Machine Learning guided workflow for Ribosome Binding Site engineering

Zhang M.[1,2,4], 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]Department of Computer Science, Australian National University
[3]CSIRO Synthetic Biology Future Science Platform, CSIRO Land and Water
[4]Data61, CSIRO

May 28, 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.

1 Introduction

One of the main tenets of synthetic biology is design, evaluation and standardisation of genetic parts [4, 6, 34]. 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 [38, 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) [37, 7, 27]. 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. [27] there are three main RBS calculators, all predicting the TIR based on the thermodynamic properties of the RBS and the ribosome [31, 22, 29]. 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 [17, 30]. 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 datasets 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, 36]. In the past few years there was a significant uptake of machine learning based approaches in synthetic biology [18]. Jervis *et al.* used support vector machines and neural network to optimise production of monoterpenoid in *Esherichia coli* [16]. 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 datasets [1, 2].
However, the use of machine learning in synthetic biology is still in its infancy and will require additional research to show its full potential. The challenges of applying machine learning techniques on synthetic biology have two folds. On the one hand, there is no large-scale and high-quality labelled data. The available RBS library [15] only covers 56 unique labelled RBS. And it is time-consuming and expensive to measure the TIR for RBS sequences. On the other hand, while the goal is to find highest TIR among the design space, one also need to explore as much as possible to learn the whole design space to avoid getting stuck in the local maximum. We addressed the challenges by applying DBTL workflow. Instead of querying labels for all RBS sequences in the design space (4,138 in total), we recommend RBS sequences to query in batches and learn the prediction model in an online form. Particularly, we use bandits algorithm for recommending RBS sequences to address the exploitation-exploration balance.
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 the goal of optimising associated protein expression level. 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 [25] and Upper Confidence Bound multi-armed Bandits algorithms [10] for prediction and recommendation respectively to analyse and optimise the initiation rates of the designed RBS . Our overall experimental goal was to maximise the Translation Initiation Rate (TIR) by querying the batches of RBS sequences while minimising the number of DBTL cycle turns that we had to do. We did this by designing a sequential experimental workflow shown in Figure 1. In the zeroth round, randomised RBS sequences and preliminary machine learning recommended designs based on literature were designed to explore the experimental space. In the subsequent rounds, designs were 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 from between 95 to 135% of TIR of our chosen very strong benchmark RBS.

2 Results

2.1 The experimental workflow

We show our DBTL workflow in Figure 1. BUILD and TEST are driven chiefly by choices made by human researchers and use of automated methods. Machine learning algorithms are applied in LEARN and DESIGN. In LEARN phase, we use the Gaussian Process regression algorithm to predict the TIR of RBS sequences comprising the experimental space. The goal is to provide predictions about TIR labels and uncertainty to assist the recommendation process. In the DESIGN phase, we have used a multi-armed Bandit recommendation algorithm that was selecting RBS designs to be tested in subsequent experimental batches.

To help us obtain reliable and reproducible results we have employed automation-heavy workflow in the BUILD and TEST phases. 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 the time required to prepare our variants.

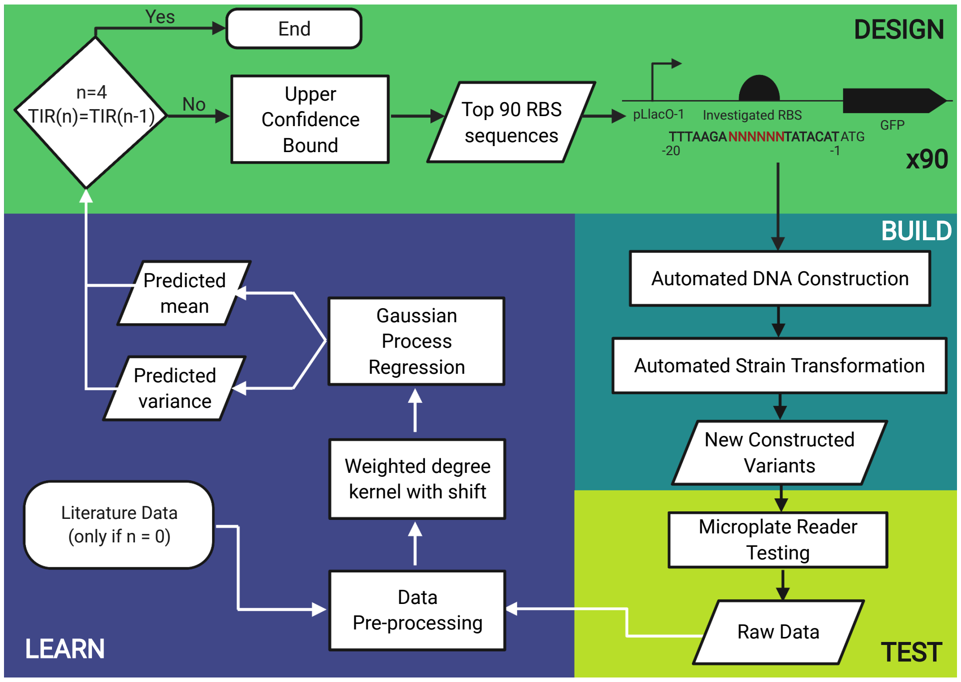


Figure 1: **Flowchart of machine learning based experimental design.** The RBS design is recommended by the Upper Confidence Bound Bandit algorithm. After generating the recommendations the RBS are built and tested using automated laboratory methods allowing for rapid construction and testing at scale.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

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.

2.2 Design of the investigated genetic device

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).
Our template RBS sequence, is 20 bps long with the sequence TTTAAGA**AGGAGA**TATACA. This sequence is a known to have high TIR and comes with the pBb series plasmids [20]. 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 benchmark sequence, i.e. TTTAAGA + NNNNNN + TATACAT, where N can be any choices of A, C, G, T. The total experimental (variant) space is then $4^6 = 4096$. We have experimentally confirmed that changing the core sequence is statistically more impactful on TIR than changes made outside of it (see Supplementary materials).

2.3 Performance of the recommendation algorithm

The design recommendations were made using the Multi-armed Bandit algorithm. In short, 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 (see below). Another feature of the bandits algorithm is that it balances two parts: the exploration of the unknown (untested) parts of the design space where high TIR RBS can be hidden, and exploitation which goal is querying areas which are known to give relatively high TIRs. One thing of note is that the bandit algorithm is stochastic, that is it exploits the probabilities of a given event occurring (in this case RBS having a specific TIR). As such, it pairs naturally with our prediction algorithm, the Gaussian Process Regression, 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:

- BPS-NC and BPS-C group: 60 RBS sequences which are subsequent single nucleotide variations 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 (Figure 8).

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

- PPM group: 30 RBS sequences randomised based on the position probability matrix (PPM) generated from all the naturally occurring RBS sequences in *E. coli* genome [35].

- Bandit-0: 60 RBS sequences recommended by our implementation of recommendation algorithm based on a data set obtained from literature [15], 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 2A shows the results for all the examined groups. In each round, we measure the TIR of benchmark RBS as the internal standard. We then obtain the normalised TIR (called *TIR ratio*) by taking the ratio between the raw TIR and the average TIR of benchmark sequences in each round (which are run in triplicate in each round). All Round 0 groups (BPS-NC, BPS-C, UNI, PPM, Bandit-0) have performed worse than our benchmark sequence in terms of TIR. The Bandit-0 group performed poorly, despite being machine learning driven, due to being trained on literature data. 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 better than the consensus Shine-Dalgarno sequence 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. Figure 2B shows the same results but divided into quantiles where the specific point for a given group is showing the highest TIR for that quantile. The gradual increase for all quantiles can be observed for all Bandit groups suggesting algorithms' better understanding of the experimental space with more data. The decreased result in the 0.9th qunatile compared to the max value for Bandit 3 group can be attributed to the increased emphasis on exploitation that has been set for the that round compared to others. We see that effect in Figure 2 c), where we coloured the data points for Bandit 1-3 groups according to their relative exploration - exploitation affinity. Those with high predicted mean are coloured blue and represents exploitation, those hued red are with high predicted uncertainty and represent exploration. We can see the RBSs with high TIRs tend to come from exploitation the design space whereas the explorative points give relatively low TIR but expand our knowledge about the unknown part of the design space. Figure 2D shows the TIRs of RBSs tested in the Bandit groups divided into bins of width of 0.1 TIR ratio. KDE plots have been overlaid

