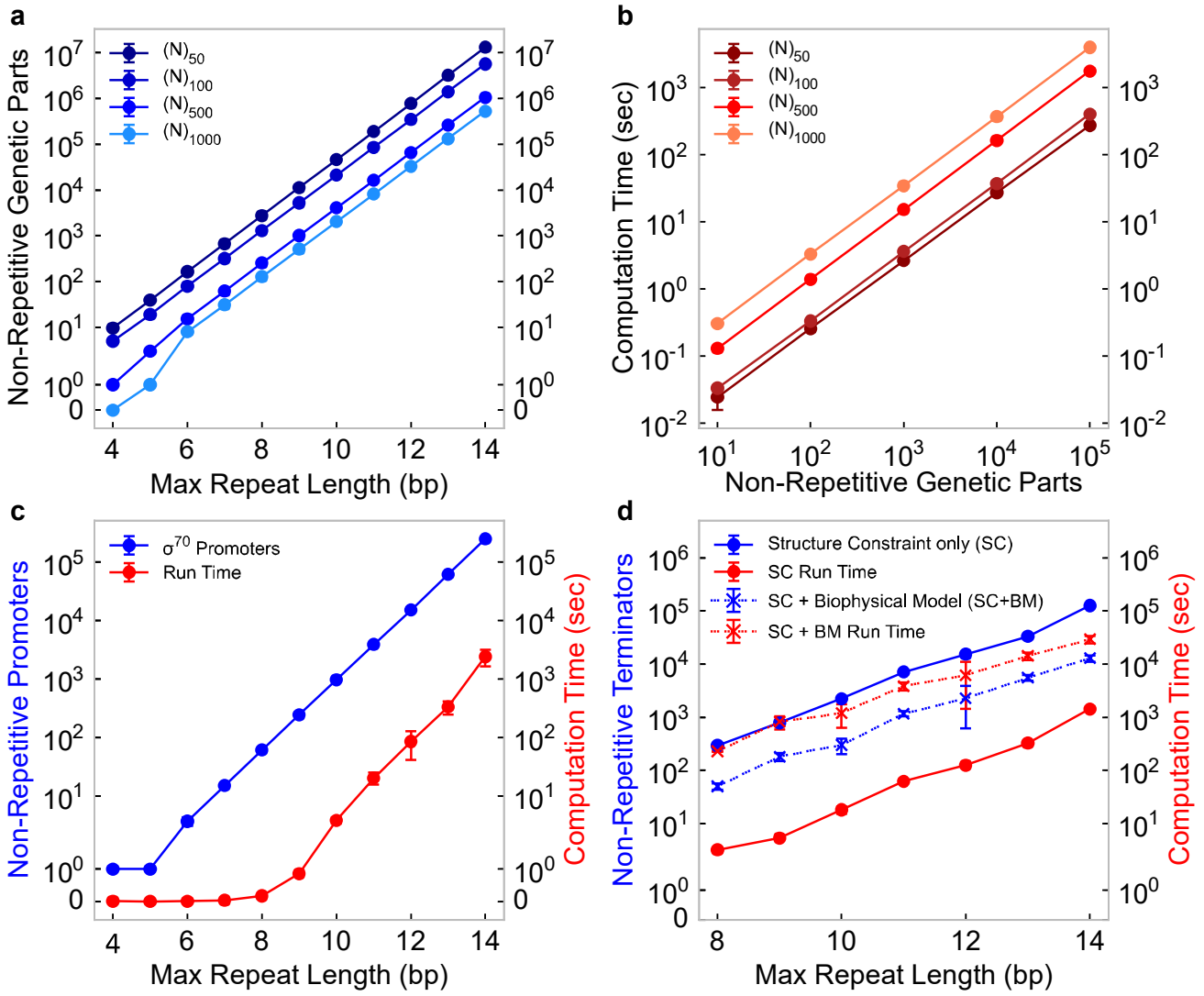
(**a**) Commercial synthesis of long DNA fragments relies on hierarchical assembly of short oligonucleotide fragments. When a large DNA construct is repetitive, its constituent fragments may hybridize at multiple locations or in unintended configurations. This leads to incomplete or mixed products, and failed assemblies. (**b**) Repeats also play a crucial role in the cellular process of homologous recombination, during which segments of DNA flanked by repeats may be exchanged, duplicated, inverted or deleted. When a repetitive genetic system inserted in an organism causes biochemical stress, the cell may respond with homologous recombination – sections of the system are deleted, effectively breaking it.



Supplementary Figure 2: Non-Repetitive Parts Calculator Finder Mode: Computation Benchmarks

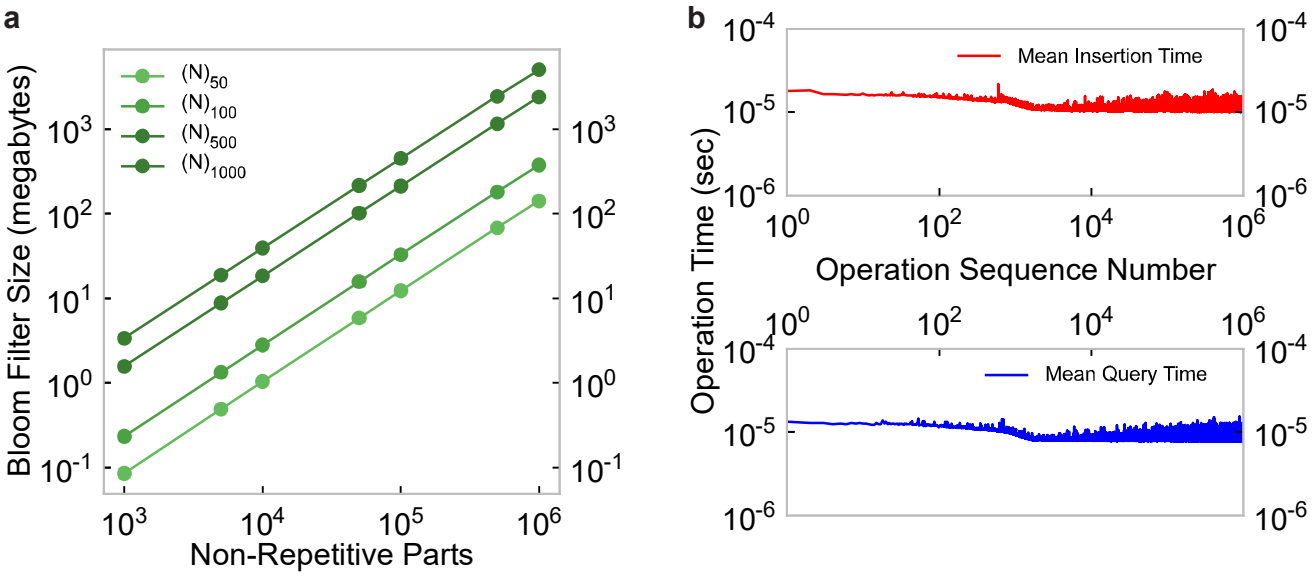
(**a**) Up to 17% larger non-repetitive toolboxes identified in simulated datasets compared to using the standard 2-approximation algorithm that closer to optimal solutions verified using the SoPlex Solver (n = 100 independent simulated datasets, error bars = 1 standard deviation from mean). (**b**) Up to 4× less computation time incurred compared to the standard linear time 2-approximation method (n = 100 independent simulated datasets, error bars = 1 s.d. from mean).

Note: SoPlex Solver does not terminate in reasonable time for large and dense graphs (≤ 24 hours).



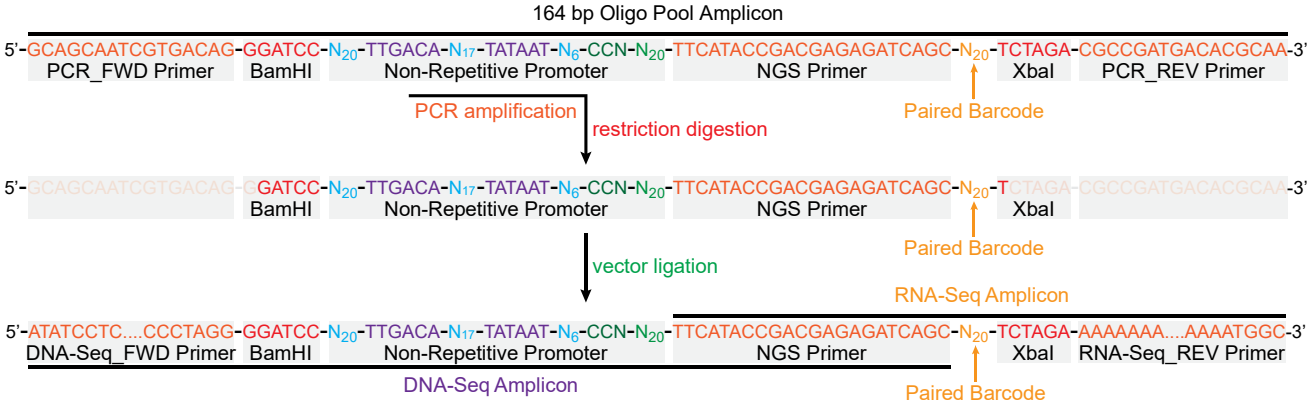
Supplementary Figure 3: Non-Repetitive Parts Calculator Maker Mode: Computation Benchmarks

(**a**) Increasing maximum repeat length (L_{MAX}) linearly allows consistent generation of exponentially large toolboxes (n = 3 independent runs, error bars = 1 standard deviation from mean). (**b**) Increasing toolbox size consistently incurs a linear increment of computation time (n = 3 independent runs, error bars = 1 s.d. from mean, L_{MAX} = 14). (**c-d**) Arbitrary design constraints (such as those for a 35 bp σ^{70} promoter, or a 43 bp rho independent terminator) retains efficient generation of large toolboxes of non-repetitive genetic parts (n = 3 independent runs, error bars = 1 s.d. from mean).



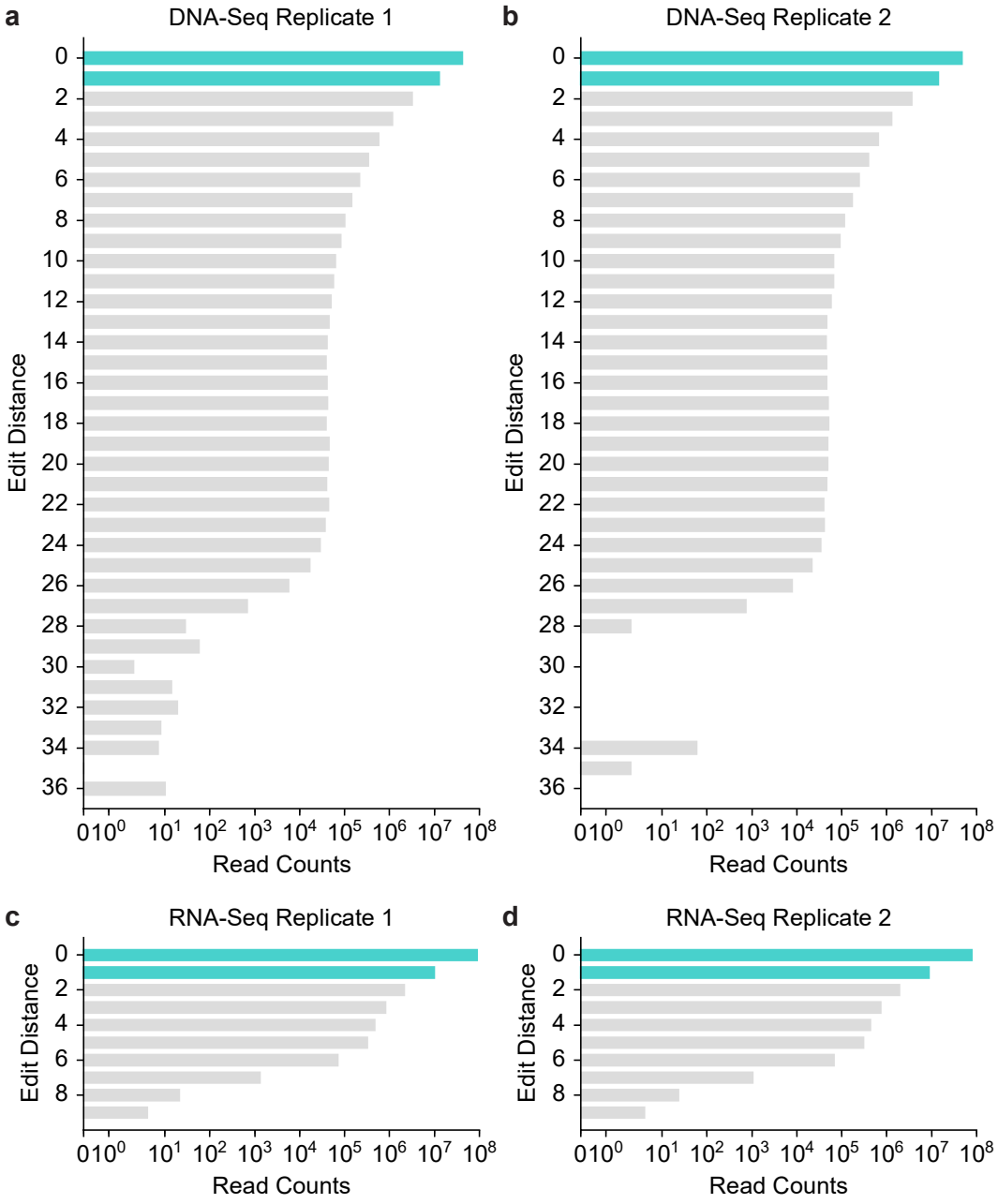
Supplementary Figure 4: Non-Repetitive Parts Calculator Maker Mode: Bloom Filter Benchmarks

(**a**) Bloom filter implementation uses a fixed number of bits per k-mer hashed, independent of k, and the memory consumed by the filter scales linearly with the desired toolbox size. (**b**) Constant insertion and query time (~10 µs) retained across operation sequences (n = 4 independent runs each with 1 million insertion or query operations, errors = 1 standard deviation from mean).



