RESEARCH

Sort-Seq Tools: software for massively parallel experiments on partially mutagenized sequences

William Ireland[1], Rob Phillips[2] and Justin B Kinney[3*]

*Correspondence: jkinney@cshl.edu
[3]Simons Center for Quantitative Biology, Cold Spring Harbor Laboratory, 11375, Cold Spring Harbor, NY
Full list of author information is available at the end of the article

Abstract

Here we introduce a software package, Sort-Seq Tools, for analyzing a variety of high-throughput mutational experiments on DNA, RNA, and proteins. Such experiments include Sort-Seq studies of bacterial promoters, massively parallel reporter assays of mammalian enhancers, and deep mutational scanning experiments of proteins. Easy-to-use methods for data simulation, data preprocessing, data analysis, quantitative modeling, and visualization are provided. These functionalities are available through a command-line interface, as well as through a Python module, `sortseq`, that is available on PyPI. Example applications of Sort-Seq Tools in the context of both real and simulated data sets are described.

Keywords: sequence analysis; transcriptional regulation; deep mutational scanning; mutual information

Introduction

Deep mutational scanning experiments are rapidly changing our experimental capabilities.

Figure 1 Experiments that can be analyzed by Sort-Seq Tools (A) Sort-Seq assays [1, 2]. A promoter of interest (blue) is used to drive the expression of a gene (green) encoding a fluorescent protein. A library of mutagenized promoters is then swapped in for the wild-type promoter. Cells expressing this construct are then sorted one-by-one into bins by FACS according to measured fluorescence. The variant promoters in each bin are then sequenced. (B) MPRA assays [3, 4, 5]. A library of variant enhancers (blue) are cloned upstream of a basal promoter and used to drive gene expression. The resulting mRNA transcripts contain enhancers-specific tags (brown). Reporter constructs are introduced into mammalian cell culture, after which RNA is extracted and expressed tags are sequences. Tags from the reporter library are also sequenced. (C) Protein deep mutational scanning assays [6, 7]. A library of protein coding sequences (green) are introduced into phage (illustrated) or some other expression system. Variant proteins are then selected according to some activity of interest, such as binding affinity (illustrated). Selected genes are then sequenced, as are genes from the starting library.

[Describe Sort-Seq] Sort-Seq provides a list of (non-unique) sequences, each with a categorical measurement of activity. [1]

[Describe MPRA] MPRA provides a list of (non-unique) sequences, each with a quantitative measure of transcriptional activity. [8]. [3]. [4].

[Describe selection-based assays] For transcription [9], For yeast ARSs [10]. For proteins [6]. [7]

[Bloom provides software to compare enrichment between two bins [11]]

[Describe data analysis] [12] [13] [1] [14]

Fitting energy matrices are the first step in fitting thermodynamic models. [15, 16]

Figure 2 Example analysis using Sort-Seq Tools. Sort-Seq Tools provides a set of functions that operate by transforming tables (stored as text files) from one type to another. The possible column headers for each input table are listed in Table 1. A single function, `draw`, provides a default visualization of each type of table, but can also accept flags that alter the type visualization produced. (A) A simple example of data analysis using Sort-Seq tools. Here, a table `data.txt` contains a list of aligned DNA binding sites for the E. coli transcription factor hunchback. The Sort-Seq Tools function `profile_freqs` transforms this table into a second table, `freqs.txt`, that lists the occurrence frequencies of each nucleotide at each position within the alignment. The function `draw --logo` then transforms this table into a sequence logo, which is stored as a .PDF file. (B) Sort-Seq Tools is designed to be used at the command-line. Every method is implemented as a subcommand of the function `sortseq`. Commands implementing the analysis in A are shown. (C) Alternatively, to reduce the use of temporary files, Sort-Seq Tools commands can be piped in series. (D) Sort-Seq Tools can also be used from within Python via the `sortseq` package (available on PyPI). Python code equivalent to the command line analysis in A and B is shown.

Figure 3 Library quality control The results of an experiment must be summarized in a table of counts and sequences for analysis with Sort-Seq Tools. For paired end reads, the tranformation from fasta or fastq files can be accomplished by using the Sort-Seq Tools `preprocess` function. (B) The output from the `profile_counts` function can be transformed into a mutation rate at each position using the `profile_mutrate` function. The mutation rate for the Sort-Seq experiment on the Lac promoter region performed by Kinney et al. (2010) is displayed. (C) The counts and sequences table can be visualized as a Zipf plot using the `draw` function. (D) The commands to perform the above analysis at the command line.

