Bayesian Optimization for Protein Sequence Design: Back to Simplicity with Gaussian Processes

Carolin Benjamins *
Leibniz University Hannover
c.benjamins@ai.uni-hannover.de

Shikha Surana
InstaDeep Ltd.

Oliver Bent
InstaDeep Ltd.

Marius Lindauer
Leibniz University Hannover
L3S Research Center

Paul Duckworth
InstaDeep Ltd.

Abstract

Bayesian optimization (BO) is a popular sequential decision making approach for maximizing black-box functions in low-data regimes. In biology, it has been used to find well-performing protein sequence candidates since gradient information is not available from *in vitro* experimentation. Recent *in silico* design methods have leveraged large pre-trained protein language models (PLMs) to predict protein fitness. However PLMs have a number of shortcomings for sequential design tasks: i) their current limitation to model uncertainty, ii) the lack of closed-form Bayesian updates in light of new experimental data, and iii) the challenge of fine-tuning on small downstream task datasets. We take a step back to traditional BO by investigating Gaussian process (GP) surrogate models with various sequence kernels, which are able to properly model uncertainty and update their belief over multi-round design tasks. We empirically evaluate our method on the sequence design benchmark ProteinGym, and demonstrate that BO with GPs is competitive with large SOTA pre-trained PLMs at a fraction of the compute budget.

1 Introduction

Evolution encodes functional information within amino acid sequences. Understanding the effects of sequence mutations on function is a fundamental problem for designing new proteins. Traditionally, directed evolution [Arnold, 1998] methods have been employed to sequentially optimize proteins to obtain better characteristics, where each round consists of random mutagenesis, gene recombinations, screening and selection. This process is slow, labor-intensive, and requires costly *in-vitro* infrastructure. Consequently, it is of significant interest to explore efficient *in-silico* sequence design methodologies.

Labeled protein sequence datasets that link amino acid sequences to quantitative measurements of relevant biological properties are increasingly available [Rao et al., 2019, Dallago et al., 2021, Trabucco et al., 2022, Notin et al., 2023a, Groth et al., 2023]. However, due to experimental limitations during dataset creation, it may only be practical to obtain measurements for dozens or a few hundred proteins at a time [Biswas et al., 2021]. To this end, the focus is on sequence design methodologies that efficiently use small labeled datasets to accelerate real-world experimentation.

A concurrent trend in protein sequence design considers large pre-trained protein language models (PLMs), which take string sequences as input, and fine-tune task-specific downstream predictive

*Work completed during an internship at InstaDeep Ltd.

AI4MAT Workshop at 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

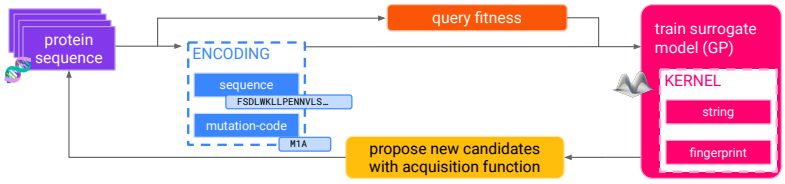

Figure 1: Traditional Bayesian optimization design loop using Gaussian process (GP) models for protein sequence design with string or fingerprint kernels. A surrogate model is trained with protein sequences and their fitness value to approximate the underlying objective function. For this, the sequences need to be represented, either via the sequence of amino-acids or as a mutation-code relative to a specific wild-type protein being optimized. The GP kernel measures similarity between protein sequences and can either directly work on the representation string or on fingerprints. Based on the surrogate model, the acquisition function proposes new and promising protein sequence candidates to evaluate next.

models; a review of the field is provided in Ruffolo and Madani [2024]. However, whilst being strong supervised fitness predictors, PLMs have a number of shortcomings for design, such as: they are computationally expensive to train and fine-tune; they have no closed-form or efficient Bayesian update rule to incorporate new experimental data; they lack formal methods to model predictive uncertainty and do not generalize well outside of training data distributions [Gruver et al., 2021, Greenman et al., 2023, Tagasovska et al., 2024]; and fine-tuning large models on small downstream task data is often challenging due to hyperparameters and overfitting.

In this work, we take a step back to simplicity and consider Gaussian process (GP) surrogate models with various string and fingerprint kernels suitable for longer protein sequences than previously possible [Griffiths et al., 2023]. Whilst GP models have been used to optimize proteins and antibody designs previously [Belanger et al., 2019, Khan et al., 2023] via Bayesian optimization (BO) [Mockus, 1989, Garnett, 2023], those works specifically focused on small regions of the sequence (at most 11 amino acids), and not the high dimensional sequence design setting required for real world protein design, which can be hundreds of amino acids. For an overview of high-dimensional BO literature, see Kim et al. [2021], Santoni et al. [2024], González-Duque et al. [2024]. Our classical BO setup demonstrates competitive performance compared to large pre-trained PLMs on a real-world benchmark, with a tremendously small fraction of the resources required for training and optimization (often running in minutes only on CPU), whilst being a general approach to optimize any protein landscape without prior knowledge.

2 Methodology

BO is a sequential decision making approach that maximizes black-box functions in a sample-efficient way. For black-box functions, we can only observe the input and the output, but not the gradients. Thus, the key idea is to sequentially learn and utilize a surrogate predictive model to efficiently explore the design space. Instead of PLMs we consider GP models to guide the design. They are the default choice in BO [Mockus, 1989, Garnett, 2023].

The choice of the kernel in the GP model represents the covariance function and determines the majority of the generalization properties, enabling posterior model uncertainty and robust out-of-distribution behaviour. We first define the problem setting, then describe protein encodings and how to translate the encodings for the kernels, and finally recap the BO design loop, which is also summarized in Figure 1.