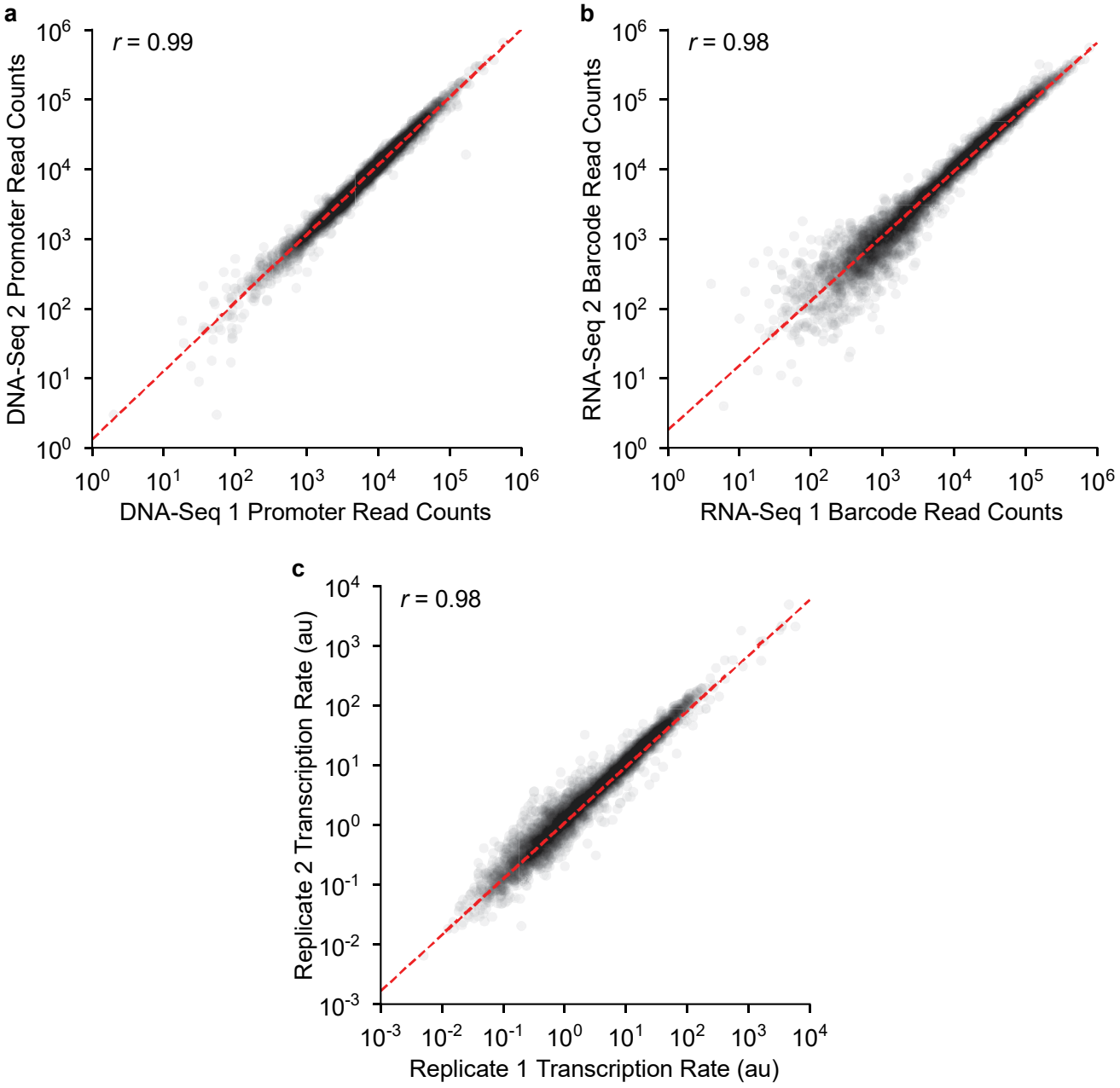
Supplementary Figure 5: *E. coli* Promoter Library Amplicon Architecture

Each promoter was uniquely barcoded such that the minimum hamming distance between any two barcodes was at least 5. All promoter constructs were synthesized together in a single oligo pool and cloned into NEB 5α *E. coli* strain in two replicates. Approximately, 4 million colonies were observed in both replicates. Promoters from plasmid DNA, and barcodes from transcript RNA were harvested from replicates and subsequently sequenced for downstream quantification.



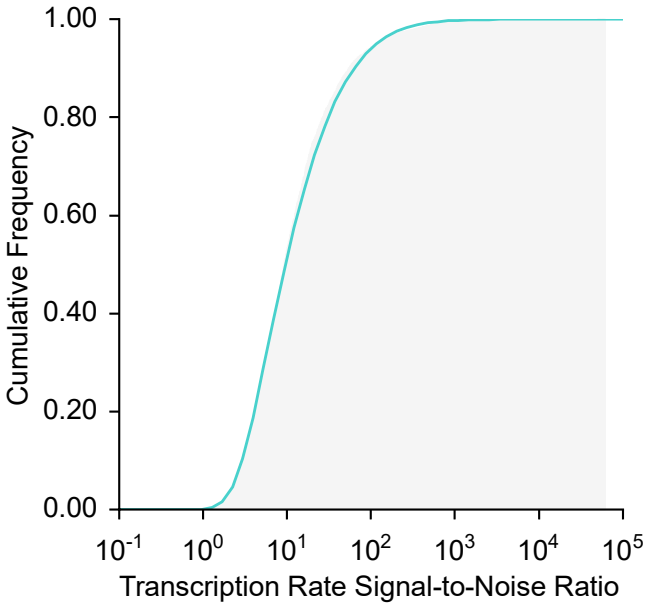
Supplementary Figure 6: *E. coli* Promoter Library Edit Distance Distribution of DNA-Seq and RNA-Seq Reads Across Replicates

(**a-b**) A total of 128 million amplicon DNA-Seq reads were uniquely mapped to the designed promoters (60 + 68 million) of which, 53 million reads in Replicate 1 and 60 million reads in Replicate 2 mapped within Levenshtein edit distance of 1 and were used for downstream quantification. **(c-d)** A total of 191 million RNA-Seq amplicon reads encompassing the paired barcodes were retained, of which 98 million reads in Replicate 1 and 85 million reads in Replicate 2 mapped within Levenshtein edit distance of 1 and used in computing the transcription rates of the promoters.



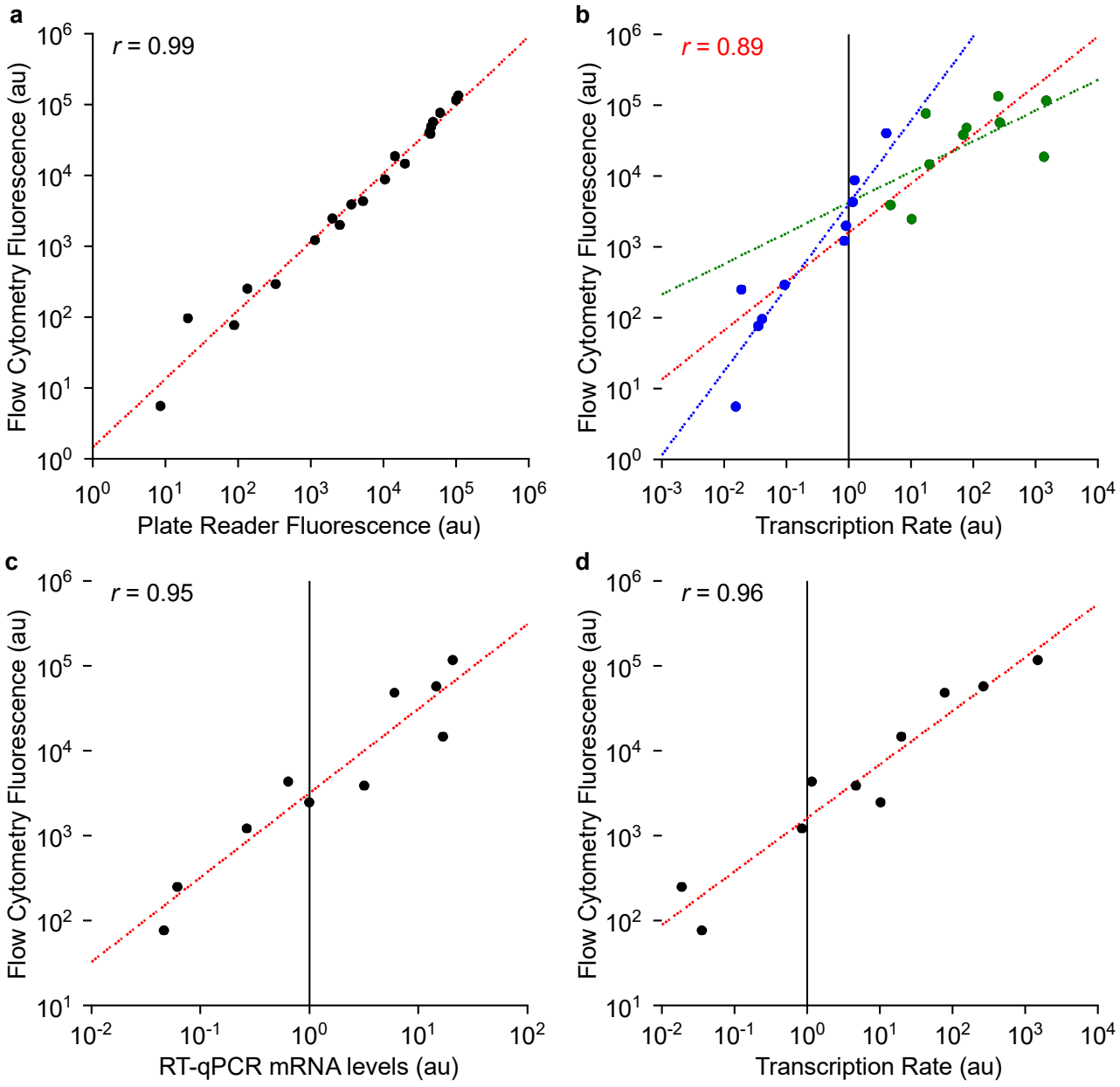
Supplementary Figure 7: NGS Replicate Read Count Correlation of *E. coli* Promoters and Paired Barcodes and their Measured Transcription Rates

Mapped read counts for (**a**) all (n = 4,350) promoters in DNA-Seq replicates and (**b**) all (n = 4,350) paired barcodes in RNA-Seq replicates were highly correlated (Pearson's r^2 = 0.98 and 0.96 respectively). The (**c**) measured transcription rates of the promoters are also correlated across the replicates (Pearson's r^2 = 0.96).



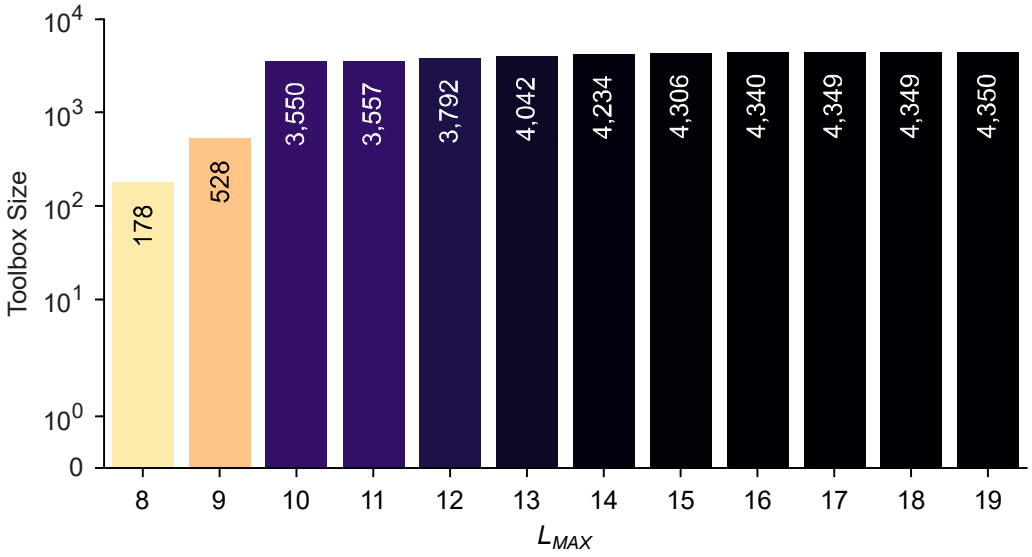
Supplementary Figure 8: Signal-to-Noise Ratio Cumulative Distribution of Measured Transcription Rates in *E. coli* Promoters

The minimum signal-to-noise ratio (SNR) observed for the promoters was 1.14, the mean and median SNR were 11.36 and 9.25 respectively, while the maximum SNR was over 62,000. Over 74.23% of the entire library satisfied the Rose criterion (SNR > 5), while over 46% of the library had SNR ≥ 10.



