Methods in Molecular Biology 1807

Springer Protocols

Hiroshi Mamitsuka *Editor*

Data Mining for Systems Biology

Methods and Protocols

Second Edition

Humana Press

METHODS IN MOLECULAR BIOLOGY

Series Editor
John M. Walker
School of Life and Medical Sciences
University of Hertfordshire
Hatfield, Hertfordshire, AL10 9AB, UK

For further volumes:
http://www.springer.com/series/7651

Data Mining for Systems Biology

Methods and Protocols

Second Edition

Edited by

Hiroshi Mamitsuka

Bioinformatics Center, Kyoto University, Uji, Kyoto, Japan; Department of Computer Science, Aalto University, Espoo, Finland

Humana Press

Editor
Hiroshi Mamitsuka
Bioinformatics Center
Kyoto University
Uji, Kyoto, Japan

Department of Computer Science
Aalto University
Espoo, Finland

ISSN 1064-3745 ISSN 1940-6029 (electronic)
Methods in Molecular Biology
ISBN 978-1-4939-8560-9 ISBN 978-1-4939-8561-6 (eBook)
https://doi.org/10.1007/978-1-4939-8561-6

Library of Congress Control Number: 2018948727

Printed on acid-free paper

This Humana Press imprint is published by the registered company Springer Science+Business Media, LLC part of Springer Nature.
The registered company address is: 233 Spring Street, New York, NY 10013, U.S.A.

Preface

Five years have passed since the first edition of *Data Mining for Systems Biology: Methods and Protocols*. In these years, we have witnessed the acceleration and diversity of the development of data mining (or, more generally, data-driven) approaches for life sciences. The first edition showed numerous data mining works for various parts of the "central dogma" of molecular biology. This principle continues, whereas some portions are more focused on new fields within or out of the central dogma that have emerged as applications. These new directions can be summarized in the following two categories: (1) genomics, particularly metagenomics and epigenomics, to deepen the knowledge of genes and genomes, i.e., the "origin" of the central dogma, and (2) metabolism (and metabolome) and also relevant medicine-oriented subjects, i.e., the "future" ahead the central dogma. These two focuses are inevitably revealed in the seventeen chapters in the volume. That is, the first nine chapters are more related with the above first point, i.e. genomics, and the rest of the chapters are rather on the other side, i.e. metabolism.

Metagenomics is an emerging research field in biology, closely relevant to lots of aspects of our life. Tools for metagenomics would be highly helpful for understanding the structure of, for example, the microbiome currently detectable everywhere. Mäklin, Corander, and Honkela (Chapter 1) show a powerful probabilistic method and software for estimating the relative abundance of species or strains in mixed samples with information by short-read high-throughput sequencing. Vervier, Mahé, and Vert (Chapter 2) present a scalable machine learning implementation based on nucleotide motifs for sequence classification task in metagenomics. The current sequence data is massive, and definitely scalability is key to analyzing those data. Lund, Tan, and Baumbach (Chapter 3) present, focusing on 16 s ribosomal RNA genes, an interactive web tool for exploring and analyzing prokaryotic distributions by integrating various metagenomics databases.

Epigenomics and chemical modification are an attention-paid topic in current molecular biology. Äijö, Bonneau, and Lähdesmäki (Chapter 4) describe a generative probabilistic model of integrative experiment analysis for multiple genomic measurements, focusing on cytosine methylation. Frisch, Gøttcke, Röttger, Tan, and Baumbach (Chapter 5) present an easy-to-use Java tool for the analysis of DNA methylation from EWAS data through a graphical user interface. Perna, Canakoglu, Pinoli, Ceri, and Wong (Chapter 6) present a web server implementing a robust, statistical method to infer physically interacting transcription factors by integrating and managing heterogeneous, large genomic datasets, such as those from ChIP-seq experiments.

Indeed, the interaction of two or more biological units, such as genes, transcription factors, and single nucleotide polymorphisms (SNPs), is clearly important in biology, while possible combinations are huge, making discovery of significant cases very hard. Terada and Tsuda (Chapter 7) provide comprehensive instructions of their software for thoroughly and efficiently detecting statistically significant combinatorial effects, such as transcriptional regulation discovery and interactions among multiple SNPs. Takahashi, duVerle, Yotsukura, Takigawa, and Mamitsuka (Chapter 8) present a tool for enumerating interactions of genes as biclusters thoroughly and further generating a network of interacting genes, where each network node corresponds to a set of genes, which can be linked to functional annotations.

Ding, Wei, and Kihara (Chapter 9) show a web server for visualizing and quantifying the functional annotations of gene ontology (GO) to make GO term annotation more biologically interpretable.

Genes can be functioning in a number of ways, while an important aspect of gene functions is revealed through the metabolic network (or metabolome) interacting with other biological molecules, mainly chemical compounds (including specific types such as glycans), which can be drugs for network disorder. Aoki-Kinoshita (Chapter 10) presents a procedure to discover glycan profiles through multiple tree alignment over glycans. Bhadra and Rousu (Chapter 11) discuss a new methodology for metabolism analysis by combining two main and complementary approaches, principal component analysis (PCA) and stoichiometric flux analysis, through an elegant regularized optimization framework. Halloran (Chapter 12) describes the Dynamic Bayesian network for Rapid Identification of Peptide (DRIP) toolkit, which methodologically improves the current, static alignment strategy in a dynamic alignment manner, for identifying the peptide responsible for producing each observed spectrum. Yamanishi (Chapter 13) introduces regular methodological protocols for deriving recent methods of sparse modeling for analyzing drug-target interaction networks. Deng, Yuan, Mamitsuka, and Zhu (Chapter 14) present a web service which combines two main approaches for predicting drug-target interactions, i.e., feature-based and similarity-based, by using a machine learning technique, Learning to Rank.

Disorders in cellular networks could cause various diseases, raising numerous medicine-related research problems with various types of data, such as medical literature, ontologies, networks, and pathways, to be tackled by data mining. Peng, Mamitsuka, and Zhu (Chapter 15) discuss the recent progress of indexing medical documents by Learning to Rank, which allows us to solve the two intrinsic problems of medical text mining: (1) only around 10 keywords for each document, out of totally about 28,000, and (2) also given information being just a word set (bag-of-words) for each document. Ata, Fang, Wu, Li, and Xiao (Chapter 16) present a method, based on metagraphs, generated by combining protein-protein interactions with keywords for proteins, particularly for building classifiers to predict genes highly related with diseases. Kanehisa (Chapter 17) shows the knowledge accumulated in the database, Kyoto Encyclopedia of Genes and Genomes (KEGG), and also the usage of web tools in KEGG, taking inferring antimicrobial resistance (AMR) as an example. The focused part of the database, KEGG Pathogen, is linked to other various parts of KEGG, such as genome sequences, high-throughput data, pathways, and ontologies. Interestingly, covering a wide variety of data in this chapter leads us back to the first main part of this book, i.e., genomics.

I expect this book to be of interest to all researchers of biology and relevant fields, such as medical, pharmaceutical, and agricultural sciences, as well as to the scientists and engineers who are (or are interested in) developing data-driven techniques, such as databases, data sciences, data mining, visualization systems, and machine learning (or more generally artificial intelligence) that now are central to the paradigm-altering discoveries being made with higher frequency.

Uji, Kyoto, Japan / Espoo, Finland *Hiroshi Mamitsuka*

Contents

Contributors

TARMO ÄIJÖ • *Center for Computational Biology, Flatiron Institute, New York, NY, USA*

KIYOKO F. AOKI-KINOSHITA • *Faculty of Science and Engineering, Soka University, Tokyo, Japan*

SEZIN KIRCALI ATA • *School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore*

JAN BAUMBACH • *Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark*

SAHELY BHADRA • *Indian Institute of Technology, Palakkad, India*

RICHARD BONNEAU • *Center for Computational Biology, Flatiron Institute, New York, NY, USA; Department of Biology, Center for Genomics and Systems Biology, New York University, New York, NY, USA; Courant Institute of Mathematical Sciences, New York University, New York, NY, USA*

ARIF CANAKOGLU • *DEIB, Politecnico di Milano, Milano, Italy*

STEFANO CERI • *DEIB, Politecnico di Milano, Milano, Italy*

JUKKA CORANDER • *Helsinki Institute for Information Technology (HIIT), Department of Mathematics and Statistics, University of Helsinki, Helsinki, Finland; Department of Biostatistics, University of Oslo, Oslo, Norway*

JIEYAO DENG • *School of Computer Science, Fudan University, Shanghai, China; Shanghai Key Lab of Intelligent Information Processing, Fudan University, Shanghai, China*

ZIYUN DING • *Department of Biological Science, Purdue University, West Lafayette, IN, USA*

DAVID A. DUVERLE • *Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, University of Tokyo, Kashiwa, Japan*

YUAN FANG • *School of Information Systems, Singapore Management University, Singapore, Singapore*

TOBIAS FRISCH • *Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark*

JONATAN GØTTCKE • *Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark*

JOHN T. HALLORAN • *Department of Public Health Sciences, University of California, Davis, CA, USA*

ANTTI HONKELA • *Helsinki Institute for Information Technology (HIIT), Department of Mathematics and Statistics, University of Helsinki, Helsinki, Finland; Department of Public Health, University of Helsinki, Helsinki, Finland*

MINORU KANEHISA • *Institute for Chemical Research, Kyoto University, Uji, Japan*

DAISUKE KIHARA • *Department of Biological Science, Purdue University, West Lafayette, IN, USA; Department of Computer Science, Purdue University, West Lafayette, IN, USA*

HARRI LÄHDESMÄKI • *Department of Computer Science, Aalto University School of Science, Aalto, Finland*

XIAO-LI LI • *Data Analytics Department, Institute for Infocomm Research, Singapore, Singapore*

JESPER LUND • *Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark*

PIERRE MAHÉ • *Bioinformatics Research Department, BioMérieux, Marcy-l'Étoile, France*

TOMMI MÄKLIN • *Helsinki Institute for Information Technology (HIIT), Department of Mathematics and Statistics, University of Helsinki, Helsinki, Finland*

HIROSHI MAMITSUKA • *Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Japan; Department of Computer Science, Aalto University, Espoo, Finland*

SHENGWEN PENG • *School of Computer Science, Fudan University, Shanghai, China; Shanghai Key Lab of Intelligent Information Processing, Fudan University, Shanghai, China*

STEFANO PERNA • *DEIB, Politecnico di Milano, Milano, Italy*

PIETRO PINOLI • *DEIB, Politecnico di Milano, Milano, Italy*

RICHARD RÖTTGER • *Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark*

JUHO ROUSU • *Helsinki Institute for Information Technology (HIIT), Department of Computer Science, Aalto University, Espoo, Finland*

KEI-ICHIRO TAKAHASHI • *Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Japan*

ICHIGAKU TAKIGAWA • *Division of Computer Science and Information Technology, Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Hokkaido, Japan*

QIHUA TAN • *Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark*

AIKA TERADA • *PRESTO, Japan Science and Technology Agency, Kawaguchi, Japan; Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, University of Tokyo, Tokyo, Japan*

KOJI TSUDA • *Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, University of Tokyo, Tokyo, Japan; Center for Materials Research by Information Integration, National Institute for Materials Science, Tsukuba, Japan; RIKEN Center for Advanced Intelligence Project, Wako, Japan*

JEAN-PHILIPPE VERT • *MINES ParisTech, PSL Research University, CBIO-Centre for Computational Biology, Fontainebleau, France; Institut Curie, Paris Cedex, France; INSERM U900, Paris, France; Département de mathématiques et applications, École normale supérieure, CNRS, PSL Research University, Paris, France*

KÉVIN VERVIER • *Department of Psychiatry, University of Iowa Hospital and Clinics, Iowa City, IA, USA*

QING WEI • *Department of Computer Science, Purdue University, West Lafayette, IN, USA*

LIMSOON WONG • *School of Computing, National University of Singapore, Singapore, Singapore*

MIN WU • *Data Analytics Department, Institute for Infocomm Research, Singapore, Singapore*

XIAOKUI XIAO • *School of Computing, National University of Singapore, Singapore, Singapore*

YOSHIHIRO YAMANISHI • *Division of System Cohort, Medical Institute of Bioregulation, Kyushu University, Fukuoka, Japan; PRESTO, Japan Science and Technology Agency, Saitama, Japan*

SOHIYA YOTSUKURA • *Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Japan*

QINGJUN YUAN • *School of Computer Science, Fudan University, Shanghai, China; Shanghai Key Lab of Intelligent Information Processing, Fudan University, Shanghai, China*
SHANFENG ZHU • *School of Computer Science, Fudan University, Shanghai, China; Shanghai Key Lab of Intelligent Information Processing, Fudan University, Shanghai, China; Center for Computational System Biology, Fudan University, Shanghai, China*

Chapter 1

Identifying Bacterial Strains from Sequencing Data

Tommi Mäklin, Jukka Corander, and Antti Honkela

Abstract

Environmental and clinical settings can host a wide variety of both bacterial species and strains in a single colony but accurate identification of the organisms is difficult. We describe BIB, a probabilistic method for estimating the relative abundances of species or strains contained in mixed samples analyzed by short read high-throughput sequencing. By grouping closely related strains together in clusters, the BIB pipeline is capable of estimating the relative abundances of the clusters contained in a sequencing sample.

Key words Bacteria, Strain identification, Abundance estimation, Metagenomics, Probabilistic modelling

1 Introduction

Infections caused by mixtures of bacterial strains can have varying symptoms or require specialized treatment [1]. Classical methods for identifying the infecting strains are based on culturing the bacteria [2] but often lose some of the strains in the process. Sequencing data can be used to identify and quantify the strains or species represented in a sample rapidly and accurately.

Existing computational methods for identifying bacterial strains can be divided based on their approach into four classes: assembly-based, single nucleotide polymorphism (SNP)-based, marker-gene-based, and alignment-based [3, 4]. Assembly-based methods have the advantage of being able to identify the genomes of novel strains contained in metagenomic samples but assembling mixed samples is challenging due to nonuniform sequencing depth of the organisms and difficulty in distinguishing between very closely related strains [4]. SNP, marker gene, and alignment based methods are restricted to identifying previously known bacteria, but in doing so they are able to leverage the large amount of existing data. The sensitivity of the methods increases as more information is used. SNP-based identification is performed by comparing SNP differences between strains which requires very high sequencing depth in metagenomic samples to identify a sufficient number of SNP sites [3]. Marker-gene-based methods can rapidly assign reads to a sequence but are only applicable to species with conserved genes across the taxa and complete genomes available [4]. Both SNP-based and marker-gene-based methods require non-trivial additional work to define good markers. Alignment-based methods such as BIB depend on a high-quality reference genome set for alignment, but these are quickly becoming available for more and more species. Given the reference, the methods can automatically make full use of all the information, leading to superior sensitivity and simplicity over SNP and marker-gene methods.

We base our strain identification on aligning reads against a set of reference genomes. To increase identifiability of reads originating from closely related strains, the reference genomes are grouped together according to some clustering or using previously defined clonal complexes or other sequence types. The resulting groups of bacteria are easier to identify than the individual strains by solving the easier task of inferring the relative abundances of the clusters. By defining a likelihood for a read to have originated from one of the clusters based on the alignments and modelling the sequencing process as a process mixing reads from the clusters to obtain a sample, mixture modelling can be applied to effectively obtain the relative abundances.

The BIB method [5] is founded upon an analogy between transcript isoform expression estimation in RNA-sequencing [6, 7] and bacterial strain abundance quantification, both of which require the estimation of the abundance of highly similar sequences (alternatively spliced transcript isoforms or different bacterial strains) from short read sequencing data. Many of the most successful methods for the RNA-sequencing transcript isoform expression estimation [8] are based on probabilistic modelling of the read generation and estimation of the mixing proportions of the different sequences [9–11]. BIB is built upon BitSeqVB [7], which is reasonably fast and provides the most accurate quantification according to a recent assessment [8].

2 Materials and Methods

2.1 Overview

The abundance estimation naturally splits into two distinct phases: *preprocessing* the reference sequences and *analysis* of the reads. Preprocessing consists of defining a collection of reference bacterial genomes to align sequencing reads against, and obtaining a grouping or clustering of the reference sequences. Depending on the species, reference sequences, and the level of detail desired, the clustering or grouping can be done, for example, using multilocus sequence typing [12], clonal complexes, or some algorithm designed to cluster bacterial genomes such as BAPS [13, 14].

Analyzing the reads R is done by obtaining an alignment against the reference sequences and then modelling the reads in a sample as a mixture of reads independently drawn from sequences contained in the reference clusters containing one or more reference genomes. We define a likelihood for a read r_n to have originated from a cluster, $I_n = k$, through the obtained alignments based on assuming that observations of the reads are independent of each other conditional on the cluster that generated them. If the indicator variable $I \sim \text{Categorical}(\theta)$ is assumed to follow a categorical distribution, the mixing proportions θ of the clusters correspond to their relative abundances in a sample.

Based on the above assumptions, the joint distribution of the variables $p(R, I, \theta)$ and the posterior distribution $p(\theta|R)$ over the mixing proportions can be written as

$$p(R, I, \theta) = p(R|I)\, p(I|\theta)\, p(\theta) = p(\theta) \prod_{n=1}^{N} p(r_n|I_n)\, p(I_n|\theta) \tag{1}$$

$$p(\theta|R) \propto p(R|\theta)\, p(\theta) = \sum_{I} p(R|I)\, p(I|\theta)\, p(\theta)\,. \tag{2}$$

We set a conjugate (to the categorical distribution) Dirichlet $(\alpha, \ldots, \alpha)$ prior on the mixing proportions to enable inference of the posterior based on the alignments. The prior parameters $\alpha = 1$ are set to one.

Obtaining the posterior distribution is done by applying variational inference similarly as in the RNA-seq case [7]. Variational Bayesian methods aim to derive an approximating distribution q to the true posterior such that

$$q(\theta, I) \approx p(\theta, I|R) \tag{3}$$

and then optimize the approximation to be as close to the true posterior as possible by minimizing the Kullback-Leibler divergence between the two. Using variational Bayes to estimate the mixing proportions instead of more traditional methods such as Markov Chain Monte Carlo sampling has the benefit of being significantly faster [7].

Incorporating the above ideas in an abundance estimation pipeline results in a method that is rapid and accurate in identifying the relative abundances of the clusters in a sample. Our provided implementation in the form of the BIB software uses one reference sequence for each cluster but the model can be extended to utilize multiple reference sequences per cluster.

2.2 Software

The described abundance estimation pipeline is available in the form of the BIB, short for Bayesian Identification of Bacteria, software consisting of Python scripts that execute the necessary steps for strain identification. Scripts are provided for building the alignment index from the reference sequences, and for obtaining the relative abundances of the reference clusters by running both the read alignment and the abundance estimation.

Some software must be installed prior to running BIB. To perform the alignment, BIB uses Bowtie2 [15], which is designed to align short reads to a reference. The abundance estimation is done by using the BitSeqVB [6, 7] software. Abundance estimation consists of two steps: computing the alignment probabilities for the alignments observed by Bowtie2 and estimating the relative abundances. We provide a script that runs both the read alignment and the two steps in abundance estimation given the reads, the alignment index, and the reference sequences. The script outputs the relative abundances of the clusters to a text file.

2.3 Sequencing Data

BIB is designed to work with high-throughput short-read sequencing data. The data are either single or paired-end reads from uncultured or cultured bacterial samples containing single or multiple species or strains. BIB estimates the relative abundance of the reference clusters in the sample. No preprocessing of the reads is necessary, as both the alignment and the probabilistic model used will filter out noisy low-quality reads.

2.4 Reference Collection and Clustering

Sequences included in the reference should be chosen after consideration of the following viewpoints:

1. The reference collection should include representatives for each species or strain believed to be in the samples.
   Rationale: BIB will assign all aligned reads to some cluster in the reference collection. Reads originating from a species or strain not represented in the reference tend to be assigned to the closest match which will confound the results.
2. Very similar strains should be clustered together.
   Rationale: Including each strain as an independent cluster will both slow down the analysis and worsen the results because the signal gets diluted, possibly leading to worse identification than with larger clusters [5].
3. Expected potentially contaminating or otherwise uninteresting species may be represented by a single catcher sequence.
4. For BIB, the single reference sequence representing the cluster should be chosen to be as representative as possible, for example by using a sequence that minimizes some distances to all others in the cluster.

5. If gapped sequences or sequences containing ambiguous bases are used in the reference, the gaps and the ambiguous characters should be removed prior to use.
6. For species with stable core genomes, the core genome alignment may be used as reference.
   Rationale: This can reduce confusion caused by mobile elements in the accessory genome [5].

When estimating the relative abundances of different species, clustering need not be performed, and the reference sequences are simply representatives for the species. For relative abundance analysis within the species, reference sequences for the species should be clustered according to biologically relevant criteria.

After producing a clustering for the reference genomes, in the BIB approach a representative genome is drawn from each of the clusters. The drawn genomes are then included in a fasta file which is fed to the indexing script *BIB_prepare_index.py* that produces the alignment index. Preprocessing the reference is complete after the index has been produced.

2.5 Read Alignment and Abundance Estimation

Both the read alignment and the abundance estimation are performed by the second *BIB_analyse_reads.py* script. The script requires as input the read file(s), the reference sequences, and the alignment index. Running the script first performs the read alignment and then runs the abundance estimation which outputs the relative abundances to a text file specified by the last argument.

