Machine Learning guided workflow for Ribosome Binding Site engineering

Zhang M.[1,2], Holowko M. B.[3], Hayman Zumpe H.[3], and Ong, C. S.[1,2,4]

[1]Machine Learning and Artificial Intelligence Future Science Platform, CSIRO
[2]Australian National University
[3]CSIRO Synthetic Biology Future Science Platform, CSIRO Land and Water
[4]Data61, CSIRO

May 12, 2021

Abstract

Fine control of gene expression can be achieved through engineering transcriptional and translation control elements, including the Ribosome Binding Site (RBS). Unfortunately, RBSs are not understood at the level of finesse required for reliable design. To address this problem, we have created a machine learning (ML) enabled workflow for the design of bacterial RBSs. We used Gaussian Process Regression for prediction and the Upper Confidence Bound-based Bandit algorithm for recommendation of genetic designs to be tested in vitro. We have integrated the ML algorithms with laboratory automation and high-throughput processes, creating a robust workflow for the design of custom RBSs. Using our workflow, we generated a novel library of diverse RBSs with a wide range of expression levels. Notably, a high number of these sites demonstrate translation initiation rates equalling or exceeding the currently known strong RBSs. Additionally, this work elucidated some important RBS design guidelines.

1 Introduction

One of the main tenets of synthetic biology is design, evaluation and standardisation of genetic parts [4, 6, 33]. This is usually done in terms of the Design-Build-Test-Learn (DBTL) cycle, where the given genetic part or organism are continually improved by going through a number of turns of the said cycle. This normally involves designing the DNA sequence in Computer Aided Design (CAD) software and then physically testing it in a laboratory. Additionally, computer modelling and prediction of part behaviour based on the designed DNA sequence or design of DNA sequence based on expected function can be used[37, 23]. Most of these models are based on either the thermodynamic properties of the involved molecules (DNA, RNA, proteins, etc.) or empirically obtained values describing a relevant to a given design property, like Translation Initiation Rate (TIR) in the case of Ribosome Binding Sites (RBS) [35, 7, 26]. However, de-novo design of small genetic elements is still challenging due to unknown relationships between sequence and performance of such elements. This means that many designers have to rely on known and characterised parts that may not be optimal for their constructs. The problem with this approach is that such part libraries can also be unreliable due to poor reliability of methods used to obtain them.
The biggest limitation for the DBTL approach currently is the Learn part of the cycle - there is very limited access to methods and software that can improve and understand designs based on the experimental results. For example, according to Reeve *et al.* there are three main RBS calculators, all predicting the TRI based on the thermodynamic properties of the RBS and the ribosome [30, 22, 28]. Reported predictions from all of these models are relatively good ($R^2 > 0.8$), but they come with a number of caveats: i) they rely on calculations of free energies that can be hard to estimate with high precision ii) in general, one of the best ways to improve the models' accuracy is by increasing the number of phenomenons taken into account, but this can lead to paradoxically decreased model accuracy due to accumulation of errors [11] and iii) by using deterministic coefficients to calculate energies one disregards often

stochastic nature of processes in the cells which again increases perceived prediction error [13]. There are also sources showing that binding energy calculations may be poor predictors of RBS strength [16, 29] . This is reinforced by studies suggesting that RNA secondary structure is potentially a more important feature in TIR determination [9, 11] .
Synthetic biology is currently going through a phase of exponential increase in volume of data produced during experiments [12]. New experimental methods heavily relying on advances in automation and microfludics allow unprecedented precision and throughput in data generation. These new data-sets can be combined with data reliant machine learning algorithms to generate new models and predictors for use in synthetic biology, vastly improving the DBTL cycle's performance [5]. In the past few years there was a significant uptake of Machine Learning based approaches in synthetic biology [17]. Jervis *et al.* used support vector machine and neural network to optimise production of monoterpenoid in *Esherichia coli* [15]. Similarly, Costello *et al.* have used a number of machine learning approaches to analyse time-series multiomics data to predict metabolic pathway behaviour [8]. There were also successful attempts at using deep learning techniques for analysis of big data-sets [1, 2]. Machine learning has been also used for prediction in proteins [36]. However, the use of machine learning in synthetic biology is still in its infancy and will require additional research to show its full potential.
Here we present how machine learning algorithms can be used as part of the DBTL cycle to predict (Learn) and recommend (Design) variants of RBS with goal of optimisation of associated protein level expression. RBS being one of the key genetic elements controlling protein expression and at the same time having a relatively short sequence is a perfect target for establishing workflows that can be later translated to more complicated systems. In this work We have used Gaussian Process Regression and Upper Confidence Bound multi-armed Bandits algorithms for prediction and recommendation respectively to analyse and optimise the initiation rates of the designed RBS [10, 24]. Our overall experimental goal was to maximise the Translation Initiation Rate (TIR) by identifying the set of RBS sequences with top TIR scores while minimising the number of DBTL cycle turns that we had to do. We did this by designing a sequential experimental workflow, where designs in the zeroth round where randomised RBS sequences designed to explore the experimental space and some preliminary machine learning recommended designs based on literature. In the subsequent rounds, designs where recommended by the algorithm based on the data obtained in the previous rounds. The designs were then physically constructed in batches of 90 to fit our automated process (see **Methods** section). After constructing, the plasmids harbouring the new genetic devices were tested in microplate reader. The results were then fed back to the algorithm for it to recommend the next round of designs. This way, we were able to build an extensive, reliable library of novel RBSs with diverse sequences. At the same time we were able to discover new RBS sequences with very high TIRs.

2 Results

Our experiments followed the Design-Build-Test-Learn cycle methodology. Figure 1 shows our DBTL cycle with the machine learning process emphasised. Principally, machine learning is used in the LEARN and DESIGN phase and the other two are driven by choices made by human researchers. In LEARN phase we use the Gaussian Process regression algorithm to predict the TIR of different RBS sequences and in the DESIGN phase we use the Upper Confidence Bound multi-armed Bandit algorithm to recommend sequences to be Built and Tested next.

2.1 Performance of the recommendation algorithm

In our genetic design, the investigated RBS controls expression of the Green Fluorescent Protein (GFP) in its mut3b variant. By controlling expression of a fluorescent protein with the RBS we can quickly assess the perceived relative TIR by measuring fluorescence of cells harbouring plasmid with the device over time. Finally, the mRNA is transcribed from an IPTG-inducible promoter pLlacO-1. By making the whole device inducible we can synchronise the start of the expression of the GFP in all the cultures by inducing them at the same time with addition of IPTG.

In *E. coli* the RBS is usually located in the 20 bases upstream of the start codon. Additionally, there is a consensus RBS core sequence called the Shine-Dalgarno sequence, which in *E. coli* is AGGAGG. Here, we put that 20 bp long sequence into focus with main emphasis being put on the 6bp core region (see detail in Figure 1).

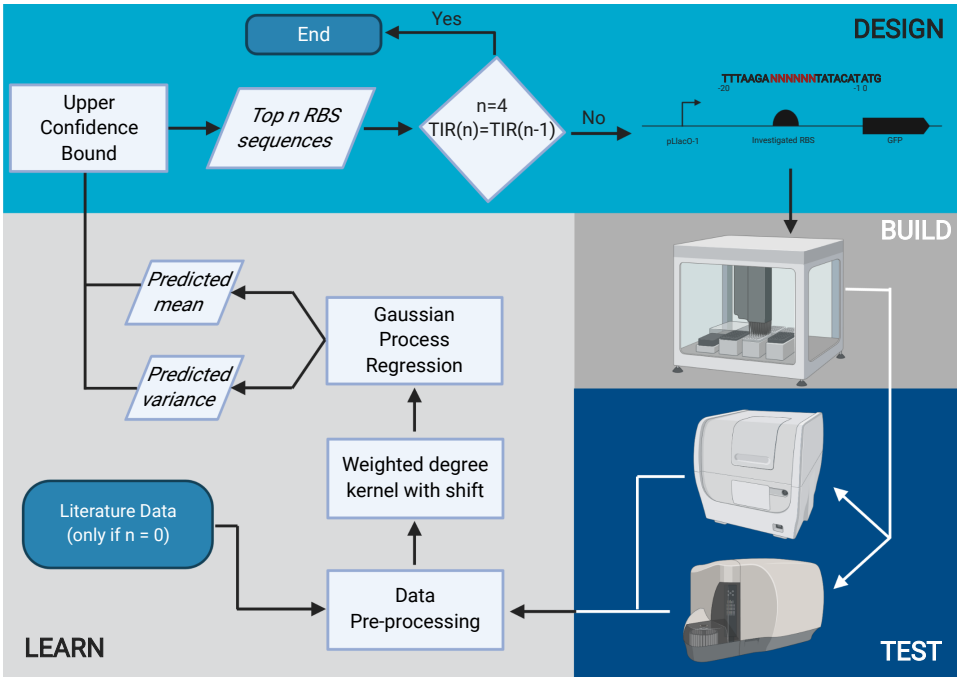


Figure 1: **Flowchart of machine learning based experimental design.** The RBS design is recommended by the Upper Confidence Bound Bandit algorithm. After generating the recommendations the RBS are built and tested using automated laboratory methods allowing for rapid construction and testing at scale. Finally, the obtained results are fed back to the prediction algorithm in the learn phase.

Our template RBS sequence is 20 bps long with the sequence TTTAAGAAGGAGATATACA. This sequence is a known high TIR RBS that comes with the pBb series plasmids [19]. Since this is the sequence against which new RBS sequenced will be benchmarked we will refer to this sequence as the benchmark sequence from now on. In our design we focus on randomising of the core at positions -8 to -13 (relative to the start codon of the GFP) nucleotides of the RBS and fix others to be the same as the consensus sequence, i.e. TTTAAGA + NNNNNN + TATACAT. We have experimentally confirmed that changing the core sequence is statistically more impactful on TIR than changes made outside of it (see Supplementary materials). Since for each of the 6 position there are 4 possibilities: A, C, G, T the total experimental (variant) space is $4^6 = 4096$.

The design recommendations were made using the Multi-armed Bandit algorithm. In short, the this algorithm is a stochastic method of probing of the experimental space. This algorithm aims at maximising the reward (output) from testing a limited number of instances from a big pool which cannot be wholly tested due to limited resources (time, computational power, capital). In our case we use the Upper Confidence Bound version of the algorithm, which focuses its recommendations on sequences that should give highest TIR based on the probabilities computed by the prediction algorithm (GPR). Another feature of the bandits algorithms is that it balances two approaches: exploration and exploitation. Exploration makes the algorithm recommend designs that will improve the predictions, whereas exploitation will recommend designs that focus on delivering the most efficient design the fastest. The two approaches can be controlled with the β parameter. We have decided that in the first iterations of the cycle it would be beneficial to skew the algorithm towards the exploration with exploitation taking increasing role in later iterations. One thing of note is that the bandit algorithm is stochastic, that is it exploits the probabilities of given event occurring (in this case RBS having a specific TIR). As such, it pairs naturally with our prediction algorithm, the Gaussian Process, which provides probability based function regression.

To generate the dataset that the algorithm could learn from we have decided to characterise a total of 450 RBS variants, which constitutes a little over 10% of the whole experimental space. To fit into our automated workflow, we have divided the 450 variants into batches of 90. In the zeroth round we have tested two batches of designs, for

total of 180 variants split as below:

1. BPS-NC and BPS-C group - 60 RBS sequences which are subsequent single nucleotide changes of all 20 nucleotides of the original, consensus sequence. This batch is designed to show us influence of such single nucleotide changes on the overall performance of the RBS and the potential impact of changes made beyond the core part.