Figure 4 Information profiling Sort-Seq Tools also provides a set of analysis which operate on more than one bin at a time. The complete list of functions which profile data set characteristics for each base pair position are summarized in Table 3. Before executing these analyses on raw fasta files, the `gatherseqs` function must be provided a list of files and bins to combine into a single table. The inputs and outputs from `gatherseqs` and other data manipulation functions are summarized in Table 2. (B) The aggregated data file with the counts of each sequence occuring in each bin. (C) Using the `profile_info` function, the mutual information between expression bin and base identity for the Kinney et al. experiment is calculated. The resulting table is displayed using the `draw` function. (D) The commands to perform the above analysis at the command line.

Figure 5 Quantitative modeling (A) A combined data set is fit to a linear energy model using the `learn_model` function. The four methods, specified by flags to the `learn_model` function, that the Sort-Seq Tools package can use to fit models are summarized in Table 4. The model predicts binding energy given any sequence. The model is visualized as both a sequence logo and a matrix. Kinney et al performed a Sort-Seq experiment on the Lac promoter. The crp site fit to this data set is displayed. (B) The commands to perform the above analysis at the command line.

Figure 6 Model Comparison (A) The `predictiveinfo` function can be used to calculate the mutual information between the predictions of your model and the results of your experiment. The inputs and outputs for the `predictiveinfo` function are shown in Table 4. Table 4 also contains the `totalinfo` function, which calculates for a sublibrary the maximum possible predictive information of a model. (B) A display of the commands to perform the analysis in (A) at the command line. A list of several data sets to test, shown in (C), and several models to use in the evaluation, shown in (D), can be used as inputs in a comparison of the predictive ability of each model on each data set. The commands to perform this analysis are shown in (E). The visualization in (F) is a compairison of crp models fit to data sets from experiments with several different concentrations of cAMP. The data is from Kinney et al. (2010).

Kinney et al. (2010) showed that such data could be used to look at far more than enrichment ratios. Given a long list of aligned variant sequences, one can create information footprints (which identify important regions within a regulatory sequence). One can also fit quantitative models to these data. Importantly, quantitative models of experimental noise are not needed for such model fitting. This

Figure 7 Tag expression analysis (A) In MPRA experiments, files of counts and tags, can be translated into a file of counts and corresponding mutated sequences. A table containing file locations, as displayed in (B), and a key that connects tags and mutated sequences, displayed in (C), are used as inputs to the `gatherseqs --tags` function to transform the raw data. (D) From the transformed data file, the log-ratio of each sequence, using pseudo-counts, or an energy model can be calculated. (E) The commands to carry out the analysis in (D) at the command line.

Figure 8 Analysis of protein sequences (A) The DNA sequences obtained from the [6, 7] protein deep sequencing experiment are translated into amino acid sequences using the `convertseqs` function. Enrichment ratios for each amino acid at each position are calculated using pseudo-counts. The result of this analysis for the [6, 7]. protein mutagenesis experiment is displayed. (B) A display of the commands to carry out the analysis in (A) at the command line.

Figure 9 Data simulation (A) During simulation, a library is generated either by randomly mutating away from a specified wild type sequence, or by specifying a matrix of base probabilities. Expression predictions are calculated by using an energy model, input either in the form of a logo or a matrix. The logo for hunchback is displayed. Noise according to a chosen model is added, and then the simulated sequences are sorted based on their simulated expression. Commands to perform all phases of simulation, as well as functions to simulate MPRA experiments and protein selection experiments are summarized in Table 5. (C) A display of the commands to perform the above analysis at the command line.

allows, for example, matrix models of transcription factor-DNA binding energy to be learned. However, there is currently no readily available software for performing such analyses.

Methods

Overview

Table 1 Column descriptions

Header	Description
`bin`	number of sequence bin (0, 1, 2, ...)
`file`	name of file containing sequence or tag data
`ct`	sequence or tag count
`seq`	sequence of DNA (default), RNA (`_rna`), or protein (`_pro`)
`tag`	expression tag
`pos`	nuclotide/residue position within a set of aligned sequences
`wt`	wild-type nucleotide or residue
`obs`	observed nucleotide or residue
`mut`	mutation rate
`le`	log enrichment
`freq`	occurrence frequency of nucleotide/residue
`val`	model parameter
`lr`	log ratio
`info`	mutual information in bits
`_0, _1 ...`	associated bin number
`_A, _C ...`	associated nucleotide or residue
`_err`	estimated uncertainty

Table 2 Data processing commands

Command	Input columns	Output columns
`gatherseqs`	`bin file`	`ct_0 ct_1 ct_2 ... seq`
`gatherseqs --tagkey key.txt`	`bin file`	`ct_0 ct_1 ct_2 ... tag seq`
`logratios`	`ct_0 ct_1 seq`	`lr lr_err seq`
`errfromtags --var x`	`x tag seq`	`x x_err seq`
`convertseqs`	`seq(,_rna,_pro)`	`seq(,_rna,_pro)`