The estimation process first computes the alignment probabilities for each read and each observed alignment before applying the BitSeqVB algorithm to obtain the posterior probabilities for the mixing proportions. Applying the algorithm to the alignments produces an output file containing the abundances where the first column in the file (mean theta) corresponds to the estimated relative abundance of the clusters while the next two characterize the posterior variance. The file will contain a line for each cluster as represented by a reference genome included in the reference collection.

The probabilistic model used by the BitSeq, and consequently the BIB implementation, includes an additional parameter modelling the probability of a sequenced read being noise [6]. This adds an extra line in the output containing this noise probability [5].

2.6 Example

To illustrate the various steps in running the BIB pipeline from scratch, we construct a reference set from four *Staphylococcus aureus* assemblies and estimate the relative abundances from sequencing data for a fifth isolate. The reference assemblies are available in the GitHub repository https://github.com/PROBIC/BIB-S-aureus-example and the sequencing reads can be downloaded from the European Nucleotide Archive with run accession SRR016122.

1. Obtain the core genome alignment.

```
progressiveMauve --output=full_alignment.xmfa
                   saur_assemblies.fasta
```

2. Extract LCBs shared by all genomes.

```
stripSubsetLCBs full_alignment.xmfa
                full_alignment.xmfa.bbcols
                core_alignment.xmfa 500 4
```

   The first number "500" is the minimum length of the LCB; the second number "4" indicates the minimum number of genomes that share an LCB.

3. Concatenate all the LCBs.

```
perl xmfa2fasta.pl --file core_alignment.xmfa
                   > core_alignment.fasta
```

4. Run hierBAPS to obtain a clustering with 1 level and a maximum of 10 clusters.

```
hierBAPS.sh exData core_alignment.fasta fasta
hierBAPS.sh hierBAPS seqs.mat 1 10 results
```

   The clustering is stored in the *results.partition.txt* file. Sequences 1 and 2 belong to cluster 1, and sequences 3 and 4 to cluster 2.

5. Select sequences 1 and 3 as the reference sequences using fastagrep which is included in the BitSeq distribution.

```
fastagrep.sh ">1 " core_alignment.fasta
             > ref_seqs.fasta
fastagrep.sh ">3 " core_alignment.fasta
             >> ref_seqs.fasta
```

6. Remove gaps from the reference sequences.

```
sed 's/-//g' ref_seqs.fasta
             > ref_seqs_gapless.fasta
```

7. Build the alignment index.

```
python BIB_prepare_index.py ref_seqs_gapless.fasta
          reference_alignment_index
```

8. Perform the read alignment and abundance estimation.

```
python BIB_analyse_reads.py SRR016122.fastq.gz
          ref_seqs_gapless.fasta
          reference_alignment_index
          SRR016122_abundances
```

Acknowledgements

This work was supported by the Academy of Finland [259440 to A.H., 251170 to J.C.].

References

1. Balmer O, Tanner M (2011) Prevalence and implications of multiple-strain infections. Lancet Infect Dis 11:868–878
2. Didelot X, Bowden R, Wilson DJ, Peto TEA, Crook DW (2012) Transforming clinical microbiology with bacterial genome sequencing. Nat Rev Genet 13:601–612
3. Brito IL, Alm EJ (2016) Tracking strains in the microbiome: insights from metagenomics and models. Front Microbiol 7:712
4. Breitwieser FP, Lu J, Salzberg SL (2017) A review of methods and databases for metagenomic classification and assembly. Brief Bioinf https://doi.org/10.1093/bib/bbx120
5. Sankar A, Malone B, Bayliss SC, Pascoe B, Méric G, Hitchings MD et al (2016) Bayesian identification of bacterial strains from sequencing data. Microb Genomics 2:e000075
6. Glaus P, Honkela A, Rattray M (2012) Identifying differentially expressed transcripts from RNA-seq data with biological variation. Bioinformatics 28:1721–1728
7. Hensman J, Papastamoulis P, Glaus P, Honkela A, Rattray M (2015) Fast and accurate approximate inference of transcript expression from RNA-seq data. Bioinformatics 31:3881–3889
8. Kanitz A, Gypas F, Gruber AJ, Gruber AR, Martin G, Zavolan M (2015) Comparative assessment of methods for the computational inference of transcript isoform abundance from RNA-seq data. Genome Biol 16:150
9. Xing Y, Yu T, Wu YN, Roy M, Kim J, Lee C (2006) An expectation-maximization algorithm for probabilistic reconstructions of full-length isoforms from splice graphs. Nucleic Acids Res 34:3150–3160
10. Jiang H, Wong WH (2009) Statistical inferences for isoform expression in RNA-Seq. Bioinformatics 25:1026–1032
11. Li B, Ruotti V, Stewart RM, Thomson JA, Dewey CN (2010) RNA-Seq gene expression estimation with read mapping uncertainty. Bioinformatics 26:493–500
12. Maiden MC, Bygraves JA, Feil E, Morelli G, Russell JE, Urwin R et al (1998) Multilocus sequence typing: a portable approach to the identification of clones within populations of pathogenic microorganisms. Proc Natl Acad Sci USA 95:3140–3145
13. Cheng L, Connor TR, Sirén J, Aanensen DM, Corander J (2013) Hierarchical and spatially explicit clustering of DNA sequences with BAPS software. Mol Biol Evol 30:1224–1228
14. Corander J, Sirén J, Arjas E (2008) Bayesian spatial modeling of genetic population structure. Comput Stat 23:111
15. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. Nat Methods 9:357–359

Chapter 2

MetaVW: Large-Scale Machine Learning for Metagenomics Sequence Classification

Kévin Vervier, Pierre Mahé, and Jean-Philippe Vert

Abstract

Metagenomics is the study of microbial community diversity, especially the uncultured microorganisms by shotgun sequencing environmental samples. As the sequencers throughput and the data volume increase, it becomes challenging to develop scalable bioinformatics tools that reconstruct microbiome structure by binning sequencing reads to reference genomes. Standard alignment-based methods, such as BWA-MEM, provide state-of-the-art performance, but we demonstrate in Vervier et al. (2016) that compositional approaches using nucleotides motifs have faster analysis time, for comparable accuracy. In this work, we describe how to use MetaVW, a scalable machine learning implementation for short sequencing reads binning, based on their k-mers profile. We provide a step-by-step guideline on how we trained the classification models and how it can easily generalize to user-defined reference genomes and specific applications. We also give additional details on what effect parameters in the algorithm have on performances.

Key words Metagenomics, Machine learning, Classification, Next-generation sequencing, Microbiology, Binning

1 Introduction

Metagenomics, the study of microbial community genomic content, has become a popular sequencing protocol to access all organisms present in a sample including those who do not grow on culture media [1]. The experimental output of a metagenomics experiment consists in a collection of short sequencing reads obtained by shotgun sequencing of the microbial DNA in the sample [2]. Recent *taxonomic binning* methods explicitly assign each read to a clade of a pre-defined taxonomy, using only the set of *k*-mers it contains [3, 4]. It is a necessary step for downstream applications which require draft-genome reconstruction or fine-grain characterization. This may notably be the case in a diagnosis context, where further analyses could aim to detect pathogen microorganisms or antibiotic resistance mechanisms [5]. However, training a machine learning model for taxonomic binning with standard libraries is computationally challenging, with potentially billions of training examples, each represented by a vector in millions of dimensions for, e.g., *k*-mers of lengths 12. Here, we demonstrate the potential of compositional approaches for taxonomic label assignment using a large-scale machine learning algorithm, called Vowpal Wabbit. We show that it provides an interesting trade-off in speed and accuracy, particularly when confronted to species absent from the reference database (rank-flexible) and for a moderate number of candidate species (700).

2 Materials

2.1 MetaVW: Metagenomics Reads Classification

The following archive[1] contains the data used in [3], as well as programs allowing to reproduce the results under UNIX or Mac OS X distributions. This archive is structured as follows:

- The *data* directory contains the sequence data used to carry out the experiments.
- The *src* directory contains script allowing to reproduce the experiments.
- The *tools* directory contains source code of utilities used to draw fragments from genomic sequences.

Details on the content of the three directories can also be found at http://projects.cbio.mines-paristech.fr/largescalemetagenomics. After downloading the project archive, first untar it. Then, run the BASH script `INSTALL.sh` that can be found in the `tools` directory. This script will process the installation of the GDL library and create the binary executables. In order to check that everything went well during the installation, please use the `tools/test/test.sh` script. If no error is detected, it will use installed tools to simulate a toy dataset.

2.2 Third-Party Softwares

To run the experiments, you need, in addition, to install the following third-party softwares:

- Vowpal Wabbit (version $\geq$ 7.7.0), the machine-learning algorithm
- Grinder (version $\geq$ 0.5.3) to simulate datasets with sequencing errors
- R (version $\geq$ 3.0.0) to analyze and plot the results

[1] http://projects.cbio.mines-paristech.fr/largescalemetagenomics/large-scale-metagenomics-1.0.tar.gz.

Our implementation uses the following libraries, which you do not need to install since they are provided in our software for convenience

- kseq library to parse FASTA/FASTQ files
- GDL library implementing several structures like lists and hash tables (note that we provide a slightly updated version that properly compiles on Mac OS X)

2.3 Reference Genome Databases for Rank-Specific Applications

In [3], we considered three databases of genomes (*small*, *medium*, and *large*) for different validation applications. For each of those databases, we carefully separated *reference* (training) genomes and *validation* genomes to avoid overoptimistic estimation of classification performance.

The *small* database consists of 356 complete genome sequences, from 51 bacterial species, described in `data/train-dataset/small-DB/reference-dataset`. For the *small* validation set, we randomly selected 52 genomes (*see* **Note 1**) and removed them from the reference database. We recommend to use the *small* database to rapidly benchmark performances and parameters impact, such as *k*-mer size. We used the *medium* and *large* databases to create classification models on a scale larger than with the *small* database. We downloaded the 5201 complete bacterial and archeal genomes available on RefSeq as of July 2014 (*see* **Note 2**). We then filtered these sequences with only keeping well-covered genera with at least three species representants and also removed short genomes (less than a million base pairs), corresponding to draft genomes, plasmids, and contigs. The filtered database consists in 774 species and 2961 genomes. For the *medium* database, we extracted 110 species with at least three strains and randomly selected one strain per species to go in the validation set. We added to the *medium* reference set genomes from 83 species with only two strains. The *large* validation set is made of one randomly selected strain from the 193 (110 + 83) species, and the remaining 2768 genomes create the *large* reference database. The FASTA files for both *medium* and *large* databases are located in `data/train-dataset`.

2.4 Reference Genome Databases for Rank-Flexible Applications

We also evaluate MetaVW models in more challenging settings and designed a *novel lineage* validation set, composed of genomes we filtered in the rank-specific application, because of lack of coverage at the genus level, with less than three species. All the species present in this database were not considered during the model training process, so one does not expect the model to provide a classification answer at the species level but above. Table 1 shows the number of strains involved in the *novel lineage* validation set from the *large* database.

Table 1
Number of strains involved in the novel-lineage study, per reachable rank

Reachable rank	Test strains considered	Test species	Reference taxa represented
Genus	584	421	69/126
Family	338	146	42/114
Order	183	147	18/54
Class	143	111	9/52
Phylum	97	81	4/16

The first column gives the number of strains considered for each reachable rank. The second column gives the number of species these strains originate from. The last column shows the number of taxa of this rank that they represent in the *large* reference database. For instance, the first row means that 584 strains coming from 421 species are reachable at the genus level, but not beneath, and represent 69 genera of the 126 genera present in the reference database

2.5 Utility Functions

In the `tools` directory, one can find `C` utility programs which most of the data processing rely on, prior to model training, and classification predictions. *Drawfrag* is used to draw random fragments (*see* **Note 3**) from reference genomes. Users can provide the fragment `size`, the expected average genome `coverage`, and the reference genomes repository. *Fasta2vw* is used to convert a FASTA file into a plain text file compliant with VW input format (*see* **Note 4**). *Spectrumpredict* computes predictions faster than the regular VW program, because it takes advantage of the particular structure of the *k*-mers (*see* Subheading 3). Each of these tools has its own help menu.

3 Methods

3.1 Read Classification Model Using Machine Learning

Model training is done using `src/main.sh` script. In this section, we provide a step-by-step description of the process, but interested users do not have to run each command separately. Figure 1 gives an overview of the training and prediction processes.

Training sets for large number of classes and high coverage do not often fit in memory. Online learning can be used to reduce the memory footprint of the model training step. It consists in generating a small subset of the whole training data and only uses one batch at a time to optimize the model weights. In the MetaVW framework, training observations are drawn from reference genomes `DB`, using the `drawfrag` tool. Those fragments have a constant length `L` (default: 200 bp), and the number of sampled fragments is defined by the mean `COVERAGE` for all genomes in the reference database. Each set of random fragments is called a training batch. Each of these short DNA sequences requires to be processed in the VW input format (*see* Fig. 1 for

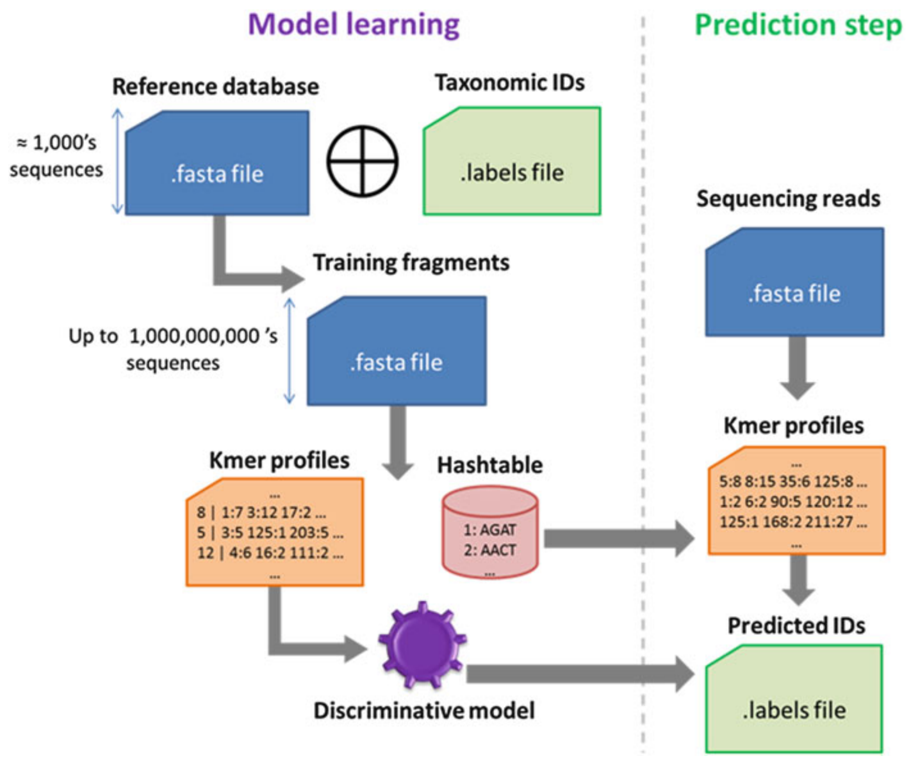


Fig. 1 MetaVW overview. The left panel represents the training step, where reference genomes are used to generate sequencing reads fed to Vowpal Wabbit algorithm to optimize the discriminative model. The right panel shows how this model is then used to make predictions on a new dataset

an example). Especially, the tool `fa2vw` converts FASTA sequence into a vector of K-mer frequencies (*see* **Note 5**). As a linear representation, MetaVW models consist each in a weight vector, reflecting how important each motif is. VW implementation uses hashing strategy to efficiently store and query the model weights. Weight optimization is obtained through supervised learning on training sequences. For each training batch, model predictions are compared with the actual taxon and weights are adjusted in case of classification error. We demonstrated in [3] that after iteratively seeing enough `NBATCHES` data (*see* **Note 6**), the model performances starts to plateau.

3.2 Use Vowpal Wabbit for Online Learning

The script `2-build-models/src/`01.main.sh is the central element of the MetaVW model learning step. Multiple VW parameters (*see* **Note 7**) were optimized using the *small* database in [3]. Especially, we focused on the impact of the number of training examples, the *k*-mer size, and the hash table size in `BITS` (*see* **Note 8**). For each iteration through a new batch of training data, we update the existing model (`-save_resume` flag), providing a backup version of the predictive model in case of the process unexpectedly stops.

3.3 Generate Training Set

One of the main advantages of VW, previously described, is its ability to process training data in small random batches. By combining the tools `drawfrag` and `fa2vw`, it is possible to directly stream training examples into VW without writing the data on disk. This procedure is also memory efficient given that only one batch at a time is loaded in memory. Therefore, the whole training data in VW format is not stored. One important step is to randomize the order the fragments are presented to the algorithm; `drawfrag` by default groups together all the fragments coming from the same reference genome, which is not desirable when streaming data in online learning. Therefore, we pipe the output of `fa2vw` into an efficient `awk/rand` combination, while keeping the mapping between the fragments and the corresponding taxon. This randomly sorted data is then directly streamed as an input for the Vowpal Wabbit algorithm.

3.4 Generate Validation Sets

In [3] we extensively evaluated MetaVW using multiple difficulty levels. First, we generate a set of error-free fragments simply by using the drawfrag tool (*see* `01.generate-dataset-fragments.sh`). Then, we proposed to evaluate how the models perform on real-world data with sequencing noise. In [3], two error models were considered. First, `02.generate-dataset-reads-homo.sh` generates the homopolymer test datasets. The other error model provided in `03.generate-dataset-reads-mutation.sh` generates the mutations test dataset. Interestingly, we found that our approach trained on error-free fragments is competitive in terms of accuracy for reasonable amounts of sequencing errors.

3.5 Rank-Specific Predictions

The classification approach described in the previous sections is called rank-specific, given that all the sampled fragments are labeled at the same taxonomic rank (e.g., species). Source code for predicting labels on a set of sequences can be found in `3-make-prediction/01.make-predictions.sh`. The computational aspect of the prediction step involves computing a score for each candidate species, defined as a dot product between the k-mer profile of the sequence to classify and the vector of weights obtained by training the predictive model. To efficiently compute this dot-product on vectors with millions of dimensions, we implemented a procedure, called *spectrumpredict*, described in [6]. With this procedure, each *A, T, G,* and *C* nucleotide is encoded by two bits, which allows to directly convert a k-mer as an integer between 0 and $4^k - 1$. It allows to compute the score directly from a FASTA sequence, without the need to convert the DNA sequence in VW hashed format, which can be time-consuming for such a large number of features. To gain computational efficiency, we reformat the VW model to a binary format (`vw-to-binary.sh`), making it faster to load in memory and process:

1. Use `tools/enumerateKmers` to create a plain text file containing all the possible *k*-mers (4^k).
2. Invert the hash table created along the VW predictive model using the flag `–invert_hash`
3. Extract the weights from the file using `tools/parseHash`

Provided that the weight vector is loaded into memory (*see* **Note 9**), the score can be computed "on the fly" while evaluating the *k*-mer profile of the sequence to be classified, by adding the contribution of the current *k*-mer to the score. To compare and evaluate multiple models, we also provide `src/3-make-predictions/src/02.generate-graphs.R` which computes five different performance indicators, as described in [3].

3.6 Rank-Flexible Predictions

In this section, we describe how taxonomic binning methods can also be used to classify reads coming from novel bacterial lineages at their appropriate taxonomic rank. For instance, a strain is said to be *reachable* at the genus level when its species is not part of the reference database but when other species of the same genus are represented. We derive a rank-flexible classifier able to choose the most adequate taxonomic rank to classify a read. It also includes a rejection option, where a read remains unclassified if too different from the training sequences distribution. Our rank-flexible algorithm is a combination of multiple rank-specific models trained at species, genus, and family levels. To assess which rank is the most suitable for a given prediction, we combine the following two criteria [7]: the *credibility* estimates if the highest predicted score (between -1 and $+1$) is sufficient enough and rejects unlikely predictions, and the *confidence* compares the two top-scoring classes and rejects ambiguous predictions. An iterative process starts from the species-level models and assigns the credible reads to a given species, whereas rejected reads at the species level are predicted by the genus-level models and so on. If a read is rejected by all rank-specific models considered, it is left unclassified. In [3], we described a procedure to optimize the credibility and confidence thresholds and show that trade-offs can be achieved in terms of precision and recall, depending on user-defined settings, including how complex is the studied microbial community and the estimated proportion of novel organisms (species). To calibrate these thresholds, we include an internal calibration step during the model training:

1. Split the reference database into a *calibration database*, obtained by sampling one strain for each species represented by several strains, and a *learning database*, defined from the remaining strains.
2. Build rank-specific models from the learning database (e.g., at the species, genus, and family ranks).

3. Optimize the thresholds for the reject option mechanism using the *calibration database*. This can be done by simulating fragments from the calibration genomes, classifying them using the different models, and optimizing the performance of the model according to the thresholds.