2. UNI - 30 RBS sequences that were uniformly randomised, i.e. equal probability of choosing either nucleotide for each position.

3. PPM - 30 RBS sequences randomised based on the position probability matrix (PPM) generated from all the naturally occurring RBS sequences in *E. coli* genome [34]

4. Bandit0 - 60 RBS sequences recommended by our implementation of recommendation algorithm based on a data set obtained from literature [14], which contains 113 non-repeated records for 56 unique RBS sequences with the respective TIR. This data set has been used due to perceived similarity of its goal to the one of this work - prediction of TIR based on phenotypic output.

In the subsequent 3 rounds, all 90 designs were generated using our machine learning algorithm based on the data obtained from the previous rounds (these groups are called Bandit 1 to 3 respectively).

Figure 2 shows the results for all the examined groups. Different randomly generated groups were run together with Bandit 0 (literature data taught) group in Round 0. Subsequently, the following rounds from 1 to 3 consisted of only Bandit recommended designs. All the randomly generated groups have performed worse then our benchmark sequence in terms of TIR. Similarly, the Bandit 0 group performed poorly, due to being taught using approximate data set. However, starting from round 1, where the bandit algorithm was fed data from the round 0 the results become much better, with a number of sequences that perform similarly and in one case better than the benchmark (by 8%). In round 2 we have observed further improvement by getting more sequences that showed TIR on levels similar to our benchmark sequence. Finally, in round 3 the algorithm identified two sequences that were 34% and 15% stronger than the benchmark sequence.

2.2 Automated RBS variant construction

For the machine learning to work effectively the analysed data set needs to show two qualities: high relative volume and high quality of data. These two elements don't have specific definitions, but in general the data set has to be going into at least hundreds data points and have to cover 5-10% of that space. Maciej: I think it varies from case to case. We need to either change it or add reference. And since quality of the obtained data has a direct and strong correlation with quality of the predictions and in effect - recommendations, one need to ensure that the obtained results represent the real value as close as possible.
To help us obtain reliable and reproducible results we have employed automation-heavy workflow for our experiments. This way we were able to eliminate a big part of sample-to-sample variation as well as human-introduced variation. Additionally, performing all the procedures directly in 96-well microplate format enabled us to significantly cut down time to prepare our variants.
In short, the genetic variations of the RBS were introduced to the plasmids with combination of PCR and isothermal assembly. The plasmids were then transformed and the resulting transformants were tested using microplate reader. Vast majority of reactions were prepared using liquid handling equipment. Similarly, colony picking was done by an automated colony picker.

2.3 Prediction of RBS performance

Our recommendation algorithm selects new designs based on our prediction algorithm. This algorithm creates a model which takes the RBS sequences as input and predicts the TIR values with uncertainty level about the prediction, based on the experimental data. For this study, we have used the Gaussian Process regression (GPR) as our prediction method. GPR is a Bayesian approach and has been widely used for experimental design [32, 27]. Such a

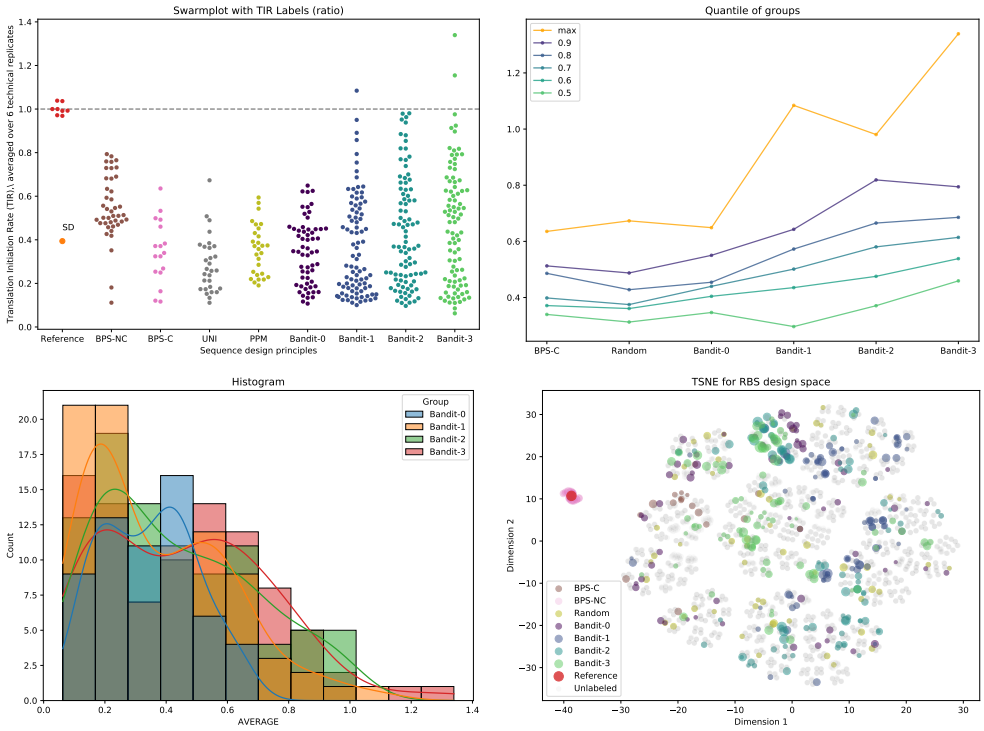


Figure 2: **TIRs of RBS groups examined in this study.** A) Swarm plot showing the obtained TIRs divided into RBS groups. BPS-NC: base-by-base changes in the non-core region. BPS-C: base-by-base changes in the core region. UNI: Randomly generated sequences with uniform distribution. PPM: Randomly generated sequences with distribution following the PPM for all natural RBS in *E. coli*. Bandit0/1/2/3 - Bandit algorithm generated results for Round 0, 1, 2 and 3 respectively. SD - Shine-Dalgarno sequence. B) Plot showing highest TIR obtained in a given quantile of results divided into groups as in A), save the random groups which were shown together due to similar distributions.

stochastic prediction method fits biological processes naturally, since they are highly stochastic as well. The explicit representation of model uncertainty provides further guide for efficient searching through large experimental space of possible sequences.

A crucial ingredient in a Gaussian Process predictor [24] is the kernel (covariance) function, which captures the similarity between data point, in our case RBS sequences. Specifically, kernel function implicitly embeds RBS sequences into high-dimensional feature space which makes the regression process easier.

For Bandit designs in Round 0, since we only had access to limited number of data points from literature, we chose to use one of the basic string kernels, the *spectrum kernel* [20] to process the core 6bp and dot product kernel [24] (with one-hot embedding) to process the 7bp flanking sequences both upstream and downstream of the core sequence. For subsequent rounds, we used the a more powerful kernel function, the *weighted degree kernel with shift* (wds) [25], which has been shown to have a strong performance in various prediction tasks [3].

Figure 3 shows how our algorithm performed in terms of predictions in each round. As expected, the predictions in Round 0 were poor due to use of approximated data. The predictions improved for the subsequent rounds, from R^2 of 0.065 for round 0 to R^2 of 0.27 for round 3. Similarly, the Spearman correlation coefficient rose from 0.27 for Round 0 to 0.48 for Round 3. Interestingly there wasn't much difference in fitness parameters between Round 1-3. Another interesting point is that despite relatively low correlations between predicted and true values our recommendation algorithm performed very well.

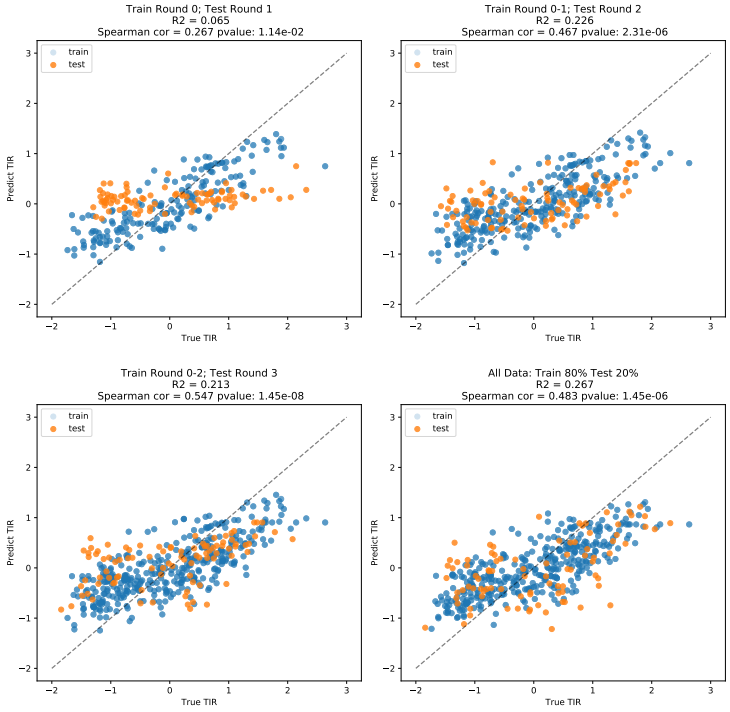
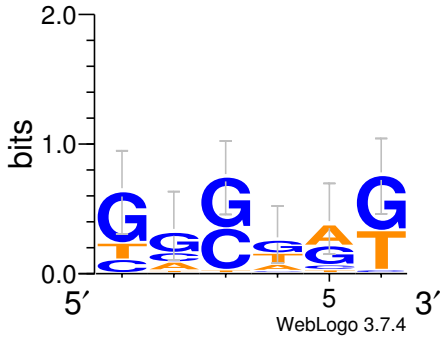


Figure 3: Scatterplot.

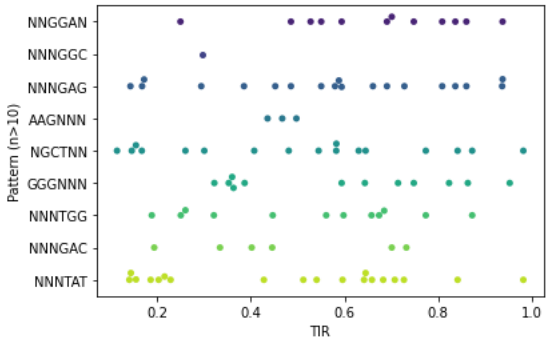
2.4 Characteristics of the obtained library and sequences

A genetic part library should display a number of characteristics to be deemed useful. First quality expected from a good library is a wide range of relevant trait values (in our case it is the TIR) so that designed genetic devices can be appropriately optimised. Secondly, it is very beneficial for the library to show a number of parts with similar numerical values for the relevant trait, but with different sequences, so that a designer would not have to settle on suboptimal part due to, for example, assembly incompatibility. Finally, a good library has to be reliable - if the numbers provided in the library are unreliable designers will be reluctant to use the data set. Figure 4 shows how our library conforms to these requirements. In short, due to focusing on exploitation distribution of our TIRs is skewed toward strong ones, but there is still a significant amount of RBSs with medium and low strengths. High diversity of our sequences is shown by lack of relative lack of bias in position weight matrices for different bins. Finally, we were able to keep the average standard deviation for each bin within XX% with 6 biological replicates for each sample, which we find satisfactory for reliability.