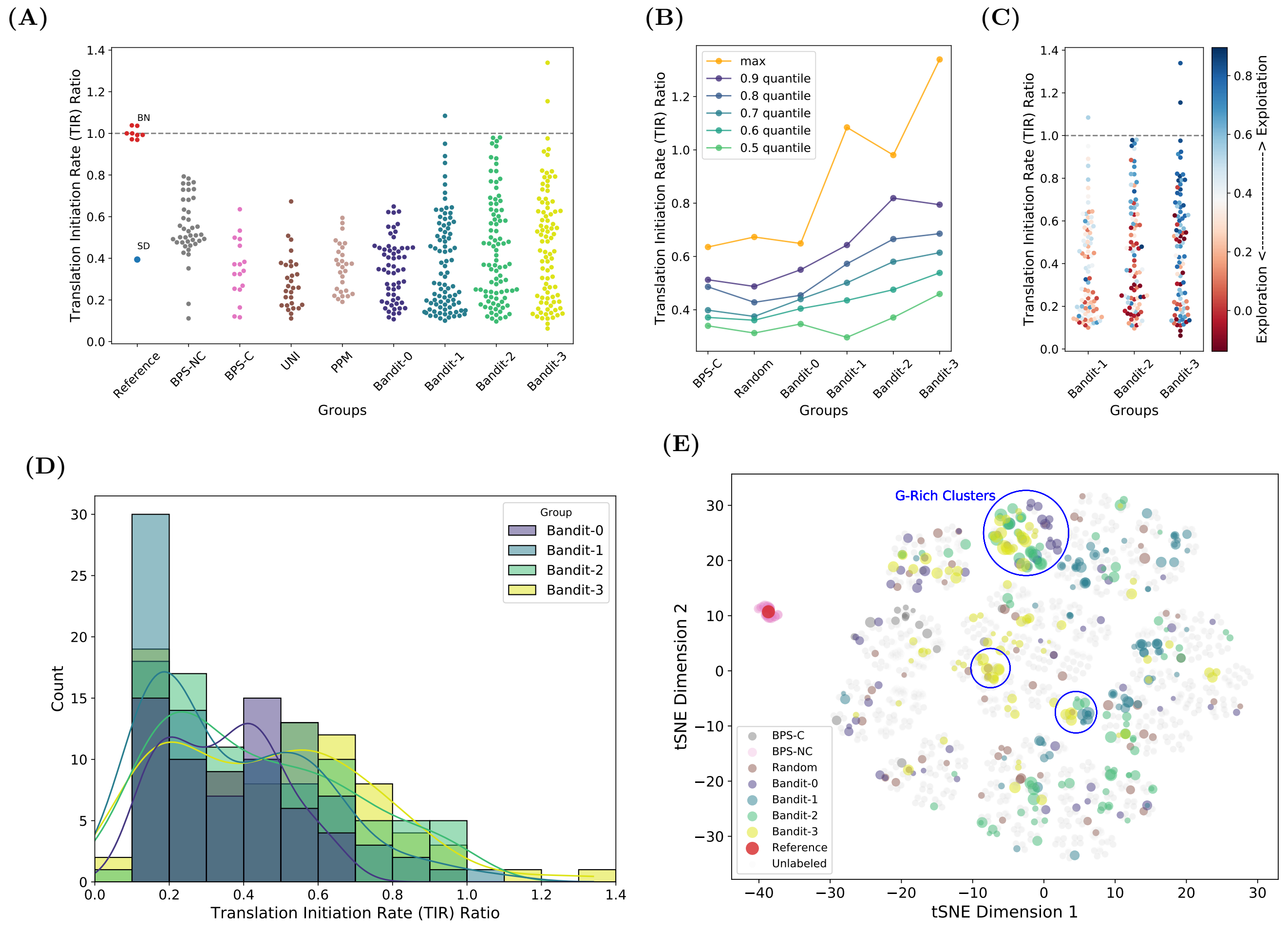


Figure 2: **TIRs of RBS groups examined in this study.** The TIR results in all subplots are shown normalised to the respective benchmark sequence sample which acts as internal standard, that is TIR of a given RBS is divided by TIR of the benchmark RBS run in the same plate. **A)** Swarm plot showing 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*. Bandit-0/1/2/3 - Bandit algorithm generated results for Round 0, 1, 2 and 3 respectively. SD - Shine-Dalgarno sequence. Dash line is set to 1 and represents the averaged benchmark sequence TIR for that group. BN - benchmark sequences for all plates. They are not all exactly 1 due to them being shown as separate samples rather than averages. **B)** Line plot showing TIR obtained in a given quantile of results divided into groups as in A). UNI and PPM are merged into Random group and BPS-NC was removed due to changes being made outside the core in this group. **C)** Exploitation v.s. Exploration for Bandit 1-3. Blue-hued points represent exploitation, those hued red represent exploration. **D)** Histogram with kernel density estimations (KDE) showing distributions of TIRs for Bandit groups. **E)** tSNE plot showing the relative distances between sequences in our design spaces as calculated by our kernel function (weighted degree kernel with shift). The area of the circle represents the experimentally obtained TIR for measured groups.

to depict the calculated density for each group. The increase of prevalence of later bandit groups in the higher bins is evident, especially for Bandit 2 and 3 constituting the bulk of results in the > 0.8 TIR ratio bins. Notably, the distributions calculated for all the groups show bivariate distribution, we discuss the possible reasons for that further in the text.

In 2E we show a tSNE plot depicting the experimental space. Each RBS is located on the plot according to its distance to other RBSs as calculated by our kernel function. We can see the RBSs recommended by Bandit groups have covered majority of the design space. A number of clusters were targeted by our recommendation algorithm. For example, the circled clusters labelled as "G-Rich Clusters" have been actively recommended by the algorithm. More specifically, sequences with more or equal to 4 guanines in any position constituted 10% of the randomly selected sequences and 5, 9, 16 and finally 25% in each of the 4 Bandit guided batches respectively.

2.4 Prediction of RBS performance

Our recommendation algorithm selects new designs based on means and uncertainties calculated by 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 [33, 28]. 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 [25] is the kernel (covariance) function, which captures the similarity between data points, 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* [21] to process the core 6bp and dot product kernel [25] (with one-hot embedding) to process the 7bp flanking sequences both upstream and downstream of the core sequence. For subsequent rounds, we used the *weighted degree kernel with shift* (WDS) [26], which has been shown to have good performance in various prediction tasks [3]. The WDS not only counts the matches of substrings of a certain length (i.e. kmers), but also takes into account the positional information and shifting of substrings. For example, two sequences A**CCTGA** and **CCTGA**A are in 1-shift. The relative distances between sequences calculated by our kernel function are shown in figure 2E.

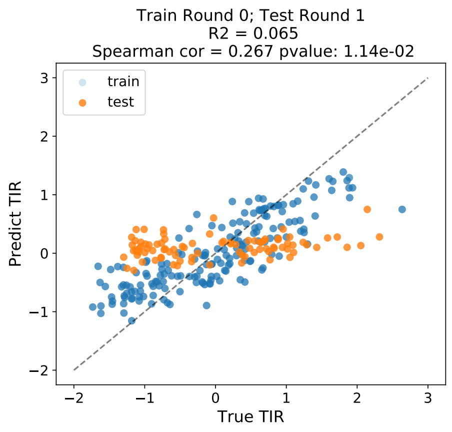
Figure 3 shows how our algorithm performed in terms of predictions in each round. As expected, the predictions in Round 0 were poor due to use of approximated data. The predictions improved for the subsequent rounds, from R^2 of 0.065 for round 0 to R^2 of 0.27 for round 3. Similarly, the Spearman correlation coefficient rose from 0.27 for Round 0 to 0.48 for Round 3. One important point to note is that the predictions are also influenced by our recommendation choices. In each round, we select a number of data points for exploration, which means that these data points, when tested, have a high chance of having real mean different to what was predicted. However, this is still very useful information for future predictions as it allows us to understand the underlying space better.