These thresholds can be set on a taxon-per-taxon basis or globally and can be further optimized for each rank. The optimization procedure we used relies on two separate steps. First, we use the same threshold for *credibility* across ranks and taxa. This threshold depends on a user-defined trade-off in terms of proportions of (i) rejected predictions, (ii) predictions made at various ranks, (iii) correctness of predictions at various ranks. This is illustrated in Fig. 2, where in panel A, as expected, the proportion of predictions made at the species level decreases as the threshold value increases, while the proportion of predictions made at upper ranks increases, as well as the rejection rate. We also note from panel B that this procedure allows to reduce the proportion of misclassified sequences, at the cost of unclassified sequences and sequences correctly classified at upper ranks. In [3], we set the global credibility threshold to 0, leading to a reasonable trade-off in terms of error and rejection rate. Second, we define the *confidence* thresholds on a taxon-by-taxon basis. Although the same kind of approach done for credibility threshold can be used, we observed very different behaviors regarding prediction

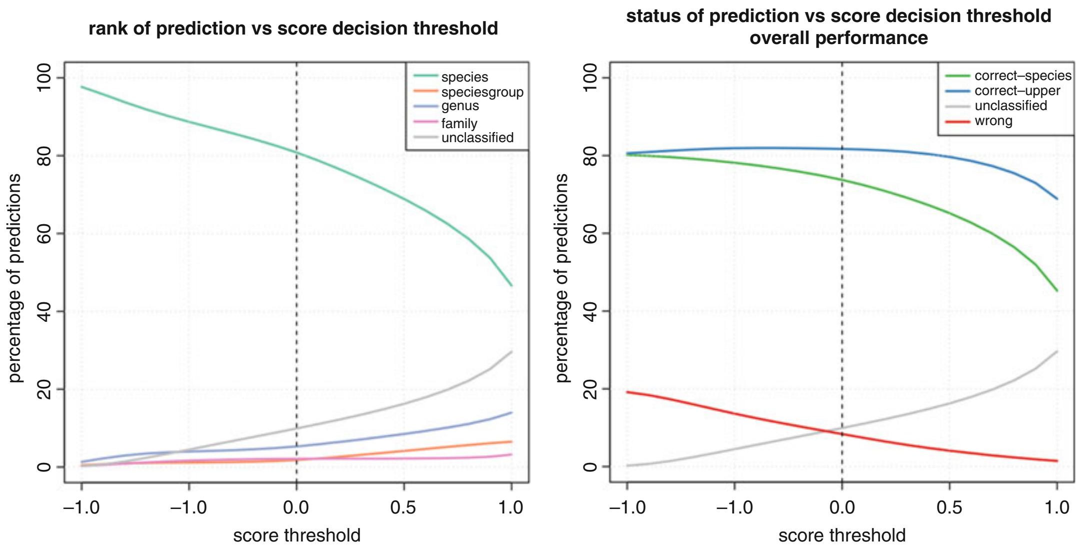


Fig. 2 Calibration procedure and impact of a global credibility threshold. A global credibility threshold taken in [−1; +1] is applied to random fragments from the calibration genomes. Left: evolution of the prediction rank, defined in terms of the proportions of rejected predictions and of predictions made at upper ranks. Right: evolution of the prediction status, defined in terms of the proportions of predictions that are rejected (gray), erroneous (red), correct at the species level (green), and correct at an upper rank (blue)

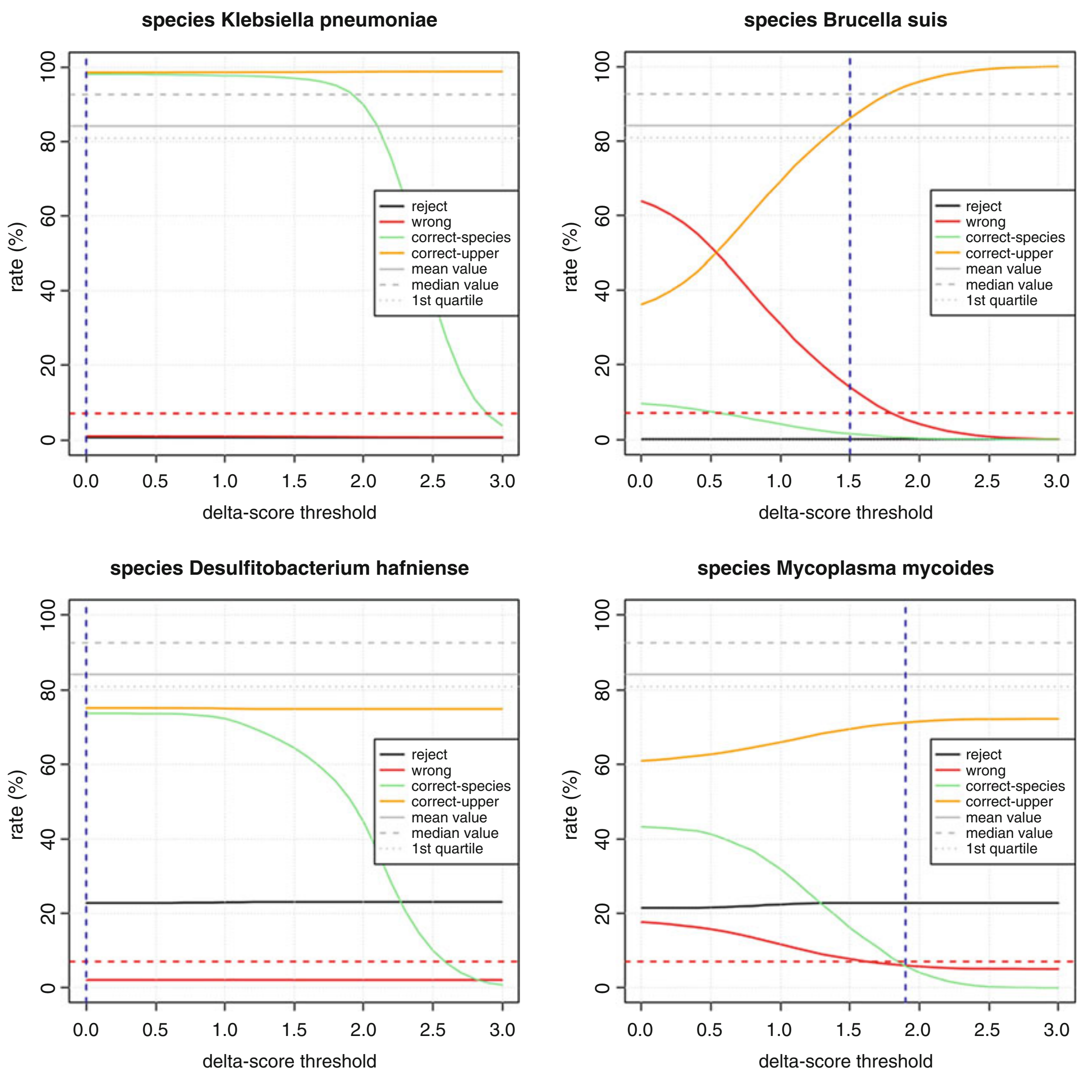


Fig. 3 Calibration procedure for taxon-by-taxon definition of the confidence threshold. Left-hand side: species for which prediction ambiguity is not an issue, hence for which confidence-based rejection does not allow to reduce the error rate. Some of these species show a good classification performance, in particular higher than a predefined target level of performance (top left). Others show a lesser level of performance (bottom left). In both cases, the confidence threshold is set to zero, as shown by the vertical blue lines. Right-hand side: species for which ambiguity is an issue and for which this confidence-based rejection has a positive effect. Top right: increasing the threshold allows to reach a target upper recall performance (solid horizontal gray line). Bottom right: although this confidence-based rejection allows to decrease the error rate, it does not allow to reach the target performance

ambiguity across species. It simply reflects that the level of genomic proximity is not constant across the taxonomy. Therefore, a global threshold is unlikely to be optimal and actually tends to degrade the performance for some taxa. Figure 3 illustrates four behaviors observed while analyzing the effect of the confidence threshold on species-level predictions. For species where ambiguity is not an issue (Fig. 3 left-hand side), the confidence threshold is set to zero. For species for which ambiguity is an issue (Fig. 3 right-hand side), the confidence-based rejection allows to reduce the rate of erroneous predictions. In the case of *B. suis* (top right), this rejection procedure allows to reach a target level of classification performance defined in terms of the average upper recall (orange curve). This target performance is defined, here, as the average value of the upper recall across species, obtained using species-specific models only (solid gray line). For such species, the confidence threshold is set to the minimum value allowing to reach the target performance. In the case of *M. mycoides* shown in the bottom right panel, while decreasing the error rate, it does not allow to reach the target performance. For such species, the confidence threshold is defined as the smallest value allowing to reach a minimum attainable error rate (up to a tolerance of 1% in absolute value). Last but not least, we note that the confidence-based rejection is unnecessary for ranks higher than species and actually degrades the level of resolution of the prediction while not reducing its error rate.

3.7 Training a New VW Model

Users might want to use their own reference database to train a MetaVW model. We make it easy to adapt 01.main.sh to user-defined parameters, where only few lines need to be modified:

1. Update `fasta` = ... line by providing the path to the user-defined reference genome database (FASTA format).
2. Update the following line `taxids` = ... by providing the path to the set of taxon IDs found in the reference genome database.
3. (Optional) Update `outputDir` = ... line with the path where the model and predictions are stored.
4. (Optional) All parameters are provided with the default value used in [3] but can also be modified in the script.

4 Notes

1. The 52 genomes are from 51 species, but two genomes are available for the *Fransiscella tularensis* species, with one actually originating from the *novicida* subspecies.
2. We used the fragment classification package utility script [8] to efficiently query RefSeq server for FASTA sequence files.
3. *Drawfrag* can take a random `seed` value as input argument, allowing reproducible fragment sampling.
4. VW input format consists in a tabulate-separated file, where each line is an input observation, starting with the corresponding class label (training only). Each column

corresponds to the relative abundance of a given k-mer in the considered DNA sequence.

5. The length of the vector can be up to 4^k, corresponding to the number of possible combinations obtained with the four nucleotides being placed in *k* different positions.
6. This process takes some time, generates large files (12 and 25 gigabytes for the small and large databases, respectively), and has a comparable memory footprint. We also observed that training time depends of number of classes (Fig. 7 of [3]).
7. L1 and L2 regularization are parameters that are set to zero by default. If one wants to apply constraints on the classification model weights, it is recommended to evaluate small values first (e.g., 10ˆ-7). Especially for L1 regularization, large values will create smaller models with few features with nonzero weights, leading to a loss of accuracy.
8. VW proceeds by hashing the input features into a vector offering at most 2ˆ32 entries. This hashing operation can induce collisions between features, following the pigeonhole principle, which can be detrimental to the model if the number of features becomes too high with respect to the size of the hash table. This issue is even more stressed in a multiclass setting, where the number of hash table entries available per model is divided by the number of classes considered. For instance, on the *small* dataset, (52-1) models are stored in the hash table, which reduces the number of entries available per model to 2ˆ32/51 4ˆ13. We have empirically observed that performance could not increase for *k* greater than 12 and actually decreased for *k*-mers greater than 15.
9. The drawback of this procedure lies in the fact that the vectors of weights defining the classification models need to be loaded into memory, which can be cumbersome in a large multiclass setting. For 193 and 774 species and k-mers of size 12, this amounted to 12 and 48 gigabytes, respectively.

Acknowledgments

This work was supported by the European Research Council (SMAC-ERC-280032 to J-P.V.).

References

1. Handelsman J (2004) Metagenomics: application of genomics to uncultured microorganisms. Microbiol Mol Biol Rev 68(4):669–685
2. Quince C et al (2017) Shotgun metagenomics, from sampling to analysis. Nat Biotechnol 35(9):833–844

3. Vervier K et al (2016) Large-scale machine learning for metagenomics sequence classification. Bioinformatics 32(7):1023–1032
4. Wood DE, Salzberg SL (2014) Kraken: ultrafast metagenomic sequence classification using exact alignments. Genome Biol 15:R46
5. Simner PJ et al (2018) Understanding the promises and hurdles of metagenomic next-generation sequencing as a diagnostic tool for infectious diseases. Clin Infect Dis 66(5): 778–788
6. Sonnenburg S et al (2006) Large scale learning with string kernels. J Mach Learn Res 7:1531–1565
7. Gammerman A, Vovk V (2007) Hedging predictions in machine learning. Comp J 50(2):151–163
8. Parks D et al (2011) Classifying short genomic fragments from novel lineages using composition and homology. BMC Bioinformatics 12:328–344

Chapter 3

Online Interactive Microbial Classification and Geospatial Distributional Analysis Using BioAtlas

Jesper Lund, Qihua Tan, and Jan Baumbach

Abstract

In recent decades, the accumulation of data on 16s ribosomal RNA genes has yielded free and public databases such as SILVA, GreenGenes, The Ribosomal Database Project, and IMG, handling massive amounts of raw data and meta information. 16s rRNA gene contains hypervariable regions with great classification power. As a result, numerous classification tools have emerged including state-of-the-art tools such as Mothur, Qiime, and the 16s classifier. However, there is a gap between the sequence databases, the taxonomy profiling tools and available meta information such as geo/body-location information. Here, we present BioAtlas, and interactive web tool for searching, exploring, and analyzing prokaryotic distributions by integration of various resources of metagenomics databases. In the following section we show how to use BioAtlas to (1) search and explore prokaryote occurrences across the geospatial map of the world, (2) investigate and hunt for occurrences across generic user-generated surface-specific maps, with an example map of a human female, with data from Bouslimani et al., and (3) classify a user-given sequences dataset through our online platform for visual exploration of the spatial abundances of the identified microbes.

Key words Taxonomic classification, 16s gene, Ribosomal RNA, Distributional analysis, Microbiology, Online tool, Maps, Data mining, Integration, Metadata

1 Introduction

Ever since the method of using the 16s ribosomal RNA (rRNA) gene's genomic-level properties for classifying prokaryotes was proceduralized [8], databases with a vast number of genomic sequences of prokaryotic origin have emerged (SILVA rRNA database project [14] (SILVA), The Ribosomal Database Project [5] (RDP), GreenGenes [7] (GG)). Due to its highly evolutionary intra-species conservation and inter-species hyper-variability [16] the gene has become the standard marker for fast and stable identification of organisms of prokaryotic origin [12]. A number of classification tools emerged: Mothur [15], Qiime [3], 16S Classifier [4], which built on the accumulated 16s RNA resources (SILVA, RDP, GG) and differential taxonomic nomenclatures within the microbial community [6]. In addition, we observe a skyrocketing increase of both manually curated and automatically created meta databases: the Genomes OnLine Database [13] (GOLD), Integrated Microbial Genomes and Microbiomes [10] (IMG), The European Bioinformatics Institute (EMBL-EBI) Metagenomics [11].

Now there is a gap between these sequence databases, the software tools for taxonomy profiling, and the databases for meta information, such as geo/body-locations, which hinders follow-up and integrated data analyses of existing data as well as new data. In order to help form a better, more integrated picture of our metagenomics data, we have developed BioAtlas [9], a novel user-friendly, browser-based analytics software for analyzing genomic and metagenomic distributions across diverse surfaces to study, e.g., the geospatial distribution on the globe, the human skin surface, and generic user-contributed maps. BioAtlas integrates numerous public online tools. It is free, fast, and easy to use, and provides a secure user interaction mode with a focus on usability and effectiveness.

1.1 Brief Description of BioAtlas Data Integration

From GOLD we have extracted and mined 21,786 microbial genomic and meta-genomic sequence projects, of which 2627 contained geospatial information (latitude, longitude, text-based). Based on these projects, and their corresponding raw 16s rRNA sequences retrieved from IMG and sequences of eukaryote origin discarded, a total of 459,304 sequences have been stored. We classified them using Mothur using the taxonomy of the SILVA database (utilizing the nomenclatures of UniEuk [1]) and SILVA's SSU Ref NR 99 (strong quality filtering, high minimum sequence length) database as reference, which resulted in 1805 distinct prokaryotes classified (92.2% classification ratio), which is spread across the 2627 markers on the globe. In addition to the geographical markers, we created skin-surface maps of the human body for both genders, using raw genomic data from Bouslimani et al. [2] consisting of *circa* 400 markers spread across the skin surface, which again was classified with Mothur yielding 851 distinct prokaryotic organisms. We store everything in an integrated data warehouse running behind the BioAtlas web servers.

2 Methods

In the following, we describe step by step how BioAtlas may be utilized to facilitate map-supported browsing of public 16S rRNA sequence data, and how to analyze user-provided sequences without requiring manual mapping to taxonomies and external databases. We concentrate on three major features of BioAtlas. These include: (1) Browsing the geographical map of the world

(Subheading 2.1), using our data warehouse knowledge base, consisting of observed locations of prokaryota based classification analysis using the markers mined from GOLD and classified with Mothur. (2) Browsing surface-specific distributions (Subheading 2.2) contributed by the user. Here we present a generic example for this kind of analysis, based on microbial observations extracted from the skin of a human female taken from Bouslimani et al. (3) Classifying strains of 16s ribosomal RNA (rRNA) by online mapping to our knowledge base, and subsequent display and filtering both the map of the world and generic surface-specific maps using the mapping results (Subheading 2.3). Here we use 2000 sequences of 16s rRNAs of actinobacteria randomly selected as online example dataset (collected from GG).

For each of the below protocols, we assume the front page of BioAtlas to be the starting point: https://bioatlas.compbio.sdu.dk/. The only prerequisite before starting is a web browser and internet connection. For a guided video and sound tour of the BioAtlas website, the user may watch the screencast video on the front page.

2.1 Browsing Geospatial Distributions of Prokaryota

On the world map page, one is presented with two interactable features of the world map. (1) The actual geographical version of the globe presented using the GoogleMaps API, with markers corresponding to locations of prokaryota sampling sites (*see* Step 2), referred to as the "map." (2) The taxonomic phylogenetic tree to the left allowing for filtering of prokaryota of interest within the map (*see* Step 8).

The map features markers for locations mapped to prokaryotes based on samples from GOLD. The locations have been extracted by either using geographic coordinate data (longitude/latitude) or text-based using GoogleMaps API's geocoding. We here present a demonstration for finding bacteria near Birmingham University (*see* Step 15) (Fig. 1).

1. To get started, click the "World Map" button in the main menu bar to go to the World map page. On the front page, it is also featured as "World map" under the "Get started" paragraph.

2. Navigating the world map and exploring markers

3. The map displays all markers currently filtered by the phylogenetic tree (*see* Point 8).
4. The world map is interactable using the mouse. Click anywhere and drag to move around.
5. The red markers on the map correspond to known locations of prokaryotic findings based on GOLD biosamples.

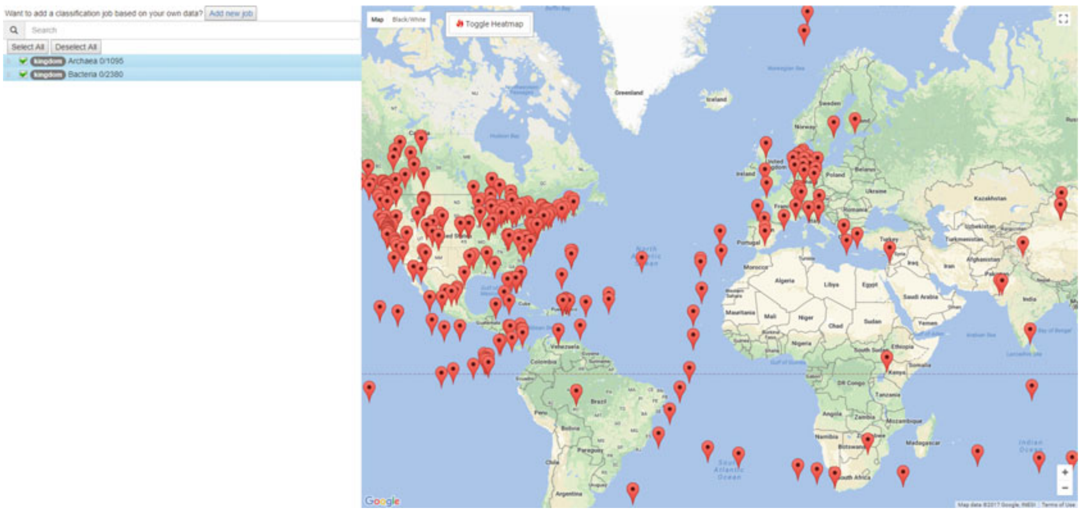
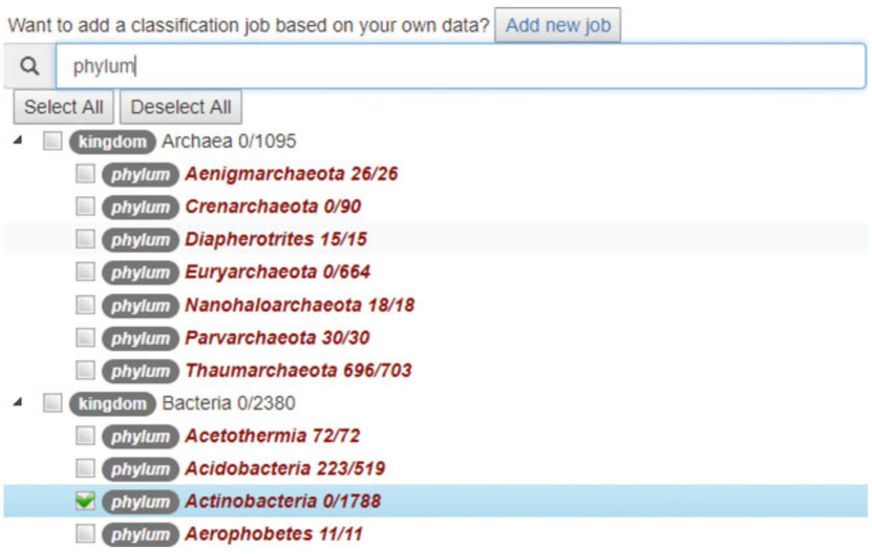


Fig. 1 Graphical overview for the geospatial world map page. Red markers: sample locations from genomics and metagenomics sequencing projects

6. Clicking a marker would bring up an overlay page with metadata and external resources (*see* **Note** 1).
7. The world map also features a heat map of the representation of prokaryotes found for each marker, with the heat color being relative to the ratio of prokaryotes found for each respective marker (*see* **Note** 2).