Problem setting Formally, in black-box optimization we aim to find the candidate $x^* \in \mathcal{X}$ in the design space $\mathcal{X}$ with the maximum fitness of a given black-box function $f : \mathcal{X} \to \mathbb{R}$: $x^* = \arg\max_{x \in \mathcal{X}} f(x)$. In our case, it is finding the highest fit protein sequence candidate with the fewest number of function evaluations. Protein sequence optimization in high-dimensions is challenging due to the combinatorial design space, i.e., there are k^l possible sequences (number of amino-acids (AAs) k and sequence length l). For this reason, we restrict the search space to a pool of candidates with corresponding fitness values.

Sequence encoding As proteins are often encoded as a string of hundreds of amino-acids (AAs), they need encoding to obtain a numerical representation. We utilize single-mutant protein landscapes from ProteinGym [Notin et al., 2023a] and investigate two different sequence representations: (i) the *sequence* string of AA tokens (between 86 and 724 in length), and (ii) the *mutation-code* to wild-type sequence. The mutation-code represents a single mutation relative to a wild-type sequence as a tuple of the AA to be replaced, the position of the mutation, and the new AA, e.g. `M12A`, resulting in a compact representation of symbols instead of a long sequence of mostly repeated AAs. To the best of our knowledge, we do not know of any work using mutation-codes as encodings for protein sequence design. In order to use fingerprint kernels (described next), we convert the sequence representation into binary vectors, where each AA in the sequence is replaced by its index in the AA alphabet. The integer index is then represented as binary, e.g. for a sequence of length 86 AAs we obtain a binary vector of length $86 \cdot 5$ (number of digits to represent the 20 AA library in binary form).

GP kernels A Gaussian process (GP) is a stochastic process that specifies the full distribution over the black-box function $f(x) \sim \mathcal{N}(m(x), k(x, x'))$, with $m(x)$ as its mean function and $k(x, x')$ as its covariance function or *kernel*. The kernel should be specified such that similar candidates yield similar predictions [Rasmussen and Williams, 2006]. Therefore, GP kernels act as an inductive bias over the underlying optimization landscape. Kernel functions have hyperparameters such as the lengthscale, which can be optimized during model training. In biology, there are kernels for fingerprints, strings, and graphs [Griffiths et al., 2023]. String kernels are directly applicable to both encoding types: sequence and mutation-code. Molecular fingerprints usually are an enumeration of subgraphs that are hashed into a binary vector. As fingerprint kernels we use Dice, Forbes [Forbes, 1925], Innerproduct, Intersection, Minmax, Otsuka, Russell-Rao, Sorgenfrei and Tanimoto. As string kernel we use the subsequence string kernel (SSK) [Lodhi et al., 2002, Cancedda et al., 2003, Beck and Cohn, 2017, Moss et al., 2020]. Although the SSK on classic protein sequences has the drawbacks of being computationally inefficient and not considering the positions of subsequences [Stanton et al., 2022], this is alleviated by using the mutation-code representation, which is a drastically shorter representation and contains the position of the mutation. All kernels are provided in GAUCHE [Griffiths et al., 2023]. For more details see Appendix A.2.

Bayesian optimisation Generally, in BO, the probabilistic surrogate model (here, a GP with fingerprint or string kernels) is an iteratively refined approximation of the black-box problem that guides the optimization process. BO starts with an initial design, obtained from quasi-random sampling strategies [Brochu et al., 2010]. With these initial evaluated candidates, the surrogate model is trained to approximate the unknown black-box function. It captures the uncertainty of the true function value for unobserved candidates. Then an iterative loop begins: the acquisition function, which is a utility function trading off exploration and exploitation, proposes the next candidates to evaluate based on the surrogate model predictions and uncertainty. In the case of proposing one candidate per round, it is the candidate with the highest acquisition function value. In the batch setting, several candidates can be proposed leveraging various criteria [Desautels et al., 2014, González et al., 2016, Wu and Frazier, 2016, Neiswanger et al., 2022]. The surrogate model is then updated with the new observation(s), and the current best candidate is updated. These steps are repeated usually for a given overall optimization budget.

3 Experiments

Experimental setup We evaluate our BO methodology on protein sequences available in the ProteinGym benchmark [Notin et al., 2023a]. We follow the evaluation protocol introduced in Notin et al. [2023b] and Hawkins-Hooker et al. [2024]. The optimization task is as follows: for a given protein "wild-type", i.e. a given fitness landscape, find the highest-scoring mutated sequence within the fewest number of trials. This benchmark focuses on single mutations from the wild-type, that is, our dataset consists of sequences where only one amino acid has been edited compared to the original sequence. As per the design setting introduced in Notin et al. [2023b], we have 10 optimisation rounds, in each round we acquire 100 new candidates and fitness values to include in the surrogate model training dataset. This setting resembles *in vitro* experiments where batches of newly proposed sequences are sent to the lab for evaluation in each round and the returned fitness values are incorporated into the model. As the acquisition function we use Log Expected Improvement [Mockus et al., 1978, Ament et al., 2023], which selects the candidates from a pool consisting of all single-mutant sequences recorded in the dataset for that wild-type. To propose a batch of candidates we select the top 100

candidates based on their acquisition value. We repeat the experiment for 3 random seeds, and compare the top-30% recall over the candidate pool, and report area-under-curve (AUC) where higher is better (details in Appendix Appendix A.5).

Baselines We compare against three large pre-trained PLM baseline approaches: ESM2 [Meier et al., 2021], PoET [Truong Jr and Bepler, 2023] and ProteinNPT [Notin et al., 2023b]. ESM2 is a 8 million parameter pre-trained masked language model. We pass the average sequence embeddings through a linear regression layer to predict fitness, and fine-tune all parameters. PoET is a family-based autoregressive model that additionally takes multiple-aligned sequences (MSA) as additional context, and pass the final token embedding through a linear regression layer to predict fitness, similarly fine-tuning all layers. ProteinNPT uses frozen-embeddings from MSA Transformer [Rao et al., 2019] and fine-tunes only the non-parametric transformer layers. We use Monte Carlo dropout [Gal and Ghahramani, 2016] to obtain uncertainty estimates, as per the open sourced code [Notin et al., 2023b]. For both ESM2 and PoET we use a greedy acquisition function, and for ProteinNPT we use UCB [Auer, 2002, Srinivas et al., 2010]. Besides PLM baselines we also compare to random search.