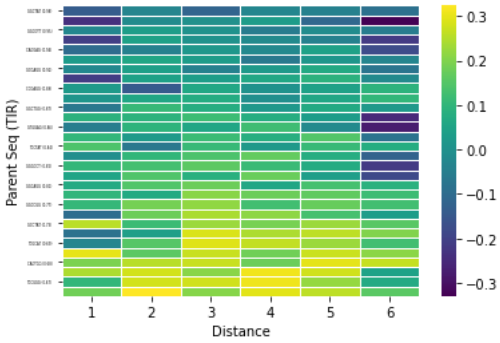
Our library is also showing two characteristics that may be useful in future RBS design efforts. One of them is the perceived editing distance between two sequences required for meaningful improvement in the TIR. We define the editing distance as Hamming distance, that is, how many positions have to be changed to get from one sequence to the other (Hamming distance of 0 means that the sequences are identical and 6 means two completely different sequences). Figure 4 shows what edit distance is required for positive change in TIR for RBS with high and medium TIR. For high TIR RBS, the minimum distance that is required for increase of TIR is 2, with edit distance between 2 and 5 giving similar results. For medium TIR RBS, a distance of just 1 is enough to produce a meaningful increase in TIR. That means that as the TIR of examined RBSs increases, the algorithm should be "encouraged" to explore sequences which are more dissimilar to the current candidates. This finding could also indicate that Adaptive Laboratory Evolution may not be able to find very strong RBSs - the low rate of natural mutations will be too slow to discover through more dissimilar sequences [18]. In other words, because the examined sequence is relatively short (6bp in a wider 20bp context) the time to accumulate 2 or 5 changes in the RBS region might be prohibitively long. In such cases, a directed process should be strongly encouraged.
Another feature noticeable in our library is prevalence of specific patterns. These include X, Y and Z (Figure X).



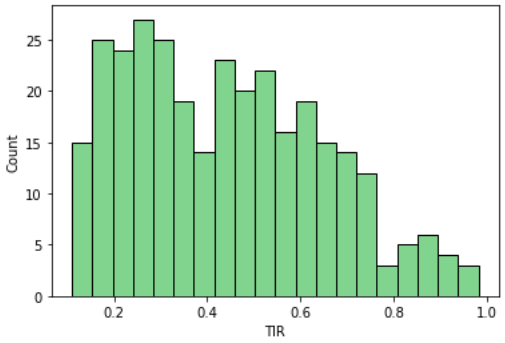
(a) $y = x$



(b) $y = 3sinx$



(c) $y = 5/x$



(d) $y = 5/x$

Figure 4: Three simple graphs

Interestingly, RBSs containing these patterns are not detected as capable of binding the relevant ribosome anti-sequence by binding energy calculations [21]. At present, there is no strong evidence why these patterns might by permissive for high TIRs, nevertheless, it is important to note that the used algorithm does not require that knowledge to effectively find them.

3 Discussion

In this work, we have shown how machine learning and high-throughput, automated laboratory methods can be used to efficiently generate a library of small parts, in this case bacterial RBS. We have used Gaussian Process regression to predict the shape of our function and Upper Confidence Bound Bandit algorithm to recommend sequences to be tested. We have investigated a number of methods of digitising the DNA sequence, finally settling on Spectrum Kernel with Shift method, which fit into our regression method. We performed bulk of our experiments using automation to increase their speed, reliability and reproducibility. By using our workflow, we have generated an extensive library of diverse RBS that can be used in the future studies.

There is a number of issues that we would like to address in the future. The biggest of them is elucidating the reason for none of our sequences being able to significantly cross the TIR of our benchmark sequence. Essentially, there are two possible reasons for this, one is due to the algorithm and one due to nature. If the problem is with the algorithm, it might due to its inability to predict TIRs higher than the ones it has been already presented with. How to solve this? It might be also possible that the TIR seen in the benchmark is simply already the strongest possible and represents a natural ceiling for it. In that case, change in the algorithm will not help. Another problem we would like to tackle is the not always good prediction of the TIR of a given RBS. In principle, the predictions should improve with more data fed into the algorithm, but that would be against the goal of the study which is to minimise the number of data points that have to be obtained.

In the future, we hope to extend the algorithm to other, more complicated genetic elements. This could include promoters and terminators. The complexity of the task quickly increases with the length of the sequence. This is because the experimental space grows exponentially with the number of examined positions and so the space becomes increasingly hard to cover with experiments. To solve this problem, a different algorithms or experimental techniques might be needed, but the general workflow can be reused.

Finally, we hope to see our RBS library being adapted by the community. We believe that its favourable characteristics will make if very useful to designers.

4 Methods

4.1 Laboratory experimental design

4.1.1 BUILD: Construction of genetic devices

Plasmid Design. The pBbB6c-GFP plasmid has been used for all our designs. This plasmid comes with GFP mut3b CDS inducible with addition of Isopropyl β-D-1-thiogalactopyranoside (IPTG). The original RBS for the GFP CDS was replaced with combination of PCR and isothermal assembly. Primers and the assembly strategy have been generated using the Teselagen DESIGN software (Teselagen Biotechnology).

PCR. PCR amplification of the cloning inserts was done using Q5 High-Fidelity 2X Master Mix (NEB, catalogue no. M0492L). 20 μL reactions were prepared by dispensing each of the 10 μM reverse primers into a well of a 96-well PCR plate using the Labcyte Echo Liquid Handler. A mastermix consisting of polymerase premix, plasmid DNA template, and the single 10 μM forward primer was prepared by and dispensed by hand and/or Labcyte Echo. Reactions were run using Touchdown PCR or standard PCR cycling methods in BioRad C1000 thermal cyclers. Capillary electrophoresis of PCR products was performed using the Agilent Technologies ZAG DNA Analyzer system. 2μL of each PCR reaction was electrophoresed using the ZAG 130 dsDNA Kit (75-20000bp) or ZAG 110 dsDNA Kit (35-5000bp) (Agilent Technologies, catalogue no. ZAG-110-5000; ZAG-130-5000). ProSize Data Analysis Software (Agilent Technologies) was used to generated gel images from the sample chromatograms and sizes were estimated by reference to the upper and lower DNA markers spiked into each sample and a DNA ladder run in well H12 of each sample plate.

Isothermal DNA Assembly. Constructs were assembled using NEBuilder HiFi DNA Assembly Master Mix (NEB, catalogue no. E2621L). Reactions consisting of the common fragment and the variable fragment were prepared using the Echo acoustic liquid handler, to a final volume of 5 or 10μL. Assemblies were run in the thermal cycler for 1 hour at 50°C, followed by an infinite hold step at 4°C. Finally, samples were incubated with addition of 50 nL of DpnI at 37°C for 90 minutes.

***E. coli* transformation.** The DH5α cell line (Thermo Fisher Scientific, catalogue no. 18265017) was made chemically competent using the Mix & Go *E. coli* Transformation Kit & Buffer Set (Zymo Research, catalogue no. T3001). 20μL of cells was aliquoted into each well of a cold 96-well PCR plate and stored at -80°C for later use. Plates of cells were thawed on a -20°C cold block before 3μL of the assembly product was added and mixed using the CyBio FeliX liquid handler. Cells were incubated on a cold block for 2-5 minutes before being plated in a 96 square grid on Omnitrays containing LB (BD, catalogue no. ***) with 34μg/mL chloramphenicol (Sigma, catalogue no. ***). Plates were incubated overnight at 37°C.

Automated colony picking and culturing. A Singer Instruments PIXL colony picker was used to select individual colonies from the transformation plates using the 490-510nm (cyan) light filter . Each selected colony was used to inoculate 1mL of selective medium in a 2mL square well 96 plate. They were then cultured overnight in 37°C with shaking (300rpm).

Glycerol stock preparation. 100μL of sterile 80% (v/v) glycerol and 100μL of overnight culture were combined in the wells of a 96 deep (2mL) round well plate using the CyBio Felix liquid handler. They were then sealed with a 96-well silicon sealing mat and transferred to a -80°C freezer.

Sequencing Strains that gave GFP fluorescence intensity readings similar to that of the original RBS were selected for sequence confirmation by capillary electrophoresis sequencing (CES) by Macrogen, Inc. (Seoul, South Korea). The strains transformed with each of the selected constructs were grown to saturation in 5 mL LB medium with chloramphenicol selection (34 μg/mL). Plasmids were extracted from the cultures using the QIAprep Spin Miniprep Kit (QIAGEN) according to the manufacturer's instructions. Plasmid concentrations were quantified using the Cytation 5 plate reader with the Take3 Micro-Volume Plate (BioTek) and all fell in the range of 100-200 ng/μL. Samples of 20 μL of undiluted plasmid DNA were sequenced using a single primer (5'-CGATATAGGCGCCAGCAA-3') that binds approximately 150 bp upstream of the RBS. Reads were aligned with the template sequence in the Teselagen software (TeselaGen Biotechnology, Inc.).

4.1.2 TEST: Culture analysis

Test strain culture. Overnight cultures were started by inoculating 1mL of LB medium supplemented with 34μg/mL chloramphenicol with 2 μL of the glycerol stock in a 96 deep (2mL) round well plate. Cultures were incubated at 37°C with shaking (300rpm) for 17 hours. The following morning, 20μL of overnight cultures were added to 980μL of fresh selection medium and these cultures were grown at 37°C with shaking in 2mL round well 96 plate. After 90 minutes, 200μL of each culture (induced with 1.0μL of 0.1M IPTG) was transferred to a flat-bottom clear polystyrene 96-well plate.

Microplate spectrophotometry. The plates were tested in Cytation5 microplate reader. Cytation 5 acquisition and incubation/shaking settings were as follows: length of run: 8h; interval: 10 min; continuous orbital shake at 237 cpm and slow orbital speed; excitation wavelength: 490/10mm; emission wavelength: 515/10 mm; bottom read; gain: 60; read height: 7mm; read speed: Sweep.

4.2 Machine learning experimental design

The flowchart of machine learning based experimental design is shown in Figure 1. To automatically design the RBS sequences in batch using machine learning, we can logically divided the workflow into two parts: 1) **LEARN**: A regression algorithm which takes the RBS sequences as input features and TIR scores as labels, trains on sequences with known labels and returns the predicted TIR scores and the respective confidence intervals. 2) **DESIGN**: An online learning approach which recommends the RBS sequences based on the predicted TIR scores and confidence intervals.

4.2.1 LEARN: Gaussian Process Regression with String Kernel

To find RBS sequences with the highest possible TIR score after a total number of rounds N, we consider our experimental design problem as sequential optimisation of an unknown reward function $f : \mathcal{D} \to \mathbb{R}$, where $\mathcal{D}$ is the

set containing all RBS sequence points, and $f(\mathbf{x})$ is the TIR score at $\mathbf{x}$. In each round t, we choose a set of m points $\mathcal{S}_t \subset \mathcal{D}$ and observe the function values at each point in the selected set $\mathcal{S}_t$, i.e. $y_i = f(\mathbf{x}_i) + \epsilon$, for all $i \in \mathcal{S}$, where ϵ is the noise (we assume that the noise is following Gaussian distribution with unknown mean and variance). This noise is influenced by the accuracy of the RBS predictor and other experimental sources of interference (e.g. time, temperature, operator, etc.).

For regression model, we have used the *Gaussian Process Regression (GPR)*. A Gaussian process regression model [24] is a Bayesian approach which provides uncertainty measurements on predictions. We model f as a sample from a *Gaussian process* $\mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, which is specified by the mean function $\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and the kernel (or covariance) function $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]$. GPR can predict both the posterior mean and posterior variance. The posterior variance represents the level of uncertainty for the prediction.