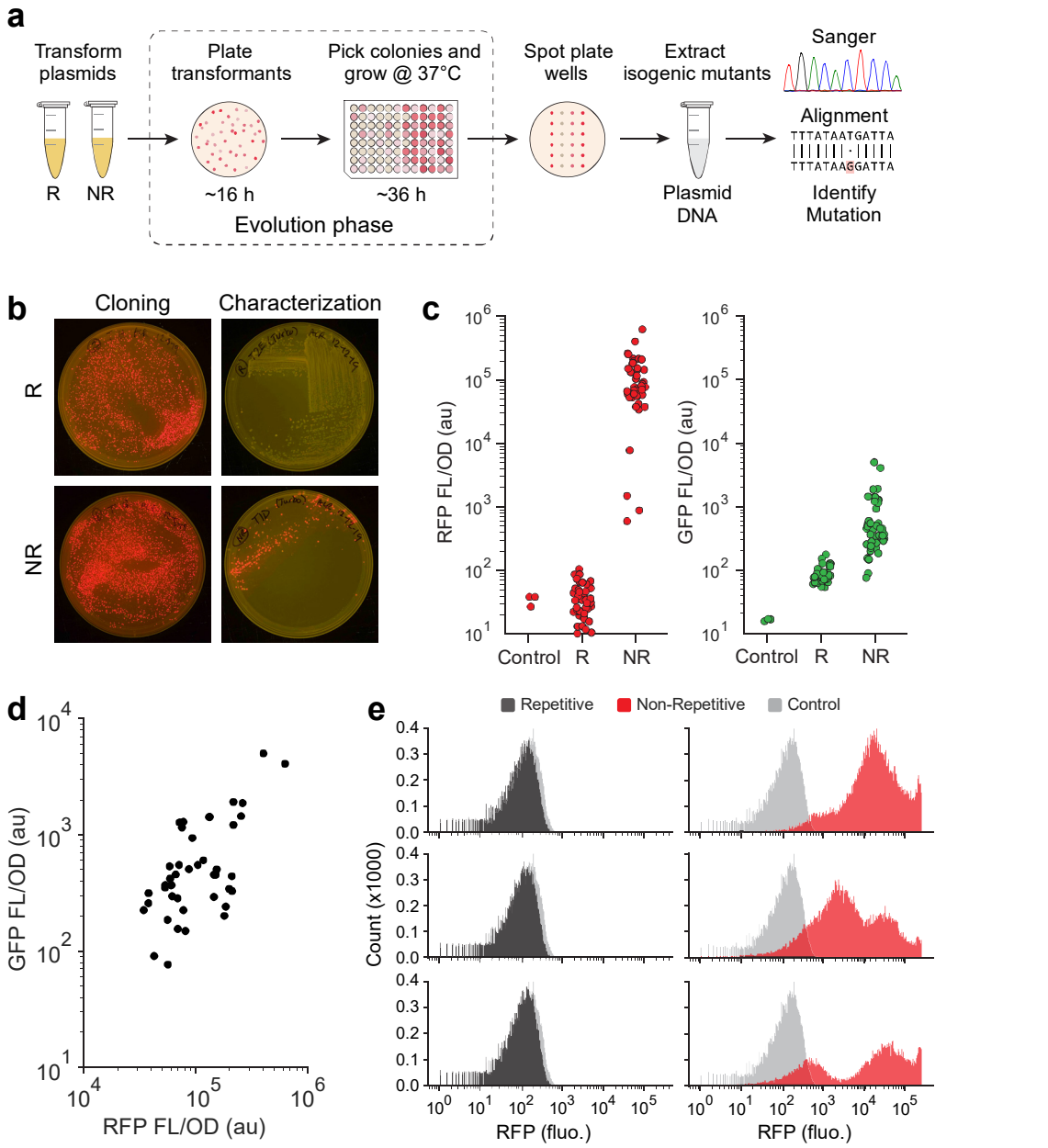
Supplementary Figure 9: Validation of *E. coli* Promoters

A subset of (n = 20) promoters spanning a large transcription rate dynamic range were characterized by measuring fluorescence and transcript levels from downstream mRFP1 reporter gene in triplicates. (**a**) The fluorescence readouts were highly correlated (Pearson's r = 0.99) in plate reader measurements and flow cytometry. (**b**) Measurements from flow cytometry were highly correlated (Pearson's r = 0.89) with transcription rates measured via NGS. All promoters with transcription rates less than 4× J23100 (blue) showed higher correlation (Pearson's r = 0.92) with measured transcription rates than promoters with higher transcription rates (green, Pearson's r = 0.63), suggesting that higher transcription rates cause excess ribosomal burden, leading to lower expression levels. (**c-d**) mRFP1 transcript levels and fluorescence measured in triplicates for (n = 10) promoters sub selected from the larger set of 20 promoters were highly correlated (Pearson's r = 0.95 and 0.96) with measured rates.



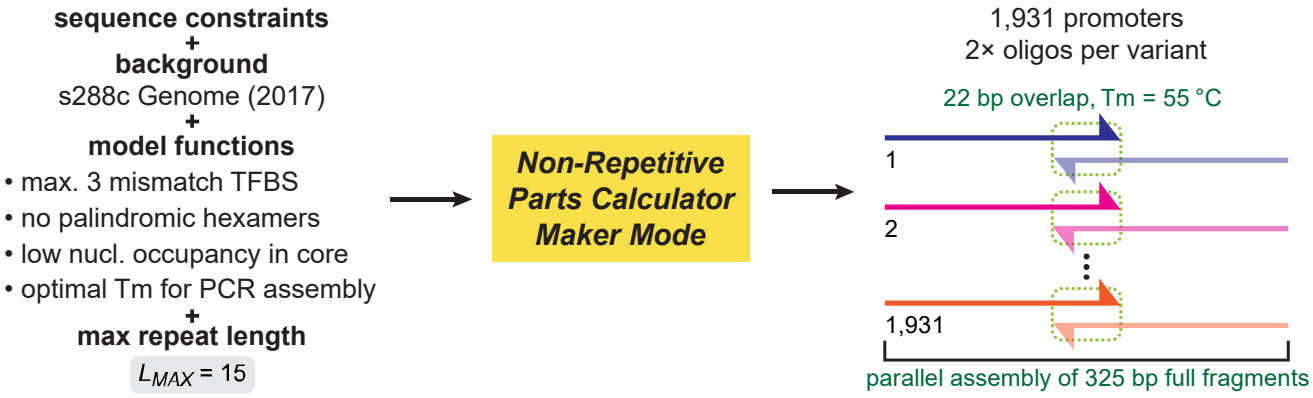
Supplementary Figure 10: Toolbox Size Distribution of *E. coli* Promoters with Increasing L_{MAX}

Up to 3,550 promoters can be used simultaneously without introducing any 11 base pair exact repeat, which is optimal for synthesizing all of these promoters together in a single DNA construct. All 4,350 promoters can be used concurrently without introducing any 20 base pair repeats, which greatly reduces the probability of homologous recombination in many bacterial species including *E. coli*.



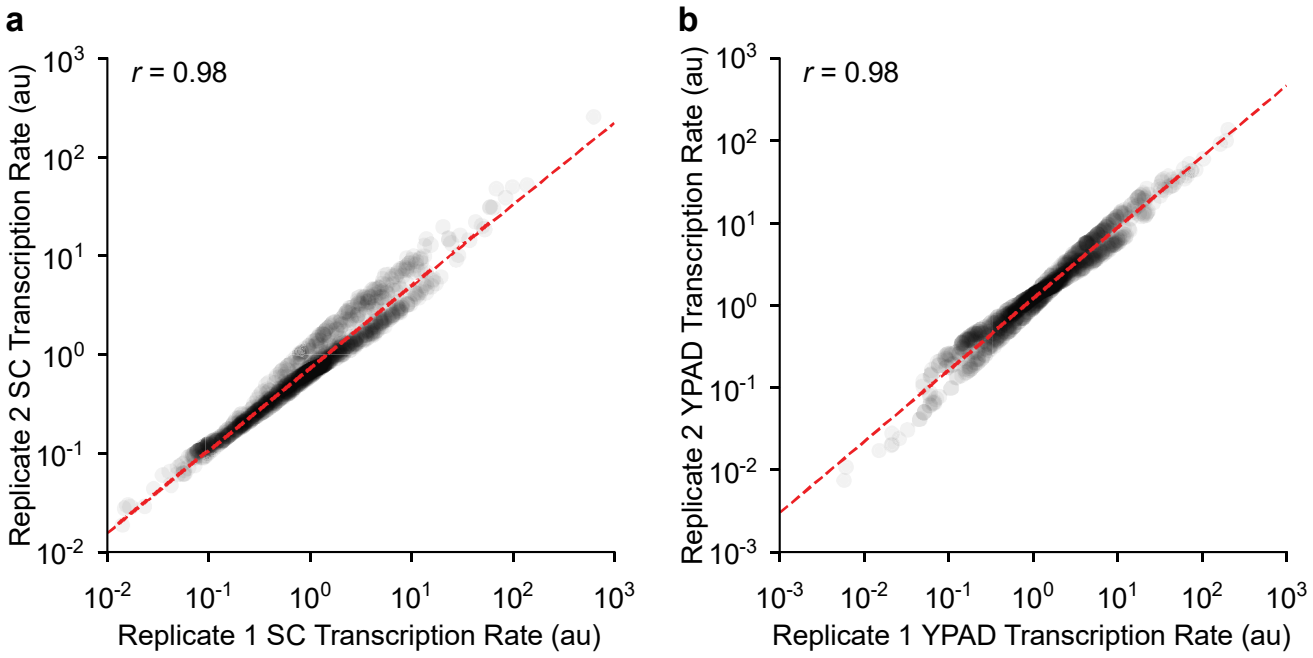
Supplementary Figure 11: Genetic Stability Assay using *E. coli* Promoters

(**a**) Cloning and characterization workflow for the genetic stability assay in triplicates. (**b**) Repetitive variants were initially red in the cloning step but subsequently mutated into white colonies during characterization. Non-repetitive variants showed consistent mRFP1 fluorescence in both stages. (**c**) After 36 hours of screening non-repetitive variants retained 10^4-fold more mRFP1 fluorescence than their non-repetitive counterparts. Interestingly, non-repetitive variants also showed higher sfGFP fluorescence, despite promoters with similar strengths before sfGFP and similar strength RBSs (TIR ~ 5000). (**d**) mRFP1 and sfGFP fluorescence levels remained correlated, suggesting transcriptional readthrough from the strong mRFP1 promoter, which could explain the higher sfGFP fluorescence in non-repetitive variants. (**e**) Flow cytometry measurements showed all repetitive variants had minimal RFP1 fluorescence (indistinguishable from white cells used as control). Non-repetitive variants showed mixed population peaks of varying levels of redness suggesting multi-modal mutation to overcome toxicity from mRFP1 over expression.



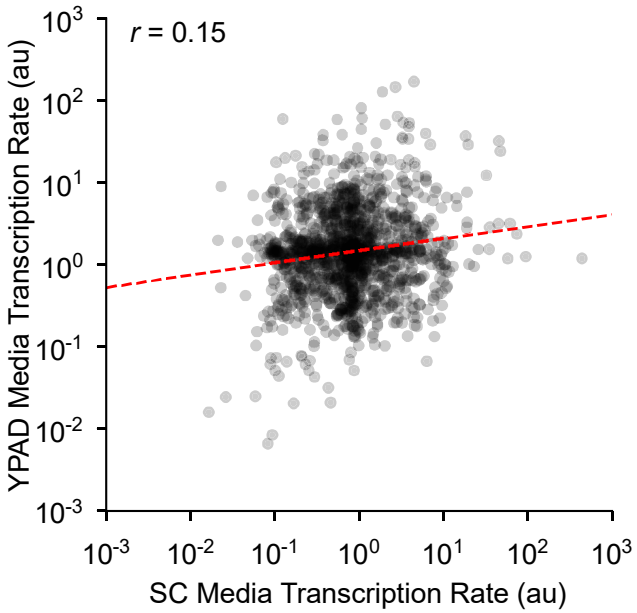
Supplementary Figure 12: Design of Non-Repetitive Yeast Promoters

A set of 13 unique sequence constraints were used along with the s288c 2017 genome assembly as background via an L_{MAX} of 15 bp. At most 3 hamming distance was allowed between the TF sites in the promoter library and the specified consensus for the TF sites. Low nucleosome occupancy was ensured using polyA and polyT nucleosome minimizer motifs, and an optimal Tm of 55 °C was ensured across a 22 bp overlap region for downstream promoter assembly.