Table 3 Profile generation commands

Command	Input columns				Output columns			
profile_counts	ct	seq			pos	ct_A	ct_C ...	
profile_counts --bin k	ct_k	seq			pos	ct_A	ct_C ...	
profile_freqs	ct	seq			pos	freq_A	freq_C ...	
profile_freqs --bin k	ct_k	seq			pos	freq_A	freq_C ...	
profile_enrichment	ct_0	ct_1	seq		pos	le_A	le_C ...	
profile_mutrates	ct	seq			pos	mut	mut_err	
profile_info	ct_0	ct_1	ct_2 ...	seq	pos	info	info_err	
fromto_mutrates	pos	ct_A	ct_C ...		wt	obs	mut	mut_err

Table 4 Modeling commands

Command	Input columns				Output columns			
learn_matrix --bvh	ct_0	ct_1	seq		pos	val_A	val_C ...	
learn_matrix --leastsq	ct_0	ct_1	ct_2 ...	seq	pos	val_A	val_C ...	
learn_matrix --MImax	ct_0	ct_1	ct_2 ...	seq	pos	val_A	val_C ...	
predictiveinfo --model m	ct_0	ct_1	ct_2 ...	seq	info	info_err		
totalinfo	ct_0	ct_1	ct_2 ...	seq	info	info_err		

Table 5 Simulation commands

Command	Input columns			Output columns			
simulate_library				ct	seq		
simulate_library --tags				ct	tag	seq	
simulate_sublib	ct	seq		ct	seq		
simulate_sort	ct	seq		ct_0	ct_1	ct_2 ...	seq
simulate_selection	ct	seq		ct_0	ct_1	seq	
simulate_expression	ct	tag	seq	ct_0	ct_1	tag	seq

Simulation

[Library creation. Entire sequence or window]
  [Sublibrary sampling]
  [Sorting]

Data visualization

[Mutation rate]
  [Information footprint]
  [Sequence logos]
  [Mutation independence]

Quantitative modeling

[Least-squares fitting]
  [Lasso fitting]
  [Information maximization Monte Carlo]

Model evaluation

[Total sequence-dependent information]
  [Predictive information]

Results

Analysis of simulated data

As discussed in Figure 9, mutated libraries and data sets can be simulated for Sort-Seq, MPRA, and protein selection experiments. In Figure 10, it is shown that we can consistently recover, using our fitting methods, the models used to generate the simulated data sets.

Analysis of Sort-Seq data from Kinney et al. (2010)

Each bin of sequences is first tested for quality using the `profile_mutrate`, `pairwise_mutrate`, and the `draw` functions. Each of the Kinney et al. (2010) data sets showed a consitent mutation rate near to the target, which indicates that the targeted diversity of sequences was achieved. There was also no mutual information between one base being mutated and another being mutated. This is an important quality control step. If there is corrolation between mutation positions, then if one of the two positions has a high value on the information footprint, the other will as well, regardless of its importance to transcription. This is because in this case, knowing the identity of one base gives you information about the identity of a second, truly important base. The data was then analyzed by using the `profile_info` function to produce information footprints, as seen in Figure 5. The footprints allowed identification of the binding sites of CRP and of RNAP. Matrix models for binding energy were individually fit to each of these sites using mutual information maximization.

Analysis of MPRA data from Melnikov et al. (2012)

The `gatherseqs` function is used to connect each mRNA sequence read to the corresponding mutated region and combine all data into one file. This file was then analyzed using the `profile_info` and `learn_model` functions to produce information footprints and energy matrices respectively. Highly informative regions in the information footprint correspond to regions that are mechanistically important to transcription such as transcription factors or the RNAP site. Energy matrices reveal the sequence depenedence of the binding energy of that transcription factor. 70% of each data set is randomly assigned to a training data set and 30% is used to test the fit. The split is created using the function `train_test_split`. The mutual information between the energy model predictions and the test data set, as calculated by the `predictiveinfo` function can be used as a metric for how well the model performed.

Analysis of protein mutational scanning data from Fowler et al. (2010)

As discussed in Figure 9, the [6, 7]. data was processed and a enrichment profile was generated. The `learn_model` was also used to produce an affinity matrix for the data set. The affinity matrix is displayed in Figure 11.

Discussion

Competing interests

The authors declare that they have no competing interests.

Author's contributions

WI, RP, and JBK designed the research. WI and JBK wrote the software. WI and JBK wrote the paper. ...

Acknowledgements

Text for this section ...