The choice of covariance (kernel) function is critical for accurate predictions, since it controls smoothness and amplitude of the function we model. To represent the RBS sequences and formulate the similarity between sequences, we use the *weighted degree kernel with shift* (WDS) [25] to specify the kernel function of GP. WDS is a type of a string kernel, which takes two sequences (strings) as inputs and outputs a scalar value which represents the similarities between the two sequences. WDS kernel does this by counting the matches of substrings of a certain length (i.e. kmers) that constitute the sequence. The maximum substring length is specified by ℓ. The WDS takes into account the positional information by counting substrings starting from different positions, where the start position is specified by l. Additionally, the WDS kernel considers the shifting of substrings, with the maximum shift specified by s. For example, for two core sequences $ACCTGA$ and $CCTGAA$, there is a common part $CCTGA$ which is begins at the 2nd nucleotide in sequence 1 and at the 1st nucleotide in sequence 2, hence the calculated shift would be 1.

Let $\mathbb{I}(\mathrm{A})$ is the indicator function, which equals 1 if A is true and 0 otherwise. Then $\mathbb{I}(\mathbf{x}_{[l+s:l+s+d]} = \mathbf{x}'_{[l:l+d]})$ counts the matches of substrings of length d between $\mathbf{x}$ starting from position $l+s$ and $\mathbf{x}'$ starting from position l. This is similarly done for $\mathbb{I}(\mathbf{x}_{[l:l+d]} = \mathbf{x}'_{[l+s:l+s+d]})$. By having these two terms considering substrings of two sequences with starting positions differing by s characters, the WDS can measure shifted positional information. When $s = 0$, the kernel function counts the matches with no shift between sequences. Let $\mathbf{x}, \mathbf{x}'$ be two RBS sequences with length L, the WDS kernel is defined as

$$k_\ell^{WDS}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \sum_{s=0, s+l \le L}^{S(l)} \delta_s \left(\mathbb{I}(\mathbf{x}_{[l+s:l+s+d]} = \mathbf{x}'_{[l:l+d]}) + (\mathbb{I}(\mathbf{x}_{[l:l+d]} = \mathbf{x}'_{[l+s:l+s+d]}) \right), \tag{1}$$

where $\beta_d = \frac{2(\ell-d+1)}{\ell(\ell+1)}, \delta_s = \frac{1}{2(s+1)}$, γ_l is a weighting parameter over the position in the sequence, where we choose to use a uniform weighting over the sequences, i.e. $\gamma_l = 1/L$. $S(l)$ determines the shift range at position l.

4.2.2 DESIGN: Batch UCB

For recommendations of RBS sequences that should be experimentally labeled next, we have used the *Upper Confidence Bound (UCB)* algorithm. On one hand, we want to exploit the function in terms of the design space, that is to pinpoint sequences that are believed to have high labels (i.e. high predicted mean); on the other hand, we also want to explore the design space where we have little information and sequences have a chance to have high labels (i.e. high predicted SD). The UCB algorithm provides such *exploitation-exploration balance* by balancing the predicted mean and SD. More precisely, UCB algorithm selects RBS sequences with the maximum upper confidence bound at round t, i.e.

$$\mathrm{argmax}_{\mathbf{x}_i \in \mathcal{D}} \left(\mu_{t-1}(\mathbf{x}_i) + \beta_t \sigma_{t-1}(\mathbf{x}_i) \right), \tag{2}$$

where β_t is a hyperparmeter balancing the exploitation and exploration, $\mu_t(\mathbf{x}_i), \sigma_t(\mathbf{x}_i)$ are the predicted mean and standard deviation (SD) at round t for the sequence $\mathbf{x}_i$. We call $\mu_{t-1}(\mathbf{x}_i) + \beta_t \sigma_{t-1}(\mathbf{x}_i)$ the *UCB score* of sequence $\mathbf{x}_i$ at round t.

Since experimentally labelling sequences is time-consuming, it is unrealistic to recommend sequence sequentially (i.e. one-by-one) and then wait for the label to be tested and used to improve the model. Instead, we can recommend RBS sequences in a batch of size n. One naive approach is to recommend sequences in design space with top n UCB scores, as shown in Figure 5(a) ($n = 2$). However, this approach may end up recommending similar sequences in the same local maximum (e.g. $x = 2, x = 2.5$ in this example). However, since we assume similar sequences would have similar labels (e.g. by knowing $x = 2$ we can gain information of $x = 2.5$ as well), we prefer to not waste time and money on labelling sequences with high similarities in the same batch.

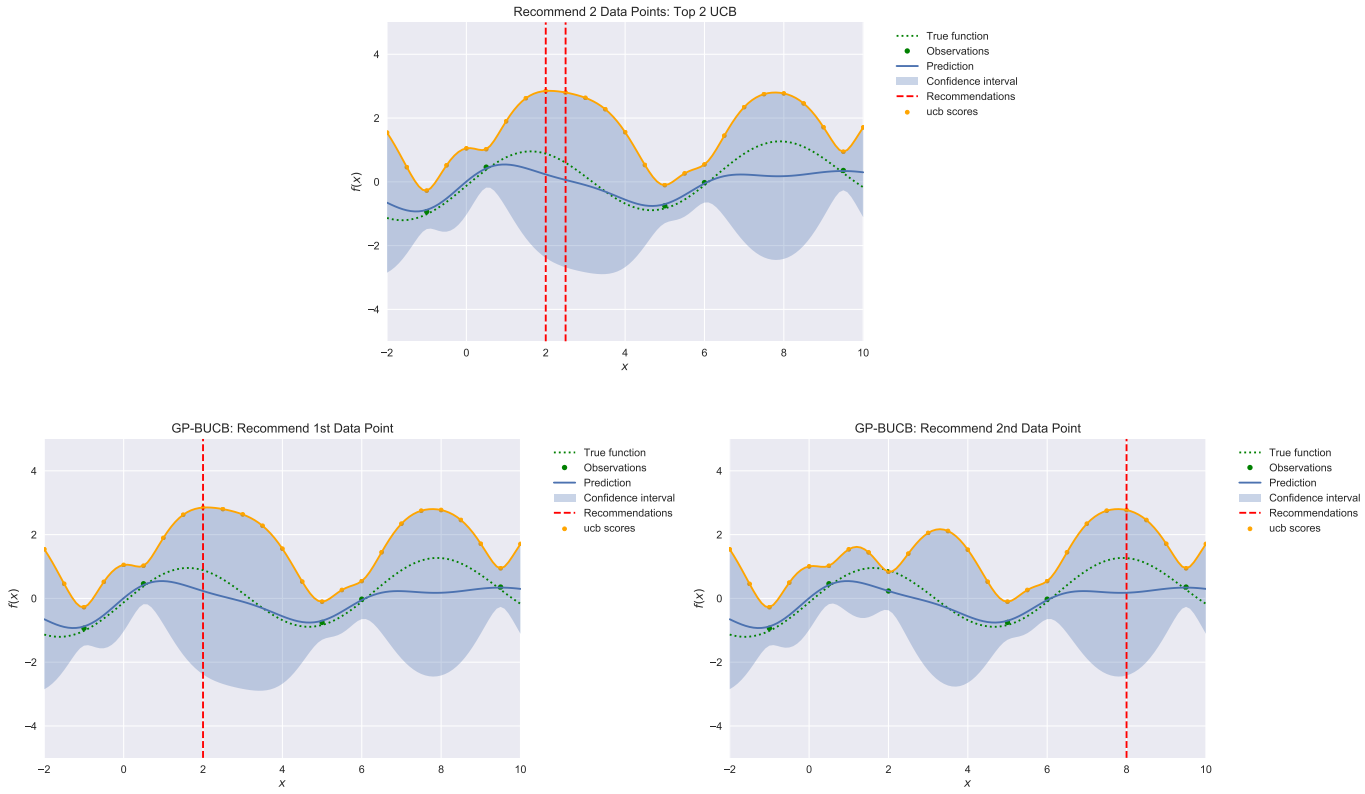


Figure 5: Batch Recommendation. We use the batch size of 2, with 5 initial observations. The design (recommendation) space is 24 uniformly distributed points in the range [-2,10], i.e. -2, -1.5, -1, ..., 9.5, 10. The confidence interval are shown with predicted mean of $\pm$ 1.96 standard deviation. (a) Top UCB recommendations. The recommendations are 2 data points with top UCB scores, constructed with GP predictions. (b)(c) GP-BUCB recommendations. (b) shows the first recommended sequence, (c) shows the new predicted confidence interval and the second recommendation based on that.

A key property of Gaussian Process regression is that the predictive variance depends only on observed points (i.e. features), but not on the labels of those observed points. We can make use of this property to design batch upper confidence bound (BUCB) algorithm [10]. That is, we can recommend sequences sequentially by updating the UCB score with the updated predicted SD with the the previously recommended points. First, GP-BUCB recommends the data point with maximum UCB score based on the predictions over initial 5 observations as shown in Figure 5(b). Then we add the recommended data point ($x = 2$) into the training data set with the predicted mean of that point as label (note it is not the true label, i.e. observation), and update the predicted variance and then we finally update the UCB scores. We then recommend the second data point based on the new UCB scores. As we can see in Figure 5(c), since we assume we have observed $x = 2$, then the new predicted variance of the data points in design space around $x = 2$ decreases, so instead of recommending a similar data point $x = 2.5$, we recommend the data point which is in a different maximum area with potentially high labels ($x = 8$). By using GP-BUCB, we can ensure that the sequences with high similarities will not be recommended in the same batch hence increasing the exploration efficiency.

Contributions

Zhang M. and Ong C. S. designed and implemented the machine learning algorithms and workflow. Holowko M. B. and Hayman Zumpe H. have designed and performed the laboratory experiments. All authors analysed the data and contributed to and reviewed the manuscript.

Competing interests

The authors declare no competing interests

Acknowledgments

The authors would like to acknowlege CSIRO's AI/ML and Synthetic Biology Future Science Platforms for providing funding for this research.

References

[1] Babak Alipanahi et al. "Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning". In: *Nature Biotechnology* 33.8 (2015), pp. 831–838. ISSN: 1546-1696. DOI: 10.1038/nbt.3300. URL: https://doi.org/10.1038/nbt.3300.

[2] Christof Angermueller et al. "Deep learning for computational biology". In: *Molecular Systems Biology* 12.7 (July 2016), p. 878. ISSN: 1744-4292. DOI: 10.15252/msb.20156651. URL: https://doi.org/10.15252/msb.20156651.

[3] Asa Ben-Hur et al. "Support Vector Machines and Kernels for Computational Biology". In: *PLOS Computational Biology* 4.10 (Oct. 2008), e1000173. URL: https://doi.org/10.1371/journal.pcbi.1000173.

[4] Jennifer A N Brophy and Christopher A Voigt. "Principles of genetic circuit design." In: *Nature methods* 11.5 (May 2014), pp. 508–520. ISSN: 1548-7105 (Electronic). DOI: 10.1038/nmeth.2926.

[5] Diogo M Camacho et al. "Next-Generation Machine Learning for Biological Networks". In: *Cell* 173.7 (2018), pp. 1581–1592. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2018.05.015. URL: http://www.sciencedirect.com/science/article/pii/S0092867418305920.

[6] Barry Canton, Anna Labno, and Drew Endy. "Refinement and standardization of synthetic biological parts and devices". In: *Nature Biotechnology* 26.7 (2008), pp. 787–793. ISSN: 1546-1696. DOI: 10.1038/nbt1413. URL: https://doi.org/10.1038/nbt1413.

[7] Ying-Ja Chen et al. "Characterization of 582 natural and synthetic terminators and quantification of their design constraints". In: *Nature Methods* 10.7 (2013), pp. 659–664. ISSN: 1548-7105. DOI: 10.1038/nmeth.2515. URL: https://doi.org/10.1038/nmeth.2515.