8. Filtering the markers

9. To the left of the world map, a phylogenetic tree is presented, with identical taxonomic nomenclature as presented by the SILVA reference database (*see* **Note** 3).
10. The tree consists of nodes, each representing a level of the tree of life.
11. It is possible to expand and compress each branch by clicking on the arrows next to it.
12. Clicking the text, or on the checkbox next to each node, allows selecting and deselecting whole branches (*see* **Note** 4).
13. This affects the markers shown on the world map to the right correspondingly, allowing one to focus on prokaryotes of interest.
14. In addition to manual filtering, it is also possible to search for a lineage of interest for fast selection and deselection of nodes, using the text input above the tree.


Want to add a classification job based on your own data? Add new job

phylum

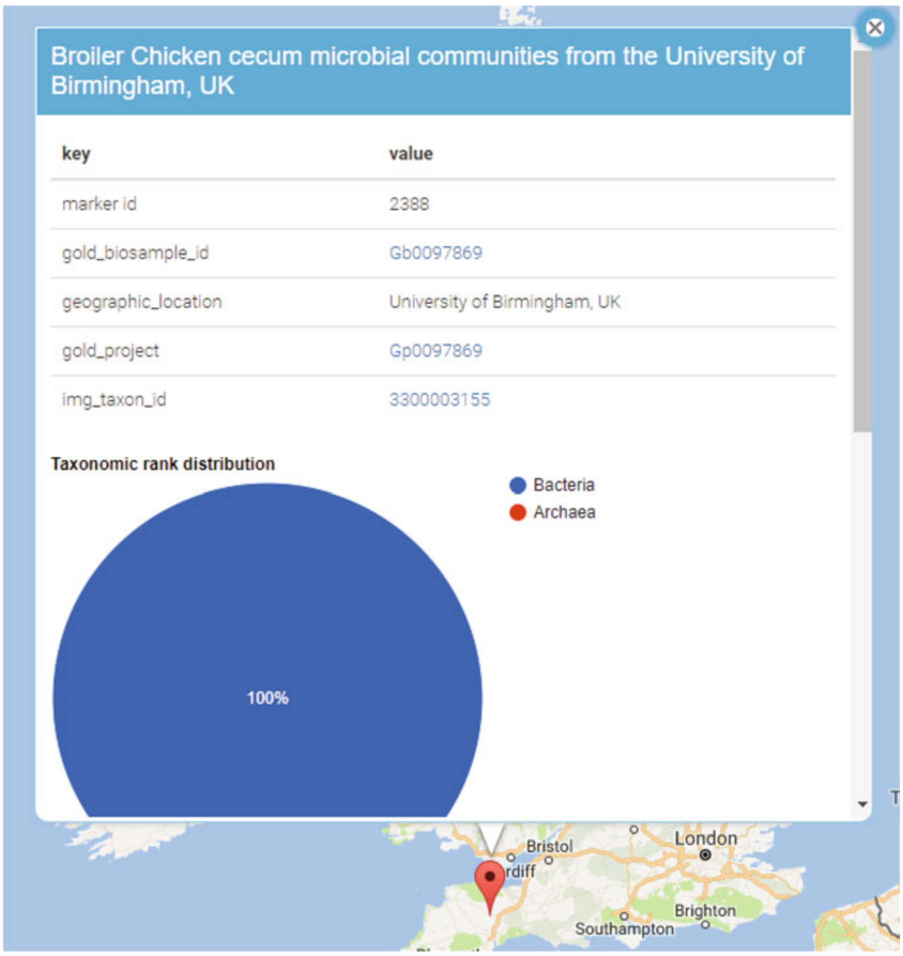
Select All | Deselect All

- kingdom Archaea 0/1095
  - phylum Aenigmarchaeota 26/26
  - phylum Crenarchaeota 0/90
  - phylum Diapherotrites 15/15
  - phylum Euryarchaeota 0/664
  - phylum Nanohaloarchaeota 18/18
  - phylum Parvarchaeota 30/30
  - phylum Thaumarchaeota 696/703
- kingdom Bacteria 0/2380
  - phylum Acetothermia 72/72
  - phylum Acidobacteria 223/519
  - ✓ phylum Actinobacteria 0/1788
  - phylum Aerophobetes 11/11

Fig. 2 Searching the taxonomic tree for filtering, with nodes of phylum rank shown

15. Working example: Exploring the world, looking for Actinobacteria (Phylum rank: Actinobacteria) near Birmingham University

16. Start by clicking the “Deselect All” button to the left, just above the taxonomic tree.
17. Filter the taxonomic tree to the left by searching for “phylum” looking for all nodes of that rank.
18. Next click the “Actinobacteria” node to select all nodes within this branch (Fig. 2) (*see* **Note** 5).
19. Zoom closer into United Kingdoms. There should be two markers present.
20. Click the Southernmost red marker near the Birmingham University (Fig. 3).
21. We see the distributions of bacteria vs archaea and the external links for this location mark. The project title: “Broiler Chicken cecum microbial communities from the University of Birmingham, UK,” suggests that this marker represents a study concerning microbial sequences were taken from the Broiler chicken cecum, of which the external links reveal to be true.
22. Click “Toggle Heatmap” button in the upper left corner and set the background to the black and white setting (Fig. 4, also *see* **Note** 6) to see a heat map representation of the microbial distribution density.



Broiler Chicken cecum microbial communities from the University of Birmingham, UK

key	value
marker id	2388
gold_biosample_id	Gb0097869
geographic_location	University of Birmingham, UK
gold_project	Gp0097869
img_taxon_id	3300003155

Taxonomic rank distribution

Fig. 3 Additional information on a specific, selected sample can be displayed by clicking on it in the map

2.2 Browsing Surface-Specific Distributions

The surface-specific maps are contributed by users together with corresponding sequences and markers, with additional information such as article references, links, and contributors (Fig. 5).

The maps behave like the world map, but with cartographical backgrounds chosen by the creator based on the surface, it represents, e.g. the skin of the human body. The markers of the surface-specific maps are organized in the same way as the markers from the geospatial map but only contain prokaryotic discovery information. The taxonomic tree is used for filtering the respective markers in the surface-specific map (*see* Subheading 2.1, Point 8).

The markers are colored based on the relative hit abundance, gradienting from blue to white. The bluer a marker is, the higher the number of hits in the phylogenetic tree. In addition to the geospatial page setup, the surface-specific maps also contain a listing of all markers, with names as described by the creator (listed to the right of the map).

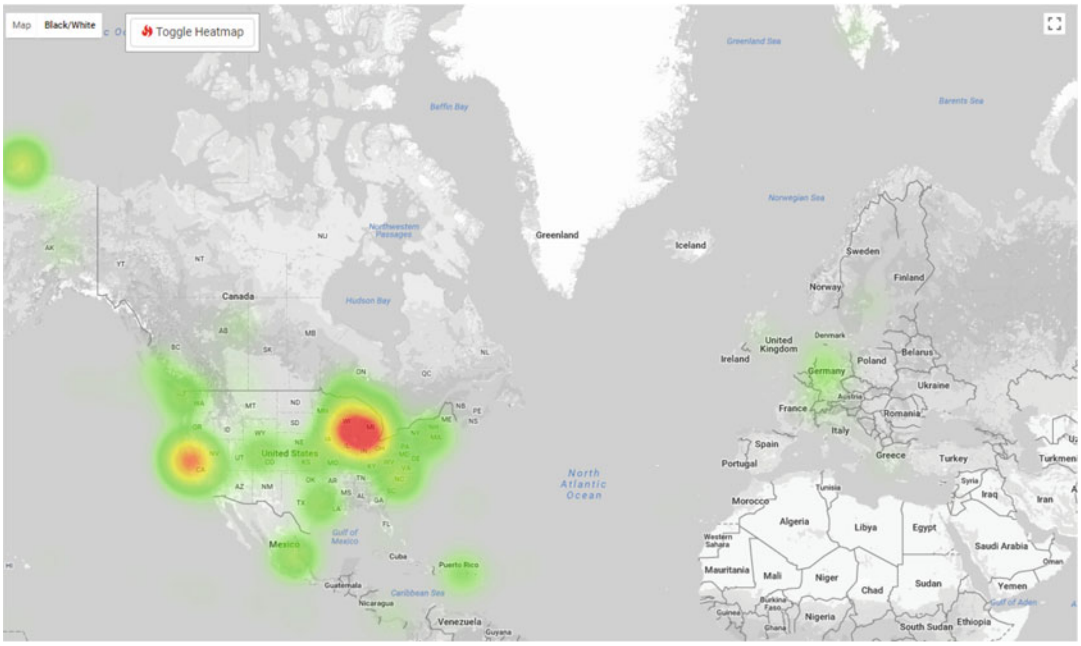
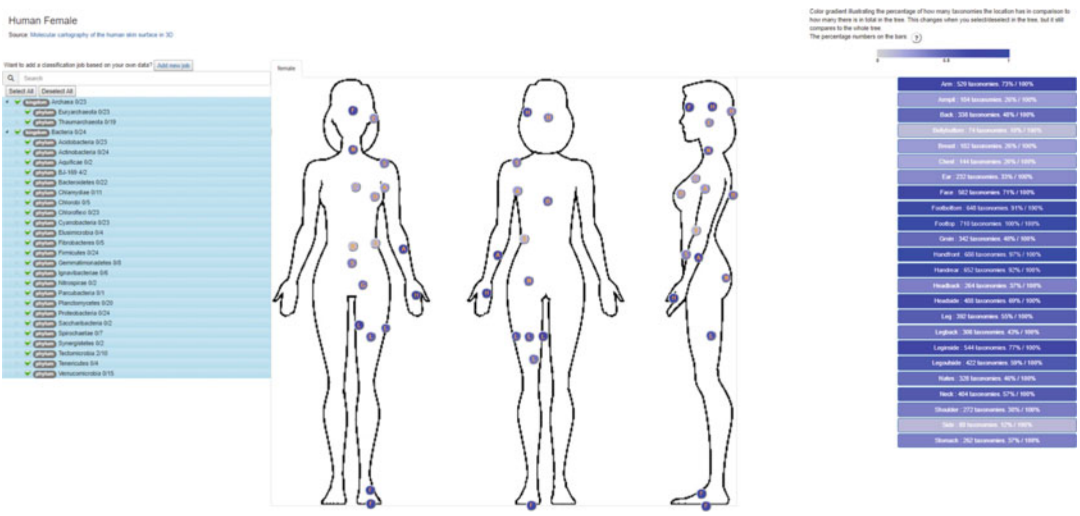


Fig. 4 Black and white representation of the world map with heatmap overlay of microbial distribution densities


Human Female

Source: Molecular cartography of the human skin surface in 3D

Want to add a classification job based on your own data? Add new job

Search

Select All Deselect All

kingdom Archaea 0/23
phylum Euryarchaeota 0/23
phylum Thaumarchaeota 0/19
kingdom Bacteria 0/24
phylum Acidobacteria 0/23
phylum Actinobacteria 0/24
phylum Aquificae 0/2
phylum BJ-169 4/2
phylum Bacteroidetes 0/22
phylum Chlamydiae 0/11
phylum Chlorobi 0/5
phylum Chloroflexi 0/23
phylum Cyanobacteria 0/23
phylum Elusimicrobia 0/4
phylum Fibrobacteres 0/5
phylum Firmicutes 0/24
phylum Gemmatimonadetes 0/8
phylum Ignavibacteriae 0/6
phylum Nitrospirae 0/2
phylum Parcubacteria 0/1
phylum Planctomycetes 0/20
phylum Proteobacteria 0/24
phylum Saccharibacteria 0/2
phylum Spirochaetae 0/7
phylum Synergistetes 0/2
phylum Tectomicrobia 2/16
phylum Tenericutes 0/4
phylum Verrucomicrobia 0/15

Female

Color gradient illustrating the percentage of how many taxonomies the location has in comparison to how many there is in total in the tree. This changes when you select/deselect in the tree, but it still compares to the whole tree.
The percentage numbers on the bars ?

Arm : 529 taxonomies 73% / 100%
Armpit : 104 taxonomies 20% / 100%
Back : 338 taxonomies 48% / 100%
Bellybutton : 74 taxonomies 10% / 100%
Breast : 182 taxonomies 26% / 100%
Chest : 144 taxonomies 20% / 100%
Ear : 232 taxonomies 33% / 100%
Face : 502 taxonomies 71% / 100%
Footbottom : 640 taxonomies 91% / 100%
Foottop : 710 taxonomies 100% / 100%
Groin : 342 taxonomies 48% / 100%
Handfront : 688 taxonomies 97% / 100%
Handrear : 652 taxonomies 92% / 100%
Headback : 264 taxonomies 37% / 100%
Headside : 488 taxonomies 69% / 100%
Leg : 392 taxonomies 55% / 100%
Legback : 308 taxonomies 43% / 100%
Leginside : 544 taxonomies 77% / 100%
Legoutside : 422 taxonomies 59% / 100%
Nares : 328 taxonomies 46% / 100%
Neck : 404 taxonomies 57% / 100%
Shoulder : 272 taxonomies 38% / 100%
Side : 83 taxonomies 12% / 100%
Stomach : 262 taxonomies 37% / 100%

Fig. 5 Graphical overview of the surface-specific user maps of the skin surface of a human female

We will not cover the surface-map creation process here but only demonstrate how to use the tool (*see* Step 3)—a full documentation is available online (https://bioatlas.compbio.sdu.dk/tutorial).

1. To get started, click the “Browse maps” button in the main menu bar to go to the custom map selection page (i.e., surface-specific). On the front page, it is also featured as “Explore usermaps,” under the “Get started” paragraph.

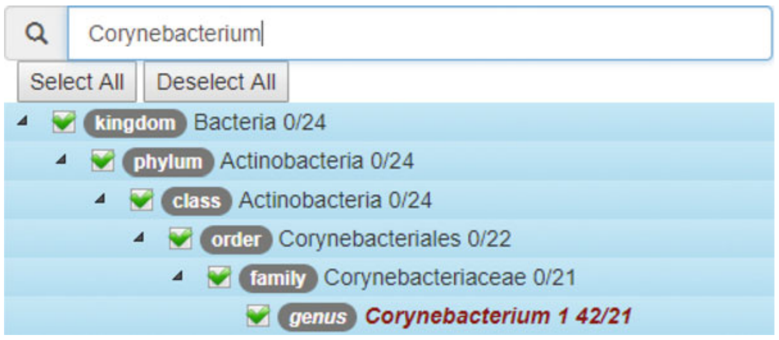

Corynebacterium
Select All
Deselect All
kingdom Bacteria 0/24
phylum Actinobacteria 0/24
class Actinobacteria 0/24
order Corynebacteriales 0/22
family Corynebacteriaceae 0/21
genus Corynebacterium 1 42/21

Fig. 6 Searching for "Corynebacterium" (genus rank) in the taxonomic tree

2. At the "Browse maps" page select any map. In order to open it click on the name or image.

3. Working example: Exploring the skin of a human female to investigate the Corynebacterium (Genus rank: Corynebacterium) distribution

4. While on the "Browse maps" page scroll down to the map named "Human Female," and click on it.
5. The map is based on the study, "Molecular cartography of the human skin surface in 3D" by Bouslimani et al. as indicated by the link in the upper left corner of your browser.
6. Click the "Deselect All" button to the left just above the taxonomic tree.
7. Filter the taxonomic tree by searching for "Corynebacterium" and click the node at the genus level (Fig. 6).
8. After filtering, the markers have turned white, but some labels are still colored in orange—meaning they have been hit (Fig. 7) (*see* **Note** 7).
9. Click on the marker for the back of the female.
10. A popup window appears and shows which prokaryotes are found at the position of this marker (Fig. 8). In our case, it is only Corynebacteria.

2.3 Classifying 16s Ribosomal RNA Sequences (with Geospatial and Surface-Specific Mapped Distributions)

The user may upload 16s ribosomal RNA sequences as FASTA formatted flat file, which will be piped into the classification procedure by BioAtlas and mapped to all surface-specific maps as well as the geospatial world map (Fig. 9). For demonstration purposes, we offer such a file with demo sequences online at the BioAtlas web site. The classifier framework utilizes a scheduler, allowing multiple users to interact with the system in parallel. Running these "jobs" normally takes some minutes.

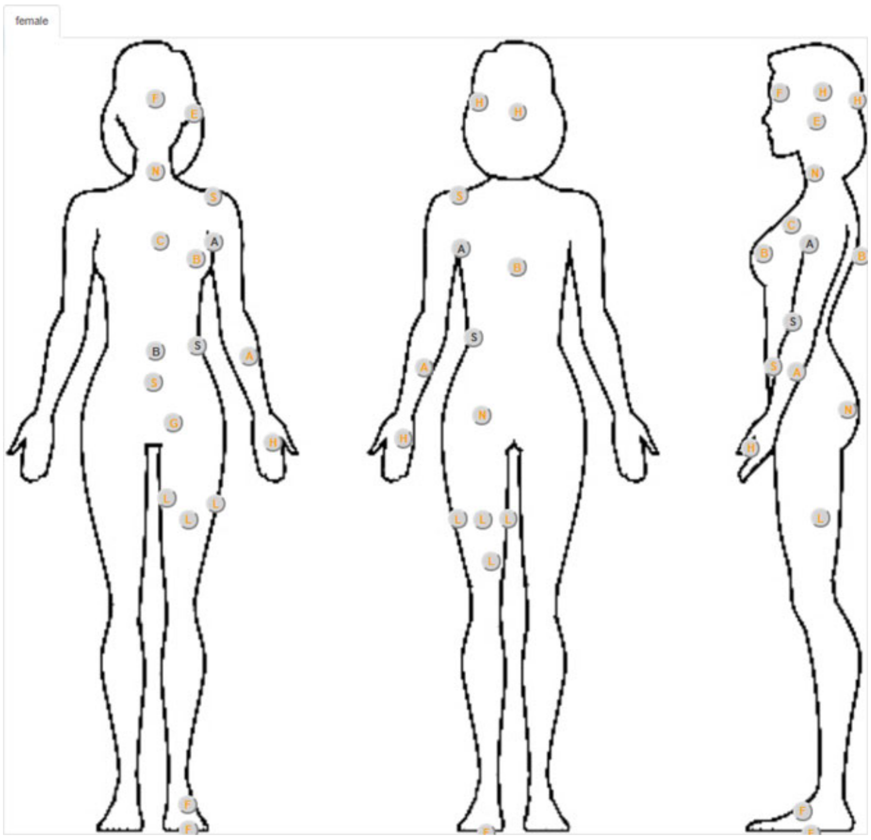


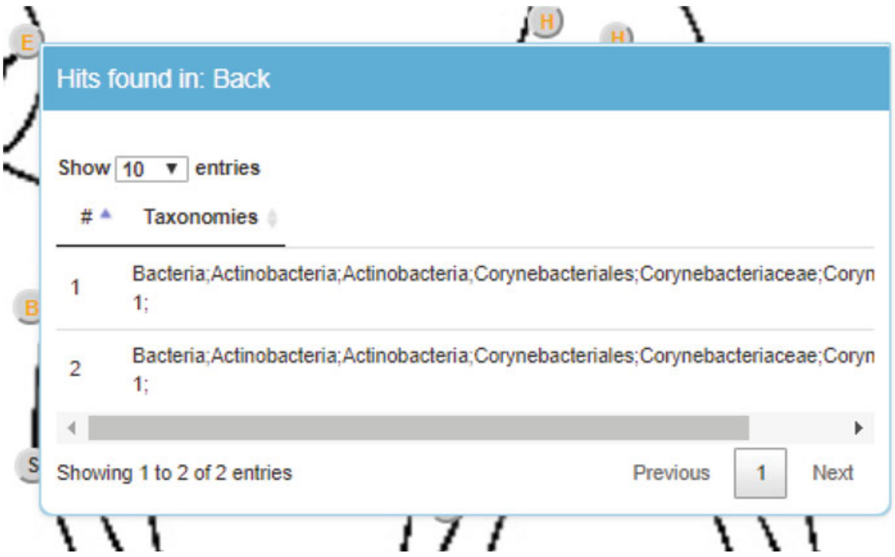
Fig. 7 The "Human Female" map including markers. Orange labels: Markers with hits. Black labels: Markers without hits

We here guide through this process using the demonstration dataset consisting of 2000 actinobacterial sequences extracted from GG as an example (Step 2).

1. To get started click "Dashboard," and find the "Add new job" button at the left (Fig. 9). At the front page it is also featured as "Upload your own data" under the "Get started" paragraph.

2. Working example: Classification and mapping of the 2000 Actinobacteria 16s rRNA sequences (Phylum rank: Actinobacteria), extracted from GG

3. On the classification setup page (Fig. 9) click the "Use test dataset 1 (Actinobacteria)" radio button (*see* **Note** 8).
4. Click the "Submit job" button at the bottom.
5. After being directed to the dashboard, the scheduler should state that it is running the job (Fig. 10).
6. When done, click the name (Fig. 11).