2.5 Characteristics of the tested sequences

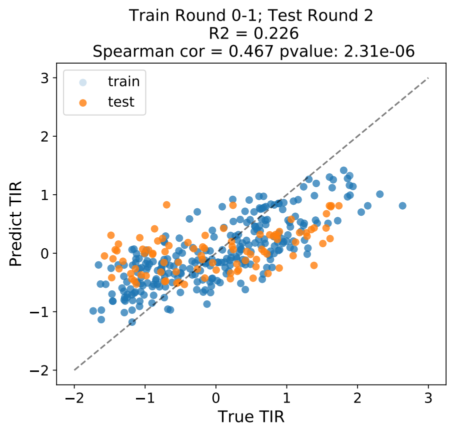
Our data set taken together can be viewed as a reliable library of RBS sequences for *E. coli*, some characteristics of which are shown in table 1. Figure 4A shows the sequence logo calculated for the Top 30 (in terms of TIR ration) of our sequences. It is generally understood that guanine rich sequences are promoting strong transcription. This expected bias towards guanine is clearly visible for all positions in our Top 30 RBSs. This result combined with the Bandits' algorithm bias towards the G rich cluster shown in figure 2D reinforces the notion that our algorithm successfully identified G rich sequences as ones with high TIR probability.

Another interesting characteristic uncovered by our research is the perceived editing distance between two sequences required for improvement in the TIR when the given RBS' TIR is already high. 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 that they are two completely different sequences). Figure 4B shows what edit distance is required for positive change in TIR for RBS with TIR > 0.75. For RBS with high TIR (> 1), the minimum distance that is required for increase of TIR is 2, with edit distance between 2 and 5 giving similar results. For RBS with medium TIR (< 1), a distance of 1 is enough to produce a meaningful increase in TIR. That means that as the TIR of examined RBSs increases, exploring sequences which are more dissimilar to the current candidates tends to give meaningful improvement. The low rate of natural mutations will be very slow to explore more dissimilar sequences on such a short distance [19], which indicate that Adaptive Laboratory Evolution may not be able to find very strong RBSs within limited budget. In other words, because the examined sequence is relatively short (6bp in a wider 20bp context) the time to accumulate 2 or more changes in the RBS region required for meaningful increase in TIR might be prohibitively long. In such cases, a directed process

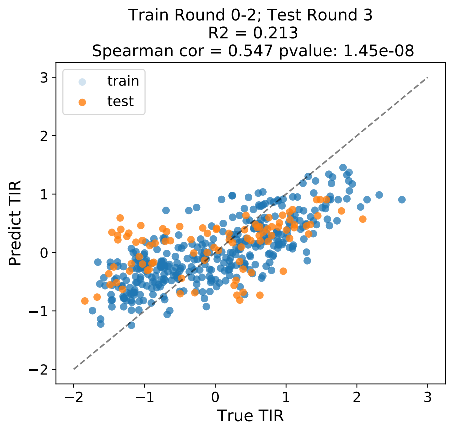
(A)



(B)



(C)



(D)

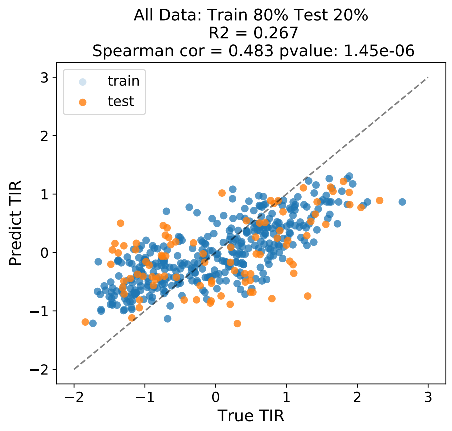


Figure 3: **Predictions generalise to held out data.** The scatter plots A-D are showing the performance of our prediction algorithm (GPR with WDS kernel, with maximum k-mer as 6 and maximum shift as 1) calculated after each of the rounds. Note the TIR values are normalised TIR based on the standardisation described in section 4.2.1, which is different from the TIR ratio reported in the Figure 2. The values of R^2 and Spearman correlation coefficient (with corresponding p-value) are all provided for each plot. The p-value is for the hypothesis test whose null hypothesis is that two sets of data are uncorrelated. The p-value decreases with round increases, which shows increasing strong evidence about the Spearman correlation between our prediction and measured TIR.

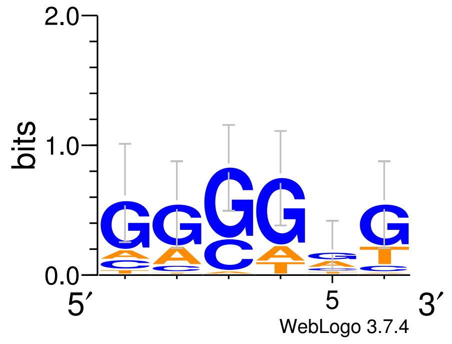
should be strongly encouraged. This is in line with common practices in e.g. protein engineering, where similar approaches, that is making more directed changes, are often observed [14].

Finally, our strong sequences did not show neither strong binding to the anti-sense sequence of the ribosome known to bind to RBS nor any obvious secondary structures that could explain their TIRs (see Supplementary). This result combined with the unexpectedly bimodal nature of KDEs in Figure 2 reinforces the notion based on the previously reported literature [17, 11] that there may be a number of different mechanisms governing the probability of effective RBS-ribosome binding.

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

(A)



(B)

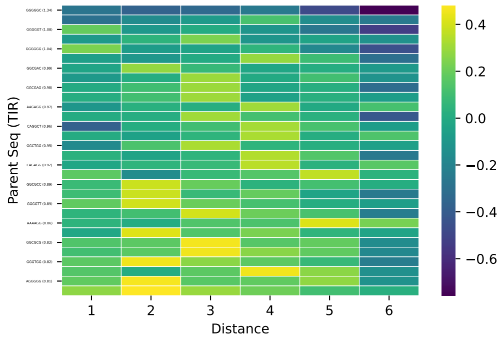


Figure 4: **Characteristics of strong RBSs.** A) Sequence logo calculated for the Top 30 tested sequences. B) Heatmap showing what edit (Hamming) distance is required for positive change in TIR for RBS with high and medium TIR. The temperature scale shows the difference between a given RBS on y axis and the RBS with strongest TIR at the given distance. Every second RBS is labelled for increased legibility.

Characteristics of the library	**Statistics**
Total experimental space	4096
Planned constructs	450
Successfully constructed	445
Sequences with CV<40%	79%
Sequences with CV<20%	27%
Efficiency of bandit design (compared with random)	2
Raw TIR range	[4.926, 105.377]
TIR ratio range	[0.063, 1.339]

Top RBS Core	**TIR Ratio**
GGGGGC	1.339
GGGGGT	1.154
GGCTAT	1.084
AGGAGA	1
GGCGTT	0.981
GGGGGG	0.979
GGCGAC	0.976
CAGGAG	0.963
GGCGAG	0.952
AGGAGG	0.394

Table 1: **Characteristics of the library.** Left table presents some of the characteristics of our library. Right table presents 10 RBS sequences with their corresponding TIR ratios, the first 9 being the strongest sequences including the benchmark sequence (AGGAGA) and the last being the Shine-Dalgarno sequence (AGGAGG). CV is coefficient of variation (STD of a sample divided by its mean). Efficiency of the bandit design is calculated by dividing the highest TIR found using machine learning by highest TIR found using random sequence generation.

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 Weighted Degree Kernel with Shift method, which fit well into our prediction method. We building and testing, we have performed bulk of our experiments using automation to increase their speed, reliability and reproducibility. By using our workflow, we have found very strong RBS sequences and we have generated an extensive library of diverse RBS that can be used in the future studies. We have achieved this despite the relatively low accuracy of our predictions, which means that the presented algorithms are robust and able to identify the right signal in a noisy environment.

There are still open questions that need to be addressed for applying machine learning in synthetic biology. For example, how can we understand the language of RBS sequences better? In other words, can we extract more biologically important information from the decisions made by our algorithms? Another example would be - given a small amount of RBS sequences tested, how can machine learning algorithm provide more accurate predictions and uncertainty measurement? Finally, another question with high impact on the performance of the workflow is - how should we choose the parameters which controls the exploration-exploitation rate?

We have found our approach bringing machine learning and synthetic biology experts very powerful. We envision to that the pairing of machine learning with high-throughput automation will keep delivering a high number of good quality datasets nad improved methods for biological engineering.

In the future, we hope to extend the algorithm to other, more complicated genetic elements. This could include, for example, promoters and terminators. However, its important to note that 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.