[8] Zak Costello and Hector Garcia Martin. "A machine learning approach to predict metabolic pathway dynamics from time-series multiomics data". In: *npj Systems Biology and Applications* 4.1 (2018), p. 19. ISSN: 2056-7189. DOI: 10.1038/s41540-018-0054-3. URL: https://doi.org/10.1038/s41540-018-0054-3.

[9] Maarten H. de Smit and Jan van Duin. "Translational initiation on structured messengers: Another role for the shine-dalgarno interaction". In: *Journal of Molecular Biology* 235.1 (1994), pp. 173–184. ISSN: 0022-2836. DOI: https://doi.org/10.1016/S0022-2836(05)80024-5. URL: https://www.sciencedirect.com/science/article/pii/S0022283605800245.

[10] Thomas Desautels, Andreas Krause, and Joel Burdick. *Parallelizing Exploration-Exploitation Tradeoffs with Gaussian Process Bandit Optimization*. 2012. arXiv: 1206.6402 [cs.LG].

[11] Amin Espah Borujeni and Howard M Salis. "Translation Initiation is Controlled by RNA Folding Kinetics via a Ribosome Drafting Mechanism". In: *Journal of the American Chemical Society* 138.22 (June 2016), pp. 7016–7023. ISSN: 0002-7863. DOI: 10.1021/jacs.6b01453. URL: https://doi.org/10.1021/jacs.6b01453.

[12] Paul S Freemont. "Synthetic biology industry: data-driven design is creating new opportunities in biotechnology". In: *Emerging Topics in Life Sciences* 3.5 (Oct. 2019), pp. 651–657. ISSN: 2397-8554. DOI: 10.1042/ETLS20190040. URL: https://doi.org/10.1042/ETLS20190040.

[13] Peter J E Goss and Jean Peccoud. "Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets". In: *Proceedings of the National Academy of Sciences* 95.12 (June 1998), 6750 LP –6755. DOI: 10.1073/pnas.95.12.6750. URL: http://www.pnas.org/content/95/12/6750.abstract.

[14] Adrian J Jervis et al. "Machine learning of designed translational control allows predictive pathway optimization in Escherichia coli". In: *ACS synthetic biology* 8.1 (2018), pp. 127–136.

[15] Adrian J Jervis et al. "Machine Learning of Designed Translational Control Allows Predictive Pathway Optimization in Escherichia coli". In: *ACS Synthetic Biology* 8.1 (Jan. 2019), pp. 127–136. DOI: 10.1021/acssynbio.8b00398. URL: https://doi.org/10.1021/acssynbio.8b00398.

[16] Allen R Buskirk Kazuki Saito Racgek Green. "Translational initiation in E. coli occurs at the correct sites genome-wide in the absence of mRNA-rRNA base-pairing". In: *eLife* 9 (Feb. 2020), e55002. URL: https://elifesciences.org/articles/55002#info.

[17] Christopher E. Lawson et al. "Machine learning for metabolic engineering: A review". In: *Metabolic Engineering* 63 (2021). Tools and Strategies of Metabolic Engineering, pp. 34–60. ISSN: 1096-7176. DOI: https://doi.org/10.1016/j.ymben.2020.10.005. URL: https://www.sciencedirect.com/science/article/pii/S109671762030166X.

[18] Heewook Lee et al. "Rate and molecular spectrum of spontaneous mutations in the bacterium Escherichia coli as determined by whole-genome sequencing". In: *Proceedings of the National Academy of Sciences* 109.41 (2012), E2774–E2783. ISSN: 0027-8424. DOI: 10.1073/pnas.1210309109. eprint: https://www.pnas.org/content/109/41/E2774.full.pdf. URL: https://www.pnas.org/content/109/41/E2774.

[19] Taek Soon Lee et al. "BglBrick vectors and datasheets: A synthetic biology platform for gene expression." eng. In: *Journal of biological engineering* 5 (Sept. 2011), p. 12. ISSN: 1754-1611 (Electronic). DOI: 10.1186/1754-1611-5-12.

[20] Christina Leslie, Eleazar Eskin, and William Stafford Noble. "The spectrum kernel: A string kernel for SVM protein classification". In: *Biocomputing 2002*. World Scientific, 2001, pp. 564–575.

[21] Martin Mann, Patrick R. Wright, and Rolf Backofen. "IntaRNA 2.0: enhanced and customizable prediction of RNA–RNA interactions". In: *Nucleic Acids Research* 45.W1 (May 2017), W435–W439. ISSN: 0305-1048. DOI: 10.1093/nar/gkx279. eprint: https://academic.oup.com/nar/article-pdf/45/W1/W435/18137215/gkx279.pdf. URL: https://doi.org/10.1093/nar/gkx279.

[22] Dokyun Na and Doheon Lee. "RBSDesigner: software for designing synthetic ribosome binding sites that yields a desired level of protein expression". In: *Bioinformatics* 26.20 (Aug. 2010), pp. 2633–2634. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btq458. URL: https://doi.org/10.1093/bioinformatics/btq458.

[23] Alec A K Nielsen et al. "Genetic circuit design automation". In: *Science* 352.6281 (Apr. 2016), aac7341. DOI: 10.1126/science.aac7341. URL: http://science.sciencemag.org/content/352/6281/aac7341.abstract.

[24] Carl Edward Rasmussen. "Gaussian Processes in Machine Learning". In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*. Ed. by Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71. ISBN: 978-3-540-28650-9. DOI: 10.1007/978-3-540-28650-9_4. URL: https://doi.org/10.1007/978-3-540-28650-9_4.

[25] G. Rätsch, S. Sonnenburg, and B. Schölkopf. "RASE: recognition of alternatively spliced exons in C.elegans". eng. In: *Bioinformatics (Oxford, England)* 21 Suppl 1 (June 2005), pp. i369–377. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bti1053.

[26] Benjamin Reeve et al. "Predicting translation initiation rates for designing synthetic biology". eng. In: *Frontiers in bioengineering and biotechnology* 2 (Jan. 2014), p. 1. ISSN: 2296-4185. DOI: 10.3389/fbioe.2014.00001. URL: https://pubmed.ncbi.nlm.nih.gov/25152877%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4126478/.

[27] Philip A. Romero, Andreas Krause, and Frances H. Arnold. "Navigating the protein fitness landscape with Gaussian processes". In: *Proceedings of the National Academy of Sciences* 110.3 (Jan. 2013), E193. DOI: 10.1073/pnas.1215251110. URL: http://www.pnas.org/content/110/3/E193.abstract.

[28] Howard M Salis, Ethan A Mirsky, and Christopher A Voigt. "Automated design of synthetic ribosome binding sites to control protein expression". In: *Nature Biotechnology* 27.10 (2009), pp. 946–950. ISSN: 1546-1696. DOI: 10.1038/nbt.1568. URL: https://doi.org/10.1038/nbt.1568.

[29] Günther F.E. Scherer et al. "The ribosome binding sites recognized by E. coli ribosomes have regions with signal character in both the leader and protein coding segments". In: *Nucleic Acids Research* 8.17 (Sept. 1980), pp. 3895–3908. ISSN: 0305-1048. DOI: 10.1093/nar/8.17.3895. eprint: https://academic.oup.com/nar/article-pdf/8/17/3895/6963796/8-17-3895.pdf. URL: https://doi.org/10.1093/nar/8.17.3895.

[30] Sang Woo Seo et al. "Predictive design of mRNA translation initiation region to control prokaryotic translation efficiency". In: *Metabolic Engineering* 15 (2013), pp. 67–74. ISSN: 1096-7176. DOI: https://doi.org/10.1016/j.ymben.2012.10.006. URL: http://www.sciencedirect.com/science/article/pii/S1096717612001188.

[31] Ryan K Shultzaberger et al. "Anatomy of Escherichia coli ribosome binding sites11Edited by D. Draper". In: *Journal of Molecular Biology* 313.1 (2001), pp. 215–228. ISSN: 0022-2836. DOI: https://doi.org/10.1006/jmbi.2001.5040. URL: http://www.sciencedirect.com/science/article/pii/S0022283601950405.

[32] Niranjan Srinivas et al. "Information-theoretic regret bounds for gaussian process optimization in the bandit setting". In: *IEEE Transactions on Information Theory* 58.5 (2012), pp. 3250–3265.

[33] Brynne C Stanton et al. "Genomic mining of prokaryotic repressors for orthogonal logic gates". In: *Nature Chemical Biology* 10.2 (2014), pp. 99–105. ISSN: 1552-4469. DOI: 10.1038/nchembio.1411. URL: https://doi.org/10.1038/nchembio.1411.

[34] Gary D. Stormo, Thomas D. Schneider, and Larry M. Gold. " Characterization of translational initiation sites in E. coli". In: *Nucleic Acids Research* 10.9 (May 1982), pp. 2971–2996. ISSN: 0305-1048. DOI: 10.1093/nar/10.9.2971. eprint: https://academic.oup.com/nar/article-pdf/10/9/2971/7067678/10-9-2971.pdf. URL: https://doi.org/10.1093/nar/10.9.2971.

[35] Tianbing Xia et al. "Thermodynamic Parameters for an Expanded Nearest-Neighbor Model for Formation of RNA Duplexes with Watson-Crick Base Pairs". In: *Biochemistry* 37.42 (Oct. 1998), pp. 14719–14735. ISSN: 0006-2960. DOI: 10.1021/bi9809425. URL: https://doi.org/10.1021/bi9809425.

[36] Kevin K Yang et al. "Learned protein embeddings for machine learning". In: *Bioinformatics* 34.15 (Mar. 2018), pp. 2642–2648. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty178. eprint: https://academic.oup.com/bioinformatics/article-pdf/34/15/2642/25230893/bty178.pdf. URL: https://doi.org/10.1093/bioinformatics/bty178.

[37] Jing Wui Yeoh et al. "An Automated Biomodel Selection System (BMSS) for Gene Circuit Designs". In: *ACS Synthetic Biology* 8.7 (July 2019), pp. 1484–1497. DOI: 10.1021/acssynbio.8b00523. URL: https://doi.org/10.1021/acssynbio.8b00523.

Supplementary

A Extended Machine Learning Discussion

A.1 Reproducible Plots

In this section, we put reproducible plots and will update the plots when we get full results.

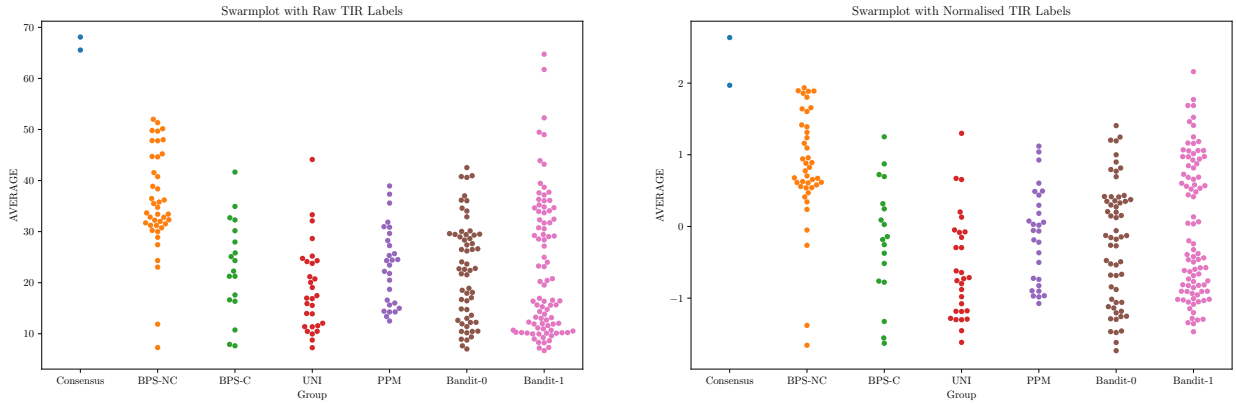


Figure 6: Swarmplots for different groups, with raw (left) and normalised (right) TIR labels (averaged over 6 replicates). Group names represent: Consensus (consensus sequence tested in different round); BPS-NC (bps noncore); BPS-C (bps core); UNI (uniformly random); PPM (position-based probability matrix); Bandit-0 (bandit design for round 0); Bandit-1 (bandit design for round 1).