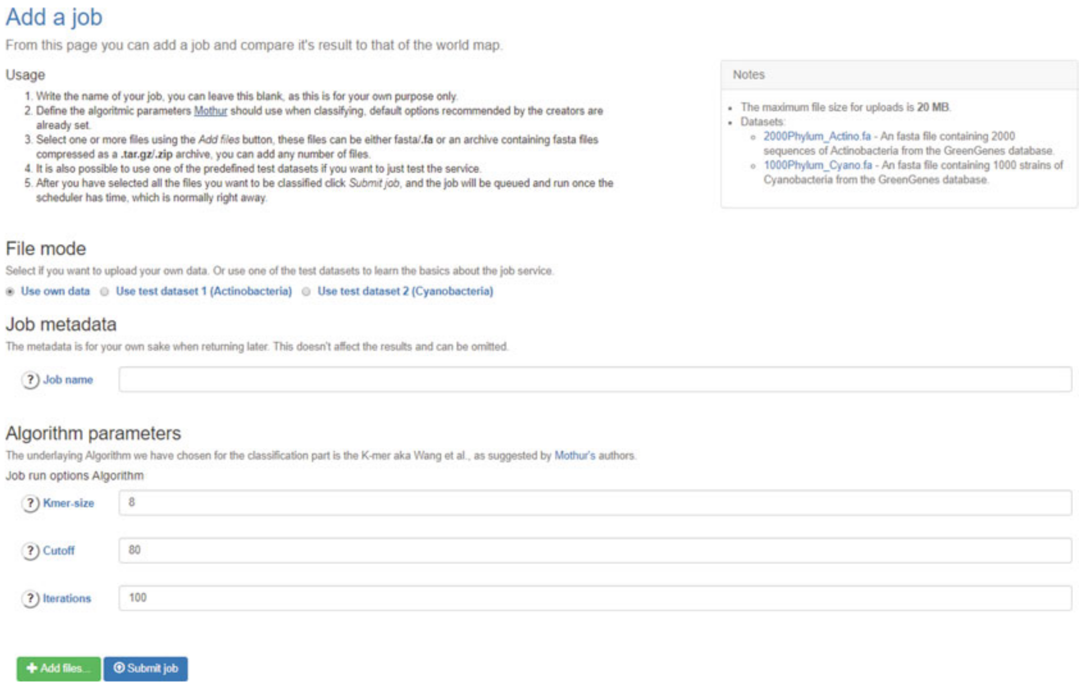
Hits found in: Back

Show 10 entries

#	Taxonomies
1	Bacteria;Actinobacteria;Actinobacteria;Corynebacteriales;Corynebacteriaceae;Coryn 1;
2	Bacteria;Actinobacteria;Actinobacteria;Corynebacteriales;Corynebacteriaceae;Coryn 1;

Showing 1 to 2 of 2 entries

Fig. 8 Popup window after clicking on the marker for the back part of the female skin surface map. Only species of the genus Corynebacteria hit there



Add a job

From this page you can add a job and compare it's result to that of the world map.

Usage

1. Write the name of your job, you can leave this blank, as this is for your own purpose only.
2. Define the algoritmic parameters Mothur should use when classifying, default options recommended by the creators are already set.
3. Select one or more files using the *Add files* button, these files can be either fasta/**.fa** or an archive containing fasta files compressed as a **.tar.gz**/**.zip** archive, you can add any number of files.
4. It is also possible to use one of the predefined test datasets if you want to just test the service.
5. After you have selected all the files you want to be classified click *Submit job*, and the job will be queued and run once the scheduler has time, which is normally right away.

Notes

- The maximum file size for uploads is **20 MB**.
- Datasets:
  - 2000Phylum_Actino.fa - An fasta file containing 2000 sequences of Actinobacteria from the GreenGenes database.
  - 1000Phylum_Cyano.fa - An fasta file containing 1000 strains of Cyanobacteria from the GreenGenes database.

File mode

Select if you want to upload your own data. Or use one of the test datasets to learn the basics about the job service.

[✓ Use own data] Use test dataset 1 (Actinobacteria) Use test dataset 2 (Cyanobacteria)

Job metadata

The metadata is for your own sake when returning later. This doesn't affect the results and can be omitted.

Job name:

Algorithm parameters

The underlaying Algorithm we have chosen for the classification part is the K-mer aka Wang et al., as suggested by Mothur's authors.

Job run options Algorithm

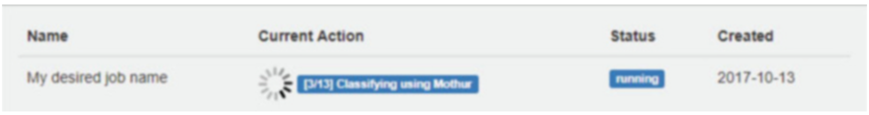
Kmer-size: 8

Cutoff: 80

Iterations: 100

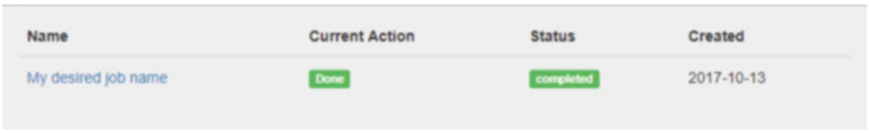
Fig. 9 Overview of the classification setup page

7. On the result page, the results of several analyses are shown: The Mothur classification results (Fig. 12) as well as Kingdom, species and taxonomic rank distributions (Figs. 13, 14, 15).
8. In addition to the classification and distributions, the geospatial and the surface-specific maps are available, prefiltered based on the classification results (*see* **Note 9**).



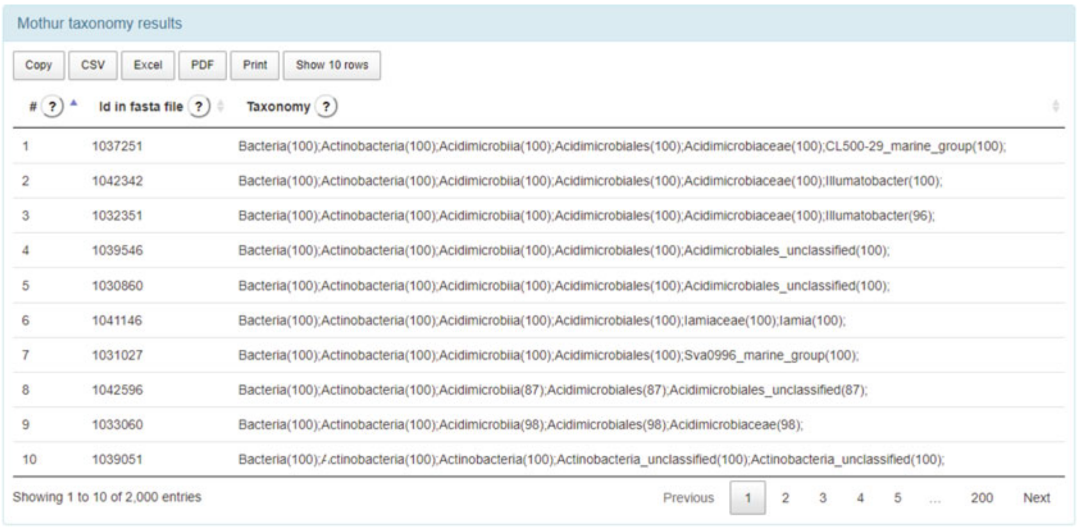
Name	Current Action	Status	Created
My desired job name	[3/13] Classifying using Mothur	running	2017-10-13

Fig. 10 Scheduler currently running the classification job



Name	Current Action	Status	Created
My desired job name	Done	completed	2017-10-13

Fig. 11 Classification job done



Mothur taxonomy results

#	Id in fasta file	Taxonomy
1	1037251	Bacteria(100);Actinobacteria(100);Acidimicrobiia(100);Acidimicrobiales(100);Acidimicrobiaceae(100);CL500-29_marine_group(100);
2	1042342	Bacteria(100);Actinobacteria(100);Acidimicrobiia(100);Acidimicrobiales(100);Acidimicrobiaceae(100);Illumatobacter(100);
3	1032351	Bacteria(100);Actinobacteria(100);Acidimicrobiia(100);Acidimicrobiales(100);Acidimicrobiaceae(100);Illumatobacter(96);
4	1039546	Bacteria(100);Actinobacteria(100);Acidimicrobiia(100);Acidimicrobiales(100);Acidimicrobiales_unclassified(100);
5	1030860	Bacteria(100);Actinobacteria(100);Acidimicrobiia(100);Acidimicrobiales(100);Acidimicrobiales_unclassified(100);
6	1041146	Bacteria(100);Actinobacteria(100);Acidimicrobiia(100);Acidimicrobiales(100);Iamiaceae(100);Iamia(100);
7	1031027	Bacteria(100);Actinobacteria(100);Acidimicrobiia(100);Acidimicrobiales(100);Sva0996_marine_group(100);
8	1042596	Bacteria(100);Actinobacteria(100);Acidimicrobiia(87);Acidimicrobiales(87);Acidimicrobiales_unclassified(87);
9	1033060	Bacteria(100);Actinobacteria(100);Acidimicrobiia(98);Acidimicrobiales(98);Acidimicrobiaceae(98);
10	1039051	Bacteria(100);Actinobacteria(100);Actinobacteria(100);Actinobacteria_unclassified(100);Actinobacteria_unclassified(100);

Showing 1 to 10 of 2,000 entries

Fig. 12 Classification results from Mothur represented as a data table

9. At the top of the results page click "Geo-map."
10. A version of the geospatial map with taxonomic tree and markers filtered by the results will appear (*see* **Note** 10).

3 Notes

1. A map marker holds information such as:
   - `gold_biosample_id` (GOLD biosample id): additional information regarding the sample place, date of sampling, and sample technology—among other things.
   - `gold_project` (GOLD project): The GOLD project for which the sample was taken. Note that one GOLD project might hold a collection of one or more samples.
   - `img_taxon_id` (IMG taxon id): Additional metadata, both manually and automatically curated. BioAtlas also provides links to RAW data stored at NCBI.

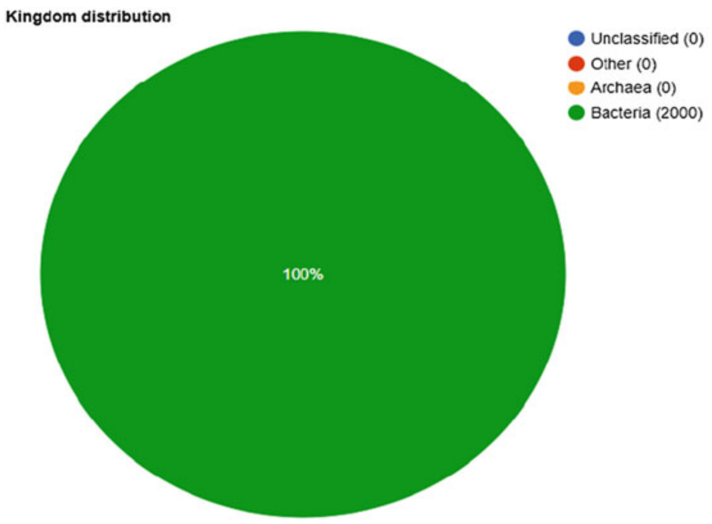


Fig. 13 Kingdom distributions

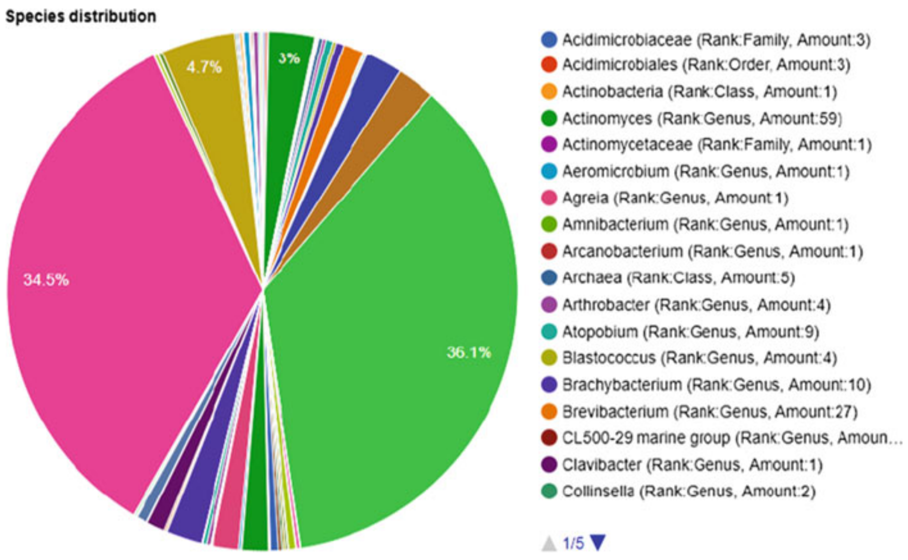


Fig. 14 Species distributions

- Taxonomic rank distribution: The "active hits" for a specific location refers to the number/ratio of hits among the selected/filtered prokaryotes in the phylogenetic tree mapping to that location. It is adjustable in font size by the buttons next to the text. Several additional metadata types such as Taxonomic rank distribution for bacteria vs. archaea, longitude, latitude, and country can be displayed as well.

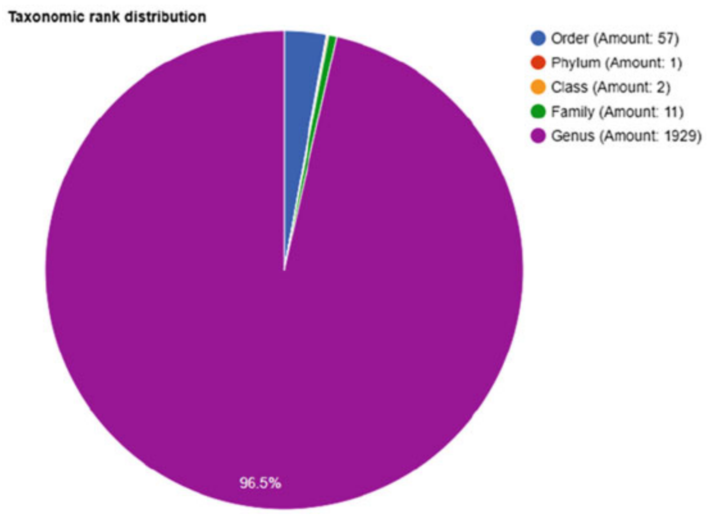


Fig. 15 Taxonomic rank distributions

2. This allows for comparing the relative abundance of prokaryotes between the location markers. The button in the upper left corner allows for switching between black/white and colored background.
3. The tree resembles a phylogenetic tree of life and is based on our integrated knowledge base for all prokaryotes found in all locations.
4. Thus, if one deselects a node all children will also be deselected, and vice versa.
5. The markers on the world map are filtered based on whether or not they map bacteria from the lineage of Actinobacteria.
6. The overall representation of heat is clearest in the United States of America, as most markers refer to sampling locations US.
7. Orange coloring of the label characters of the markers means that the marker has been hit during mapping—in contrast to black markers, which mean no hit.
8. Fill out job name if desired. Use “Kmer-size” 8, “Cutoff” 80 and “Iterations” 100 (recommended settings).
9. Filtering is done by mapping the Mothur classification results to the tree of life. If the prokaryotes from the mapping results are not found within a specific, selected maps, they are discarded in the visualization of the respective map.
10. The same holds for surface-specific maps by clicking on “User-Map.”

Acknowledgements

JBL is grateful for financial support from his VILLUM Young Investigator Grant. Funding for open access charge: VILLUM Young Investigator Grant of Jan Baumbach (Young Investigator Grant nr. 13154).

JBL is grateful for financial support from his Velux Foundation research grant (Research grant nr. 000121540), supporting his Ph.D. project.

Conflict of Interest Statement None declared.

References

1. Berney C, Ciuprina A, Bender S, Brodie J, Edgcomb V, Kim E, Rajan J, Parfrey LW, Adl S, Audic S et al (2017) Unieuk: time to speak a common language in protistology! J Eukaryot Microbiol 64(3):407–411
2. Bouslimani A, Porto C, Rath CM, Wang M, Guo Y, Gonzalez A, Berg-Lyon D, Ackermann G, Christensen GJM, Nakatsuji T et al (2015) Molecular cartography of the human skin surface in 3d. Proc Natl Acad Sci 112(17):E2120–E2129
3. Caporaso JG, Kuczynski J, Stombaugh J, Bittinger K, Bushman FD, Costello EK, Fierer N, Peña AG, Goodrich JK, Gordon JI et al (2010) Qiime allows analysis of high-throughput community sequencing data. Nat Methods 7(5):335–336
4. Chaudhary N, Sharma AK, Agarwal P, Gupta A, Sharma VK (2015) 16s classifier: a tool for fast and accurate taxonomic classification of 16s rRNA hypervariable regions in metagenomic datasets. PLoS One 10(2):e0116106
5. Cole JR, Wang Q, Fish JA, Chai B, McGarrell DM, Sun Y, Brown CT, Porras-Alfaro A, Kuske CR, Tiedje JM (2013) Ribosomal database project: data and tools for high throughput rRNA analysis. Nucleic Acids Res 42(D1):D633–D642
6. de Queiroz K (1997) The linnaean hierarchy and the evolutionization of taxonomy, with emphasis on the problem of nomenclature. Aliso J Syst Evol Bot 15(2):125–144
7. DeSantis TZ, Hugenholtz P, Larsen N, Rojas M, Brodie EL, Keller K, Huber T, Dalevi D, Hu P, Andersen GL (2006) Greengenes, a chimera-checked 16s rRNA gene database and workbench compatible with ARB. Appl Environ Microbiol 72(7):5069–5072
8. Giovannoni SJ, Britschgi TB, Moyer CL, Field KG (1990) Genetic diversity in sargasso sea bacterioplankton. Nature 345(6270):60–63
9. Lund JB, List M, Baumbach J (2017) Interactive microbial distribution analysis using bioatlas. Nucleic Acids Res. https://doi.org/10.1093/nar/gkx304
10. Markowitz VM, Chen IMA, Palaniappan K, Chu K, Szeto E, Grechkin Y, Ratner A, Jacob B, Huang J, Williams P et al (2011) Img: the integrated microbial genomes database and comparative analysis system. Nucleic Acids Res 40(D1):D115–D122
11. Mitchell A, Bucchini F, Cochrane G, Denise H, Hoopen Pt, Fraser M, Pesseat S, Potter S, Scheremetjew M, Sterk P et al (2015) Ebi metagenomics in 2016-an expanding and evolving resource for the analysis and archiving of metagenomic data. Nucleic Acids Res 44(D1):D595–D603
12. Mizrahi-Man O, Davenport ER, Gilad Y (2013) Taxonomic classification of bacterial 16s rRNA genes using short sequencing reads: evaluation of effective study designs. PloS One 8(1):e53608
13. Pagani I, Liolios K, Jansson J, Chen IMA, Smirnova T, Nosrat B, Markowitz VM, Kyrpides NC (2011) The genomes online database (gold) v. 4: status of genomic and metagenomic projects and their associated metadata. Nucleic Acids Res 40(D1):D571–D579

14. Quast C, Pruesse E, Yilmaz P, Gerken J, Schweer T, Yarza P, Peplies J, Glöckner FO (2012) The silva ribosomal rna gene database project: improved data processing and web-based tools. Nucleic Acids Res 41(D1):D590–D596
15. Schloss PD, Westcott SL, Ryabin T, Hall JR, Hartmann M, Hollister EB, Lesniewski RA, Oakley BB, Parks DH, Robinson CJ et al (2009) Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. Appl Environ Microbiol 75(23):7537–7541
16. Yarza P, Yilmaz P, Pruesse E, Glöckner FO, Ludwig W, Schleifer KH, Whitman WB, Euzéby J, Amann R, Rosselló-Móra R (2014) Uniting the classification of cultured and uncultured bacteria and archaea using 16s rrna gene sequences. Nat Rev Microbiol 12(9):635

Chapter 4

Generative Models for Quantification of DNA Modifications

Tarmo Äijö, Richard Bonneau, and Harri Lähdesmäki

Abstract

There are multiple chemical modifications of cytosine that are important to the regulation and ultimately the functional expression of the genome. To date no single experiment can capture these separate modifications, and integrative experimental designs are needed to fully characterize cytosine methylation and chemical modification. This chapter describes a generative probabilistic model, *Lux*, for integrative analysis of cytosine methylation and its oxidized variants. *Lux* simultaneously analyzes partially orthogonal bisulfite sequencing data sets to estimate proportions of different cytosine methylation modifications and estimate multiple cytosine modifications for a single sample by integrating across experimental designs composed of multiple parallel destructive genomic measurements. *Lux* also considers the variation in measurements introduced by different imperfect experimental steps; the experimental variation can be quantified by using appropriate spike-in controls, allowing *Lux* to deconvolve the measurements and recover accurately the underlying signal.

Key words DNA methylation, Bayesian analysis, Hierarchical generative modeling, 5-methylcytosine oxidation, Bisulfite sequencing, BS-seq/oxBS-seq/TAB-seq/fCAB-seq/CAB-seq/redBS-seq/MAB-seq

1 Introduction

Epigenetic modification of cytosine (C), 5-methylcytosine (5mC), and its oxidation products 5-hydroxymethylcytosine (5hmC), 5-formylcytosine (5fC), and 5-carboxylcytosine (5caC) play a role in regulation and organization of the genome (and ultimately health) through many pathways [1–3]. Epigenetic modification of cytosine governs active demethylation via thymine DNA glycosylase-dependent base excision repair [4], and these modifications can serve as distinct epigenetic markers recognized by marker-specific regulatory interactions (e.g., many transcription factors have methylation responsive binding sites) [5, 6]. In order to study these different pathways and their role in health, an accurate and comprehensive estimation of methylation levels is required. Here, we describe a generative probabilistic model motivated by experimental protocols and its use for analyzing bisulfite-based sequencing data [7, 8]. The explicit modeling of experimental variation together with proper spike-in control allows accurate estimation of parameters and accompanied uncertainties through Bayesian analysis [7, 8]. The explicit probabilistic model also provides clear routes for integration of other data types and for correction of other aspects of experimental design as future work.