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 10 μL. 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. 8

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

Two types of machine learning algorithms can be applied to drive the experimental design workflow shown in Figure 1. One type of machine learning algorithms is prediction (**LEARN**), which helps us learn for function of TIR with respect to RBS sequences. The other type of machine learning algorithm is recommendation (**DESIGN**), which recommends RBS sequences to query in each batch based on the predictions from LEARN. In round t, the prediction and design is based on the results obtained in all previous rounds. The implementation of the machine learning algorithms is in Python 3.7 and scikit-learn library [24]. In the following, we describe our machine learning pipeline (Bandit 1-3) by the order of data pre-processing, prediction and string kernels, recommendation. The pipeline for Bandit-0 is reported in Supplementary A.

4.2.1 Data pre-processing

For each RBS sequence, we measured the TIR of 6 biological replicates, 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. For label pre-processing, we first adjust the TIR values in each round by the round-wise reference values. This is because, we found that TIR values in each round for the same sequence can be different (for reference RBS, STD $\approx$ 17.7) due to the measurement noise. So we repeatedly measured the reference RBS in each round to rectify the precision of other measured RBS in the same round. More precisely, in round t, we subtract the TIR mean of reference RBS measured in round t from all TIR values measured in the same round, for each replicates respectively. We then pre-processed the data by taking a logarithm transformation and standardisation of the adjusted TIR label for each replicates respectively, After normalisation, each replicate has zero mean and unit variance. Furthermore, we also normalised the kernel matrix used for prediction by centering and unit-norm normalisation, which is reviewed in details in Supplementary B.2.1.

4.2.2 Prediction: 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 $f : \mathcal{D} \rightarrow \mathbb{R}$, 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 Gaussian random noise with unknown mean and variance.

For regression model, we have used a Bayesian non-parametric approach called *Gaussian Process Regression (GPR)* [25]. 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 function (also called *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 standard deviation. The posterior standard deviation represents the level of uncertainty for the prediction. We review the predictions of GPR in Supplementary B.1.

The choice of kernel function is critical for accurate predictions, since it controls smoothness and amplitude of the function to be modelled. To represent the RBS sequences and formulate the similarity between sequences, we use the *weighted degree kernel with shift* (WDS) [26] to specify the kernel function of GP. WDS is a type of string kernels, which take two sequences as inputs and output 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. This could be useful when there is a shift between two sequences. For example, two sequences A**CCTGA** and **CCTGA**A are in 1-shift.

We now define WDS kernel. Let $\mathbb{I}(A)$ be the indicator function, which equals 1 if A is true and 0 otherwise. Then $\mathbb{I}(\mathbf{x}_{[l+s:l+s+d]} = \mathbf{x}'_{[l:l+d]})$ indicates whether two substrings of length d matches, 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

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

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 chose to use a uniform weighting over the sequences, i.e. $\gamma_l = 1/L$. $S(l)$ determines the maximum shift at position l. In our design, the length of RBS sequence $L = 20$. we chose $\ell = 6$, and $S(l) = 1$ for all l.

4.2.3 Recommendation: Batch Upper Confidence Bound Bandit

To recommend the RBS sequences to query in next round, we have used the *Upper Confidence Bound (UCB)* batch algorithm [33], which provides such *exploitation-exploration balance* by balancing the predicted mean and standard deviation. On the 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 standard deviation). More precisely, UCB algorithm selects RBS sequences with the maximum upper confidence bound at round t, i.e.

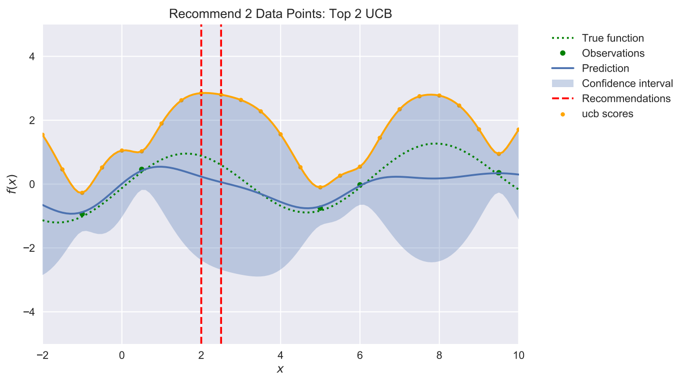
$$\text{argmax}_{\mathbf{x}_i \in \mathcal{D}} \left(\mu_{t-1}(\mathbf{x}_i) + \beta_t \sigma_{t-1}(\mathbf{x}_i) \right), \quad (2)$$

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 $\mathbf{x}_i$. 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.

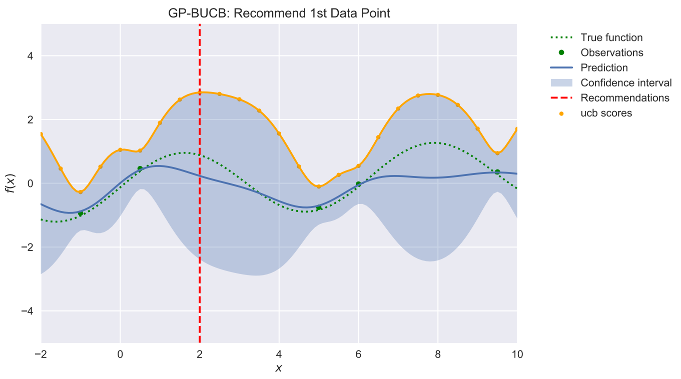
Since experimentally labelling sequences is time-consuming, it is unrealistic to recommend sequence sequentially (i.e. one-by-one) and then wait for the label to be tested and used to improve the model. Instead, we can recommend RBS sequences in a batch of size m. One naive approach is to recommend sequences in design space with top m UCB scores, as shown in Figure 5(A) ($m = 2$). 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 GPR assumes 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.

A key property of Gaussian Process regression is that the predictive standard deviation depends only on observed points (i.e. features), but not on the labels of those observed points. One can make use of this property to design batch upper confidence bound (BUCB) algorithm [10]. The strategy is to recommend sequences sequentially by updating the UCB score with the updated predicted standard deviation with the previously recommended points. As illustrated in Figure 5 (B), the algorithm recommends the data point with maximum UCB score based on the predictions over initial 5 observations. Then we add the recommended data point ($x = 2$) into the training dataset

(A)



(B)



(C)

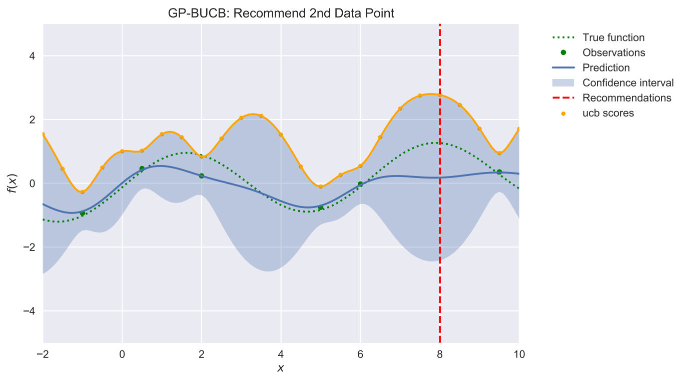


Figure 5: Batch Recommendation Illustration. We use the batch size of 2, with 5 initial observations. The design space is 24 uniformly distributed points in the range [-2,10], i.e. -2, -1.5, -1, ..., 9.5, 10. The confidence interval are shown with predicted mean of ± 1.96 standard deviation. (A) Top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

with the predicted mean of that point as label (note it is not the true label, i.e. observation), and update the predicted standard deviation and then we finally update the UCB scores. The second data point is then recommended 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 standard deviation 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 ($x = 8$). As summary, the exploration efficiency is improved since the recommended sequences in one batch will tend to be in different design area so that the information gain is maximised.

Code and data availability

All code and data required to reproduce the results is available at Github: `https://github.com/mholowko/SynbioML`

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. Holowko M. B. and Ong C. S. conceived and planned the project. All authors analysed the data, contributed to and reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Acknowledgments

The authors would like to acknowlege CSIRO's Machine Learning and Artificial Intelligence, and Synthetic Biology Future Science Platforms for providing funding for this research. The authors would also like to thank CSIRO BioFoundry for help with performing the experiments.

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 W Burdick. "Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization". In: *Journal of Machine Learning Research* 15 (2014), pp. 3873–3923.