Protein Design Results In Figure 2 (left) we plot the multi-round sequence design top 30% recall and AUC for the baseline PLMs and best performing GP models: sequence (seq.) and mutant-code (mut.). This is averaged over eight single-mutant ProteinGym landscapes. We present per-landscape results in Appendix A.7, including all GP kernel variants alongside the baseline methods.

We can clearly see that PoET and ProteinNPT are the current SOTA on the ProteinGym sequence design benchmark. However, it is worth considering that the baseline PLMs have access to two sources of privileged information: (i) a large pre-training corpus of millions of unlabeled protein sequences, and (ii) PoET and ProteinNPT additionally rely on MSA sequences to create their embeddings (which for ProteinNPT are frozen-embeddings from the MSA Transformer). This provides a significant advantage to those two PLMs as demonstrated in our results.

Runtime Results Figure 2 (right) presents the multi-round design run-time (on a log x-axis) interpreted as the total cost of hardware for each design method, averaged over the eight landscapes. Note that this does not include the considerable pre-training compute budget required for the PLM baselines. We can see that the fingerprint kernels on mutation-code representation are extremely fast, running entirely on CPU. Similarly, the sequence-based GP kernels also are significantly more efficient than the fine-tuning regimes of PLMs. The traditional GP methods only require a fraction of the total compute budget as opposed to the PLM baselines, which require many GPU hours for both pre-training and fine-tuning per round. Runtimes and cost are to be found in more detail in Appendix Table 2.

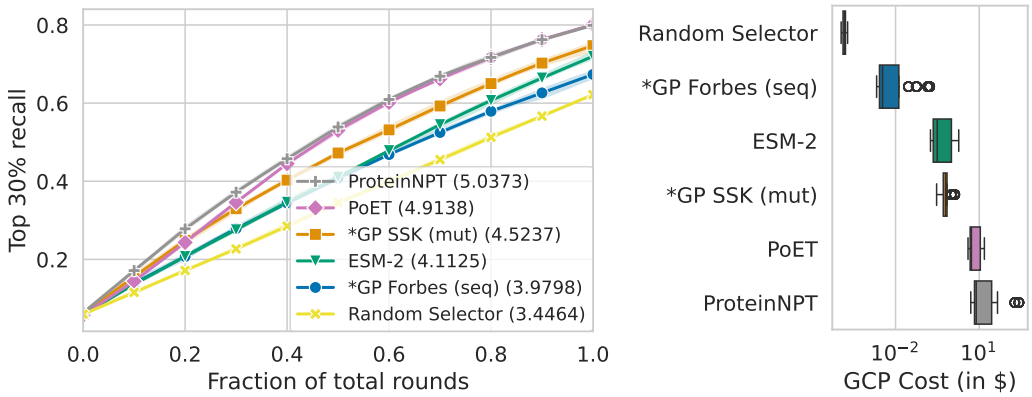

Figure 2: Multi-round design averaged over eight single-mutant protein landscapes. Left: Top-30% recall (mean and 95%-CI). Our methods are highlighted with *. Right: Wall-clock runtime interpreted across hardware as compute costs. Our GP with string (SSK) or fingerprint (Forbes) kernels are competitive with PLM baselines whilst only requiring a fraction of runtime and no pre-training.

4 Limitations and Future Work

This workshop paper comprises of work-in-progress, and there are limitations and avenues to build upon in future work. One limitation of our method is that it does not leverage the additional knowledge encoded in the pre-trained PLMs. Therefore, we consider using PLMs to inform a learned kernel function, as in Khan et al. [2022], or a prior mean function. We explore using zero-shot PLM predictions as a prior mean function in Benjamins et al. [2024], improving performance over using a constant mean. In addition, string kernels might be even more expressive when the hyperparameters of the kernel can be optimized. This has not been possible so far with high dimensional protein sequences due to GPU memory constraints, creating a need for a more efficient implementation. For future work, we plan to extend and evaluate our methodology for multi-mutant landscapes. Another avenue is to investigate BO based on learned embeddings which are updated during optimization together with end-to-end optimization of the acquisition function, similar to Stanton et al. [2022]. Additionally, we intend to investigate the exploration-exploitation trade-off specifically maximizing information over batches, as explored in [Belanger et al., 2019], either by automatically adjusting the trade-off over time as in [Benjamins et al., 2023], or by employing diversity-generating acquisition functions [Neiswanger et al., 2022].

5 Conclusion

This work identifies a promising research avenue for traditional BO with GPs to tackle multi-round protein design. With appropriate kernels, longer protein sequences than previously possibly can be optimised. We demonstrate the competitiveness of BO with classic GPs compared to SOTA PLM approaches that require large pre-training datasets, substantial fine-tuning budgets, and often privileged MSA sequences; whilst only requiring a fraction of the compute budget. We believe probabilistic surrogates hold value, especially in large design spaces where generalisation and uncertainty-driven acquisitions are important.