We describe the usual analysis workflow of BS-seq and oxBS-seq data using Lux, containing all the analysis steps starting from raw sequencing reads ending to methylation estimates and calling of differentially methylated cytosines. Our method is composed of the following steps: *(i)* quality control of raw bisulfite sequencing reads, *(ii)* mapping of bisulfite reads and extraction of bisulfite conversion frequencies, *(iii)* preparation of inputs for Lux, *(iv)* running Lux analysis, and *(v)* inspection of methylation level estimates and detection of differentially methylated cytosines. In addition, we present our recommended quality control procedures along the description of the analysis workflow.

2 Materials

2.1 Bisulfite Sequencing

Bisulfite sequencing (BS-seq) and its variants have been widely used to probe cytosine methylation [9]. For instance, BS-seq distinguishes 5mC and 5hmC from C, 5fC, and 5caC through selective deamination of C, 5fC, and 5caC to produce uracil by sodium bisulfite, whereas 5mC and 5hmC are protected from the conversion [10]. Different variations of BS-seq protocol change the conversion landscape for differential separation of cytosine modifications [11–16], such as oxidative bisulfite sequencing (oxBS-seq) that separates 5hmC from 5mC by oxidization of 5hmC to 5fC prior to sodium bisulfite treatment [11]. Importantly, each sample has to be assayed with multiple experimental protocols (BS-seq, oxBS-seq, etc.) for comprehensive quantification of cytosine methylation modifications. Finally, these different variants of BS-seq involve sensitive and imperfect chemical reactions, potentially leading to experiment-specific biases.

2.2 A Generative Probabilistic Model of Bisulfite Sequencing-Based Assays

To define the effects of experimental steps of bisulfite-based protocols on outcomes, we utilize tree diagrams whose structures represent the sequential steps of the experimental protocols. The layers in the model represent the compounded effect of the sequential steps on the signal. To illustrate this in practice, we consider the oxBS-seq protocol in Fig. 1. Briefly, the considered chemical reactions are the events that lead to the branching of the

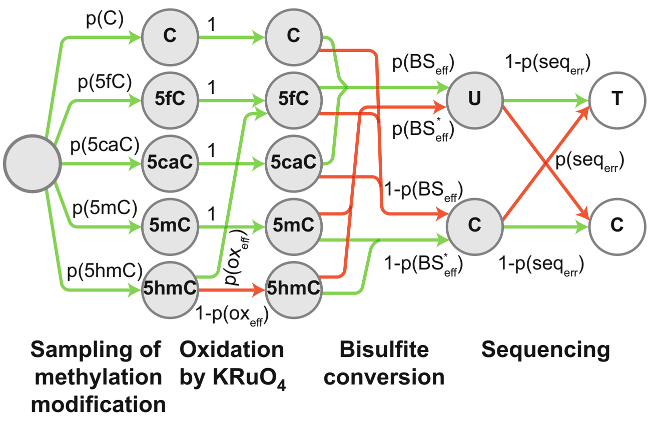


Fig. 1 The different steps of the oxBS-seq protocol are illustrated. The possible transitions are stated in the terms of the experimental parameters; for instance, $p(ox_{eff})$ is the probability that a given 5hmC is oxidized to 5fC by $KRuO_4$. Transitions representing successful and unsuccessful chemical reactions are colored with green and red, respectively

tree representing the possible outcomes, leading finally to the leaf nodes, which are either C (not converted) or T (converted) outcomes. For instance, the parameter $p(ox_{eff})$ defines the probability of successful oxidation of 5hmC to 5fC, and the probability of the other possible outcome, unsuccessful oxidation, is $1\text{-}p(ox_{eff})$. Notably, the oxBS-seq diagram reverts to the BS-seq diagram when the oxidation step is discarded ($p(ox_{eff}) = 0$).

The probabilities p(C), p(5mC), p(5hmC), p(5fC), and p(5caC) represent the sample-specific methylation distribution per cytosine ($p(C) + p(5mC) + p(5hmC) + p(5fC) + p(5caC) = 1$), which in the end we wish to estimate. In the case of perfect experiments, the probability of obtaining C readout, p_c, would be completely defined by the underlying methylation distribution of a given cytosine. However, in the case of imperfect experiments, the success rates of chemical reactions will affect the outcomes; for instance, low bisulfite conversion rate will increase the number of C outcomes (Fig. 1).

Using the law of conditional probability, the probability of a series of events occurring, leading to a given leaf nodes in Fig. 1, is equal to the product of the leaf node probability and its parent nodes' probabilities. For example, consider a series of events where *(i)* a hydroxymethylated cytosine (5hmC) is chosen from a sample, *(ii)* 5hmC is correctly oxidized to 5fC by $KRuO_4$, *(iii)* 5fC is correctly converted to uracil by sodium bisulfite treatment, and *(iv)* the uracil is correctly read as a thymine (T) during sequencing: the probability of such a series of events is

$$p(5hmC)\, p(ox_{eff})\, p(BS_{eff}) \left(1 - p\left(seq_{err}\right)\right).$$

Similarly, consider the series of events as above with an exception that the uracil is incorrectly read as cytosine (C): the probability of such a series of events is

$$p(5hmC)\, p(ox_{eff})\, p(BS_{eff})\, p\left(seq_{err}\right).$$

By applying the law of total probabilities, we can state the probability of different final outcomes (C and T) by collapsing all the series of events that lead to the particular outcome [7]. These total probabilities are used to calculate the likelihood of data under binomial model.

2.3 Likelihood Model for Bisulfite Sequencing Data

Different bisulfite-based sequencing assays reveal only whether a cytosine was converted (T) or not (C) in the experiment (following specific chemical treatments or sequences of treatment specific to a given C-methylation-type specific assay); therefore a single outcome (C or T per cytosine per read) can be modeled as a Bernoulli random variable, $Y \sim \text{Bernoulli}(p_c)$, where p_c is the probability for the C outcome from a given assay. If we assume that outcomes (reads) per cytosine are independent and identically distributed, then the number of C (not converted) outcomes in N outcomes, N_C, is a binomial random variable, $N_C \sim \text{Binomial}(p_c, N)$. If different bisulfite-based methylation assays, such as BS-seq and oxBS-seq, are applied to the same biological sample, then we can safely assume that the underlying methylation levels are the same in these different BS-seq and oxBS-seq experiments. This implies that random variables, N_C, from different assays are conditionally independent given assay-specific parameters p_c and the methylation distribution, p(C), p(5mC), p(5hmC), p(5fC), and p(5caC), thus allowing us to estimate different methylation modifications from a combination of assays.

2.4 Model Inference

Let us first consider the combination of BS-seq and oxBS-seq assays to estimate the levels and locations of C, 5mC, and 5hmC modifications in a single sample. To estimate the values of the sample-specific experimental parameters, $p(ox_{eff})$, $p(BS_{eff})$, $p(BS^{*}_{eff})$, and $p(seq_{err})$, we and others have developed an approach in which DNA controls (sequences different from host genome) with known methylation distribution are spiked into the samples [7, 8, 17]. Next when analyzing the sequencing data, we incorporate strong prior knowledge on the methylation status of the DNA controls to enable calibration of the experimental parameters. Simultaneously, the calibrated experimental parameters enable accurate estimation of the methylation distributions of the wild-type cytosines. In statistical terms, we assign prior distributions for all the parameters and derive the posterior distributions over them conditioned on the data; the posterior distributions are estimated using a Hamiltonian Monte Carlo (HMC) approach with the No-U-Turn Sampler (NUTS) [18–20] as implemented in a Stan program.

3 Methods

3.1 Quality Control of Sequencing Reads

We recommend doing quality control of sequencing reads prior to read mapping using FastQC [21] or similar tools. This quality control step will reveal if there are any overrepresented sequences such as adapters, which should be removed before mapping, or whether base calling error rates are higher than normal, especially toward the end of the sequences, in which case we recommend trimming the raw sequence reads. Notably, thymines are expected to be overrepresented over cytosines due to the bisulfite conversion of unmethylated cytosines.

3.2 Reference Genome Specification and Index Building

The pipeline described here uses Bismark to map bisulfite sequencing reads [22]. The reference genome sequence used in mapping the sequencing reads has to contain the host genome sequence and the template sequences of the incorporated DNA controls. To do this, we incorporate the control sequence templates in the FASTA file containing the host genome. Then, we prepare two bisulfite converted versions of the reference genome in which either C → T or G → A conversions are done and build corresponding Burrows-Wheeler indices for the Bowtie 2 mapping (bismark_genome_preparation) (*see* **Note 1**). Note that this step has to be done only once per host genome and control sequences combination.

3.3 Sequence Read Mapping

The Burrows-Wheeler indices prepared in the previous step are used in the mapping of the bisulfite-based sequence reads. Similar to the reference genome construction, first the reads are fully converted to allow mapping against the reference genomes (C → T and G → A) while keeping track which bases were converted so that afterward we can tell which bases were converted (C → T) in the assay (bismark). It is important to specify in the mapping whether the sequencing libraries are directional or nondirectional. Each sequencing sample will be mapped separately.

3.4 Bisulfite Conversion Frequency Extraction

After mapping the reads, we quantify the frequencies of observing converted and non-converted reads at the level of single cytosine from the Bismark alignment output for each sample. To do this, we use the bismark_methylation_extractor script distributed with the Bismark software. We recommend extracting counts for all the cytosines independent of the context (CpG/CHG/CHH) (--counts --CX) (*see* **Note 2**). We highly recommend using the bedGraph output (--bedGraph), which makes it easy to collect count data over multiple samples.

3.5 Input Preparation for Lux Analysis

Lux requires number of C and total (C + T) readouts per cytosine per sample, which can be parsed easily from the bedGraph output files as the chromosome and chromosomal position information together with the counts are provided (bedtools) [23]. Additionally, prior information on methylation levels, specifically vital for control cytosines, has to be supplied. Altogether, four files should be generated; we will describe the files in detail below:

(a) Count data format

Control and wild-type cytosines data are stored in separate files (e.g., control_data.tsv and data.tsv). Replicate-specific bisulfite-based data are grouped together; for instance, data from BS-seq and oxBS-seq assays per cytosine and replicate are represented by quadruplet of integers (N_C from BS, N from BS, N_C from oxBS, N from oxBS). Each control and wild-type cytosine has to be present in every considered replicate. Data are stored in tab-separated values files where each cytosine (*see* **Note 3**) has its own line; i.e., control_data.tsv and data.tsv have as many lines as there are control and wild-type cytosines, respectively. Each replicate is represented by aforementioned quadruplet; for instance, in the case of four replicates, each row has 16 integers separated by tabs.

(b) Prior data format

The prior information over the control cytosines is supplied through Dirichlet priors in terms of pseudo-counts. Pseudo-counts are defined in tab-separated values file (e.g., control_prior.tsv) in which each control cytosine has its own line; the same prior is assumed for each of the replicate. In the case of BS-seq and oxBS-seq assays, our methylation variables have three dimensions (as C, 5fC, and 5caC cannot be distinguished from each other), and thus we define prior over p(C or 5fC or 5caC), p(5mC), and p(5hmC). That is, each row has three positive numbers that represent pseudo-counts for p(C or 5fC or 5caC), p(5mC), and p(5hmC) in that order. We express our prior knowledge that the C and 5mC controls are stable and relatively pure, whereas 5hmC controls are less pure and exhibit greater variation (Table 1).

Also, wild-type cytosines require prior distribution on methylation levels. The format of the file (e.g., prior.tsv) is identical to the format of the prior file for control cytosines. Each wild-type cytosine requires its own prior definition, but in most cases identical priors are assumed across cytosines, and the same priors are used across replicates. In the case of BS-seq and oxBS-seq assays, we define prior information on p(C or 5fC or 5caC),

Table 1
Pseudo-counts for different methylation control oligonucleotides

	C control	5mC control	5hmC control
Pseudo-counts	$\alpha = [1000, 1, 1]$	$\alpha = [1, 1000, 1]$	$\alpha = [6, 2, 72]$

The listed values are pseudo-count parameters of Dirichlet distribution; the means of the distributions are $\alpha_i/\Sigma\alpha$. The 5hmC control cytosines are assumed to be less pure than the C and 5mC control cytosines

p(5mC), and p(5hmC). Each row has three positive numbers that represent pseudo-counts for p(C or 5fC or 5caC), p(5mC), and p(5hmC) in that order. We use a relatively uninformative prior, that is, ([p(C or 5fC or 5caC), p(5mC), p(5hmC)]~Dirichlet(α), where $\alpha = [0.8, 0.8, 0.8]$) (*see* **Notes 4** and **5**). The file should have as many rows as there are wild-type cytosines.

3.6 Running Lux

(a) Installing Stan

The Python interface to Lux described in this note uses PyStan [24] to access Stan computation. In addition to PyStan, also NumPy and SciPy have to be installed; we recommend using pip to install the required Python modules (`pip install pystan numpy scipy`).

In addition, we recommend installing the stansummary script that is part of the CmdStan distribution [25]. The script stansummary can be used to summarize output files from Lux analysis.

(b) Running Lux

Running Lux using our PyStan wrapper [26] is straightforward, as PyStan will carry out the model compilation, input data transfer, and results collection. Lux analysis can be executed by a single command after the input files have been prepared as described above:

```
python lux.py -d data.tsv -p prior.tsv \
-cd control_data.tsv \
-cp control_data.tsv \
-o ./output.csv
```

in which we have defined the names of input files and output file as command-line arguments. By default, 4 chains and 2000 iterations per chain are run (*see* **Note 6**). When the analysis is completed, there will be files (./output_0.csv, ./output_1.csv, ./output_2.csv, and ./output_3.csv) containing posterior samples including warm-up samples for the parameters per chain. Importantly, one should always inspect whether HMC simulations have converged; this can be done by running the stansummary script (e.g., `$STAN_HOME/bin/stansummary output_0.csv output_1.csv output_2.csv output_3.csv`) and studying the Rhat values (Rhat values should be <1.1) [27] (*see* **Note 7**). When necessary, computational complexity can be reduced (*see* **Note 8**).

The output files can be read directly in Python (e.g., `pandas.read_csv('output_0.csv',comment='#')`) or R (e.g., `read.csv('output_0.csv',comment.char='#')`); note that when using general-purpose CSV readers, one should discard the warm-up samples (by default the first half) before running further analysis. We also provide a programmatic access to the Lux calculation (*see* **Note 9**).

(c) Inspection of estimated values of experimental parameters

The first thing to do after the analysis is finished is to check the estimated values of the experimental parameters per sample. The variables bsEff, oxEff, bsBEff, and seqErr denote bisulfite conversion efficiency ($p(BS_{eff})$), oxidation efficiency ($p(ox_{eff})$), inaccurate bisulfite conversion efficiency ($p(BS^*_{eff})$), and sequencing error probability ($p(seq_{err})$), respectively (*see* also Fig. 1). To study the estimated experimental parameters, we can use the stansummary script (e.g., `$STAN_HOME/bin/stansummary output_0.csv output_1.csv output_2.csv output_3.csv`), which will summarize the posterior samples per parameter and print the summary to standard output. Examples of estimated posterior distributions of experimental parameters from experimental data [8] are illustrated in Fig. 2. Ideally, probabilities for inaccurate bisulfite conversion of 5mC and 5hmC (bsBEff) and sequencing error (seqErr) should be small ($\sim 10^{-3}$–10^{-2}), whereas the probabilities for bisulfite conversion (bsEff) should be close to 1 and the probabilities for oxidation of 5hmC (oxEff) should be closer to 1 than 0; we usually see oxEff varying between 0.7 and 0.95. For instance, estimated oxidation efficiency values would be close to zero if BS-seq and oxBS-seq samples were mislabeled. Importantly, unexpected or highly variable between samples experimental parameter values might suggest that something has gone wrong with the experiment, and thus the experimental procedures should be checked and optimized.

(d) Inspection of methylation estimates

The stansummary call mentioned above will also print information on methylation distributions. We recommend starting by comparing the methylation estimates of the control cytosines (theta_control) with their expected values (prior knowledge, $\alpha_i/\Sigma\alpha$). Examples of estimated methylation levels of control cytosines from experimental data [8] are shown in Fig. 3; these three control cytosines behave as expected across three samples. If the estimated values are far from expected, then that might suggest

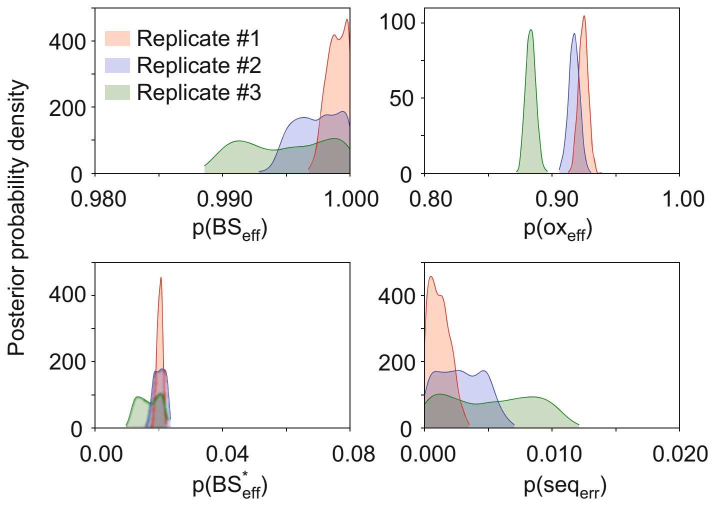


Fig. 2 Variation in experimental parameters, bisulfite conversion efficiency (*top left*), oxidation efficiency (*top right*), inaccurate bisulfite conversion (*bottom left*), and sequencing error (*bottom right*), between samples is illustrated. In this example, three samples are considered. The densities are estimated using a kernel density estimation on the posterior samples

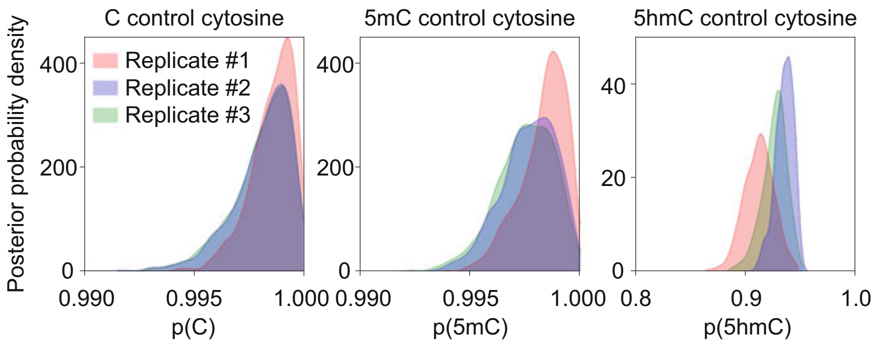


Fig. 3 Variation in methylation in control cytosines is illustrated. In this example, three control cytosines (C in left, 5mC in middle, and 5hmC in left) and three samples are considered. For each control cytosine, only its main methylation modification level is visualized; e.g., in the case of 5hmC control cytosine, only 5hmC level is visualized (third panel). The densities are estimated using a kernel density estimation on the posterior samples

that something has gone wrong in the experiment; for instance, synthesis of control oligonucleotides could have failed, or there is a mismatch between data and prior.

Next, we study the methylation distributions of wild-type cytosines (theta) per sample. Usually, most of the cytosines are either nearly fully unmethylated (C level is ~1) or methylated (5mC level is ~1). Only a small subset of cytosines, depending on the studied cell types, express 5hmC, usually at low levels (from ~0.1 to ~0.3). Unexpectedly high 5hmC levels could be due to mislabeling of BS-seq and oxBS-seq samples (e.g., p(ox$_{eff}$) is low in this case); especially, this should be checked if the signal is observed only in one experiment. It is worthwhile to check methylation levels at group level (mu). Additionally, unexpected methylation levels can be due to C → T mutations. Finally, it is often informative to visualize methylation distributions across chromosomes.

3.7 Differential Methylation Detection

Bayes factor analysis quantifies the relative evidence for one model compared to another model [28]. In our setting, the two considered models correspond to the cases of similar and differential methylation. We utilize the Savage-Dickey density ratio to approximate Bayes factors for detecting significant changes between methylation level distributions between conditions using group-level estimates (mu) [7, 29]. Briefly, the Savage-Dickey density ratio quantifies how different methylation distributions between a pair of conditions are before (a priori) and after (a posteriori) data. The a priori difference can be calculated analytically [7], but the a posteriori difference is approximated from the posterior samples using a kernel density estimation approach [7]. The provided script bf.py (`bf.py -c1 output_c1_0.csv output_c1_1.csv output_c1_2.csv output_c1_3.csv -c2 output_c2_0.csv output_c2_1.csv output_c2_2.csv output_c2_3.csv`) takes chain output files for two conditions as input and calculates and reports approximate Bayes factor for each of the cytosines. Note that the script bf.py discards warm-up samples automatically. We also provide a programmatic access to the Bayes factor calculation (*see* **Note 10**).

Differentially methylated cytosines can be defined as those that have Bayes factor greater than a specific threshold. For example, Jeffreys' interpretation of Bayes factors [30] can be used to guide the choice of the threshold value (*see* Table 2). Instead of using any specific cutoff, Bayes factors can be simply used to rank the findings. Then the comprehensive biological interpretation of the methylation changes can begin for identifying active epigenetic regulation of gene expression. For instance, the cytosines expressing differential methylation are putative regulatory regions through which transcriptional regulation takes place [31, 32]. However, relating changes in methylation to phenotypes usually requires other data [33].