[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] Christian Jäckel, Peter Kast, and Donald Hilvert. "Protein Design by Directed Evolution". In: *Annual Review of Biophysics* 37.1 (2008). PMID: 18573077, pp. 153–173. DOI: 10.1146/annurev.biophys.37.032807.125832. eprint: https://doi.org/10.1146/annurev.biophys.37.032807.125832. URL: https://doi.org/10.1146/annurev.biophys.37.032807.125832.

[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 (2018), pp. 127–136.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[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] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[25] 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.

[26] 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.

[27] 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/.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] Ryan K Shultzaberger et al. “Anatomy of Escherichia coli ribosome binding sites”. 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`.

[33] 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.

[34] 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`.

[35] 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`.

[36] Radivojević T et al. “ A machine learning Automated Recommendation Tool for synthetic biology”. In: *Nat Commun.* 25.11 (2020), p. 4879. DOI: `10.1038/s41467-020-18008-4`. URL: `https://www.nature.com/articles/s41467-020-18008-4`.

[37] 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`.

[38] 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 Machine Learning Design Pipeline

In this section, we report in details about our machine learning design pipeline. The raw 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.

explain somewhere why we have a different pipeline for round 0. add some analysis on literature data? to verify 1) our library is "better" 2) it's hard to do ML on available library. some analysis in my mind (for literature data): number of data points, number of replicates (I remembered there only one replicates, if not, then std), label histogram, coverage of RBS. And we need to address the label in literature data and ours is "different", in the way that our pipeline is different for bandit-0 design

Bandit-0: the design of round-0 is based on the literature data [15]. We first normalise the raw TIR to values between 0 and 1. We applied the Gaussian Process Regression with noise parameter $\alpha = 1e - 10$. We chose to use one of the basic string kernels, the *spectrum kernel* [21] to process the core 6bp and dot product kernel [25] (with one-hot embedding) to process the 7bp flanking sequences both upstream and downstream of the core sequence. The design size is 60 with UCB parameter $\beta = 1$.

Bandit-1 to Bandit-3: the t + 1 round design is based on the t^th $(t \geq 1)$ 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. 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.

B Machine Learning Methods

B.1 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 $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. We consider the case where the observations are noisy, i.e. $\{(\mathbf{x}_i, y_i) | i = 1, \dots, n\}$, 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 $\operatorname{cov}(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \alpha^2 \delta_{pq}$, 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

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

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 $\mathbf{f}_* | X, \mathbf{y}, X_* \sim \mathcal{N}(\overline{\mathbf{f}}_*, cov(\mathbf{f}_*))$, where

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

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

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).

B.2 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.*

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

  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

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

- *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 $\beta_d = \frac{2(\ell-d+1)}{\ell(\ell+1)}$,

$$\begin{aligned} k_\ell^{WD}(\mathbf{x}, \mathbf{x}') &= \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l k_d^{Spec}(\mathbf{x}_{[l:l+d]}, \mathbf{x}'_{[l:l+d]}) & (10) \\ &= \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \phi_d^{Spec}(\mathbf{x}_{[l:l+d]})^T \phi_d^{Spec}(\mathbf{x}'_{[l:l+d]}) & (11) \\ &= \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \mathbb{I}(\mathbf{x}_{[l:l+d]} = \mathbf{x}'_{[l:l+d]}), & (12) \end{aligned}$$

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

- *Weighted Degree Kernel With Shift.*

$$\begin{aligned} k_\ell^{WDS}(\mathbf{x}, \mathbf{x}') &= \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \sum_{s=0, s+l \le L}^{S(l)} \delta_s \left(k_d^{Spec}(\mathbf{x}_{[l+s:l+s+d]}, \mathbf{x}'_{[l:l+d]}) + (k_d^{Spec}(\mathbf{x}_{[l:l+d]}, \mathbf{x}'_{[l+s:l+s+d]})\right) & (13) \\ &= \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \sum_{s=0, s+l \le L}^{S(l)} \delta_s \left(\mathbb{I}(\mathbf{x}_{[l+s:l+s+d]} = \mathbf{x}'_{[l:l+d]}) + (\mathbb{I}(\mathbf{x}_{[l:l+d]} = \mathbf{x}'_{[l+s:l+s+d]})\right), & (14) \end{aligned}$$

  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:

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

B.2.1 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

$$\phi^C(\mathbf{x}_i) = \phi(\mathbf{x}_i) - \bar{\Phi}(X) = \phi(\mathbf{x}_i) - \frac{1}{n'}\sum_{s=1}^{n'} \phi(\mathbf{x}_s). \tag{15}$$

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

$$k^C(\mathbf{x}_i, \mathbf{x}_j) =< \phi^C(\mathbf{x}_i), \phi^C(\mathbf{x}_j) > \tag{16}$$

$$= \left(\phi(\mathbf{x}_i) - \frac{1}{n}\sum_{s=1}^{n} \phi(\mathbf{x}_s)\right)^T \left(\phi(\mathbf{x}_j) - \frac{1}{n'}\sum_{s'=1}^{n'} \phi(\mathbf{x}_{s'})\right) \tag{17}$$

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

$$= k(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{n}\sum_{s=1}^{n} k(\mathbf{x}_s, \mathbf{x}_j) - \frac{1}{n'}\sum_{s'=1}^{n'} k(\mathbf{x}_i, \mathbf{x}_{s'}) + \frac{1}{n^2}\sum_{s=1}^{n}\sum_{s'=1}^{n'} k(\mathbf{x}_s, \mathbf{x}_{s'}) \tag{19}$$

- 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

$$\phi^{UN}(\mathbf{x}_i) = \frac{\phi(\mathbf{x}_i)}{||\phi(\mathbf{x}_i)||}. \tag{20}$$

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

$$k^{UN}(\mathbf{x}_i, \mathbf{x}_j) =< \frac{\phi(\mathbf{x}_i)}{||\phi(\mathbf{x}_i)||}, \frac{\phi(\mathbf{x}_j)}{||\phi(\mathbf{x}_j)||} > \tag{21}$$

$$= \frac{\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)}{||\phi(\mathbf{x}_i)|| \times ||\phi(\mathbf{x}_j)||} \tag{22}$$

$$= \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{k(\mathbf{x}_i, \mathbf{x}_i)k(\mathbf{x}_j, \mathbf{x}_j)}} \tag{23}$$

- 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

$$\phi^{UV}(\mathbf{x}_i) = \frac{\phi(\mathbf{x}_i)}{\sqrt{Var(\Phi(X))}}. \tag{24}$$

The corresponding kernel representation is

$$k^{UV}(\mathbf{x}_i, \mathbf{x}_j) =< \frac{\phi(\mathbf{x}_i)}{\sqrt{Var(\Phi(X))}}, \frac{\phi(\mathbf{x}_j)}{\sqrt{Var(\Phi(X'))}} > \tag{25}$$

$$= \frac{\phi(\mathbf{x}_i)^T \mathbf{x}_j}{\sqrt{Var(\Phi(X))Var(\Phi(\mathbf{X}'))}} \tag{26}$$

$$= \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{\frac{1}{n}\sum_{s=1}^{n} k^C(\mathbf{x}_s, \mathbf{x}_s)\frac{1}{n}\sum_{s'=1}^{n'} k^C(\mathbf{x}_{s'}, \mathbf{x}_{s'})}} \tag{27}$$

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.

B.3 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.

$$\text{fb}[t] = \begin{cases} 0 & : t \in \{1, \ldots, B\} \\ B & : t \in \{B+1, \ldots, 2B\} \\ 2B & : t \in \{2B+1, \ldots, 3B\} \\ & \vdots \end{cases} \tag{28}$$

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

$$\text{argmax}_{\mathbf{x}_i \in \mathcal{D}} \left(\mu_{fb[t]}(\mathbf{x}_i) + \beta_t \sigma_{t-1}(\mathbf{x}_i)\right). \tag{29}$$

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.