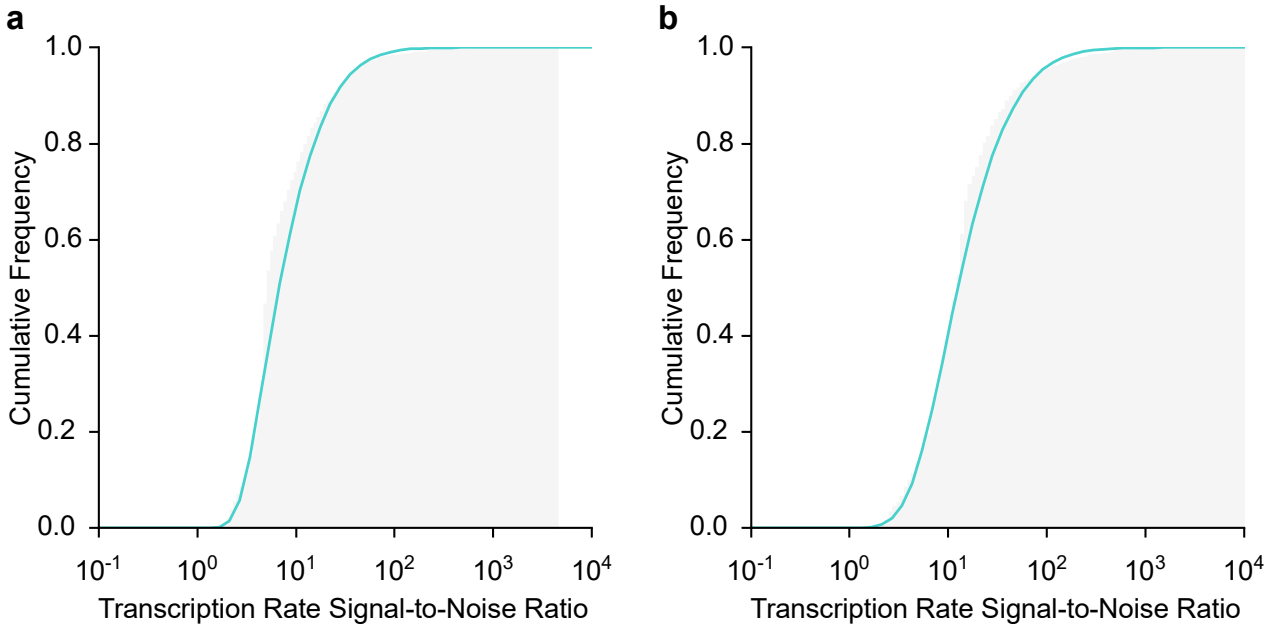
Supplementary Figure 13: Normalized Transcription Rate Correlation of Yeast Promoters in Both Media Conditions

Characterization of Yeast Promoters in (**a**) SC media (n = 1,668) and (**b**) YPAD media (n = 1,678) are highly correlated in normalized transcription rates (Pearson's *r* = 0.98) for both conditions.



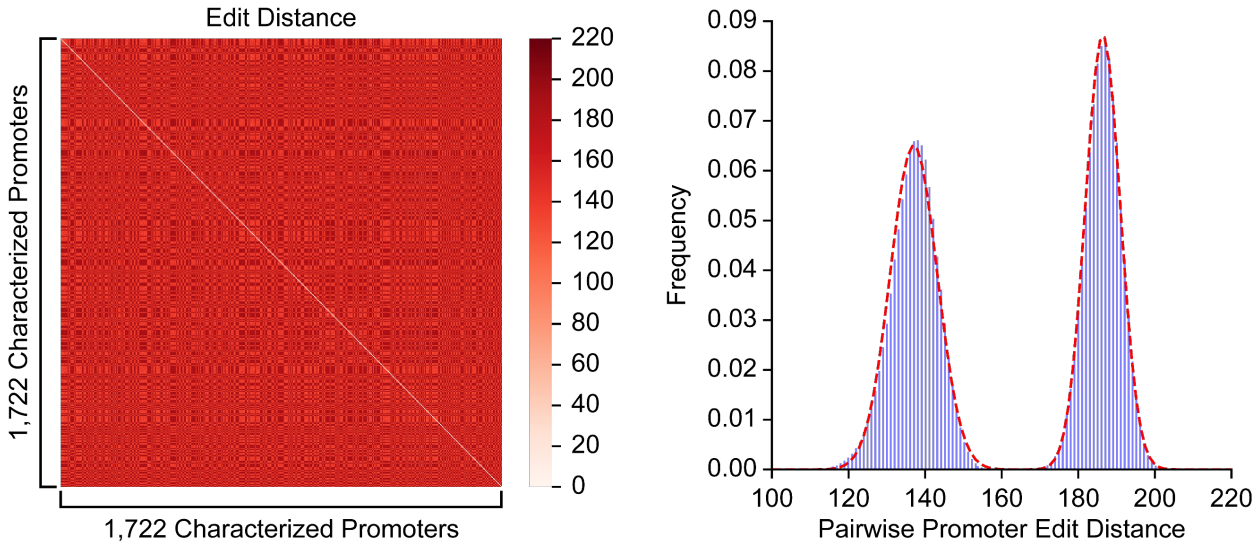
Supplementary Figure 14: Transcription Rate Correlation between Media for Characterized Yeast Promoters

Computed normalized transcription rates of 1,624 that were characterized and common to both media conditions show low correlation (Pearson's $r = 0.15$).



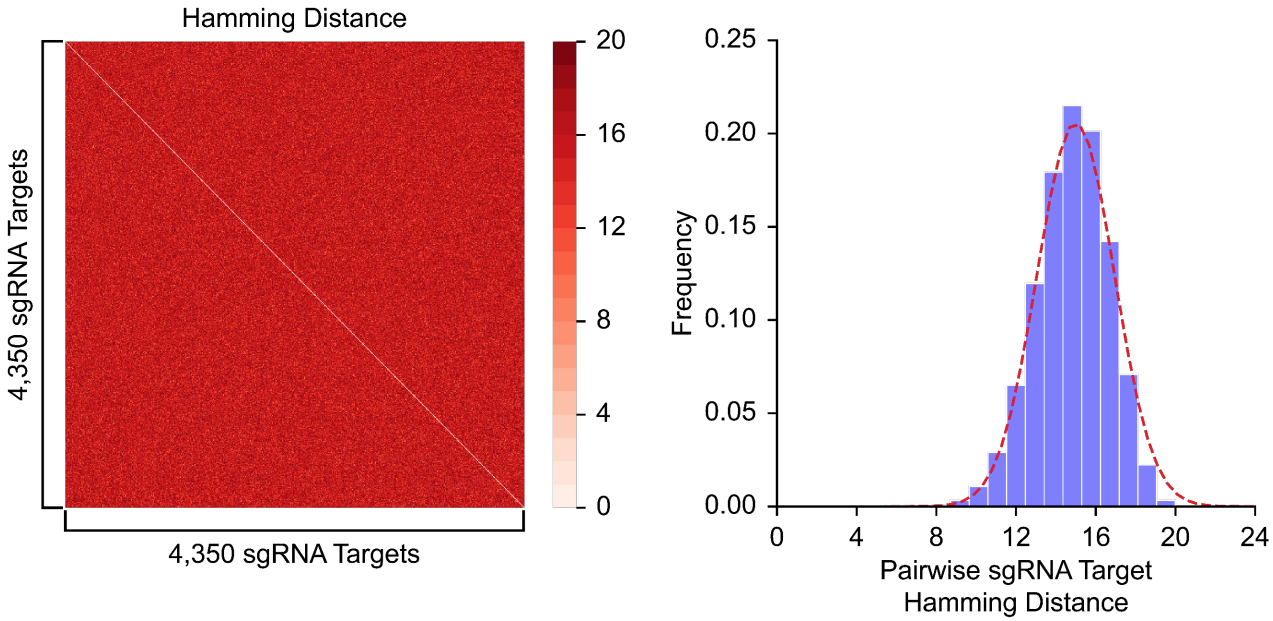
Supplementary Figure 15: Signal to Noise Ratio of Measured Transcription Rates of Yeast Promoters in SC and YPAD Media

(**a**) Over 55.99% of the promoters characterized in SC media satisfied the Rose criterion (SNR > 5), compared to (**b**) over 87.12% of the promoters characterized in YPAD media had SNR > 5.



Supplementary Figure 16: Edit Distance Matrix and Distribution for Yeast Promoters

Pairwise edit distance of all 1,722 promoters characterized in either SC or YPAD media show at minimum an edit distance of 100. The edit distance distribution is bi-modal due to distinct use of polyA and polyT motifs in designing the promoters. The expected edit distance between promoters with similar nucleosome minimizer motifs is 137.02 ± 6.13, while the expected distance between promoters with different minimizer motifs is 186.39 ± 4.57.

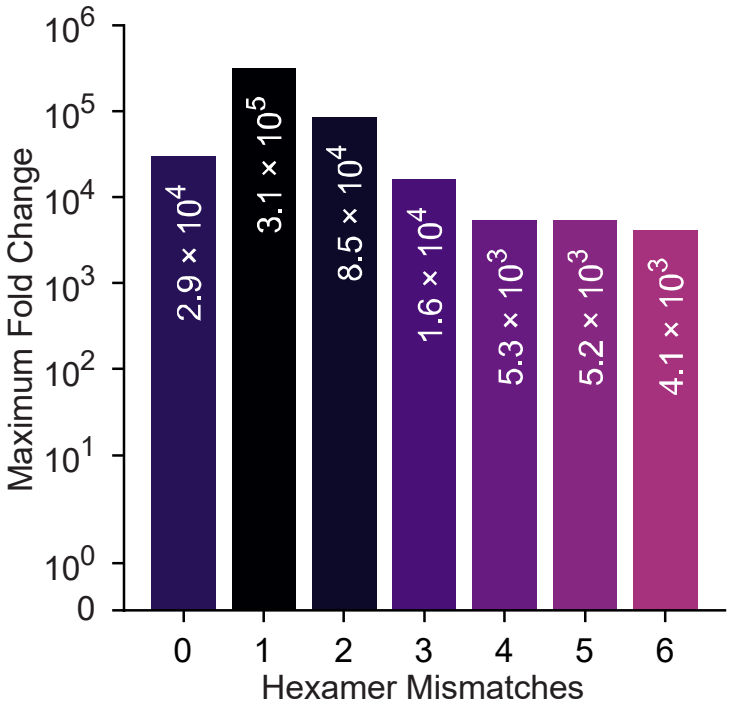


Supplementary Figure 17: sgRNA Target Sequence Pairwise Hamming Distance Distribution in *E. coli* Promoters

Combinatorial analysis showed that for every pair of 20 base pair non-repetitive sgRNA target sequences associated with our promoters, the expected hamming distance was 15 ± 1.94. This implies that in 95.45% of the cases, a single guide targeting a specific promoter will also trigger an off-target repression of another promoter in our library, if and only if 55.6 to 94.44% sequence mismatch in the off-target candidate is tolerated by CRISPRi.

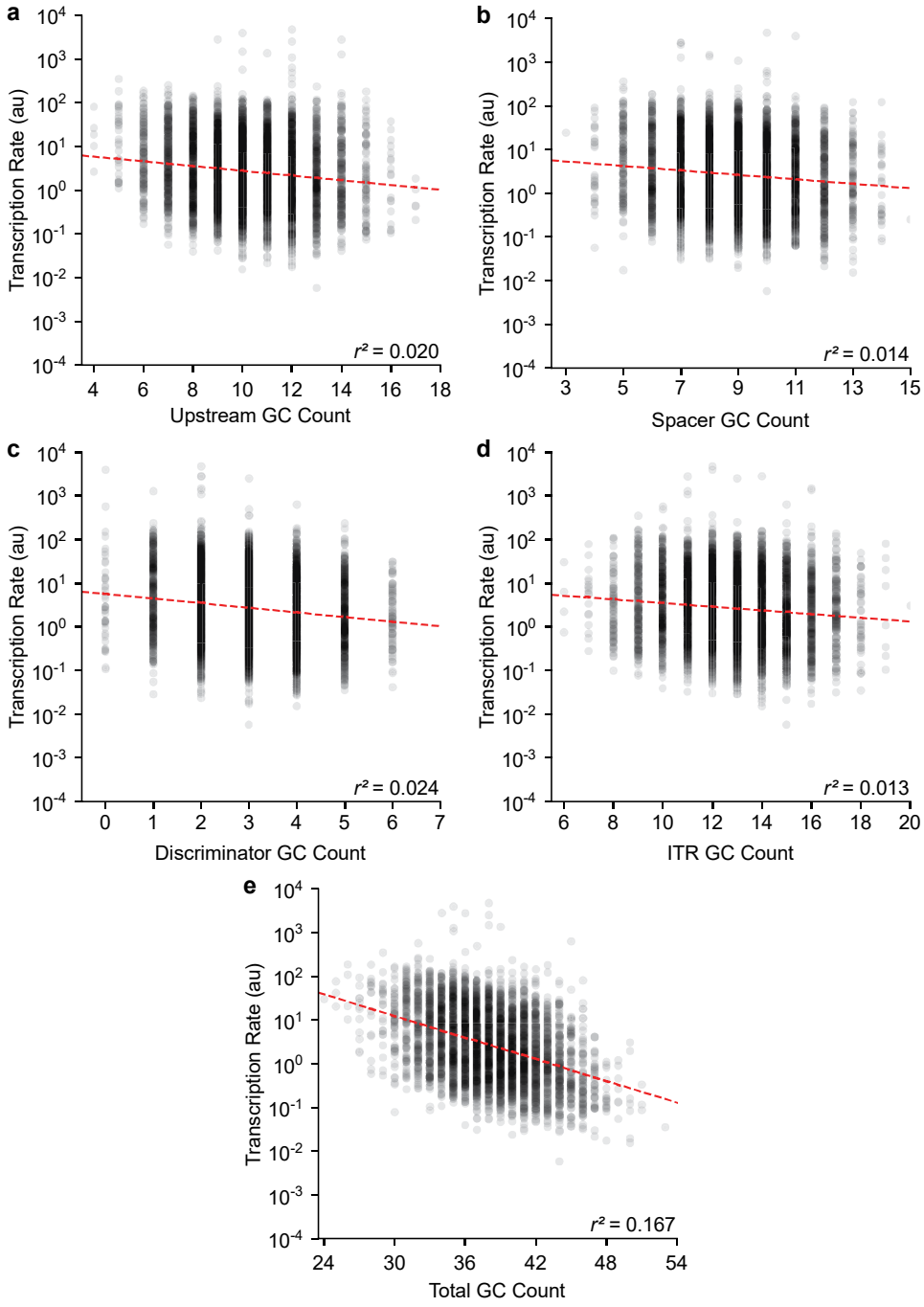
Supplementary Figure 18: Non-Repetitive *E. coli* Promoter Library WebLogo

Expected entropy at each position in the promoters is 0.20 bits, while the -35 hexamer, -10 hexamer and PAM entropies are 0.87 bits, 0.97 bits and 1.33 bits respectively. In particular, the -10 hexamer was mutated less frequently, with least mutations made in the sixth nucleotide of -10 hexamer. Entropy per base without the constant components is 0.0014 bits.