References

S. Ament, S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy. Unexpected improvements to expected improvement for bayesian optimization. In *NeurIPS 2023*, 2023. 3, A.3

F. Arnold. Design by directed evolution. *Accounts of chemical research*, 31(3):125–131, 1998. 1

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3: 397–422, 2002. 3

M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. Wilson, and E. Bakshy. Botorch: A framework for efficient monte-carlo Bayesian optimization. In Larochelle et al. [2020]. A.3

D. Beck and T. Cohn. Learning kernels over strings using gaussian processes. In *IJCNLP 2017*, pages 67–73, 2017. 2, A.2

D. Belanger, S. Vora, Z. Mariet, R. Deshpande, D. Dohan, C. Angermueller, K. Murphy, O. Chapelle, and L. Colwell. Biological sequence design using batched bayesian optimization. In *NeurIPS 2019 Workshop on Machine Learning and the Physical Sciences*, 2019. 1, 4

C. Benjamins, E. Raponi, A. Jankovic, C. Doerr, and M. Lindauer. Self-adjusting weighted expected improvement for bayesian optimization. In A. Faust, C. White, F. Hutter, R. Garnett, and J. Gardner, editors, *Proceedings of the Second International Conference on Automated Machine Learning*. Proceedings of Machine Learning Research, 2023. 4

C. Benjamins, S. Surana, O. Bent, M. Lindauer, and P. Duckworth. Bayesian optimisation for protein sequence design: Gaussian processes with zero-shot protein language model prior mean. In *Machine Learning in Structural Biology Workshop at NeurIPS 2024*, 2024. 4

S. Biswas, G. Khimulya, E. Alley, K. Esvelt, and G. Church. Low-N protein engineering with data-efficient deep learning. *Nature Methods*, 18(4):389–396, 2021. 1

E. Brochu, V. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv:1012.2599v1 [cs.LG]*, 2010. 2

N. Cancedda, E. Gaussier, C. Goutte, and J.M. Renders. Word sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082, 2003. 2, A.2

C. Dallago, J. Mou, K. Johnston, B. Wittmann, N. Bhattacharya, S. Goldman, A. Madani, and K. Yang. FLIP: Benchmark tasks in fitness landscape inference for proteins. *bioRxiv*, 2021. 1

T. Desautels, A. Krause, and J. Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *JMLR*, 15(1):3873–3923, 2014. 2

S. Forbes. Method of determining and measuring the associative relations of species. *Science*, 61 (1585):518–524, 1925. 2, A.2

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML 2016*, volume 48, pages 1050–1059, 2016. 3

R. Garnett. *Bayesian Optimization*. Cambridge University Press, 2023. Available for free at `https://bayesoptbook.com/`. 1, 2

J. González, Z. Dai, P. Hennig, and N. Lawrence. Batch bayesian optimization via local penalization. In *AISTATS 2016*, 2016. 2

M. González-Duque, R. Michael, S. Bartels, Y. Zainchkovskyy, S. Hauberg, and W. Boomsma. A survey and benchmark of high-dimensional bayesian optimization of discrete sequences. *arXiv preprint arXiv:2406.04739*, 2024. 1

K. Greenman, A. Amini, and K. Yang. Benchmarking uncertainty quantification for protein engineering. *bioRxiv*, 2023. doi: 10.1101/2023.04.17.536962. 1

R. Griffiths, L. Klarner, H. Moss, A. Ravuri, S. Truong, Y. Du, S. Stanton, G. Tom, B. Rankovic, A. Rokkum Jamasb, A. Deshwal, J. Schwartz, A. Tripp, G. Kell, S. Frieder, A. Bourached, A. Chan, J. Moss, C. Guo, J. Peter Dürholt, S. Chaurasia, J. Won Park, F. Strieth-Kalthoff, A. Lee, B. Cheng, A. Aspuru-Guzik, P. Schwaller, and J. Tang. GAUCHE: A library for gaussian processes in chemistry. In *NeurIPS 2023*, 2023. 1, 2, A.2, A.3

P. Groth, R. Michael, P. Tian, J. Salomon, and W. Boomsma. FLOP: Tasks for fitness landscapes of protein families using sequence- and structure-based representations, 2023. 1

Nate Gruver, Samuel Stanton, Polina Kirichenko, Marc Finzi, Phillip Maffettone, Vivek Myers, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Effective Surrogate Models for Protein Design with Bayesian Optimization. In *The 2021 ICML Workshop on Computational Biology*, 2021. 1

A. Hawkins-Hooker, J. Kmec, O. Bent, and P. Duckworth. Likelihood-based fine-tuning of protein language models for few-shot fitness prediction and design. *bioRxiv*, pages 2024–05, 2024. 3

A. Khan, A. Cowen-Rivers, A. Grosnit, D. Deik, P. Robert, V. Greiff, E. Smorodina, P. Rawat, R. Akbar, K. Dreczkowski, R. Tutunov, D. Bou-Ammar, J. Wang, A.s Storkey, and H. Bou-Ammar. Toward real-world automated antibody design with combinatorial Bayesian optimization. *Cell Reports Methods*, 3:100374, 2023. 1

M. Khan, A. Cowen-Rivers, D. Deik, A. Grosnit, K. Dreczkowski, P. Robert, V. Greiff, R. Tutunov, D. Bou-Ammar, Jun Wang, and Haitham Bou-Ammar. Antbo: Towards real-world automated antibody design with combinatorial bayesian optimisation. *CoRR*, abs/2201.12570, 2022. 4

S. Kim, P. Lu, C. Loh, J. Smith, J Snoek, and M. Soljačić. Deep learning for bayesian optimization of scientific problems with high-dimensional structure. *arXiv preprint arXiv:2104.11667*, 2021. 1

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015. A.6

H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H. Lin, editors. *Proceedings of the 34th International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*, 2020. 5

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444, 2002. 2, A.2

J. Meier, R. Rao, R. Verkuil, J. Liu, T. Sercu, and A. Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. In *NeurIPS 2021*, volume 34, pages 29287–29303, 2021. 3

J. Mockus. *Bayesian Approach to Global Optimization. Theory and Applications*. Kluwer Academic Publishers, 1989. 1, 2

J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129), 1978. 3, A.3

H. Moss, D. Beck, J. Gonzàles, D. Leslie, and P. Rayson. BOSS: Bayesian optimization over string spaces. In Larochelle et al. [2020]. 2