A.2 Gaussian Process Regression

A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution. We define mean function $\mu(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$ as

$$\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{3}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]. \tag{4}$$

A Gaussian process is specified by its mean function and convariance function as $f(\mathbf{x}) \sim \mathcal{GP}\left(\mu(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right)$. We consider the case where the observations are noisy, i.e. $\{(\mathbf{x}_i, y_i)|i = 1, \ldots, n\}$, where $y_i = f(\mathbf{x}_i) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \alpha^2)$. The Gaussian noise is independent identically distributed, and the prior on the noisy observations is then $\operatorname{cov}\left(y_p, y_q\right) = k\left(\mathbf{x}_p, \mathbf{x}_q\right) + \alpha^2 \delta_{pq}$, where δ_{pq} is a Kronecker delta which is one if $p = q$ and zero otherwise. It is equivalent to a diagonal matrix $\alpha^2 I$ on the kernel matrix evaluated on the training points.

For n_* test points X_*, we assume the prior over the functions values as a random Gaussian vector $\mathbf{f}_* \sim \mathcal{N}(\mathbf{0}, K(X_*, X_*))$. Then the joint distribution of the observed target values and the function values at the test points under the prior as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) + \alpha^2 I & K\left(X, X_*\right) \\ K\left(X_*, X\right) & K\left(X_*, X_*\right) \end{bmatrix}\right) \tag{5}$$

where $K(X, X_*)$ denotes the $n \times n_*$ covariance/Kernel matrix evaluated at all pairs of training and testing points, similarly for other kernel matrices. Then the posterior of the test points (i.e. predictive distributions) is given by the conditional distribution $\mathbf{f}_* | X, \mathbf{y}, X_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, cov(\mathbf{f}_*))$, where

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}\left[\mathbf{f}_* \mid X, \mathbf{y}, X_*\right] = K\left(X_*, X\right)\left[K(X, X) + \alpha^2 I\right]^{-1} \mathbf{y} \tag{6}$$

$$\operatorname{cov}\left(\mathbf{f}_*\right) = K\left(X_*, X_*\right) - K\left(X_*, X\right)\left[K(X, X) + \alpha^2 I\right]^{-1} K\left(X, X_*\right) \tag{7}$$

For noisy test targets $\mathbf{y}_*$, we can compute the predictive distribution by adding $\alpha^2 I$ to the variance term $cov(\mathbf{f}_*)$ in Eq. (7).

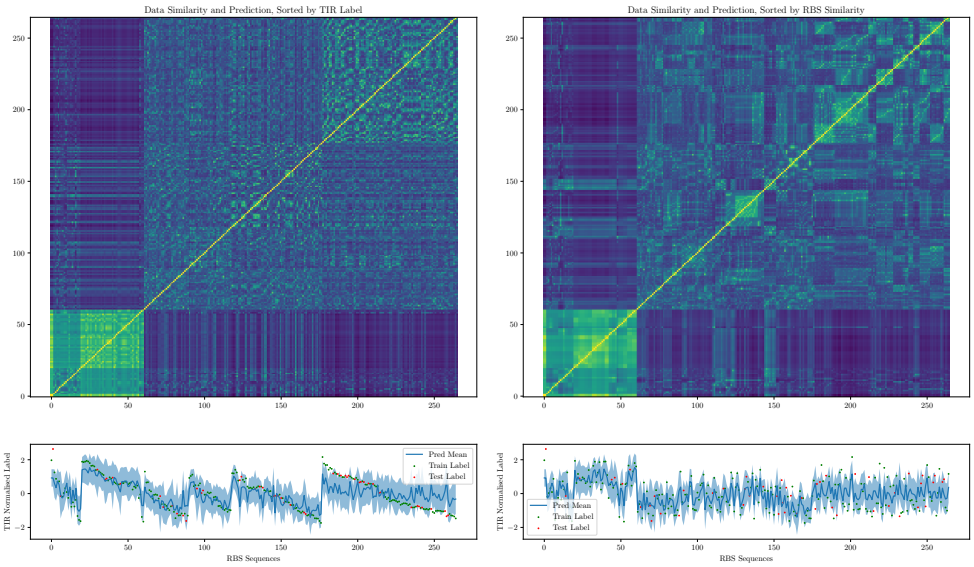


Figure 7: Kernel Heatmap and Predictions. Sequences are grouped as Consensus (1-2), BPS-C (3-20), BPS-NC (21-61), UNI (62-90), PPM (91-118), Bandit-0 (119-177), Bandit-1 (178-265). Inside of each group, sequences are clustered and sorted in terms of TIR labels (left) or RBS similarity (right). The first row shows the similarity measured by weighted degree kernel with shift, the second shows the predicted mean and uncertainty (1.95 standard deviation).

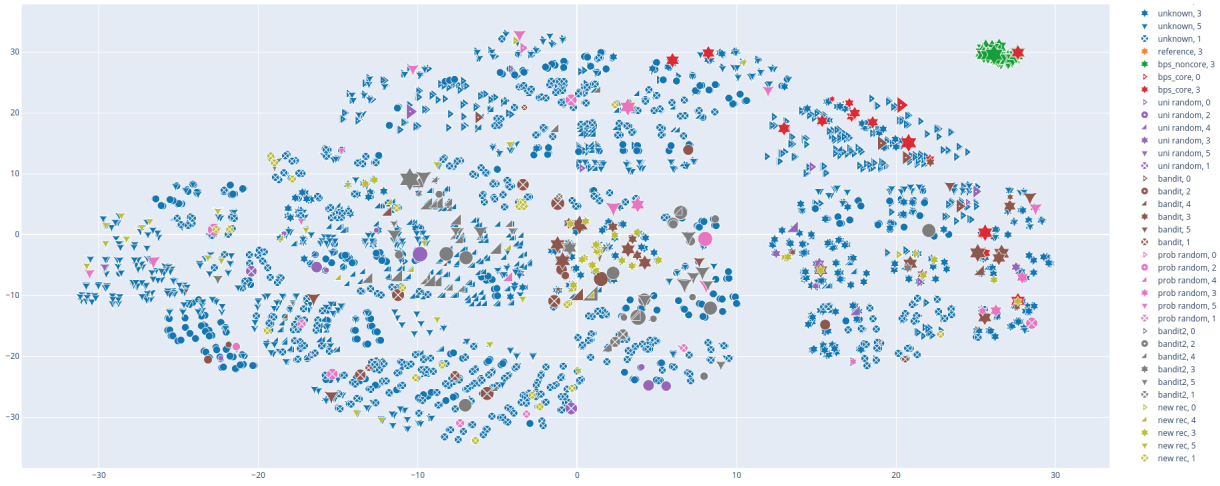


Figure 8: TSNE of RBS sequences in design space with clustering. The distance is calculated based on the weighted degree kernel on RBS sequences. Colours indicate different groups, shapes indicate different clusters.

A.3 Choices of Kernels

The choice of covariance function is critical for the performance of Gaussian process regression, we show a number of different string kernels tested in this study below:

- *Spectrum Kernel.*

$$k_\ell^{\text{Spec}}(X, X') = \left\langle \phi_\ell^{\text{Spec}}(\mathbf{x}), \phi_\ell^{\text{Spec}}(\mathbf{x}') \right\rangle = \phi_\ell^{\text{Spec}}(\mathbf{x})^T \phi_\ell^{\text{Spec}}(\mathbf{x}') . \tag{8}$$

  where $\mathbf{x}, \mathbf{x}'$ are two RBS sequences in $\mathcal{D}$ over an alphabet Σ. We denote the number of letters in the alphabet as $|\Sigma|$. $\phi_\ell^{\text{spec}}(\mathbf{x})$ maps the sequence X into a $|\Sigma|^\ell$ dimensional feature space, where each dimension is the count of the number of one of the $|\Sigma|^\ell$ possible strings s of length ℓ. Let X, X' be two metrics which include n sequences, and $\Phi_d^{Spec}(X) \in \mathbb{R}^{n \times |\Sigma|^\ell}$, then the spectrum kernel over metrics is

$$K_\ell^{\text{Spec}}(X, X') = \Phi_\ell^{\text{Spec}}(X) \Phi_\ell^{\text{Spec}}(X')^T . \tag{9}$$

- *Sum of Spectrum Kernel,* considers weighted sum over different parts of the string.

- *Mixed Spectrum Kernel,* considers weighted sum over different substring length, with $\beta_d = \frac{2(\ell-d+1)}{\ell(\ell+1)}$,

$$k_\ell^{MixedSpec}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{\ell} \beta_d k_d^{Spec}(\mathbf{x}, \mathbf{x}') \tag{10}$$

- *Weighted Degree Kernel,* considers positional information. WD kernel counts the match of kmers at corresponding positions in two sequences. For sequences with fixed length L and weighted degree kernel considers substrings starting at each position $l = 1, ..., L$, with $\beta_d = \frac{2(\ell-d+1)}{\ell(\ell+1)}$,

$$k_\ell^{WD}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l k_d^{Spec}(\mathbf{x}_{[l:l+d]}, \mathbf{x}'_{[l:l+d]}) \tag{11}$$

$$= \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \phi_d^{Spec}(\mathbf{x}_{[l:l+d]})^T \phi_d^{Spec}(\mathbf{x}'_{[l:l+d]}) \tag{12}$$

$$= \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \mathbb{I}(\mathbf{x}_{[l:l+d]} = \mathbf{x}'_{[l:l+d]}), \tag{13}$$

  where $\mathbb{I}(\text{true}) = 1$ and 0 otherwise.

- *Weighted Degree Kernel With Shift.*

$$k_\ell^{WDS}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \sum_{s=0, s+l \le L}^{S(l)} \delta_s \left(k_d^{Spec}(\mathbf{x}_{[l+s:l+s+d]}, \mathbf{x}'_{[l:l+d]}) + (k_d^{Spec}(\mathbf{x}_{[l:l+d]}, \mathbf{x}'_{[l+s:l+s+d]}) \right) \tag{14}$$

$$= \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \sum_{s=0, s+l \le L}^{S(l)} \delta_s \left(\mathbb{I}(\mathbf{x}_{[l+s:l+s+d]} = \mathbf{x}'_{[l:l+d]}) + (\mathbb{I}(\mathbf{x}_{[l:l+d]} = \mathbf{x}'_{[l+s:l+s+d]}) \right), \tag{15}$$

  where $\beta_d = \frac{2(\ell-d+1)}{\ell(\ell+1)}, \delta_s = \frac{1}{2(s+1)}$, γ_l is a weighting over the position in the sequence, where we choose to use a uniform weighting over the sequences, i.e. $\gamma_l = 1/L$. $S(l)$ determines the shift range at position l.

From kernel to distance:

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{k(\mathbf{x}, \mathbf{x}) + k(\mathbf{x}', \mathbf{x}') - 2k(\mathbf{x}, \mathbf{x}')}$$

A.3.1 Normalisation of Kernel

As part of data pre-processing, the range of all features should be normalised so that each feature contributes approximately proportionately to the predictive model. The kernel matrix is represented by the inner product of the underlying feature vectors, it needs to be normalised before being used in the downstream regression models. Up-scaling (down-scaling) features can be understood as down-scaling (up-scaling) regularizers such that they penalise the features less (more).