C Supplementary Plots

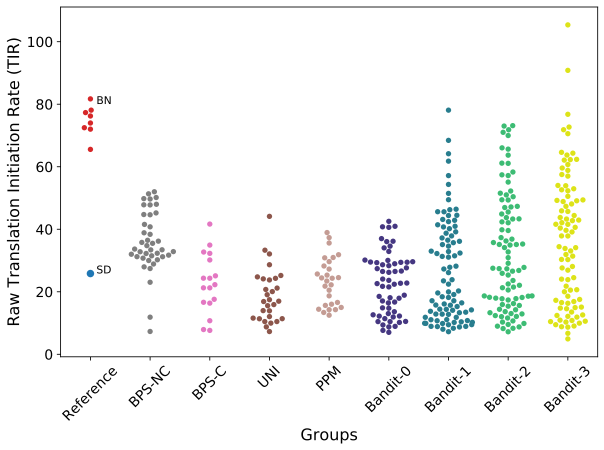


Figure 6: Swarmplots for different groups, with raw TIR labels (averaged over 6 replicates). Group names represent: Consensus (consensus sequence tested in different round); BPS-NC (bps noncore); BPS-C (bps core); UNI (uniformly random); PPM (position-based probability matrix); Bandit-0 (bandit design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

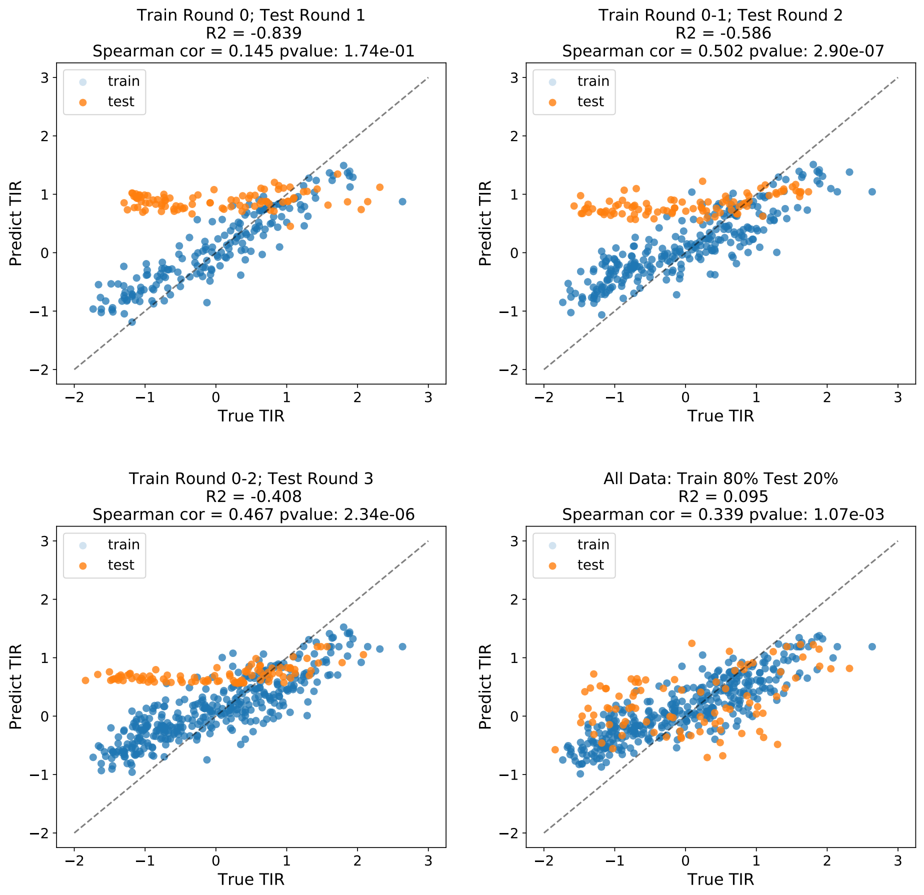


Figure 7: scatter abc1 TT

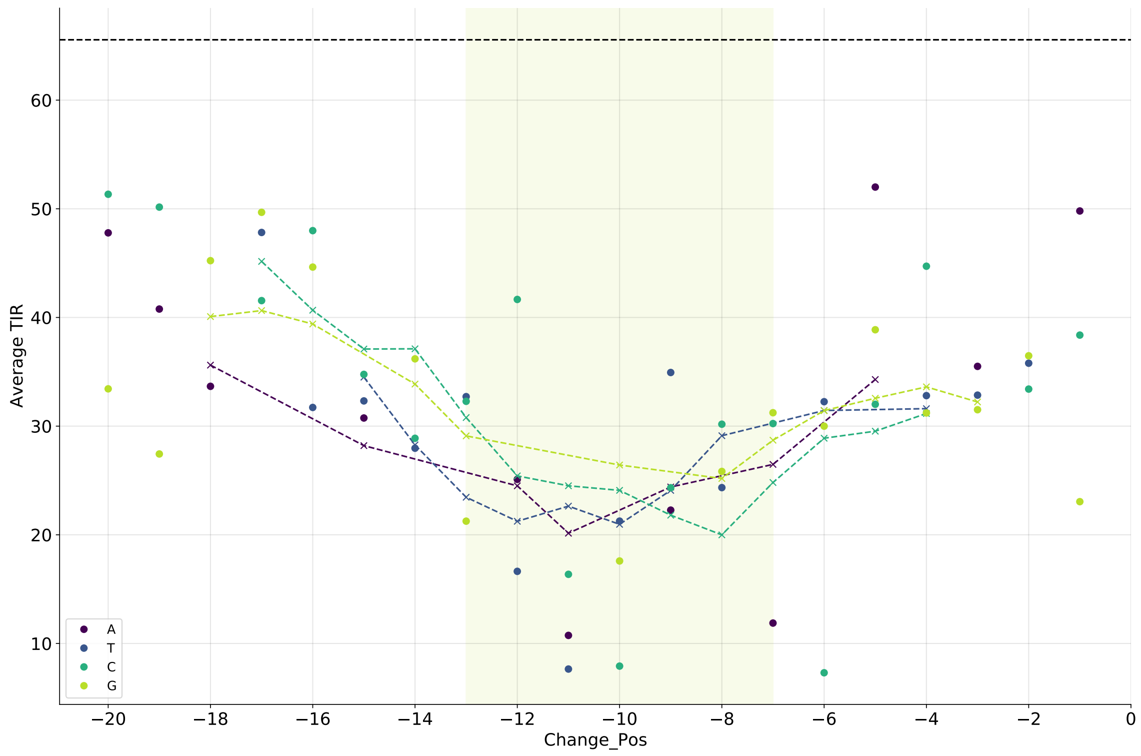


Figure 8: **Comparison of base change impact on TIR in core versus non-core region.** The core region is highligthed in light green and the lines are rolling averages for each base. The top dotted line shows the TIR for the benchmark sequence, where dots represent a change at a given position to a given base, which is colour coded. 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 [32] 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 8), 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.