4 Notes

1. Bismark can be defined to use Bowtie instead of Bowtie 2. In this case, the reference genome indices have to be constructed for Bowtie (bismark_genome_preparation --bowtie1). Additionally, Bismark has to be defined to use Bowtie instead of Bowtie 2 in mapping (bismark --bowtie1).

Table 2
Jeffreys' interpretation of Bayes factors ($\log_{10}$) is given in the table

	Strength of evidence					
	Negative	Barely worth mentioning	Substantial	Strong	Very strong	Decisive
$\log_{10}(\text{BF})$	<0	0–1/2	1/2–1	1–3/2	3/2–2	>2

Bayes factors quantify the relative strength of evidence that the data provide for the alternative hypothesis against the null hypothesis. Therefore, Bayes factors can be used in model selection and differential methylation detection

2. Computation and memory requirements are lower if only the cytosines in CpG context are considered (the argument --CX is not defined). Depending on the application and biological question, focusing on the CpG cytosines, where most of the cytosine methylation is located, might be a reasonable approach.
3. In the case of low sequencing depth, counts from different strands representing a single CpG dinucleotide could be pooled to increase statistical power. However, the pooling will lead to non-strand-specific CpG methylation estimates.
4. If sequencing depth is low, then it might be useful to change the hyperparameter (α) controlling the prior for wild-type cytosine methylation. The rationale would be to make the prior less informative, and consequently, the experimental data will have more weight. However, at the limit (Jeffrey's prior), the Savage-Dickey density ratio is not applicable.
5. The Savage-Dickey approximation uses the methylation prior distribution to calculate the a priori methylation distribution differences. Moreover, the Savage-Dickey approximation involves calculating the ratio between a priori and a posteriori densities. Thus, the approximated Bayes factors depend directly on the prior distribution. In principle, less informative (i.e., probability mass is distributed more widely) priors will lead to greater Bayes factors.
6. When desired, number of chains and number of iterations per chain can be specified using command-line arguments --iter (or -i) and --chains (or -n), respectively.
7. In order to reach convergence, one should try to increase the number of iterations per chain, for instance, by doubling the number of iterations until convergence is reached.
8. The simultaneous estimation of experimental parameters and methylation distributions in genome--wide setting might be computationally too expensive. Therefore, in these cases, we recommend a two--step approach: first, the experimental parameter posterior distributions are estimated without considering any wild-type cytosine data, and second, wild-

type cytosine data is analyzed using either point estimates of experimental parameters (when the corresponding distributions are tight) or approximated parametric forms of the posterior distributions (when the corresponding distributions are not tight). Additionally, distributed computing can be used to speed up the analysis by distributing the computation of distinct sets of cytosines.

9. Some users might find that the RStan and PyStan interfaces provide more integrated workflows than CmdStan. For instance, handling of input data and visualization (*see*, e.g., BayesPlot and ShinyStan) could be done directly in Python or R. We provide functions for generating inputs from arrays for Lux, running Lux, and calculating Bayes factors for use in Python.
10. Let us assume we have input (count and prior) for two conditions, a and b, and we wish to compare them. Moreover, assume that the data is stored in the same format as described above in the following NumPy arrays: counts_a, prior, control_counts_a, control_prior_a, counts_b, control_counts_b, and control_prior_b. Then, we can generate the required input variables, run the Lux analysis, and calculate the Bayes factors as follows:

```
  # import some routines from lux
  from lux import generate_inputs, run_lux, \
calculate_bayes_factors

  # generate data and init variables for A and B conditions
  # from input data for lux
  data_a,init_a = generate_inputs(counts_a,prior, \
  control_counts_a, control_prior_a,from_files=False)
  data_b,init_b = generate_inputs(counts_b,prior, \
  control_counts_b, control_prior_b,from_files=False)

  # run Lux analysis for A and B conditions
  fit_a,samples_a = run_lux(data_a,init_a,iter=2000, \
  chains=4, sample_file='./output_a.csv')

  fit_b,samples_b = run_lux(data_b,init_b,iter=2000, \
  chains=4, sample_file='./output_b.csv')
  # calculate Bayes factors
  bfs = calculate_bayes_factors(samples_a['mu'], \
samples_b['mu'])
```

Note that the extract method we use in the run_lux routine automatically discards the warm-up samples; i.e., samples_a and samples_b are ready to be used.

References

1. Kohli RM, Zhang Y (2013) TET enzymes, TDG and the dynamics of DNA demethylation. Nature 502(7472):472. https://doi.org/10.1038/nature12750
2. Pastor WA, Aravind L, Rao A (2013) TETonic shift: biological roles of TET proteins in DNA demethylation and transcription. Nat Rev Mol Cell Biol 14(6):341. https://doi.org/10.1038/nrm3589
3. Wu X, Zhang Y (2017) TET-mediated active DNA demethylation: mechanism, function and beyond. Nat Rev Genet 18(9):517–534
4. Shen L, Wu H, Diep D, Yamaguchi S, D'Alessio AC, Fung H-L et al (2013) Genome-wide analysis reveals TET-and TDG-dependent 5-methylcytosine oxidation dynamics. Cell 153(3):692–706
5. Spruijt CG, Gnerlich F, Smits AH, Pfaffeneder T, Jansen PW, Bauer C (2013) Dynamic readers for 5-(hydroxy)methylcytosine and its oxidized derivatives. Cell 152(5):1146–1159. https://doi.org/10.1016/j.cell.2013.02.004
6. Yin Y, Morgunova E, Jolma A, Kaasinen E, Sahu B, Khund-Sayeed S et al (2017) Impact of cytosine methylation on DNA binding specificities of human transcription factors. Science 356(6337):eaaj2239. http://www.sciencemag.org/lookup/doi/10.1126/science.aaj2239
7. Äijö T, Huang Y, Mannerström H, Chavez L, Tsagaratou A, Rao A et al (2016) A probabilistic generative model for quantification of DNA modifications enables analysis of demethylation pathways. Genome Biol 17(1):49. https://doi.org/10.1186/s13059-016-0911-6
8. Äijö T, Yue X, Rao A, Lähdesmäki H (2016) LuxGLM: a probabilistic covariate model for quantification of DNA methylation modifications with complex experimental designs. Bioinformatics 32(17):i511–i519
9. Plongthongkum N, Diep DH, Zhang K (2014) Advances in the profiling of DNA modifications: cytosine methylation and beyond. Nat Rev Genet 15(10):647–661. https://doi.org/10.1038/nrg3772
10. Huang Y, Pastor WA, Shen Y, Tahiliani M, Liu DR, Rao A (2010) The behaviour of 5-hydroxymethylcytosine in bisulfite sequencing. PLoS One 5(1):e8888. https://doi.org/10.1371/journal.pone.0008888
11. Booth MJ, Branco MR, Ficz G, Oxley D, Krueger F, Reik W (2012) Quantitative sequencing of 5-methylcytosine and 5-hydroxymethylcytosine at single-base resolution. Science 336(6083):934–937. https://doi.org/10.1126/science.1220671
12. Yu M, Hon GC, Szulwach KE, Song CX, Zhang L, Kim A (2012) Base-resolution analysis of 5-hydroxymethylcytosine in the mammalian genome. Cell 149(6):1368–1380. https://doi.org/10.1016/j.cell.2012.04.027
13. Song CX, Szulwach KE, Dai Q, Fu Y, Mao SQ, Lin L (2013) Genome-wide profiling of 5-formylcytosine reveals its roles in epigenetic priming. Cell 153(3):678–691. https://doi.org/10.1016/j.cell.2013.04.001
14. Booth MJ, Marsico G, Bachman M, Beraldi D, Balasubramanian S (2014) Quantitative sequencing of 5-formylcytosine in DNA at single-base resolution. Nat Chem 6(5):435–440. https://doi.org/10.1038/nchem.1893
15. Lu X, Song CX, Szulwach K, Wang Z, Weidenbacher P, Jin P (2013) Chemical modification-assisted bisulfite sequencing (CAB-Seq) for 5-carboxylcytosine detection in DNA. J Am Chem Soc 135(25):9315–9317. https://doi.org/10.1021/ja4044856
16. Wu H, Wu X, Shen L, Zhang Y (2014) Single-base resolution analysis of active DNA demethylation using methylase-assisted bisulfite sequencing. Nat Biotechnol 32(12):1231–1240. https://doi.org/10.1038/nbt.3073
17. Yu M, Hon GC, Szulwach KE, Song C-X, Jin P, Ren B et al (2012) Tet-assisted bisulfite sequencing of 5-hydroxymethylcytosine. Nat Protoc 7(12):2159–2170. https://doi.org/10.1038/nprot.2012.137
18. Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M et al (2017) Stan: a probabilistic programming language. J Stat Softw 76(1):1–32. https://www.jstatsoft.org/v076/i01
19. Hoffman MD, Gelman A (2014) The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. J Mach Learn Res 15(1):1593–1623
20. Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2013) Bayesian data analysis, 3rd edn. Taylor & Francis. (Chapman & Hall/CRC Texts in Statistical Science), London. https://books.google.com/books?id=ZXL6AQAAQBAJ
21. Andrews S (2010) FastQC: a quality control tool for high throughput sequence data [Internet]. http://www. bioinformatics.babraham.ac.uk/projects/fastqc/

22. Krueger F, Andrews SR (2011) Bismark: a flexible aligner and methylation caller for bisulfite-Seq applications. Bioinformatics 27(11):1571–1572. https://doi.org/ 10.1093/bioinformatics/btr167
23. Quinlan AR, Hall IM (2010) BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics 26(6):841–842. https://doi.org/10.1093/bioinformatics/btq033
24. Stan Development Team (2017) PyStan: the Python interface to Stan [Internet]. http://mc-stan.org
25. Stan Development Team (2017) CmdStan: the command-line interface to Stan
26. Äijö T, Mannerström H (2017) Lux: an integrative hierarchical Bayesian modeli for analyzing bisulphite based sequencing data [Internet]. https://github.com/tare/Lux/
27. Gelman A, Rubin DB (1992) Inference from iterative simulation using multiple sequences. Stat Sci 7(4):457–472. http://projecteuclid.org/euclid.ss/1177011136
28. Kass RE, Raftery AE (1995) Bayes factors. J Am Stat Assoc 90(430):773–795
29. Dickey JM (1971) The weighted likelihood ratio, linear hypotheses on normal location parameters. Ann Math Stat 42(1):204–223
30. Jeffreys H (1998) Theory of probability, 3rd edn. Oxford University Press, New York, p xii+459; (Oxford Classic Texts in the Physical Sciences)
31. Hon GC, Rajagopal N, Shen Y, McCleary DF, Yue F, Dang MD et al (2013) Epigenetic memory at embryonic enhancers identified in DNA methylation maps from adult mouse tissues. Nat Genet 45(10):1198–1206. http://www.nature.com/doifinder/10.1038/ng.2746
32. Tsagaratou A, Äijö T, Lio C-WJ, Yue X, Huang Y, Jacobsen SE et al (2014) Dissecting the dynamic changes of 5-hydroxymethylcytosine in T-cell development and differentiation. Proc Natl Acad Sci 111(32):E3306–E3315. http://www.pnas.org/cgi/doi/10.1073/pnas.1412327111
33. Ritchie MD, Holzinger ER, Li R, Pendergrass SA, Kim D (2015) Methods of integrating data to uncover genotype–phenotype interactions. Nat Rev Genet 16(2):85–97. http://www.nature.com/doifinder/10.1038/nrg3868

Chapter 5

DiMmer: Discovery of Differentially Methylated Regions in Epigenome-Wide Association Study (EWAS) Data

Tobias Frisch, Jonatan Gøttcke, Richard Röttger, Qihua Tan, and Jan Baumbach

Abstract

DNA-methylation has a strong influence on gene expression such that differences in methylation are associated with a wide range of diseases. Array-based approaches like the Illumina 450 K or 850 K EPIC chips have been used in a wide range of studies mostly comparing a disease group with healthy control, but also to correlate with survival times, for instance. Processing, normalization, and analysis of raw data require extensive knowledge in statistics and programming languages such as R. Here we introduce DiMmer, an easy-to-use Java tool for the analysis of EWAS. A graphical user interface guides the user through preprocessing, normalization, testing for differentially methylated CpGs, and finally the discovery of differentially methylated regions (DMRs). The software performs randomization tests to compute empirical *P*-values, corrects for multiple testing, and requires no prior knowledge in programming. All computed results are provided as plots or tables and can be easily exported. DiMmer is thus a powerful one-stop-shop for EWAS data analysis.

Key words DNA modification, Methylation, Epigenetic, Epigenome-wide association studies, Differentially methylated regions

1 Introduction

DNA-methylation describes the addition of a methyl group to adenine or cytosine bases within the DNA molecule. For eukaryotes especially the methylation of cytosines influences the accessibility of the DNA double-strand and is consequently involved in expression regulation [3, 10]. Since the four bases (A, C, T, G) are not equally distributed over the genome scientists focus primarily on the so-called CpG rich regions usually defined to have a GC-content of more than 50% and observed/expected CpG ratios greater than 0.6 [5].

Approaches to measure DNA-methylation levels can be categorized into next generation sequencing (NGS)-based or array-based methods. The former are principally able to detect methylation levels genome wide, not only in a CpG- but also in CHG- and CHH-context where H stands for A, T, or C. However, the resulting data is much more complex to handle and the expenses are linked to the sequencing depth. An additional bias might be introduced based on the preparation steps and utilized by the sequencing methodology [9]. Array-based methods are limited by the amount of observable methylation sites but are well known for several years and have proven to be more cost-efficient [13]. They are the quasi-standard to date, although NGS-based approaches are expected to be seen in higher numbers in the future.

The arrays provided by Illumina (HumanMethylation 450 K or MethylationEPIC 850 K BeadChip) have been widely used in epigenetic studies [15]. The 850 K array is the newest version built on the structure of the 450 K array and provides over 850,000 methylation sides that are no longer limited to CpG islands. Using the Illumina technology produces datasets that require three major steps in order to reveal differences in methylation between sample groups:

1. Normalization and Correction
2. CpG significance
3. Search for differentially methylated regions (DMRs)

In this chapter we introduce DiMmer [1], a Java-based software that provides the functionality necessary to read and analyze raw Illumina output. The only requirement to run this software is a working Java Runtime Environment (JRE) [12]. It is a one-stop-shop that, with only few mouse clicks, extracts DMRs associated with a phenotype of interest (or confounder) from Illumina raw data.

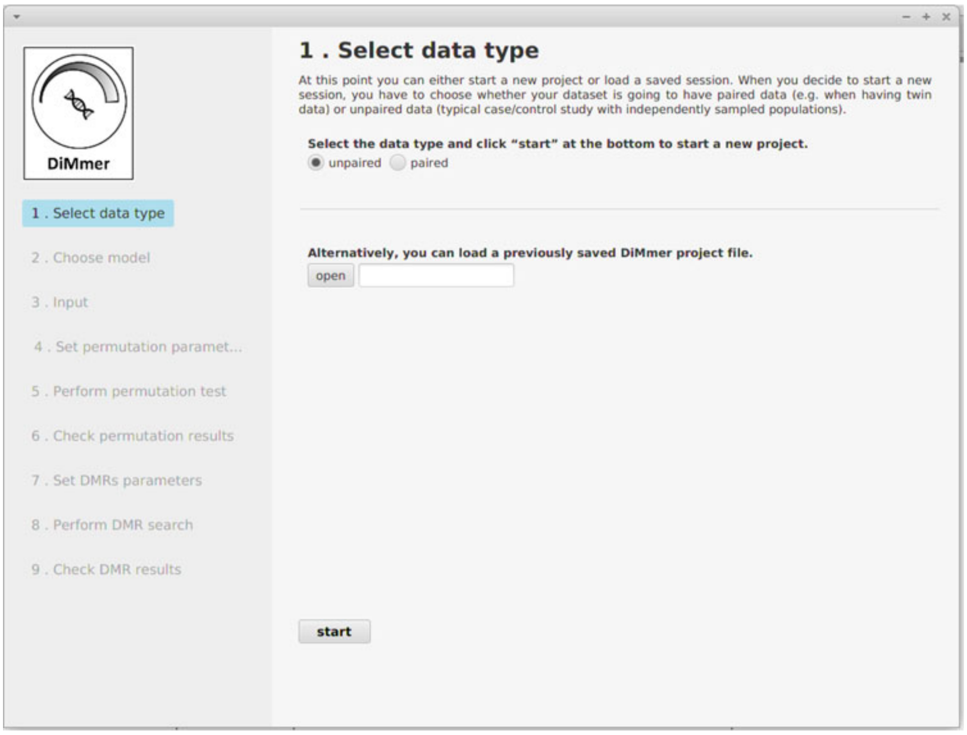
2 DiMmer Methods and Workflow

The software provides a user interface that is designed to guide through the whole analysis process. Even for larger datasets DiMmer is able to run on a standard laptop computer—but we recommend to use a machine that is equipped with 16 GB of memory (RAM). The user interface also provides information and guidelines regarding all options and on how to interpret the visualized results.

In the following section we provide an overview about the nine data analysis steps (Fig. 1) and all selections necessary to analyze your dataset. Additionally we provide background knowledge about the implemented algorithms.

2.1 CpG Significance

In the first step we have to read the given raw data files, normalize, and test for significant CpGs between samples based on the study design. A permutation will be performed in order to evaluate



1 . Select data type

At this point you can either start a new project or load a saved session. When you decide to start a new session, you have to choose whether your dataset is going to have paired data (e.g. when having twin data) or unpaired data (typical case/control study with independently sampled populations).

Select the data type and click "start" at the bottom to start a new project.

[✓ unpaired] paired

Alternatively, you can load a previously saved DiMmer project file.

Fig. 1 Starting screen of DiMmer showing the first out of nine steps necessary to process the data

statistical significance of single CpG sites. In addition, we will correct for multiple testing and visually interpret the results.

2.1.1 *Study Design*

In the first place we have to decide whether the experimental data consists of paired or unpaired samples. The majority of studies focus on independent samples although this potentially introduces problems with environmental confounders [14]. An example for a dependent dataset would be a twin study containing monozygotic twins where we are able to control for genetic factors. The decision between paired and unpaired will later influence the label permutation tests.

In the next step we have to choose a suitable model which is able to reflect our study design. The regression model is able to reveal different methylated CpGs based on continuous variables such as age, weight, or survival time. Additionally, confounding factors such as differences in cell composition can be easily included in this model type. Especially for blood samples it has been shown that cell composition is a critical factor since DNA-methylation differs strongly between different cell types [8]. Given that we are able to estimate the cell composition [7] this cell frequency should be included in the regression model. When dealing with a standard case/control study containing healthy and disease sample the t-test is recommended. Here it is assumed that the sample data

Table 1
Example structure for the annotation file

Sentrix_ID	Sentrix_Position	Status	Group_ID	Pair_ID	Gender_ID
9969489068	R01C01	0	Disease	P1	m
9969489068	R02C01	1	Healthy	P1	f

follows a normal distribution. The test static evaluates if the given groups show significant difference in the mean of their distribution. Although this model requires a binary classification of the samples, we have to distinguish between three cases: We can generally test for (1) differentially methylated CpGs between the groups (select both) and (2) hyper- or hypomethylated CpGs (select left/right).

2.1.2 Input

In screen three (Fig. 1) we are required to define our input data. As input we take the raw data from the Illumina files (*.idata) that are produced with HumanMethylation 450 K BeadChip (or Infinium MethylationEPIC 850 K). Based on those files we are able to calculate β-values for every probe as defined in Eq. 1. In this formula $y_{i,M}$ represents the intensity value of the methylated spot while $y_{i,U}$ corresponds to the unmethylated locus leading to values between 0 (unmethylated) and 1 (fully methylated).

$$\beta_i = \frac{\max(y_{i,M}, 0)}{y_{i,U} + y_{i,M} + 100} \tag{1}$$

In addition a comma separated annotation file providing information about the samples is required. The first two columns in the file are labeled *Sentrix_ID, Sentrix_Position* and are necessary to locate and access the corresponding sample IDAT file. As shown in Table 1 there are two columns (*Group_ID* and *sample*) necessary to split all samples into two groups. In case of a paired study the column *Pair_ID* reflects the connection between samples. Additional columns might be added that represent other phenotypes of interest or confounding factors.

2.1.3 Pre-processing

After defining input and output directory in screen three (Fig. 1) all necessary information for the pre-processing has been collected. In this step the DiMmer software calculates β-values as previously defined and automatically applies normalization and correction methods thereby accounting for the technical bias of the array experiments. All those steps are performed automatically by the software providing simple but effective pre-processing of the data without the user interference.

Background Correction

The background correction is based on the control probes provided by the Illumina arrays. Using those spots we are able to calculate the signal distribution of the background which then will be subtracted from every probe.

Quality Control

After correcting for the background a basic quality control is performed. Here we filter out loci whose intensity is significantly lower than the intensity values of all other probes. The default cutoff for Formula 2 is hereby set to 0.01.

$$P(i) = \sum_{j=1}^{m} \frac{I\left[\text{Intensity}(j) > \text{Intensity}(i)\right]}{m} \tag{2}$$

Quantile Normalization

Here, we use a stratified quantile normalization as described in [2] to normalize for different distributions within the β-values.

2.1.4 Permutation

In the next step DiMmer applies the previously specified model to the remaining CpGs. In order to evaluate the significance of our results we perform a permutation test. In screen 4 (Fig. 1) we are therefore specifying the amount of permutations that should be performed and define the resources available on your computer. Afterwards the t-test is applied to calculate the CpGs that are separating the user defined groups. In order to retrieve empirical P-values the labels are permuted n times and for every of those permutations the selected model is applied again.