Here we consider two approaches for kernel normalisation: centering and unit norm. We will show how to convert the normalisation in terms of feature vectors to normalisation in terms of kernel matrices. As defined before, consider $\mathbf{x}, \mathbf{x}'$ are two RBS sequences in $\mathcal{D}$ over an alphabet Σ. We denote $\phi(\mathbf{x}_i)$ as a column feature vector of sequence $\mathbf{x}_i$, where a feature function $\phi : \mathbf{x} \to \mathbb{R}^d$. Assume there is total of n sequences in the data X (n' sequences in the data X'). We illustrate centering and unit norm normalisation below.

- Centering. Defining the mean vector as $\bar{\Phi}(X) = \frac{1}{n}\sum_{s=1}^{n}\phi(\mathbf{x}_s) \in \mathbb{R}^d$, the centered feature vector $\phi^C(\mathbf{x}_i) \in \mathbb{R}^d$ of $\mathbf{x}_i$ is

$$\phi^C(\mathbf{x}_i) = \phi(\mathbf{x}_i) - \bar{\Phi}(X) = \phi(\mathbf{x}_i) - \frac{1}{n'}\sum_{s=1}^{n'}\phi(\mathbf{x}_s). \tag{16}$$

  The corresponding centering kernel value between $\mathbf{x}_i$ and $\mathbf{x}_j$ is then

$$k^C(\mathbf{x}_i, \mathbf{x}_j) =< \phi^C(\mathbf{x}_i), \phi^C(\mathbf{x}_j) > \tag{17}$$

$$= \left(\phi(\mathbf{x}_i) - \frac{1}{n}\sum_{s=1}^{n}\phi(\mathbf{x}_s)\right)^T \left(\phi(\mathbf{x}_j) - \frac{1}{n'}\sum_{s'=1}^{n'}\phi(\mathbf{x}_{s'})\right) \tag{18}$$

$$= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) - \left(\frac{1}{n}\sum_{s=1}^{n}\phi(\mathbf{x}_s)\right)^T \phi(\mathbf{x}_j) - \phi(\mathbf{x}_i)^T \left(\frac{1}{n}\sum_{s'=1}^{n'}\phi(\mathbf{x}_{s'})\right) + \left(\frac{1}{n}\sum_{s=1}^{n}\phi(\mathbf{x}_s)\right)^T \left(\frac{1}{n'}\sum_{s'=1}^{n'}\phi(\mathbf{x}_{s'})\right) \tag{19}$$

$$= k(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{n}\sum_{s=1}^{n}k(\mathbf{x}_s, \mathbf{x}_j) - \frac{1}{n'}\sum_{s'=1}^{n'}k(\mathbf{x}_i, \mathbf{x}_{s'}) + \frac{1}{n^2}\sum_{s=1}^{n}\sum_{s'=1}^{n'}k(\mathbf{x}_s, \mathbf{x}_{s'}) \tag{20}$$

- Unit Norm. Define the (l_2) norm of a feature vector as $||\phi(\mathbf{x})|| = \sqrt{\sum_{m=1}^{d}\phi_d(\mathbf{x})^2} = \sqrt{k(\mathbf{x}, \mathbf{x})} \in \mathbb{R}^+$, then the unit norm feature vector $\phi^{UN}(\mathbf{x}_i) \in \mathbb{R}^d$ of $\mathbf{x}_i$ is

$$\phi^{UN}(\mathbf{x}_i) = \frac{\phi(\mathbf{x}_i)}{||\phi(\mathbf{x}_i)||}. \tag{21}$$

  The corresponding unit norm kernel value between $\mathbf{x}_i$ and $\mathbf{x}_j$ is then

$$k^{UN}(\mathbf{x}_i, \mathbf{x}_j) =< \frac{\phi(\mathbf{x}_i)}{||\phi(\mathbf{x}_i)||}, \frac{\phi(\mathbf{x}_j)}{||\phi(\mathbf{x}_j)||} > \tag{22}$$

$$= \frac{\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)}{||\phi(\mathbf{x}_i)|| \times ||\phi(\mathbf{x}_j)||} \tag{23}$$

$$= \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{k(\mathbf{x}_i, \mathbf{x}_i)k(\mathbf{x}_j, \mathbf{x}_j)}} \tag{24}$$

- Unit Variance. After the centering and unit norm normalisation, the kernel matrix is unit variance as well. In the following, we show transformations of the unit variance (with centering) normalisation. Define the variance vector $Var(\Phi(X)) = \frac{1}{n}\sum_{s=1}^{n}||\phi(\mathbf{x}_s) - \bar{\Phi}(X)||^2 = \frac{1}{n}\sum_{s=1}^{n}||\phi(\mathbf{x}_s) - \sum_{s'=1}^{n}(\phi(\mathbf{x}'_s))||^2 = \frac{1}{n}\sum_{s=1}^{n}k^C(\mathbf{x}_s, \mathbf{x}_s) \in \mathbb{R}$, the unit variance feature vector $\phi^{UV}(\mathbf{x}_i) \in \mathbb{R}^d$ of $\mathbf{x}_i$ is

$$\phi^{UV}(\mathbf{x}_i) = \frac{\phi(\mathbf{x}_i)}{\sqrt{Var(\Phi(X))}}. \tag{25}$$

The corresponding kernel representation is

$$k^{UV}(\mathbf{x}_i, \mathbf{x}_j) = < \frac{\phi(\mathbf{x}_i)}{\sqrt{Var(\Phi(X))}}, \frac{\phi(\mathbf{x}_j)}{\sqrt{Var(\Phi(X'))}} > \tag{26}$$

$$= \frac{\phi(\mathbf{x}_i)^T \mathbf{x}_j}{\sqrt{Var(\Phi(X))Var(\Phi(\mathbf{X}'))}} \tag{27}$$

$$= \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{\frac{1}{n}\sum_{s=1}^{n} k^C(\mathbf{x}_s, \mathbf{x}_s) \frac{1}{n}\sum_{s'=1}^{n'} k^C(\mathbf{x}_{s'}, \mathbf{x}_{s'})}} \tag{28}$$

After centering and unit norm, $\frac{1}{n}\sum_{s=1}^{n} k^C(\mathbf{x}_s, \mathbf{x}_s) = k(\mathbf{x}_i, \mathbf{x}_i)$, which implies that after centering and unit norm, the kernel matrix is already unit variance normalised.

For the Gaussian Process regression, we make of use of two kernel matrices: the kernel function between the training data itself, i.e. $K(X_{train}, X_{train})$; and the kernel function taking the training data and testing data as inputs, i.e. $K(X_{test}, X_{train})$. We will state two ways of normalisation those two kind of matrices:

- Normalise training and testing data separately. This approach is preferred for most of the machine learning algorithms since it follows the rule that we have no information about testing data while training. Then for centering, one should subtract the mean vector over the training data for both kinds of matrices. For unit norm normalisation, when one calculates $K^{UN}(X_{test}, X_{train})$, the two terms inside of square root: $k(\mathbf{x}_i, \mathbf{x}_i)$ is taken from $K(X_{test}, X_{test})[i, i]$, and $k(\mathbf{x}_j, \mathbf{x}_j)$ is taken from $K(X_{train}, X_{train})[j, j]$.
- Normalise training and testing data together, i.e. normalise $K(X_{train+test}, X_{train+test})$, then extra the parts we need from the normalised matrix. This approach is suitable in a case where one already knows the whole of testing features. For centering, one should subtract the mean vector over the whole matrix $\Phi(X_{train+test})$. The unit norm normalisation is the same as in the previous case.

For our experiment, we fix the design space before training, i.e. the testing features are already known before testing. So we choose to normalise the kernel matrix over the training and testing data together, by first applying centering and then unit norm normalisation.

A.4 Batch Recommendation

For recommending RBS sequences to label, we consider the Upper Confidence Bound (UCB) algorithm, selecting RBS sequences with the maximum upper confidence bound at round t, i.e.

$$\text{argmax}_{\mathbf{x}_i \in \mathcal{D}} \left(\mu_{t-1}(\mathbf{x}_i) + \beta_t \sigma_{t-1}(\mathbf{x}_i)\right), \tag{2}$$

where β_t is a hyperparmeter balancing the exploitation and exploration, $\mu_t(\mathbf{x}_i), \sigma_t(\mathbf{x}_i)$ are the predicted mean and standard deviation at round t for the sequence $\mathbf{x}_i$.

Since labelling sequences is time-consuming, it is unrealistic to recommend sequence sequentially (i.e. one-by-one) and waiting for the label after each prediction. Therefore we consider recommending sequences in batch and using Gaussian Process Batch Upper Confidence Bound (GP-BUCB) algorithm [10]. With batches of size B, the feedback mapping $fb[t] = \lfloor (t-1)/B \rfloor B$, i.e.

$$\text{fb}[t] = \begin{cases} 0 & : t \in \{1, \ldots, B\} \\ B & : t \in \{B+1, \ldots, 2B\} \\ 2B & : t \in \{2B+1, \ldots, 3B\} \\ \vdots & \end{cases} \tag{29}$$

A key property of Gaussian Process regression is that the predictive variance in Eq. (7) only depends on observed points (i.e. features), but not on the labels of these observed points. So one can compute the posterior variance without actually observing the labels. The GP-BUCB policy is to select sequences that

$$\text{argmax}_{\mathbf{x}_i \in \mathcal{D}} \left(\mu_{fb[t]}(\mathbf{x}_i) + \beta_t \sigma_{t-1}(\mathbf{x}_i)\right). \tag{30}$$

And only update $y_{t'} = f(\boldsymbol{x}_{t'}) + \varepsilon_{t'}$ for $t' \in \{\text{fb}[t]+1, \ldots, \text{fb}[t+1]\}$ at the end of each batch ($\text{fb}[t] < \text{fb}[t+1]$). This is equivalent to sequential GP-UCB with *hallucinated observations* $\boldsymbol{y}_{\text{fb}[t]+1:t-1} = \left[\mu_{\text{fb}[t]}\left(\boldsymbol{x}_{\text{fb}[t]+1}\right), \ldots, \mu_{\text{fb}[t]}\left(\boldsymbol{x}_{t-1}\right)\right]$, while the posterior variance decreases.

A.5 Design Pipeline

The n + 1 round design is based on the nth round result, where each sequence has 6 replicates with TIR labels. We pre-processed the data by taking a logarithm transformation and standardisation of the raw TIR label for each replicates respectively, where TIR is calculated as a derivative of GFP fluoresence divided by OD600 of culture over 4h counting from the start of log phase of growth. After normalisation, each replicate has zero mean and unit variance.

For prediction, we use Gaussian process regression, with training on all normalised replicates and predicting on the design space (6-base core part design) except known sequences. We assume the observation are noisy, where the noise is under centered normal distribution with standard deviation α. We model the covariance matrix using the weighted degree kernel with shift. We normalise the kernel with centering and unit norm in terms of the whole kernel constructed by both first round result and design space. The hyperparameter for kernel, including maximum substring length l, maximum shift length s, and the noise standard deviation α of Gaussian process model are choose based on 10-repeat 5-fold cross validation. We choose $l = 6, s = 1, \alpha = 2$ for the second round design.

For recommendation, we use batch upper confidence bound introduced by GP-BUCB algorithm [10]. The upper confidence bound is constructed by predicted mean plus 2 predicted standard deviation. We recommend 90 sequences from the design space.

A.6 Intuition behind UCB and visualisation

- Exploitation and exploration explanation.
- Visualise coverage by clustering plot.
- Table for in-clustering mean and variance.