Author details

[1]Department of Physics, California Institute of Technology, 91125, Pasadena, CA. [2]Department of Applied Physics, California Institute of Technology, 91125, Pasadena, CA. [3]Simons Center for Quantitative Biology, Cold Spring Harbor Laboratory, 11375, Cold Spring Harbor, NY.

References

1. Kinney, J.B., Murugan, A., Callan, C.G., Cox, E.C.: Using deep sequencing to characterize the biophysical mechanism of a transcriptional regulatory sequence. Proc. Natl. Acad. Sci. USA **107**(20), 9158–9163 (2010)
2. Sharon, E., Kalma, Y., Sharp, A., Raveh-Sadka, T., Levo, M., Zeevi, D., Keren, L., Yakhini, Z., Weinberger, A., Segal, E.: Inferring gene regulatory logic from high-throughput measurements of thousands of systematically designed promoters. Nature Biotechnology **30**(6), 521–530 (2012)
3. Melnikov, A., Murugan, A., Zhang, X., Tesileanu, T., Wang, L., Rogov, P., Feizi, S., Gnirke, A., Callan, C.G., Kinney, J.B., Kellis, M., Lander, E.S., Mikkelsen, T.S.: Systematic dissection and optimization of inducible enhancers in human cells using a massively parallel reporter assay. Nature Biotechnology **30**(3), 271–277 (2012)
4. Patwardhan, R.P., Hiatt, J.B., Witten, D.M., Kim, M.J., Smith, R.P., May, D., Lee, C., Andrie, J.M., Lee, S.-I., Cooper, G.M., Ahituv, N., Pennacchio, L.A., Shendure, J.: Massively parallel functional dissection of mammalian enhancers in vivo. Nature Biotechnology **30**(3), 265–270 (2012)
5. Kwasnieski, J.C., Mogno, I., Myers, C.A., Corbo, J.C., Cohen, B.A.: Complex effects of nucleotide variants in a mammalian cis-regulatory element. Proceedings of the National Academy of Sciences of the United States of America **109**(47), 19498–19503 (2012)
6. Fowler, D.M., Araya, C.L., Fleishman, S.J., Kellogg, E.H., Stephany, J.J., Baker, D., Fields, S.: High-resolution mapping of protein sequence-function relationships. Nature Methods **7**(9), 741–746 (2010)
7. Fowler, D.M., Fields, S.: Deep mutational scanning: a new style of protein science. Nature Methods **11**(8), 801–807 (2014)
8. Patwardhan, R.P., Lee, C., Litvin, O., Young, D.L., Pe'er, D., Shendure, J.: High-resolution analysis of DNA regulatory elements by synthetic saturation mutagenesis. Nature Biotechnology **27**(12), 1173–1175 (2009)
9. Findlay, G.M., Boyle, E.A., Hause, R.J., Klein, J.C., Shendure, J.: Saturation editing of genomic regions by multiplex homology-directed repair. Nature **513**(7516), 120–123 (2014)
10. Liachko, I., Youngblood, R.A., Keich, U., Dunham, M.J.: High-resolution mapping, characterization, and optimization of autonomously replicating sequences in yeast. Genome Res **23**(4), 698–704 (2013)
11. Bloom, J.D.: Software for the analysis and visualization of deep mutational scanning data. BMC Bioinformatics **16**, 168 (2015)
12. Atwal, G.S., Kinney, J.B.: Learning quantitative sequence-function relationships from high-throughput biological data. arXiv:1212.3647 [q-bio.QM] (2015). 1506.00054v1
13. Kinney, J.B., Tkacik, G., Callan, C.G.: Precise physical models of protein-DNA interaction from high-throughput data. Proc. Natl. Acad. Sci. USA **104**(2), 501–506 (2007)
14. Kinney, J.B., Atwal, G.S.: Parametric inference in the large data limit using maximally informative models. Neural Comput. **26**(4), 637–653 (2014)
15. Bintu, L., Buchler, N., Garcia, H., Gerland, U., Hwa, T., Kondev, J., Phillips, R.: Transcriptional regulation by the numbers: models. Curr Opin Genet Dev **15**(2), 116–124 (2005)
16. Bintu, L., Buchler, N.E., Garcia, H.G., Gerland, U., Hwa, T., Kondev, J., Kuhlman, T., Phillips, R.: Transcriptional regulation by the numbers: applications. Curr Opin Genet Dev **15**(2), 125–135 (2005)

Additional Files

Additional file 1 — Sample additional file title
Additional file descriptions text (including details of how to view the file, if it is in a non-standard format or the file extension). This might refer to a multi-page table or a figure.

Additional file 2 — Sample additional file title
Additional file descriptions text.