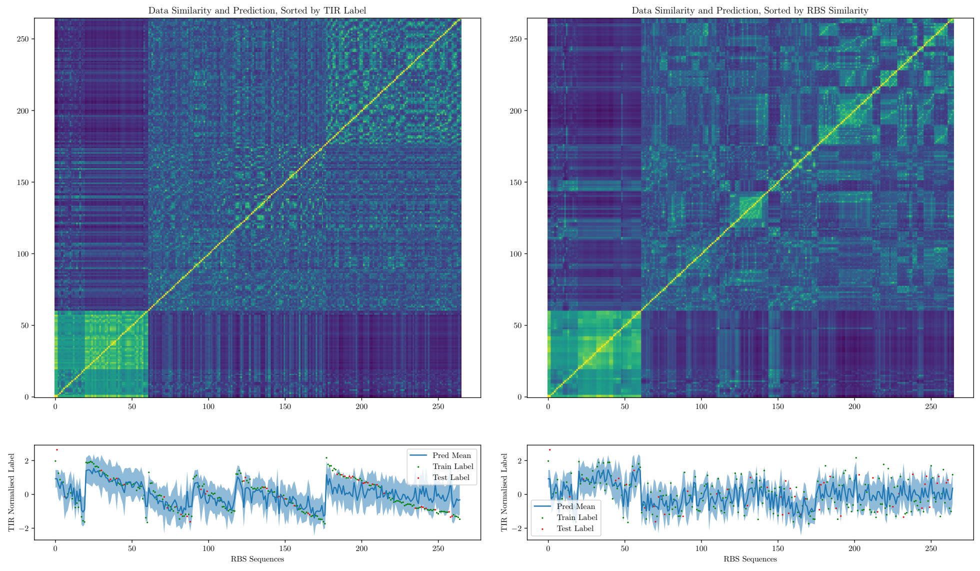
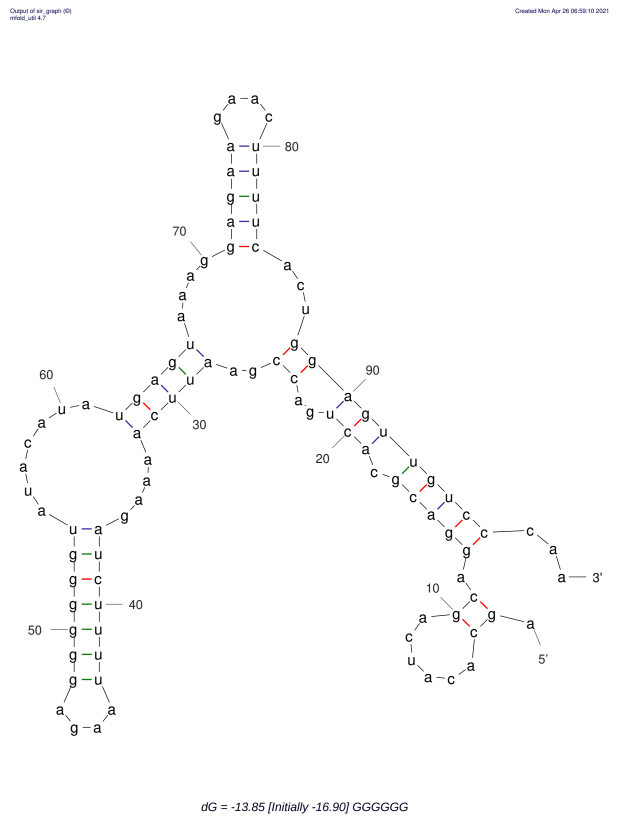
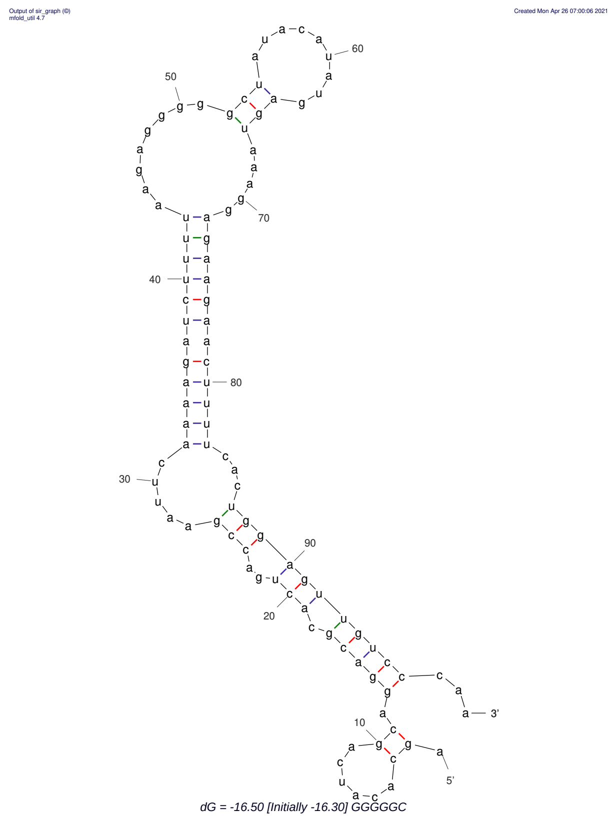


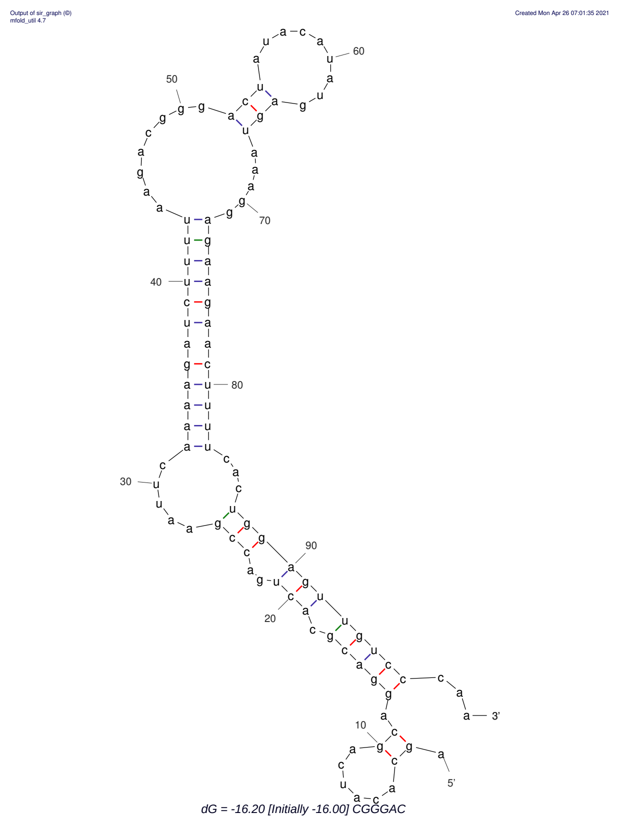
Figure 9: Kernel Heatmap and Predictions. Sequences are grouped as Consensus (1-2), BPS-C (3-20), BPS-NC (21-61), UNI (62-90), PPM (91-118), Bandit-0 (119-177), Bandit-1 (178-265). Inside of each group, sequences are clustered and sorted in terms of TIR labels (left) or RBS similarity (right). The first row shows the similarity measured by weighted degree kernel with shift, the second shows the predicted mean and uncertainty (1.95 standard deviation).


Output of sir_graph (©)
mfold_util 4.7
Created Mon Apr 26 06:59:10 2021
5'
3'
dG = -13.85 [Initially -16.90] GGGGGG

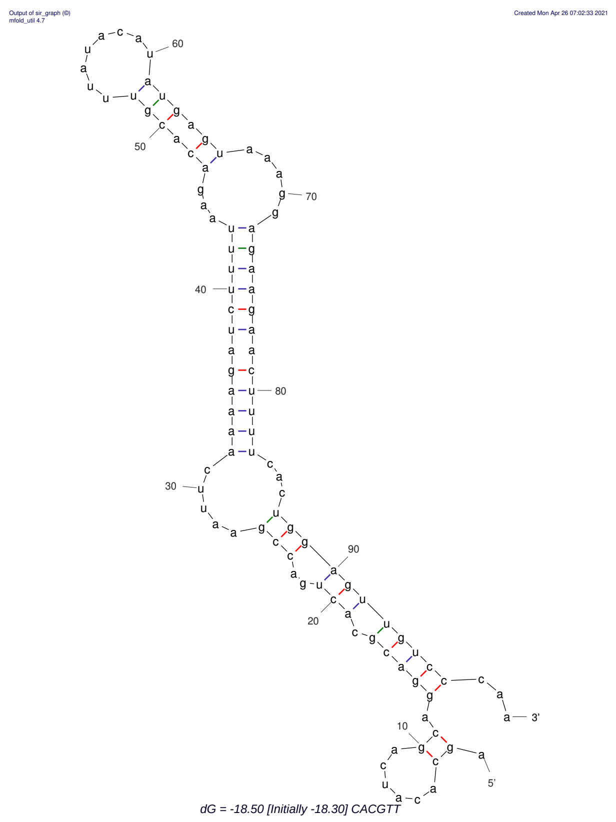
(A) $y = x$


Output of sir_graph (©)
mfold_util 4.7
Created Mon Apr 26 07:00:06 2021
5'
3'
dG = -16.50 [Initially -16.30] GGGGGC

(B) $y = 3sinx$


Output of sir_graph (©)
mfold_util 4.7
Created Mon Apr 26 07:01:35 2021
5'
3'
dG = -16.20 [Initially -16.00] CGGGAC

(C) $y = 5/x$


Output of sir_graph (©)
mfold_util 4.7
Created Mon Apr 26 07:02:33 2021
5'
3'
dG = -18.50 [Initially -18.30] CACGTT

(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).