H. Moss, D. Leslie, and P. Rayson. MUMBO: Multi-task max-value Bayesian optimization. In F. Hutter, K. Kersting, J. Lijffijt, and I. Valera, editors, *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'20)*, volume 12459, pages 447–462, 2021. A.2

W. Neiswanger, L. Yu, S. Zhao, C. Meng, and S. Ermon. Generalizing bayesian optimization with decision-theoretic entropies. In *NeurIPS 2022*, 2022. 2, 4

P. Notin, A. Kollasch, D. Ritter, L. van Niekerk, S. Paul, H. Spinner, N. Rollins, A. Shaw, R. Orenbuch, R. Weitzman, J. Frazer, M. Dias, D. Franceschi, Y. Gal, and D. Marks. ProteinGym: Large-scale benchmarks for protein fitness prediction and design. In *Advances in Neural Information Processing Systems*, volume 36, pages 64331–64379, 2023a. 1, 2, 3, A.1, 1

P. Notin, R. Weitzman, D. Marks, and Y. Gal. Proteinnpt: Improving protein property prediction and design with non-parametric transformers. In *NeurIPS 2023*, volume 36, pages 33529–33563, 2023b. 3, A.1, A.6

R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, P. Chen, J. Canny, P. Abbeel, and Y. Song. Evaluating Protein Transfer Learning with TAPE. In *NeurIPS 2019*, volume 32, 2019. 1, 3

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. 2

J. Ruffolo and A. Madani. Designing proteins with language models. *nature biotechnology*, 42(2): 200–202, 2024. 1

M. Santoni, E. Raponi, R. Leone, and C. Doerr. Comparison of high-dimensional bayesian optimization algorithms on bbob. *ACM Transactions on Evolutionary Learning*, 4(3):1–33, 2024. 1

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*, pages 1015–1022. Omnipress, 2010. 3

S. Stanton, W. Maddox, N. Gruver, P. Maffettone, E. Delaney, P. Greenside, and A.G. Wilson. Accelerating Bayesian Optimization for Biological Sequence Design with Denoising Autoencoders. In *ICML*, pages 20459–20478, 2022. 2, 4

N. Tagasovska, J. Park, M. Kirchmeyer, N. Frey, A. Watkins, A. Ismail, A. Jamasb, E. Lee, T. Bryson, S. Ra, et al. Antibody domainbed: Out-of-distribution generalization in therapeutic protein design. In *ICLR*, 2024. 1

B. Trabucco, X. Geng, A. Kumar, and S. Levine. Design-bench: Benchmarks for data-driven offline model-based optimization. In *ICML*, pages 21658–21676, 2022. 1

T. Fei Truong Jr and T. Bepler. PoET: A generative model of protein families as sequences-of-sequences. In *NeurIPS*, 2023. 3

J. Wu and P. Frazier. The parallel knowledge gradient method for batch Bayesian optimization. In D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2016. 2

A Appendix / supplemental material

A.1 Protein landscapes

We use the set of eight single-mutant landscapes selected for ablations and hyperparameter selection in Notin et al. [2023b] included in ProteinGym [Notin et al., 2023a], see Table 1 for an overview of each landscape.

Table 1: Single-mutant Protein Landscapes from ProteinGym [Notin et al., 2023a]

Landscape	Sequence Length	Number of Sequences
TAT-HV1BR-Fernandes-2016	86	1577
REV-HV1H2-Fernandes-2016	116	2147
RL40A-YEAST-Roscoe-2013	128	1195
CALM1-HUMAN-Weile-2017	149	1813
DYR-ECOLI-Thompson-2019	159	2363
BLAT-ECOLX-Jacquier-2013	286	989
P53-HUMAN-Giacomelli-2018-WT-Nutlin	393	7467
DLG4-RAT-McLaughlin-2012	724	1576

A.2 String and Fingerprint Kernels

The kernel $k(x, x'))$ measures similarity between inputs and acts as an inductive bias over the underlying optimization landscape. Kernel functions have hyperparameters such as the lengthscale, which can be optimized during model training. String kernels operate on strings or a sequence of symbols and mostly compare the similarity of their sub-strings. In our case the string is either a sequence of an alphabet of 20 amino acids (length l up to hundreds) or the mutation-code (length $l = 3$) of an alphabet of 20 amino acids plus the number of possible mutation locations. More formally, our alphabet $\mathcal{A}$ is the set of k symbols $\mathcal{A} = \{s_0, ..., s_k\}$ ($|\mathcal{A}| = k$). Then our vector space is $\mathcal{A}^l$ with l being the sequence length. Each element $x \in \mathcal{A}^l$ is a vector $x = (x_0, x_1, ..., x_l)$ with $x_i \in \mathcal{A} \, \forall i = 1, 2, ..., l$.

The subsequence string kernel (SSK) [Lodhi et al., 2002, Cancedda et al., 2003, Beck and Cohn, 2017, Moss et al., 2021] we use as provided by GAUCHE [Griffiths et al., 2023] compares sub-strings of length n (set to $n = 5$). The sub-sequences are used as features and can be non-contiguous. An SSK (n-th) order between to strings x and x' is defined as:

$$k_{\text{SSK}}(x, x') = \sum_{u \in \mathcal{A}^n} c_u(x) \cdot c_u(x')$$

with

$$c_u(s) = \lambda_m^{|u|} \sum_{1<i_1<\cdots<i_{|u|}<|s|} \lambda_g^{i_{|u|}-i_1} \mathbb{1}_u((s_{i_1}, \ldots, s_{i_{|u|}})) \,,$$

where $\mathcal{A}^n$ is the set of all possible ordered collections containing up to n characters from the alphabet $\mathcal{A}$, $\mathbb{1}_x(x')$ the indicator function if strings x and x' are equal. The contribution of a subsequence u to a string s is measured by $c_u(s)$. The kernel hyperparameters are the match decay $\lambda_m \in [0, 1]$ and the gap decay $\lambda_g \in [0, 1]$. They control the weighting of long and/or highly non-contiguous sub-strings. The kernel is normalized $\tilde{k}_{\text{SSK}}(x, x') = k_{\text{SSK}}(x, x')/\sqrt{k_{\text{SSK}}(x, x)k_{\text{SSK}}(x', x')}$ to be able to meaningfully compare strings of varied lengths.