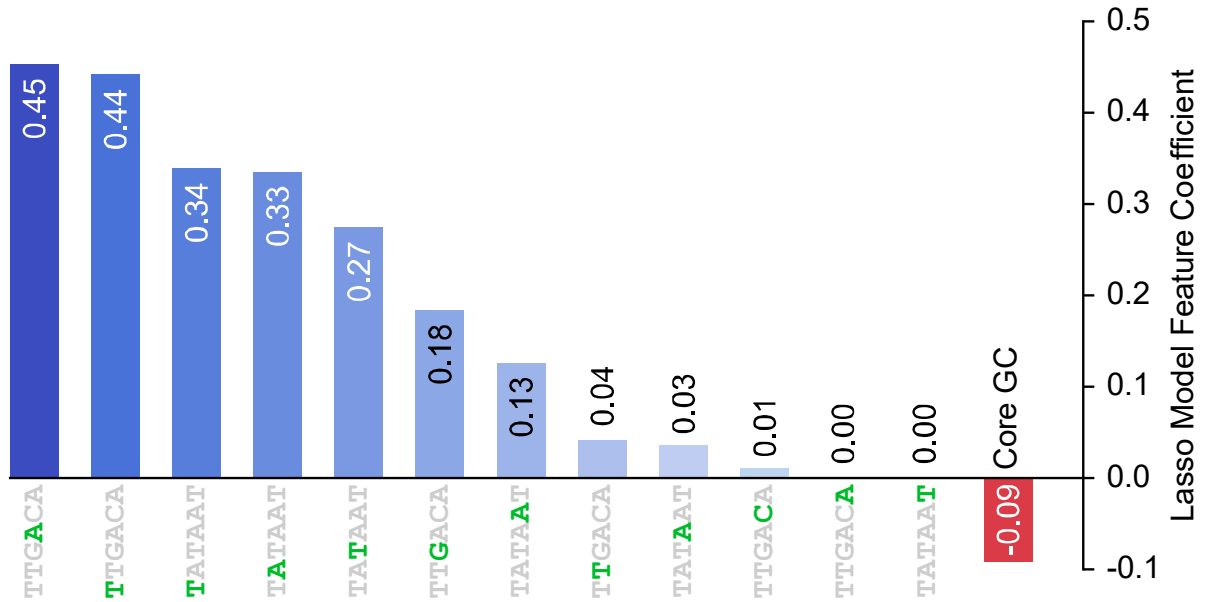
Supplementary Figure 19: Maximum Fold Change Across *E. coli* Toolbox 2 Promoters Grouped by Hexamer Mismatches

While most 0-mismatch promoters in the second toolbox have transcription rates higher than J23100, the maximum observed fold-change is still greater than 10,000-fold. There is a sharp increase in the fold-change from 0-mismatch promoters to 1-mismatch promoters, beyond which increasing the number of mismatches results in both steady decline in transcription rates and fold-change.



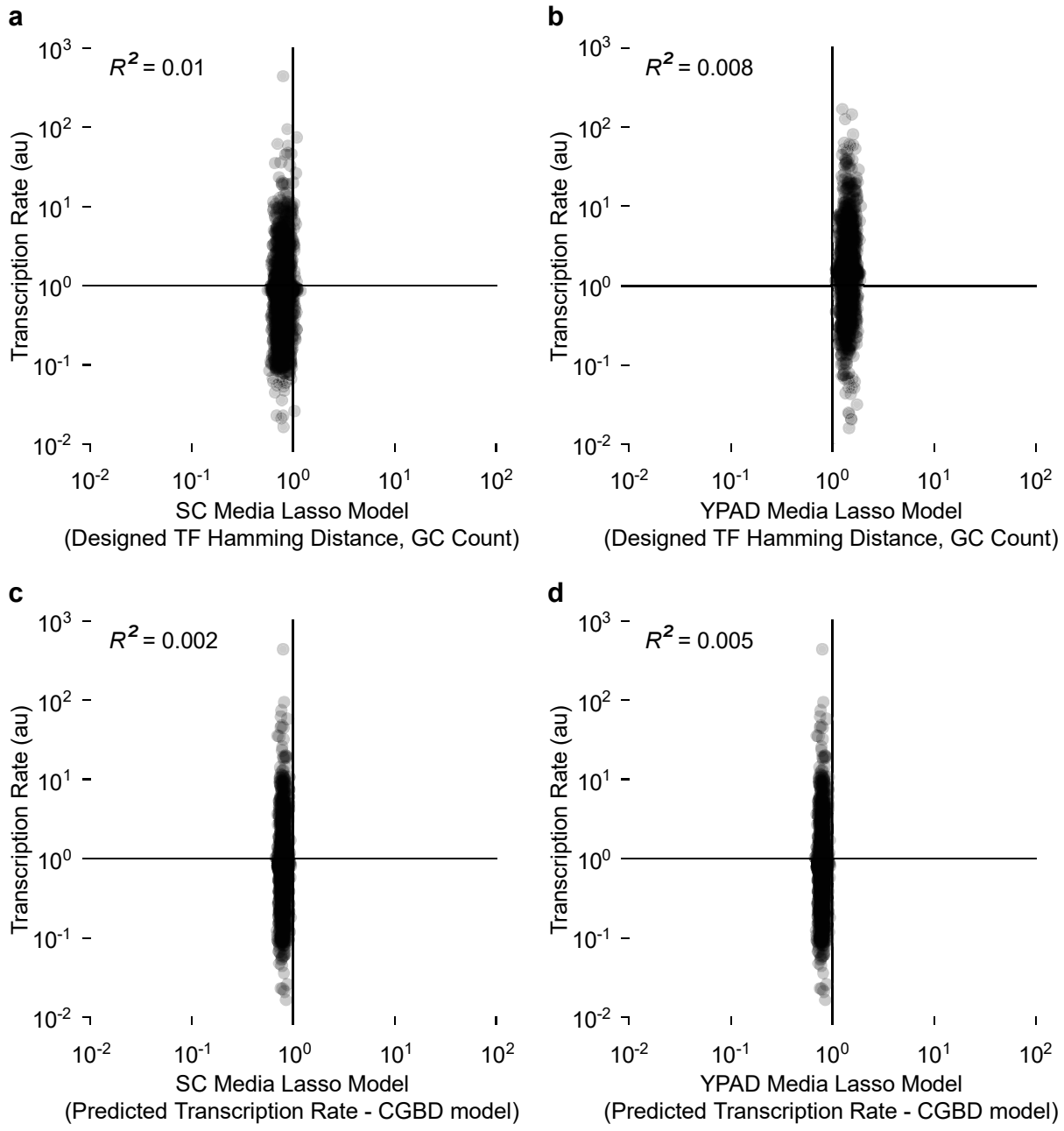
Supplementary Figure 20: Component-wise GC Content Correlation with Measured Transcription Rates of *E. coli* Promoters

GC content of individual components for all (n = 4,350) promoters, such as the (**a**) upstream insulator, (**b**) spacer between the -35 and -10 hexamers, (**c**) discriminator beyond the -10 hexamer and (**d**) the initially transcribed region consisting of the PAM and sgRNA Target site, show a negative correlation with measured transcription rates (Pearson's r = -0.14, -0.12, -0.15, and -0.11 respectively, p-value < 10^{-13}), but do not explain the observed variance in transcription rates adequately. In contrast, the transcription rate correlation with (**e**) total GC content of full promoter sequences is more negative (Pearson's r = -0.41, p value = 10^{-174}) and explains about 16.70% of the observed variance.



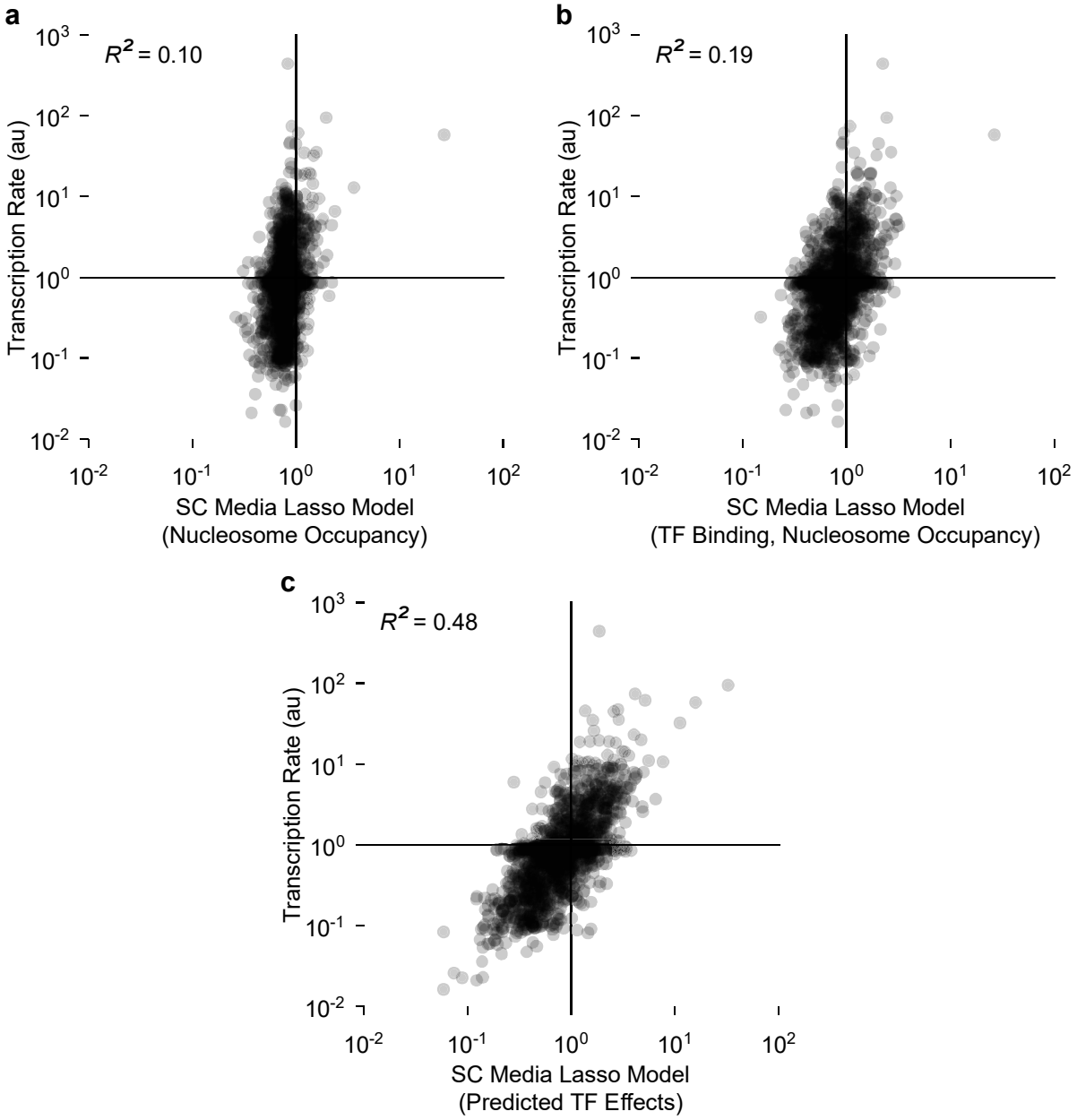
Supplementary Figure 21: Feature Coefficient Values in Trained Lasso Model for *E. coli* Promoter Library

The feature co-efficient values for conservation of bases in the hexamers, except the last two nucleotides in both -35 and -10 hexamers are positive (i.e. hexamers are mostly conserved in promoters with higher transcription rates). Conservation of the first three nucleotides in the -10 hexamer and the first and fourth nucleotide in the -35 hexamer have the highest importance. It is possible that the last nucleotide in -10 hexamer is deemed unimportant for transcription rate prediction because it was not mutated in more than 90% of the promoters in the full library. GC content of the full promoter is relatively less important overall but as a feature ranks higher than the importance of conserving the second and fifth nucleotide in the -35 hexamer and the fourth nucleotide in the -10 hexamer, and negatively affects transcription rates, as expected.