A.7 Result analysis

- violinplot.
- regression performance plot, table.
- kernel matrix plot.

A.8 Statistics of the core vs non-core regions of the RBS

In our study, we have tested a set of sequnces designed to confirm the notion that changing bases within the RBS core (6 bases) is statistically more influencing the TIR than the changes made outside the core. This hypothesis has been build based on reported biases towards certain bases present in the core of the RBS but absent outside of it. For example, according to [31] there is a strong bias towards A and G bases in the core region of the RBS. Similarly outside of the 6 bases of the core in the wider 20 bp context of the RBS there is no significant bias towards any particular base which suggest that these bases do not contribute to the overal TIR of a given RBS. This effect is shown in (Figure 9), which shows results for our set X of sequences. The value of Welch's t-test between the mean TIR in core and non-core groups is -4.8780 with p-value ¡ 0.0001 and 34 degrees of Freedom.

B Supplementary Figures

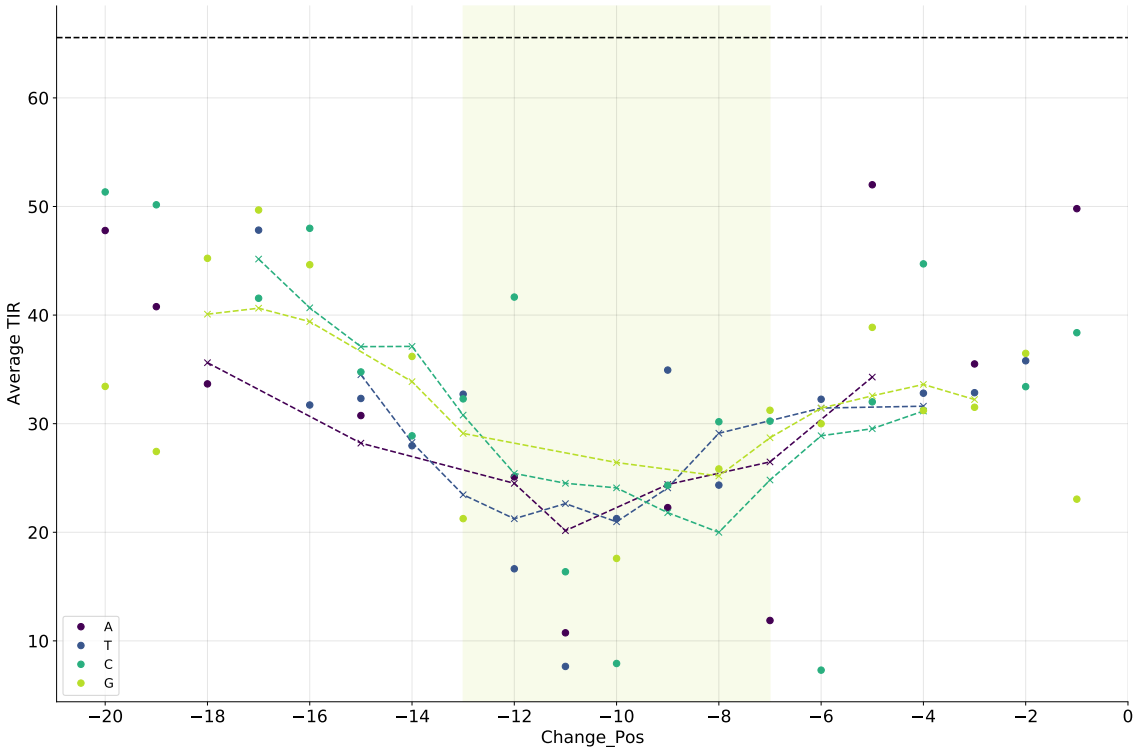


Figure 9: **Comparison of base change impact on TIR in core versus non-core region.** The core region is highligthed in light green and the lines are rolling averages for each base. The top dotted line shows the TIR for the benchmark sequence, where dots represent a change at a given position to a given base, which is colour coded.

C Mixture of Absolute Gaussian

The raw TIR scores in each round are positive and have two modes. We assume they are mixture of two absolute Gaussian distributions. If we can fit the mixture of absolute Gaussian distributions, we can answer the following questions:

- is the left peak decreasing in height with each round?
- is the left peak going left?
- Is the right peak increasing in height with each round?
- is the right peak going right?

The density of absolute Gaussian (also called *folded normal distribution*) is

$$f_Y\left(x;\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} + \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x+\mu)^2}{2\sigma^2}} \tag{31}$$

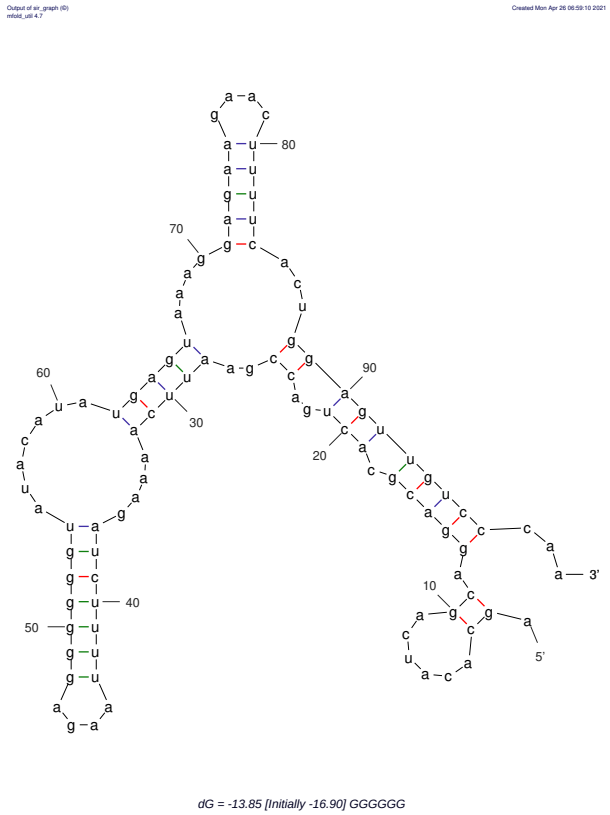
for $x \geq 0$ and 0 everywhere else. The mean and variance of the absolute Gaussian distribution is

$$\mu_Y = \sigma\sqrt{\frac{2}{\pi}} \exp\left(\frac{-\mu^2}{2\sigma^2}\right) + \mu \operatorname{erf}\left(\frac{\mu}{\sqrt{2\sigma^2}}\right); \tag{32}$$

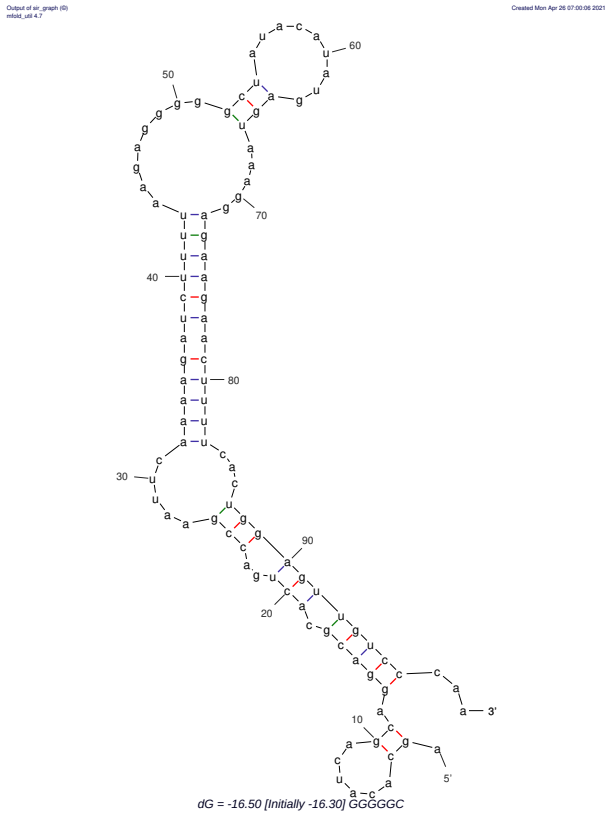
$$\sigma_Y^2 = \mu^2 + \sigma^2 - \mu_Y^2, \tag{33}$$

where erf is the error function $\operatorname{erf} z = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$.

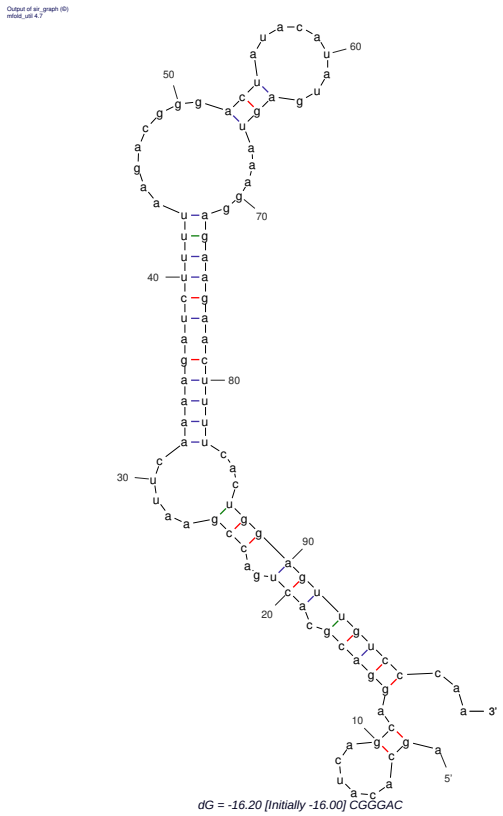
We consider the model of the mixture of two absolute Gaussian distributions as



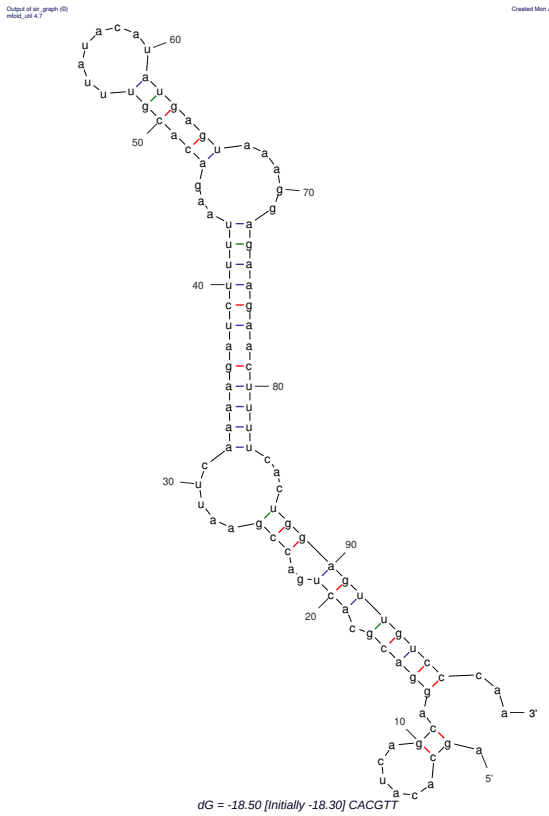
(a) $y = x$



(b) $y = 3sinx$



(c) $y = 5/x$



(d) $y = 5/x$

Figure 10: **Folding predictionds for four different RBS.** Here we show the predicted, energetically most favourable structures for four of our RBS with the following cores: GGGGGG, GGGGGC, CGGGAC, CACGTT and their immediate upstream and downstream background sequence. There is no discerbile and consistent difference between strong ones (the first two) and weak ones (the last two).