Fingerprint kernels do not operate on the alphabet of symbols but on a binary vector, thus $x, x' \in \{0, 1\}^d$ (d length of vector). For example, the Forbes kernel [Forbes, 1925] is defined as

$$k_{Forbes}(x, x') = \sigma_f^2 \cdot \frac{d \cdot \langle x, x' \rangle}{\|x\| + \|x'\|}$$

with $\|\cdot\|$ the Euclidean norm.

A.3 Hyperparameters

For our BO with GPs we use Botorch [Balandat et al., 2020] with prior zero mean, LogEI [Mockus et al., 1978, Ament et al., 2023] and kernels from GAUCHE [Griffiths et al., 2023]. We standardize the response values and use standard hyperparameters.

A.4 Hardware

For our experiments we used NVIDIA-A100-SXM4-80GB GPU for the PLM baselines and BO with the SSK kernel. For the remaining experiments we used AMD EPYC 7452 CPUs.

A.5 Metrics

Top 30% Recall We define the top 30% recall as follows: The number of acquired candidates who have a fitness value higher equal the threshold divided by the number of candidates in the pool with a fitness value higher than the threshold. The threshold marks the lowest fitness value of the top 30% of all candidates in the pool. It represents how many relevant items have been retrieved so far. For the design curves aggregating results over all tasks we first average over tasks and plot the 95%-CI over the seeds. To account for the different number of rounds we normalize the rounds and interpolate the values accordingly. The per-task design curves are based on the raw data.

AUC We calculate the area-under-curve (AUC) as the integral of the top 30% recall over rounds. For the aggregation over tasks we average the top 30% recall over tasks first and then calculate AUC per seed.

A.6 Baseline models

Hyperparameters for the fine-tuning PLM methods is consistent with the practice used to select hyperparameters for the baselines from ProteinNPT.

ESM-2 and PoET models were fine-tuned using the Adam optimizer [Kingma and Ba, 2015] using gradient accumulation with an effective batch size of 32. Learning rates were selected in each case after a sweep over the values $\{10^{-4}, 3 \cdot 10^{-5}, 10^{-5}\}$ on the eight single mutant landscapes.

Linear regression heads were added to embeddings extracted from PoET and ESM-2. In the former case, we used final token embeddings, and in the latter case we averaged embeddings across the sequence dimension before feeding them to the regression head.

ProteinNPT was run using the code released by [Notin et al., 2023b]. We refer to [Notin et al., 2023b] for further details. We used the Monte Carlo dropout uncertainty quantification strategy proposed by [Notin et al., 2023b] for ProteinNPT. Notin et al. [2023b] report best results with a 'hybrid' uncertainty quantification strategy, however this strategy is not implemented in the publicly available code.

A.7 Design curves and AUC Per Landscape

Upon inspecting the following individual landscape results, the general trend is that the PLMs ProteinNPT and PoET are most performant. However, when we also consider cost and expertise required for pre-training, fine-tuning and inference and in-context learning for PLMs, our GP setup is more accessible. Per landscape the exact order of performance changes and some landscapes are more challenging than others (P53-HUMAN-Giacomelli-2018-WT-Nutlin vs. RL40A-YEAST-Roscoe-2013). When we compare the encoding, the mutation-code is more performant than sequence encoding which matches the intuition that the mutation-code carries semantic information, i.e. what amino acid has been switched where.