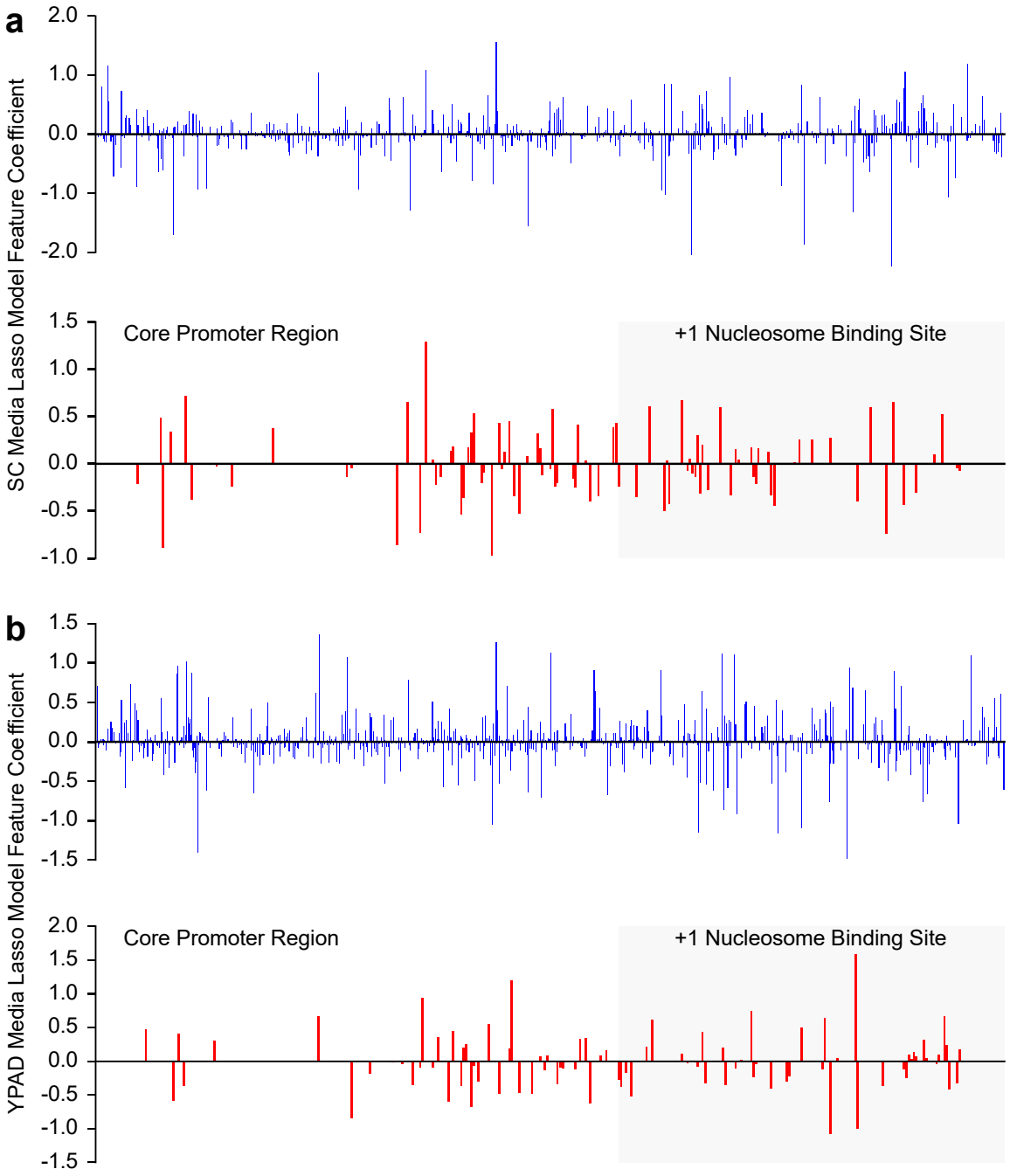
Supplementary Figure 22: Insufficient Lasso Models of Transcription Rates for Yeast Promoters

For all promoters in YPAD (n = 1,678) and SC media (n = 1,668) (**a**-**b**) categorical mismatches in designed TF binding sites, and GC content were insufficient in explaining the variance observed in measured transcription rates of promoters in both SC and YPAD media. Using (**c**-**d**) the predicted transcription rates from the CGBD model prediction alone was also insufficient in explaining the characterized transcription rates of the yeast promoters in both conditions.



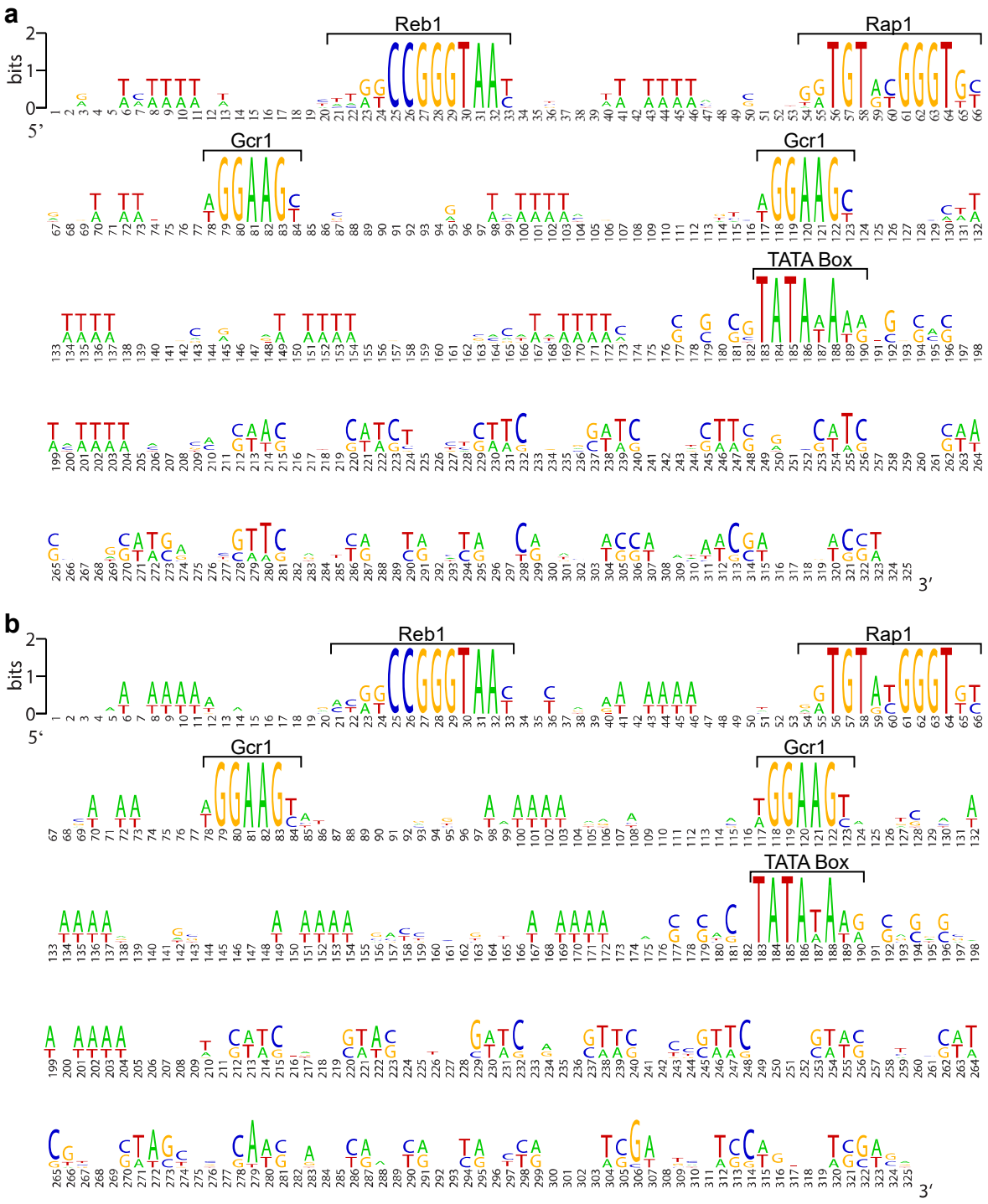
Supplementary Figure 23: Improved Lasso Models of Transcription Rates of Yeast Promoters

For all 1,668 promoters characterized in SC media (**a**) using Nucleosome Occupancy Probabilities as a feature, in SC media, explained 11% of the observed variance. (**b**) Augmenting the Lasso model with calculated TF binding probabilities furthered the variance explained to 19%. (**c**) Using the predicted TF effects bound to the yeast promoters from CGDB model as features resulted in a Lasso model that could explain up to 48% of the observed variance.



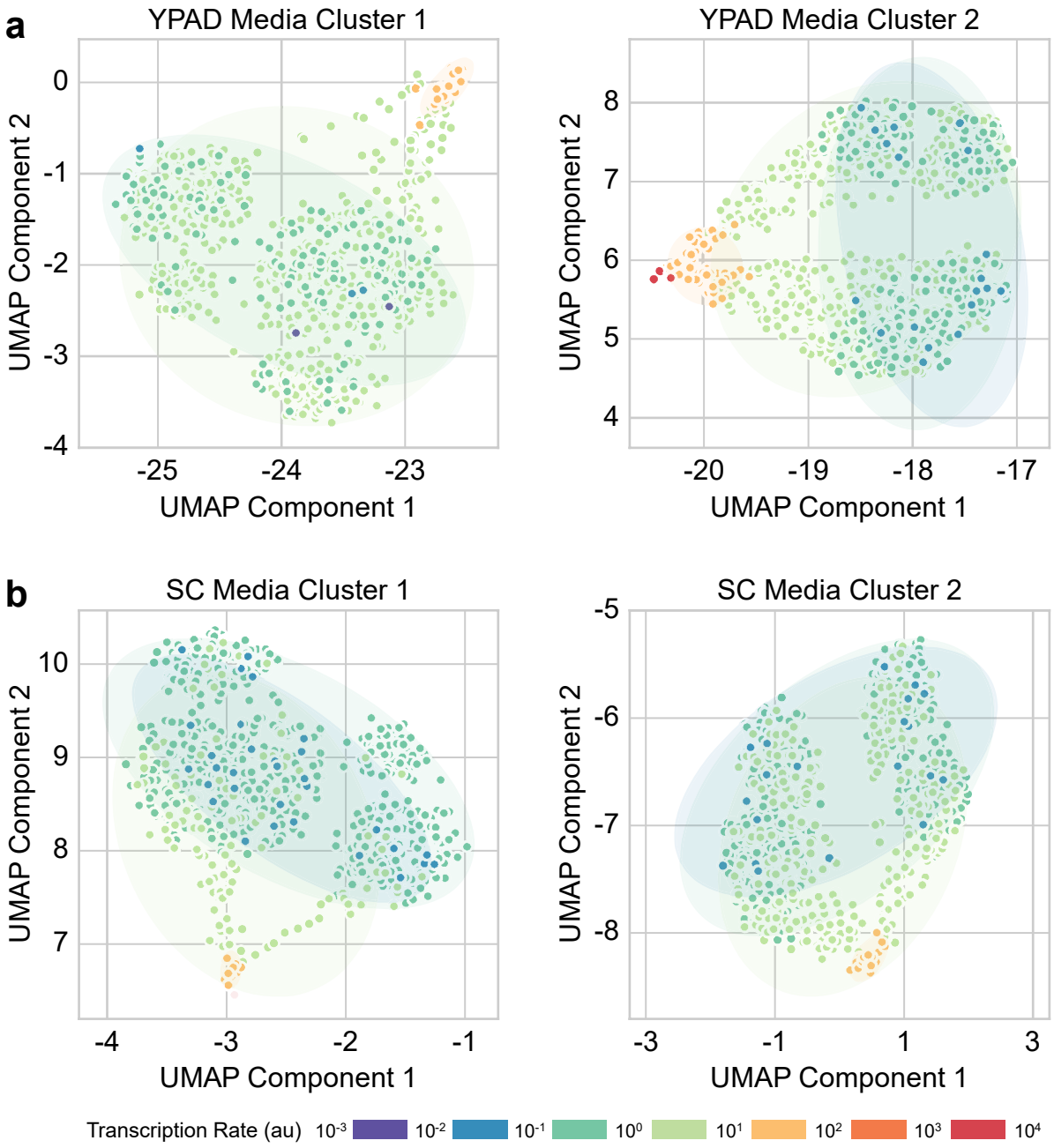
Supplementary Figure 24: Final Lasso Model Feature Coefficient Values for Yeast Promoters

(**a**) In SC media, approximately 44.10% of the contribution comes from significantly positive TF effect features, while positive nucleosome occupancy features contribute 9.02% to the model. Negative TF effects contribute 33.10% while negative nucleosome occupancy features 13.78%. (**b**) Comparatively, in YPAD media, the nucleosome occupancy features are mostly zeroed out in the core region, with more significantly positive features (positive correlation with transcription rate) occurring in the +1 NBS region. Positive TF effects contribute 41.62%, negative TF effects contribute 29.27%, positive nucleosome features contribute 12.43% and negative nucleosome features contribute 16.68%.



Supplementary Figure 25: WebLogos of Top and Bottom 10 Promoter Characterized in YPAD Media

As designed both (**a**) top 10 promoters and (**b**) bottom 10 promoters have highly conserved Reb1, Rap1, and Gcr1 sites and TATA boxes. However, the strong promoters are enriched in polyT nucleosome minimizer motifs, compared to polyA motifs enriched in weaker promoters.



Supplementary Figure 26: UMAP Analysis of Yeast Promoters

Two component UMAP analysis of promoters in (**a**) YPAD media (n = 1,678) and (**b**) SC media (n = 1,668) show two distinct clusters in both conditions, due to use of polyA and polyT motifs in the designed sequences – promoters with polyA motifs and promoters with polyT motifs were clustered together. Despite differences in sequence composition, both types of promoters in both conditions engender promoters with diverse activity, and each cluster share overlapping sub clusters similar to that observed in the *E. coli* promoters. There are few promoters that are strong, and usually such promoters have very similar composition, but there are many more possible weaker promoters and they are diverse.