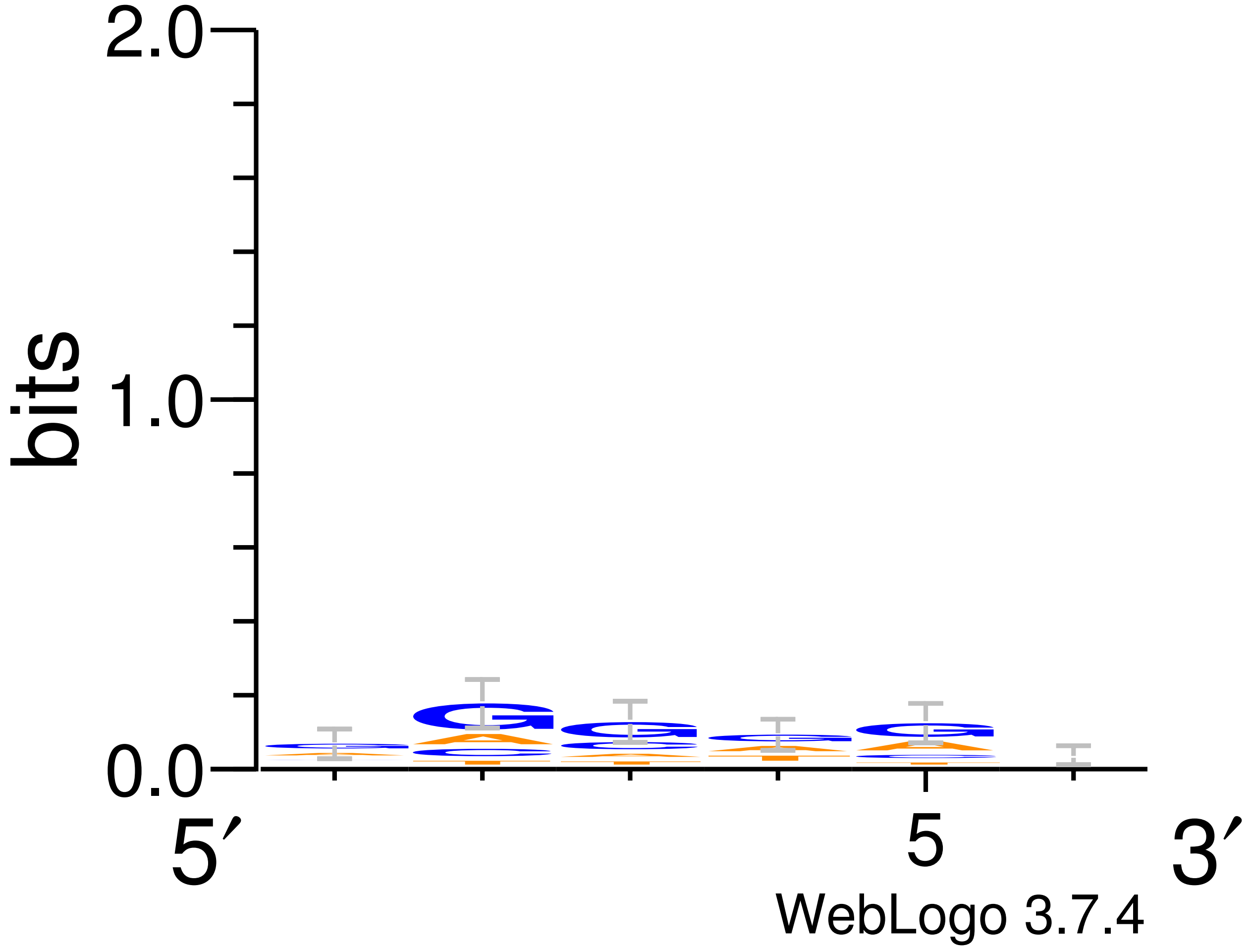


Figure 11: **Sequence logo for all tested sequences.**

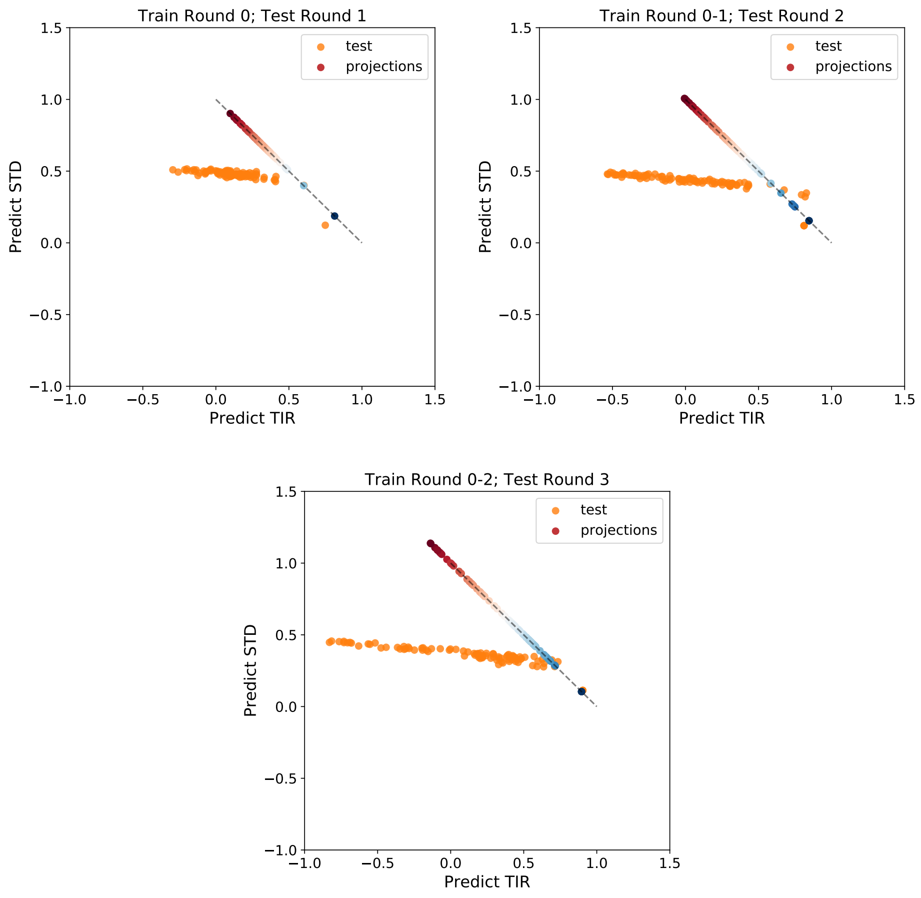


Figure 12: Exploration v.s. Exploitation.

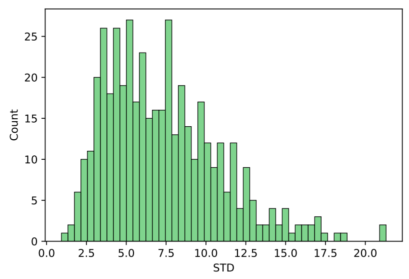
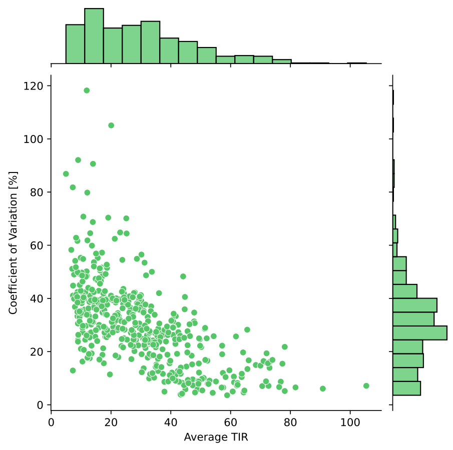


Figure 13: Histogram of standard deviations.

(A)



(B)

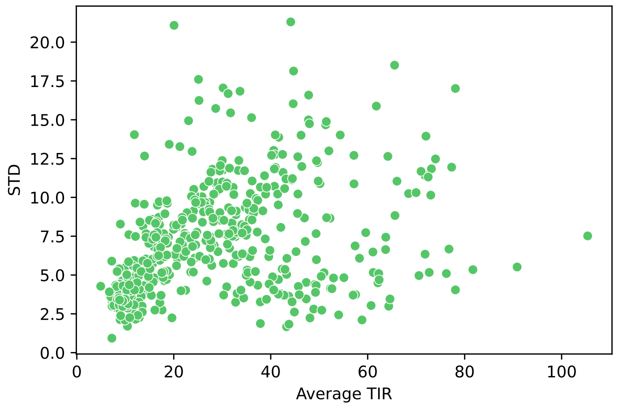


Figure 14: **Distribution of standard deviation of tested samples** A) Joint plot showing the distribution of relative error calculated as the average TIR from 6 replicates divided by the given samples standard deviation (coefficient of variation). B) Scatter plot showing the standard deviation for each RBS plotted against its averaged TIR. Standard deviation and average TIR is given in terms of raw number.