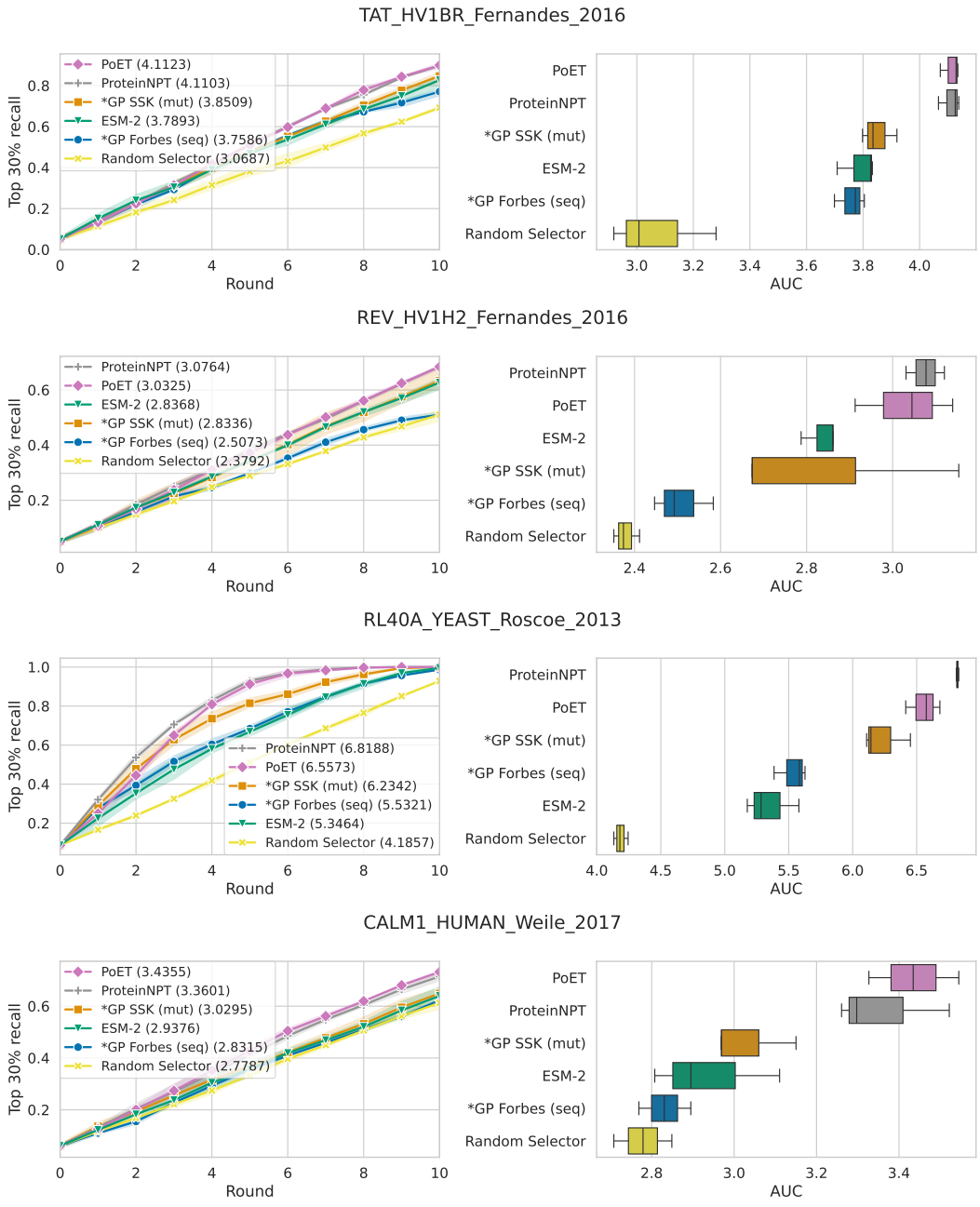

Figure 3: Design curves for individual single mutant landscapes for baselines and top performing GP kernels (seq: sequence encoding, mut: mutation encoding).

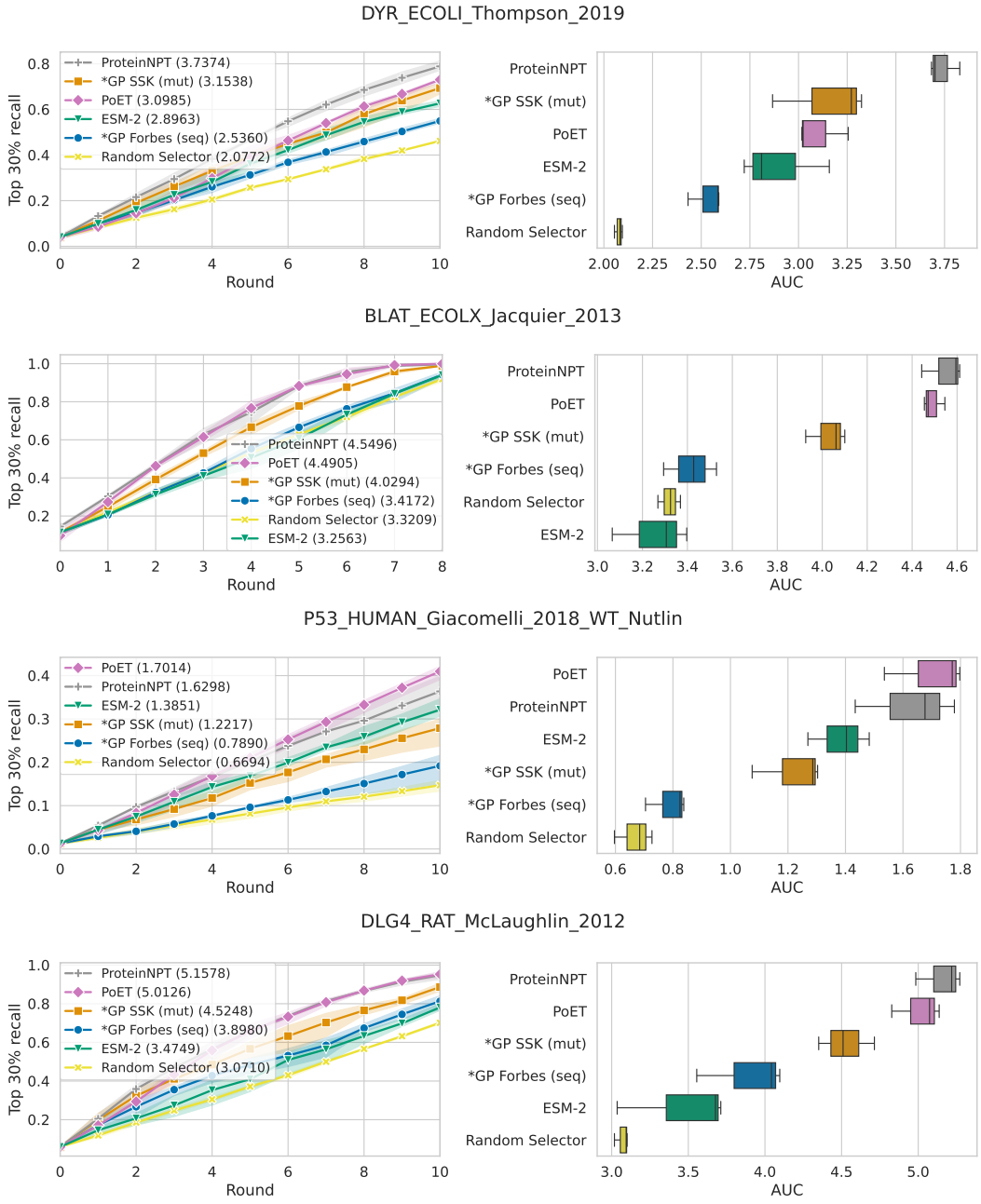

Figure 4: Design curves for individual single mutant landscapes for baselines and top performing GP kernels (seq: sequence encoding, mut: mutation encoding).