Supplementary Table 1: NRP Calculator Finder Mode Toolbox Benchmarks

[**parts**] The total number of parts initially in the given toolbox.

[**nodes**], [**edges**] Total number of nodes and edges in the formulated repeat graph, after deleting parts with internal repeats.

[**FinderMode**] Total number of Non-Repetitive Parts (NRPs) recovered from the repeat graph and the wall time consumed using Non-Repetitive Parts Calculator Finder Mode.

[**Std.2-apx**] Total number of Non-Repetitive Parts recovered and wall time consumed when using the standard linear time 2-approximation minimum vertex cover algorithm on the repeat graph.

	parts	**nodes**	**edges**	**FinderMode**	**Std.2-apx**
E. coli Promoters	11,049	6,945	11,139,141	13 NRPs 255.35 sec	9 NRPs 325.50 sec
E. coli Terminators	647	469	2,660	154 NRPs 0.0265 sec	107 NRPs 0.0675 sec
S. cerevisiae Promoters	23	13	5	10 NRPs 0.0003 sec	9 NRPs 0.0004 sec
S. cerevisiae 5’ UTRs	5,032	3,894	1,792,494	104 NRPs 12.73 sec	80 NRPs 46.89 sec
S. cerevisiae 3’ UTRs	66	41	220	14 NRPs 0.0025 sec	12 NRPs 1.28 sec

Supplementary Table 2: Non-Repetitive Parts Design Constraint Examples

[**type**], [**constraint**] The part type or category, and sequence and/or structure constraints examples that may be input to NRP Calculator.

[**size**], [**time**] Total number parts and computation time (in seconds) incurred in designed toolboxes by NRP Calculator in Maker Mode using constraints.

type	**constraints**	**size**	**time**
Minimal Pol II Promoters	SSRCGCCTATAAANNNNNNNNNNNNNNNNNNNNNNNNNNTCAKTYNNNNNNNNNNNNNNNNNNNNNNNNRGWYV	512	4.95
Insulated Bacterial RBSs	NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNUAAGGAGGANNNNNAUG .(((((....)))))............xxxxxxxxxxxxxxxxx	749	129.02
Intrinsic Terminators	NNNSNSNSNSNSNSNSNSNSNNNNNNNNNNNNNNNNNNNNNNNNNNNNWWWWWWWWWWWWWWWWWW ...((((((((((((((((((.xxxx.)))))))))))))))))).....	32,896	303.55
Toehold RNA Switches	**NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN**AGGAGGANNNNNNAUGNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN xxxxxxxxxxxxxx(((((((((xxx((((((xxxxxxx))))))xxx)))))))))xxxxxx..... * Trigger RNA Sequence in Bold	1,026	241.54
Cas9 sgRNA handles	NNNNNNGNNNNNNNNNNNNNANGUNNNNNNNNAGGCNNGNCCGNNNNNNNNNNNNNNNNNNNN ((((((..((((....)))).....)))))))..((.....))xxxxxxx.((((....)))).	24,813	11001

Supplementary Table 3: Design Constrains used for generating Non-Repetitive *E. coli* Promoter Library

[**tbox**] Toolbox identifier, 1 = Toolbox 1, 2 = Toolbox 2 and 3 = Toolbox 3.

[**mx**] Number of mismatches in promoter hexamers as specified in the sequence constraint (0 to 12).

[**constraint**] The specified sequence constraint in IUPAC degenerate nucleotide code. Individual promoter hexamers and the PAM motifs are in bold.

[**parts**] Total number of parts generated using a particular sequence constraint.

tbox	mx	constraint	parts
1	0	NNNNNNNNNNNNNNNNNNNN**TTGACA**NNNNNNNNNNNNNNNNN**TATAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	800
2	0	NNNNNNNNNNNNNNNNNNNN**TTGACA**NNNNNNNNNNNNNNNNN**TATAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	500
	1	NNNNNNNNNNNNNNNNNNNN**TTGACB**NNNNNNNNNNNNNNNNN**TATAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	24
		NNNNNNNNNNNNNNNNNNNN**TTGADA**NNNNNNNNNNNNNNNNN**TATAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	15
		NNNNNNNNNNNNNNNNNNNN**TTGBCA**NNNNNNNNNNNNNNNNN**TATAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	25
		NNNNNNNNNNNNNNNNNNNN**TTGACA**NNNNNNNNNNNNNNNNN**TAVAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	76
		NNNNNNNNNNNNNNNNNNNN**TTGACA**NNNNNNNNNNNNNNNNN**TATBAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	100
		NNNNNNNNNNNNNNNNNNNN**TTGACA**NNNNNNNNNNNNNNNNN**TATABT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	100
		NNNNNNNNNNNNNNNNNNNN**VTGACA**NNNNNNNNNNNNNNNNN**TATAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	25
		NNNNNNNNNNNNNNNNNNNN**TVGACA**NNNNNNNNNNNNNNNNN**TATAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	25

	NNNNNNNNNNNNNNNNNNNN**TTHACA**NNNNNNNNNNNNNNNNN**TATAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	25
	NNNNNNNNNNNNNNNNNNNN**TTGACA**NNNNNNNNNNNNNNNNNN**VATAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	20
	NNNNNNNNNNNNNNNNNNNN**TTGACA**NNNNNNNNNNNNNNNNNN**TBTAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	35
	NNNNNNNNNNNNNNNNNNNN**TTGACA**NNNNNNNNNNNNNNNNNN**TATAAV**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	30
	NNNNNNNNNNNNNNNNNNNN**VVGACA**NNNNNNNNNNNNNNNNNN**TATAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	161
2	NNNNNNNNNNNNNNNNNNNN**TTGACA**NNNNNNNNNNNNNNNNNN**VBTAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	139
	NNNNNNNNNNNNNNNNNNNN**TTGACB**NNNNNNNNNNNNNNNNNN**VATAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	200
	NNNNNNNNNNNNNNNNNNNN**TTGBDA**NNNNNNNNNNNNNNNNNN**TAVAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	150
	NNNNNNNNNNNNNNNNNNNN**TTGBDA**NNNNNNNNNNNNNNNNNN**TATBAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	50
3	NNNNNNNNNNNNNNNNNNNN**TTGBDA**NNNNNNNNNNNNNNNNNN**TATABT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	150
	NNNNNNNNNNNNNNNNNNNN**TTHBCA**NNNNNNNNNNNNNNNNNN**TAVAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	150
	NNNNNNNNNNNNNNNNNNNN**TTGBDA**NNNNNNNNNNNNNNNNNN**TAVBAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	50
4	NNNNNNNNNNNNNNNNNNNN**TTGBDA**NNNNNNNNNNNNNNNNNN**TATBBT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	50
	NNNNNNNNNNNNNNNNNNNN**TTGBDB**NNNNNNNNNNNNNNNNNN**TAVAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	250

		NNNNNNNNNNNNNNNNNNNN**TTHBDA**NNNNNNNNNNNNNNNNN**TAVAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	150
	5	NNNNNNNNNNNNNNNNNNNN**TTGBDB**NNNNNNNNNNNNNNNNN**TAVBAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	350
		NNNNNNNNNNNNNNNNNNNN**TTGADB**NNNNNNNNNNNNNNNNN**TAVBBT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	50
		NNNNNNNNNNNNNNNNNNNN**VVHACA**NNNNNNNNNNNNNNNNN**VBTAAT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	75
		NNNNNNNNNNNNNNNNNNNN**TVHACA**NNNNNNNNNNNNNNNNN**VBTAAV**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	25
	6	NNNNNNNNNNNNNNNNNNNN**TTGBDB**NNNNNNNNNNNNNNNNN**TAVBBT**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	250
		NNNNNNNNNNNNNNNNNNNN**VVHACA**NNNNNNNNNNNNNNNNN**VBTAAV**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	125
		NNNNNNNNNNNNNNNNNNNN**VVHACA**NNNNNNNNNNNNNNNNN**VBTAAV**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	125
3	12	NNNNNNNNNNNNNNNNNNNN**SSMSGS**NNNNNNNNNNNNNNNNN**VBSSKV**NNNNNN**CCN**NNNNNNNNNNNNNNNNNNN	50

Supplementary Table 4: List of Hexamer-like Sequences Prevented in Upstream Elements, Spacers and ITR of *E. coli* Promoter Sequences

TTGACA	TTTACA	TTGCCA	TTGTCA	TTGATA	TTGAGA
TTGAAA	TTGACT	TTGACG	TTGACC	TTGAAT	
TATAAT	TAAAAT	TAGAAT	TACAAT	TATTAT	TATCAT
TATGAT	TATACT	TATATT	TATAGT	TGTCAA	ATTATA
TATAAC	TATAAG	TATAAA	TAATAT	TATATA	TATTTA
ATATAT	TAAATT	TTAAAT			

Supplementary Table 5: *E. coli* Promoter Library Read Counting Statistics

[**avgDNASeq**] Mean of DNA-Seq read count per promoter across DNA-Seq Replicates 1 and 2.

[**avgRNASeq**] Mean of RNA-Seq read count per promoter across RNA-Seq Replicates 1 and 2.

	avgDNASeq	**avgRNASeq**
Minimum Read Count	20.83	40.07
25th Percentile Read Count	2,573.50	981.45
Median Read Count	5,860.90	3,804.20
75th Percentile Read Count	13,121.22	23,252.53
Mean Read Count	12,234.52	22,636.73
Max Read Count	779,544.03	900,729.16

Supplementary Table 6: Comparative Measurements for all 20 Secondary Validated *E. coli* Promoters

[**pid**] Promoter identifier as listed in Supplementary Data 3, where they are prefixed with 'SLP2018-'.

[**plate_flu**] Computed geometric mean of mRFP fluorescence (au) from variant triplicates measured via TECAN Spark plate reader (n = 3).

[**plate_sd**] Associated geometric standard deviation of values measured via plate reader.

[**flow_flu**] Computed geometric mean of mRFP fluorescence (au) from variant triplicates measured via BD LSRFortessa cell analyzer (n = 3).

[**flow_sd**] Associated geometric standard deviation of triplicate fluorescence values measured via flow cytometry.

[**txrate**] Computed normalized transcription rates as described in Supplementary Data 3 (n = 2).

[**txrate_sd]** Associated arithmetic standard deviation of computed transcription rates.

[**rtqpcr**] Computed geometric mean of double delta CT values (au) from variant triplicates, normalized against the promoter SLP2018-2-1491 which has similar transcription rate 10× J23100 (n = 3).

[**rtqpcr_sd**] Associated geometric standard deviation of relative mRNA levels measured via RT-qpCR.

pid	plate_flu	plate_sd	flow_flu	flow_sd	txrate	txrate_sd	rtqpcr	rtqpcr_sd
2-101	99591.35	1.155235	116451	1.059196	1476.09	271.4698	20.58533	6.12417
2-342	14396.81	1.087066	18699.14	1.060978	1356.236	184.1213	NaN	NaN
2-162	47965.74	1.177529	57002.11	1.027976	263.8819	25.5311	14.58095	4.692563
1-573	107025.9	1.097649	134595.7	1.018085	250.3508	28.55964	NaN	NaN
2-931	45059.24	1.201266	48153.21	1.004352	77.46358	1.303809	6.024414	2.387041
2-582	44288.19	1.153161	38475.42	1.017627	68.43635	1.547175	NaN	NaN
2-807	19697.74	1.043107	14640.12	1.001419	19.64102	0.003633	16.8362	5.730509

2-778	60149.21	1.232624	76325.3	1.016723	17.14267	0.179126	NaN	NaN
2-1491	1978.925	1.026838	2478.846	1.080015	10.17321	0.364363	1	0.400403
2-1365	3619.535	1.156042	3885.958	1.064622	4.667053	0.010448	3.186244	0.999896
2-1326	42825.52	1.111804	40393.41	1.039671	3.985228	0.302706	NaN	NaN
2-1444	10442.78	1.125378	8746.103	1.044221	1.230024	0.006905	NaN	NaN
2-1405	5205.865	1.202177	4337.35	1.121003	1.150139	0.010039	0.638692	0.217136
2-1765	2515.317	1.075295	1998.949	1.098507	0.895562	0.048023	NaN	NaN
2-1577	1143.37	1.218099	1219.084	1.022587	0.846897	0.063464	0.266002	0.104698
2-1895	329.9048	1.173922	292.2834	1.111347	0.092904	0.003172	NaN	NaN
2-1689	134.0155	1.140954	250.2364	2.002361	0.01877	0.001585	0.061415	0.01877
2-2396	20.40849	1.225722	96.2692	2.085258	0.040512	0.001709	NaN	NaN
2-2586	88.58015	1.00932	76.95144	1.260645	0.035217	0.004879	0.046148	0.01397
3-42	8.562003	1.803308	5.549232	1.049031	0.015307	0.002924	NaN	NaN

Supplementary Table 7: Identified Mutations from Sanger Sequencing

A detailed report of the 15 observed mutations in the non-repetitive dual reporter system (FTV-GSA-NR) including the mutation type, its location relative to genetic landmarks, its position relative to the reference plasmids, the sequence mutation that occurred and clarifying details.