Figure 5: Final AUC of baselines and all GP kernels. Only SSK is a string kernel, the rest is a fingerprint kernel.

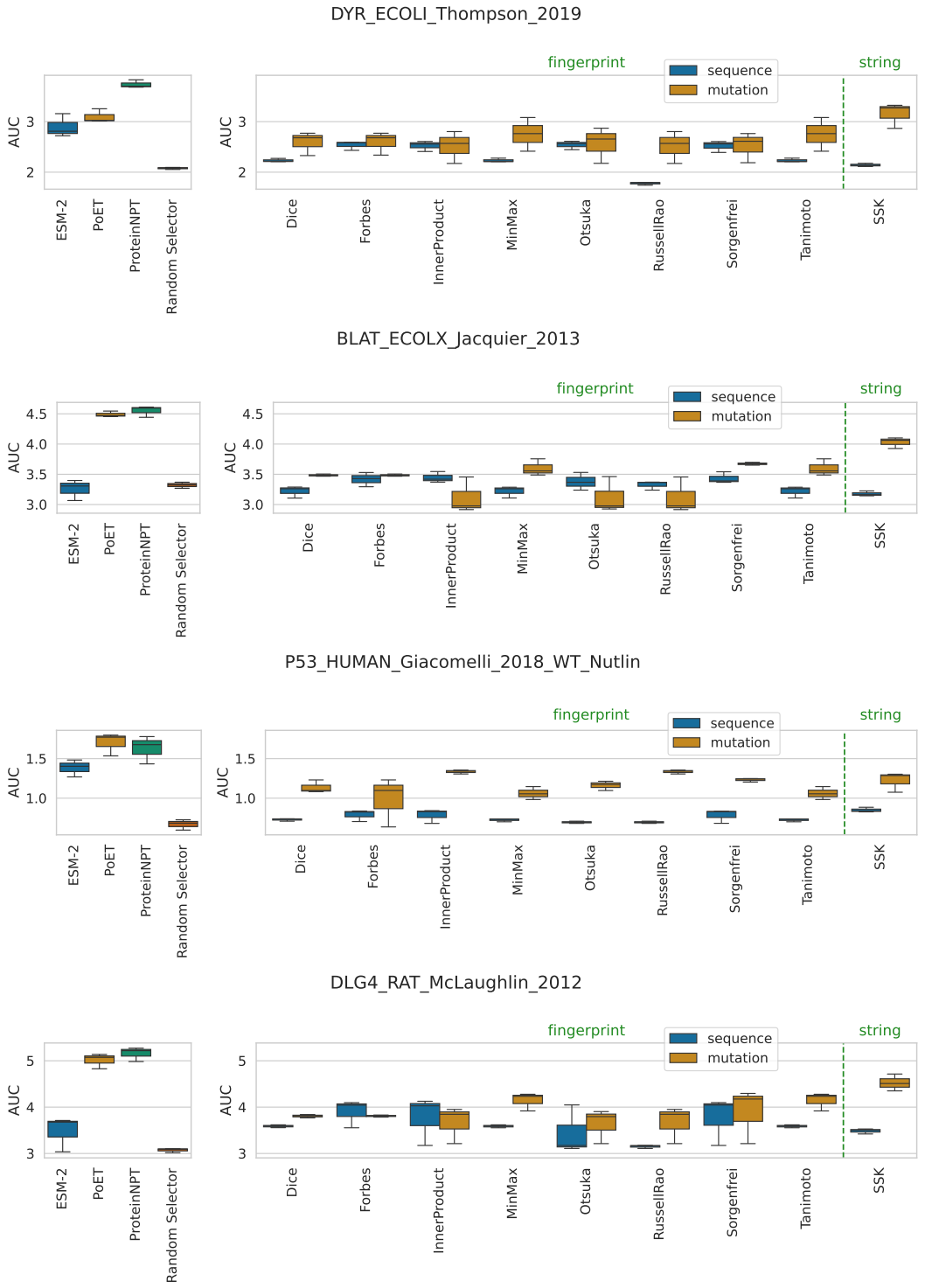

Figure 6: Final AUC of baselines and all GP kernels. Only SSK is a string kernel, the rest is a fingerprint kernel.

Table 2: Final mean AUC averaged over all 8 landscapes and 3 seeds of all surrogate models together with run time and cost. Ordered by AUC. Best GP variant marked with '*'.

	AUC	Runtime (h)	Cost ($)
ProteinNPT	5.0373	5.5038	30.7112
PoET	4.9138	1.2601	7.0314
ESM-2	4.1125	0.1014	0.5659
Random Selector	3.4464	0.0005	0.0001
*GP Forbes (seq)	3.9798	0.0993	0.0285
GP InnerProduct (seq)	3.9267	0.0877	0.0252
GP Sorgenfrei (seq)	3.9259	0.1006	0.0289
GP Otsuka (seq)	3.8541	0.0879	0.0252
GP MinMax (seq)	3.6946	0.0870	0.0250
GP Tanimoto (seq)	3.6946	0.1065	0.0306
GP Dice (seq)	3.6945	0.0853	0.0245
GP SSK (seq)	3.6447	2.6948	0.7734
GP RussellRao (seq)	3.4050	0.0826	0.0237
*GP SSK (mut)	4.5237	2.3998	0.6887
GP MinMax (mut)	4.2354	0.0078	0.0022
GP Tanimoto (mut)	4.2354	0.0085	0.0024
GP Sorgenfrei (mut)	4.1281	0.0085	0.0024
GP Dice (mut)	3.9977	0.0081	0.0023
GP Forbes (mut)	3.9762	0.0094	0.0027
GP RussellRao (mut)	3.8959	0.0087	0.0025
GP InnerProduct (mut)	3.8884	0.0064	0.0018
GP Otsuka (mut)	3.8704	0.0099	0.0028