	Mutation Type	Location	Position	Mutation	Details
1	substitution	-10 hexamer	65	TATAAG	-
2	substitution	-10 hexamer	65	TATAAG	-
3	substitution	-10 hexamer	65	TATAAG	-
4	substitution	-10 hexamer	65	TATAAG	-
5	substitution	-10 hexamer	65	TATAAG	-
6	substitution	-35 hexamer	42	CTGACA	-
7	substitution	codon 41	346	TAA	Nonsense mutation (GAA to TAA)
8	insertion	codon 20	283	TCTT	Frameshift mutation (+1)
9	large insertion	mRFP1	234	Sequence[1]	IS1-like element IS1A family transposase
10	small deletion	promoter	64-81	-	TTATAATGATTTACCCCCA
11	small deletion	promoter	45-105	-	Sequence[2]
12	small deletion	-10 hexamer	64-81	-	TTATAATGATTTACCCCCA
13	large deletion	mRFP1	117-1429	-	See alignment, cause unclear
14	large deletion	mRFP1	153-1963	-	Deletion between repeat CTCTACAAATAAT
15	large insertion	mRFP1-sfGFP	37	-	Cryptic element added

[1]TTATTGATAGTGTTTTATGTTCAGATAATGCCCGATGACTTTGTCATGCAGCTCCACCGATTTTGAGAACGACAGCGACTT
CCGTCCCAGCCGTGCCAGGTGCTGCCTCAGATTCAGGTTATGCCGCTCAATTCGCTGCGTATATCGCTTGCTGATTACGT
GCAGCTTTCCCTTCAGGCGGGATTCATACAGCGGCCAGCCATCCGTCATCCATATCACCACGTCAAAGGGTGACAGCAG
GCTCATAAGACGCCCCAGCGTCGCCATAGTGCGTTCACCGAATACGTGCGCAACAACCGTCTTCCGGAGACTGTCATAC
GCGTAAAACAGCCAGCGCTGGCGCGATTTAGCCCCGACATAGCCCCACTGTTCGTCCATTTCCGCGCAGACGATGACGT
CACTGCCCGGCTGTATGCGCGAGGTTACCGACTGCGGCCTGAGTTTTTTAAGTGACGTAAAATCGTGTTGAGGCCAACGC
CCATAATGCGGGCTGTTGCCCGGCATCCAACGCCATTCATGGCCATATCAATGATTTTCTGGTGCGTACCGGGTTGAGAA
GCGGTGTAAGTGAACTGCAGTTGCCATGTTTT

[2]ACACTCCAAGAAGGTCCTTTTATAATGATTACCCCCATCGGGCATTTCCGGCGCTAGCTGT

Supplementary Table 8: List of Transcription Factor Binding Site and Nucleosome Depleting Motifs Used in Yeast Core Promoter Regions

CAATCCGGGTAAC	CAATCCGGGTAAC	AGGAAGC	TATAWAWR
HHRKCCGGGTAAY	DRTGTRYGGGTKY	WGGAAGY	
TTTTTTTTTY	YTTTYTTWYN	AAAAAAAAAR	RAAARAASRN
YTTTTTTTYN	YTTYYTTWYN	RAAAAAAARN	RAARRAASRN
RAARRAASRN	KTYTTTTYYB	AARAAAARAA	MARAAAARRA

Supplementary Table 9: Design Constrains used for generating Non-Repetitive Yeast Promoter Library

[**cid**] Identifier for individual sequence constraints used.

[**constraint**] The full sequence constraint in IUPAC code.

[**parts**] Total number of parts generated for the given sequence constraint.

cid	constraint	parts
1	NNNNTTTTTTTTTYNNNNNN**CAATCCGGGTAAC**NNNNNNTTTTTTTTTYNNNN**GGTGTATGGGTG** **T**NNTTTTTTNNN**AGGAAGC**NNNNNNNNNNNNTTTTTTTTTYNNNNNNNNNN**AGGAAGC**NNNNNNN TTTTTTTTTYNNNNNNNTTTTTTTTTYNNNNNNNNTTTTTTTTTYNSNSNSN**TATAWAWR**NSNSN SNTTTTTTTTTYNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	13
2	NNNNYTTTTTTTYNNNNNNN**CAATCCGGGTAAC**NNNNNNYTTTTTTTYNNNNN**GGTGTATGGGTG** **T**NNYTTTTYNNN**AGGAAGC**NNNNNNNNNNNNYTTTTTTTYNNNNNNNNNNN**AGGAAGC**NNNNNNN YTTTTTTTYNNNNNNNNYTTTTTTTYNNNNNNNNNYTTTTTTTYNNSNSNSN**TATAWAWR**NSNSN SNYTTTTTTTYNNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	16
3	NNNNTTYTTTTYTTNNNNNN**CAATCCGGGTAAC**NNNNNNTTYTTTTYTTNNNN**GGTGTATGGGTG** **T**NNTTYTTTNNN**AGGAAGC**NNNNNNNNNNNNTTYTTTTYTTNNNNNNNNNN**AGGAAGC**NNNNNNN TTYTTTTYTTNNNNNNNTTYTTTTYTTNNNNNNNNTTYTTTTYTTNSNSNSN**TATAWAWR**NSNSN SNTTYTTTTYTTNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	2
4	NNNNKTYTTTTYYBNNNNNN**CAATCCGGGTAAC**NNNNNNKTYTTTTYYBNNNN**GGTGTATGGGTG** **T**NNKTYTTYNNN**AGGAAGC**NNNNNNNNNNNNKTYTTTTYYBNNNNNNNNNN**AGGAAGC**NNNNNNN	15

	KTYTTTTYYBNNNNNNNKTYTTTTYYBNNNNNNNNKTYTTTTYYBNSNSNSN**TATAWAWR**NSNSN SNKTYTTTTYYBNNNNSWWSNNNNSWWSNNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	
5	NNNNAAAAAAAAARNNNNNN**CAATCCGGGTAAC**NNNNNNAAAAAAAAARNNNN**GGTGTATGGGTG** **T**NNAAAAAANNN**AGGAAGC**NNNNNNNNNNNNAAAAAAAAARNNNNNNNNNN**AGGAAGC**NNNNNNN AAAAAAAAARNNNNNNNAAAAAAAAARNNNNNNNNAAAAAAAAARNSNSNSN**TATAWAWR**NSNSN SNAAAAAAAAARNNNNSWWSNNNNSWWSNNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	16
6	NNNNTTTTTTTTTYNNNNNNN**HHRKCCGGGTAAY**NNNNNNTTTTTTTTTYNNNN**DRTGTRYGGGTK** **Y**NNTTTTTTNNN**WGGAAGY**NNNNNNNNNNNNTTTTTTTTTYNNNNNNNNNNN**WGGAAGY**NNNNNNN TTTTTTTTTYNNNNNNNTTTTTTTTTYNNNNNNNNTTTTTTTTTYNSNSNSN**TATAWAWR**NSNSN SNTTTTTTTTTYNNNNSWWSNNNNSWWSNNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	207
7	NNNNYTTTTTTTYNNNNNNN**HHRKCCGGGTAAY**NNNNNNYTTTTTTTYNNNNN**DRTGTRYGGGTK** **Y**NNYTTTTYNNN**WGGAAGY**NNNNNNNNNNNNYTTTTTTTYNNNNNNNNNNN**WGGAAGY**NNNNNNN YTTTTTTTYNNNNNNNNYTTTTTTTYNNNNNNNNNYTTTTTTTYNNSNSNSN**TATAWAWR**NSNSN SNYTTTTTTTYNNNNNSWWSNNNNSWWSNNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	223
8	NNNNTTYTTTTYTTNNNNNN**HHRKCCGGGTAAY**NNNNNNTTYTTTTYTTNNNN**DRTGTRYGGGTK** **Y**NNTTYTTTNNN**WGGAAGY**NNNNNNNNNNNNTTYTTTTYTTNNNNNNNNNN**WGGAAGY**NNNNNNN TTYTTTTYTTNNNNNNNTTYTTTTYTTNNNNNNNNTTYTTTTYTTNSNSNSN**TATAWAWR**NSNSN SNTTYTTTTYTTNNNNSWWSNNNNSWWSNNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	169
9	NNNNKTYTTTTYYBNNNNNN**HHRKCCGGGTAAY**NNNNNNKTYTTTTYYBNNNN**DRTGTRYGGGTK** **Y**NNKTYTTYNNN**WGGAAGY**NNNNNNNNNNNNKTYTTTTYYBNNNNNNNNNN**WGGAAGY**NNNNNNN KTYTTTTYYBNNNNNNNKTYTTTTYYBNNNNNNNNKTYTTTTYYBNSNSNSN**TATAWAWR**NSNSN SNKTYTTTTYYBNNNNSWWSNNNNSWWSNNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	393
10	NNNNAAAAAAAAARNNNNNN**HHRKCCGGGTAAY**NNNNNNAAAAAAAAARNNNN**DRTGTRYGGGTK** **Y**NNAAAAAANNN**WGGAAGY**NNNNNNNNNNNNAAAAAAAAARNNNNNNNNNN**WGGAAGY**NNNNNNN AAAAAAAAARNNNNNNNAAAAAAAAARNNNNNNNNAAAAAAAAARNSNSNSN**TATAWAWR**NSNSN SNAAAAAAAAARNNNNSWWSNNNNSWWSNNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	183
11	NNNNRAAAAAAARNNNNNNN**HHRKCCGGGTAAY**NNNNNNRAAAAAAARNNNNN**DRTGTRYGGGTK** **Y**NNRAAAARNNN**WGGAAGY**NNNNNNNNNNNNRAAAAAAARNNNNNNNNNNN**WGGAAGY**NNNNNNN RAAAAAAARNNNNNNNNRAAAAAAARNNNNNNNNNRAAAAAAARNNSNSNSN**TATAWAWR**NSNSN SNRAAAAAAARNNNNNSWWSNNNNSWWSNNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	205
12	NNNNAARAAAARAANNNNNN**HHRKCCGGGTAAY**NNNNNNAARAAAARAANNNN**DRTGTRYGGGTK** **Y**NNAARAAANNN**WGGAAGY**NNNNNNNNNNNNAARAAAARAANNNNNNNNNNN**WGGAAGY**NNNNNNN AARAAAARAANNNNNNNAARAAAARAANNNNNNNNAARAAAARAANSNSNSN**TATAWAWR**NSNSN SNAARAAAARAANNNNSWWSNNNNSWWSNNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	166
13	NNNNMARAAAARRANNNNNN**HHRKCCGGGTAAY**NNNNNNMARAAAARRANNNN**DRTGTRYGGGTK** **Y**NNMARAARNNN**WGGAAGY**NNNNNNNNNNNNMARAAAARRANNNNNNNNNNN**WGGAAGY**NNNNNNN MARAAAARRANNNNNNNMARAAAARRANNNNNNNNMARAAAARRANSNSNSN**TATAWAWR**NSNSN SNMARAAAARRANNNNSWWSNNNNSWWSNNNNNSWWSNNNNSWWSNNNNSWWSNNNNSWWSNNNN NSWWSNNNNSWWSNNNNSWWSNNNNYRNNYRNNYRNNYRNNNNWSSWNNNNWSSWNNNNWSSWNN	323

Supplementary Table 10: Yeast Promoter-Barcode Association Statistics

[**counts**] Read counts associated with barcode to promoter mapping.

	counts
Total Reads Sequenced	1,209,667.00
Reads Mapped	912,988.00
Unique Promoters Referenced	1,825.00
Mean Barcodes per Promoter	181.59
Unreferenced Promoters	91.00
Unique Barcodes Referenced	331,347.00
Mean Promoters per Barcode	1.00
Unreferenced Barcodes	60.00

Supplementary Table 11: SC and YPAD Media Read Counting Statistics

[**SCavgDNASeq**], [**YPADavgDNAseq**] Mean of DNA-Seq read count per promoter across two DNA-Seq Replicates in SC and YPAD media respectively.

[**SCavgRNASeq**], [**YPADavgRNAseq**] Mean of RNA-Seq read count per promoter across two RNA-Seq Replicates in SC and YPAD media respectively.

	SCavgDNASeq	**SCavgRNASeq**	**YPADavgDNASeq**	**YPADavgRNASeq**
Minimum Read Count	498.81	597.54	558.72	531.76
25th Percentile Read Count	563.89	644.32	611.98	565.62
Median Read Count	3,429.91	2,670.13	2,989.46	1,960.45
75th Percentile Read Count	7,809.12	7,865.00	7,924.82	7,407.07
Mean Read Count	7,963.64	10,991.01	10,207.32	7,722.76
Max Read Count	120,133.02	325,943.97	246,293.95	91,706.37

Supplementary Table 12: Primers Used in Experimental Characterization

[**pname**] The name of the primer as referenced in the Methods and Supplementary Information.

[**pseq**] The full DNA sequence associated with a primer name.

pname	pseq
PCR_FWD	GCAGCAATCGTGACAG
PCR_REV	TTGCGTGTCATCGGCG
DNA-Seq_FWD	TATCCTCGAGCGCGGAATTCCCTAGG
DNA-Seq_REV (NGS reverse complement)	GCTGATCTCTCGTCGGTATGAA
RNA-Seq_FWD (NGS)	TTCATACCGACGAGAGATCAGC
RNA-Seq_REV	GCCATTTTTTACCTCCTTATTTTTTTT
Link_FWD	TTGACTGTAATATCTGGAGCTGCGGAATACAAG
Link_MID / Seq_FWD	GACATCGCTGACTCCCAC
Link_REV	CCAGTGAATAATTCTTCACCTTTAGAC
Seq_REV	ATTGGGACAACACCAGTGAATAATTCTTCACCTTTAGAC
FTVF2	GCTGAACGGTCTGGTTATAGG
GFPRV	CCATAGGTCAGGGTGGTAACCA