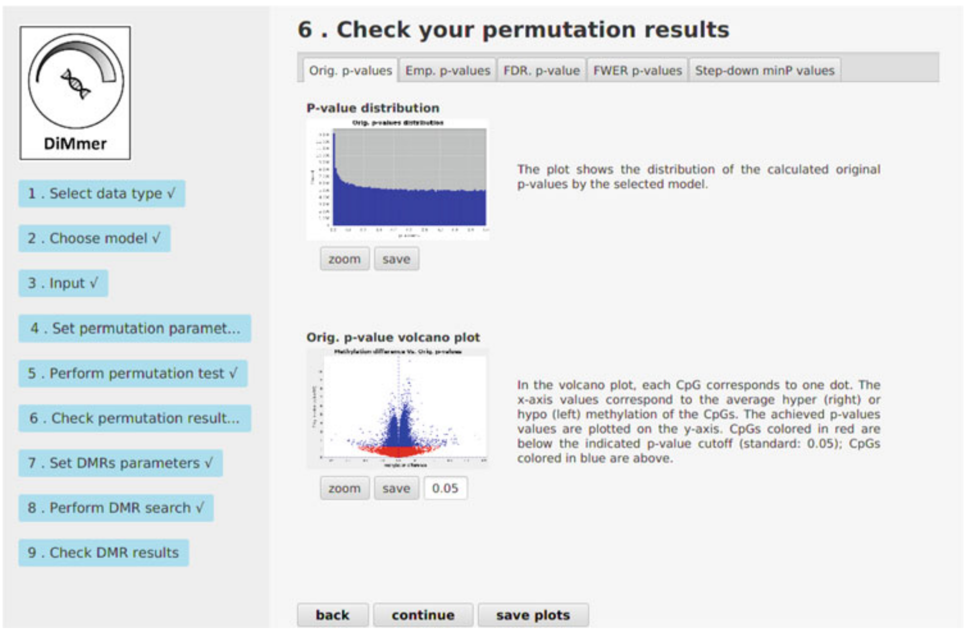
Afterwards the user is provided with a wide range of figures visualizing the results as shown in Fig. 2. Based on those results one may select a method to correct for multiple testing. Additionally, the CpGs that are significantly differentially methylated between the user-defined groups will be selected by a cutoff value (Fig. 3). Therefore a careful evaluation of the permutation results is recommended since they have a strong impact on the downstream DMR search (Subheading 2.2).

Original P-values

In the first tab of the results section (Fig. 2) of screen 6 information about the original P-values is presented. These values are the direct result of the underlying test statistic and have not been corrected for multiple testing.

Empirical P-values

The empirical P-values are the frequencies of observing P-values obtained by random permutations that are better (i.e., lower) than the original P-values calculated by the test statistic. If no label (or few) permutation is receiving a better P-value than the original labeling, we have a highly significant CpG. In opposition, finding significant P-values for a CpG site after random permutation of the labels indicates that the CpG is rather sample unique but not specific for the phenotype under investigation (high empirical P-value).



6 . Check your permutation results

P-value distribution

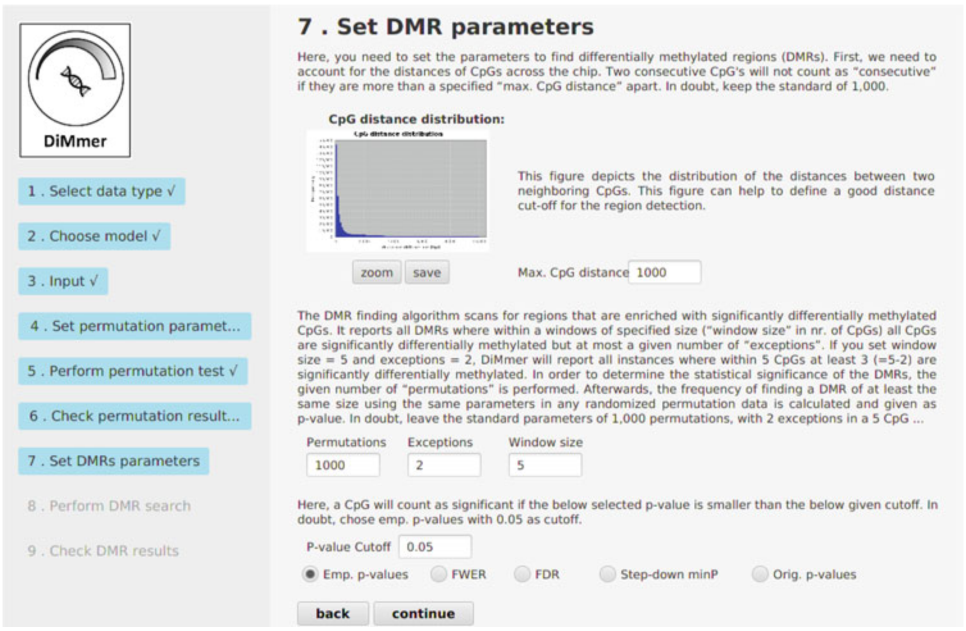
The plot shows the distribution of the calculated original p-values by the selected model.

Orig. p-value volcano plot

In the volcano plot, each CpG corresponds to one dot. The x-axis values correspond to the average hyper (right) or hypo (left) methylation of the CpGs. The achieved p-values values are plotted on the y-axis. CpGs colored in red are below the indicated p-value cutoff (standard: 0.05); CpGs colored in blue are above.

0.05

Fig. 2 In screen six we show the results after random label permutation in order to reveal differentially methylated CpGs. Based on this the user is able to decide which P-values and threshold should be used for the DMR finding algorithm



7 . Set DMR parameters

Here, you need to set the parameters to find differentially methylated regions (DMRs). First, we need to account for the distances of CpGs across the chip. Two consecutive CpG's will not count as "consecutive" if they are more than a specified "max. CpG distance" apart. In doubt, keep the standard of 1,000.

CpG distance distribution:

This figure depicts the distribution of the distances between two neighboring CpGs. This figure can help to define a good distance cut-off for the region detection.

Max. CpG distance 1000

The DMR finding algorithm scans for regions that are enriched with significantly differentially methylated CpGs. It reports all DMRs where within a windows of specified size ("window size" in nr. of CpGs) all CpGs are significantly differentially methylated but at most a given number of "exceptions". If you set window size = 5 and exceptions = 2, DiMmer will report all instances where within 5 CpGs at least 3 (=5-2) are significantly differentially methylated. In order to determine the statistical significance of the DMRs, the given number of "permutations" is performed. Afterwards, the frequency of finding a DMR of at least the same size using the same parameters in any randomized permutation data is calculated and given as p-value. In doubt, leave the standard parameters of 1,000 permutations, with 2 exceptions in a 5 CpG ...

Permutations	Exceptions	Window size
1000	2	5

Here, a CpG will count as significant if the below selected p-value is smaller than the below given cutoff. In doubt, chose emp. p-values with 0.05 as cutoff.

P-value Cutoff 0.05

[✓ Emp. p-values] FWER FDR Step-down minP Orig. p-values

Fig. 3 Screen 7 of the interface where the user is supposed to give the parameters for the DMR search: (1) The maximal distances between two consecutive CpGs in order to be allowed to belong to a reported DMR. (2) The number of permutations, window size, and how many exceptions should be allowed. (3) The P-values of interest and the selected cutoff

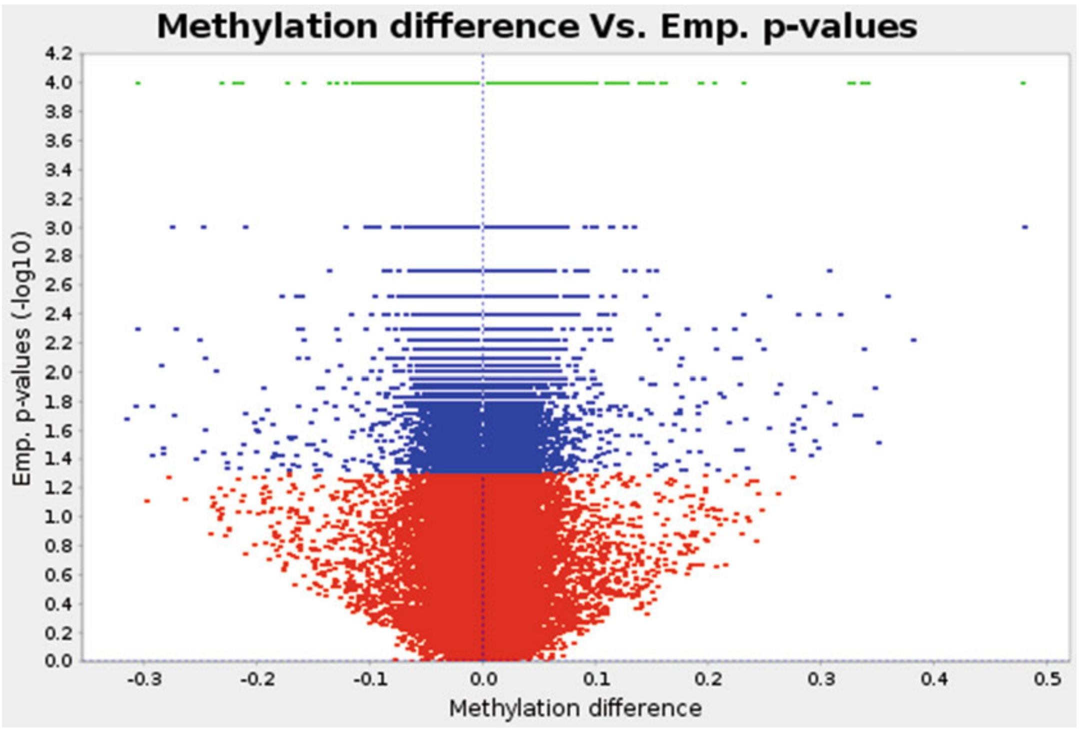


Fig. 4 Volcano plot of the empirical P-value after 1000 permutations on the example dataset. Every dot represents a CpG site and is colored red (P-value ≥ 0.05) or blue (P-value ≤ 0.05) indicating the significance for group separation

Findings of every correction method are visualized in three different plots. In Fig. 4 a volcano plot is shown that visualizes the log-fold-change on the x-axis and a selected (here empirical) P-value on the y-axis. It is generally expected that CpGs with a highly significant value also have a high change of differential methylation between user-defined groups. Furthermore, volcano plots indicate the amount of significant CpGs represented by blue and green dots (threshold 0.05), thereby giving a feeling on how to choose the threshold for downstream analysis.

In addition, it is interesting to compare the original P-value with the newly calculated normalized one. Every CpG in Fig. 5 that is on or above the diagonal line is at least as significant after correcting for multiple testing. In this figure we show the results for the empirical P-value which almost looks like a step function. This can be explained by the limited amount of permutation we have performed here.

Finally, in Fig. 6, the amount of estimated P-values is plotted in a histogram. This is usually equally distributed and can also give an impression on the amount of remaining CpGs for a chosen cutoff.

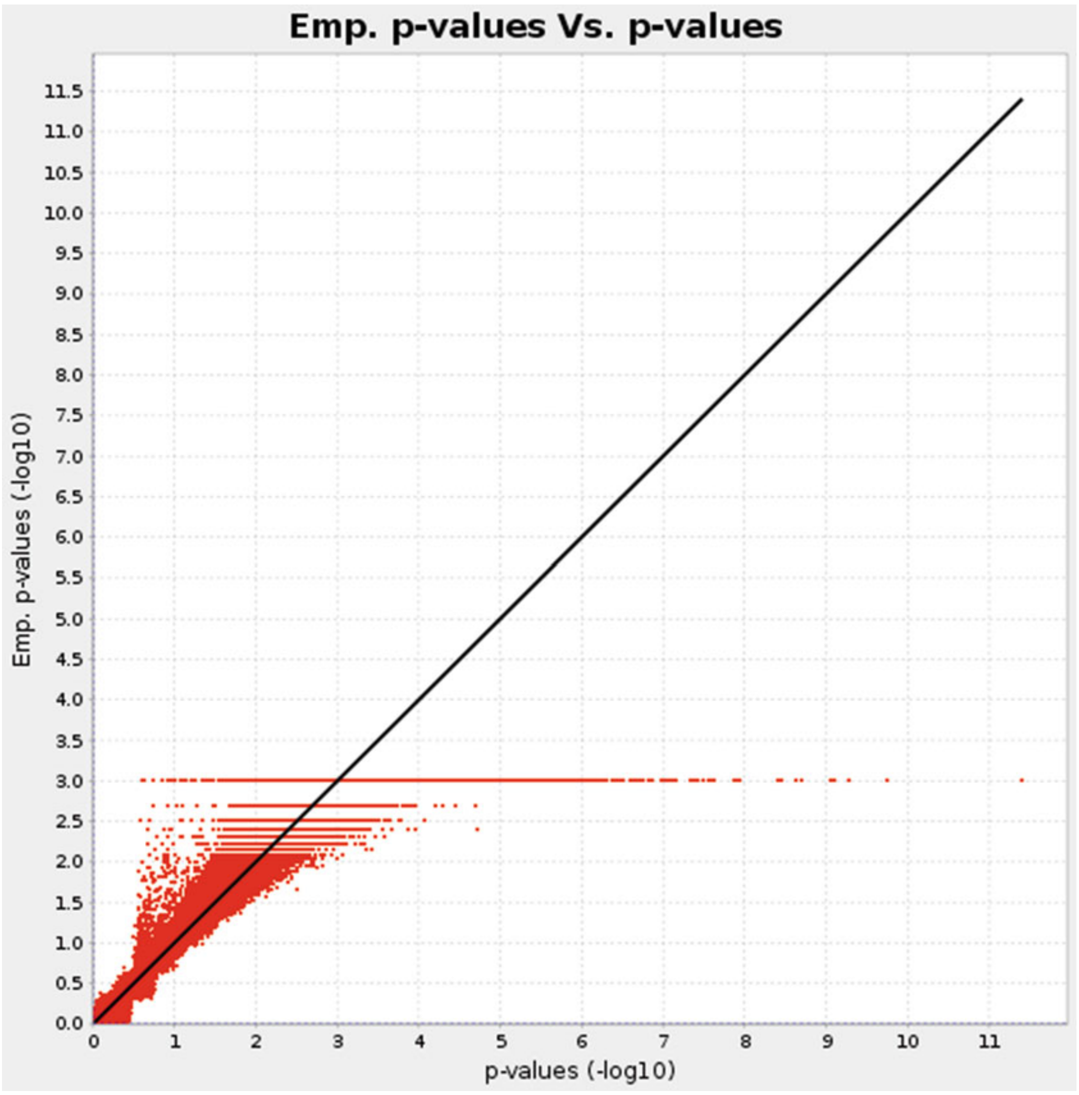


Fig. 5 Scatter plot showing original against empirical P-values after 1000 permutations

False Discovery Rate and Family-Wise Error Rate

We implemented the Benjamini-Hochberg (BH) [6] method to adjust for multiple testing by adjusting the false discovery rate (FDR).

$$p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(M)} \tag{3}$$

$$L = \max\left\{ j : p_{(j)} \leq \alpha * \frac{j}{M} \right\} \tag{4}$$

As shown in Eq. 4 this method calculates a new threshold $p_{(L)}$ based on the sorted P-values. Thus, it is reducing the type 1 error (false positives) where a CpG is predicted to be differentially methylated while it is actually not.

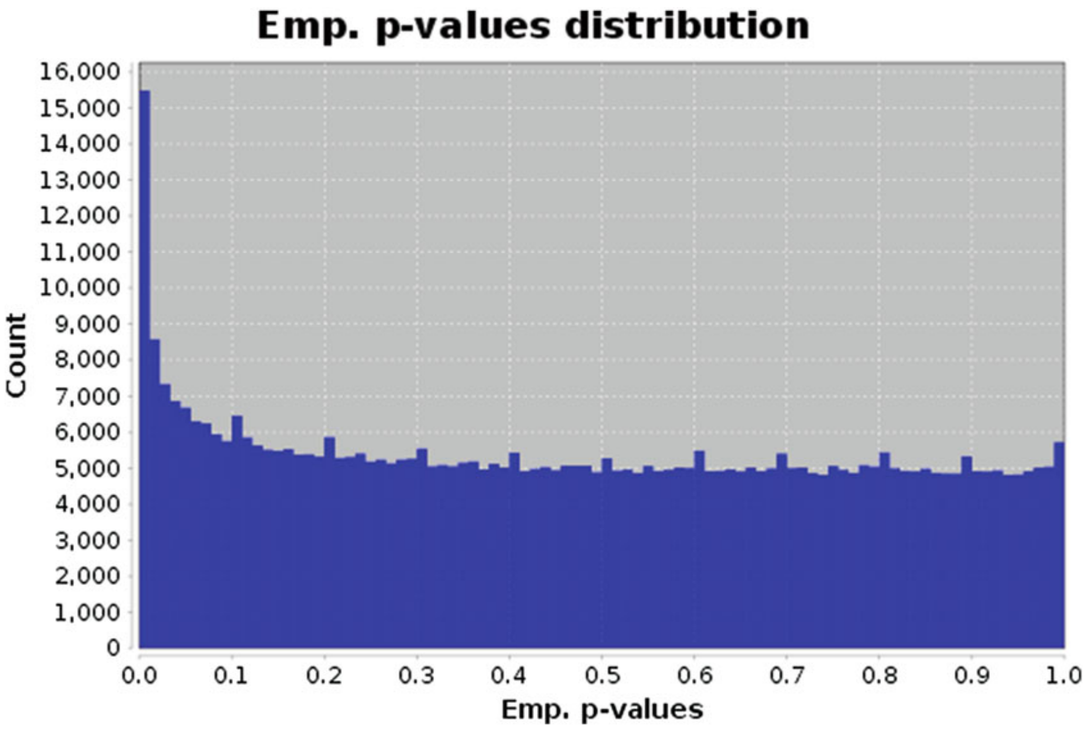


Fig. 6 Histogram plot showing empirical P-values after 1000 permutations based on the example dataset available on the DiMmer website

$$p_j \leq \frac{\alpha}{M} \tag{5}$$

We also offer the more traditional Bonferroni correction method using the family-wise error rate (FWER). It is much more conservative than BH especially since the number of possible CpGs ($M = 850{,}000$) will lead to a very low significance threshold.

2.2 Differentially Methylated Regions

A differentially methylated region (DMR) represents a set of adjacent CpGs usually reflecting functional regions of the genome. It has become general practice to hunt for DMRs instead of single CpGs to reach a higher level of robustness. Depending on the conducted study a DMR can be specific for tissue (iDMR), cancer (cDMR), or aging (aDMR) [14]. In average the length is expected to be below 1 kb. However, regions with more than 1Mb have been reported [4].

Our package uses a window based approach in order to evaluate which differentially methylated CpGs can be accumulated into one DMR (Fig. 7). In order to find all maximal DMRs with at most k exceptions a window of size w is moved over the genome. As long as the number of not differentially methylated CpGs is smaller than k the DMR is extended. Window size and number of exceptions are hereby intuitive and not independent parameters. Increasing w

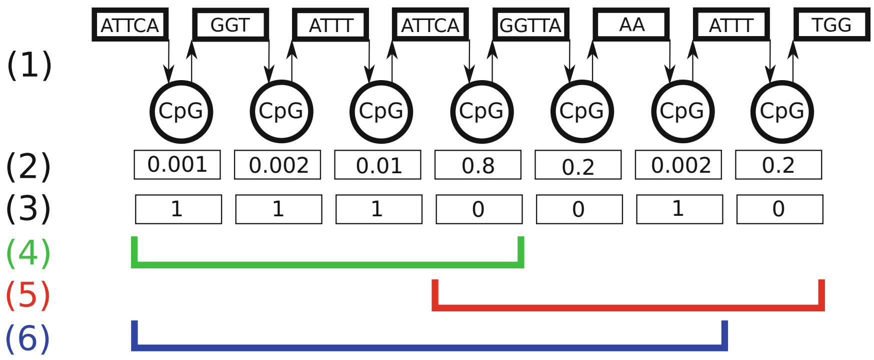


Fig. 7 Visualization of the DiMmer work flow: (1) Based on the Illumina Chip we know where the CpGs are located in the human genome. (2) Testing reveals whether CpGs show differences in methylation between the groups leading to P-values that are corrected for multiple testing. (3) The user decides which correction method should be applied and gives a threshold leading to a binary array where significant CpGs are marked with 1 and 0 otherwise. (4) The algorithm slides with a window of size $k = 4$ over the array. (5) If there are more than $k \leq 2$ insignificant CpGs, the algorithm stops leading to a DMR as shown in blue (6)

will consequently lead to less DMRs reported since the probability to see the maximal number of exceptions is increased. According to that increasing the number of exceptions k will mainly influence the length of revealed DMRs.

In order to evaluate whether the calculated DMRs (especially their length) are significant the tool performs a permutation test by randomly shuffling the CpG vector. For every permutation the new set of differentially methylated regions is calculated in order to compare the length of those regions. It is expected that the DMRs found in the permuted datasets are much lower than in the original one. To access the quality of a DMR the score is defined as the ratio between the amount of differentially methylated CpGs and the overall length of the DM-region. In Fig. 8 the distribution of the score (number of CpGs divided by the length of the DMR) is shown. Here we can see that the best score, achieved by a DMR in the original dataset is above 0.56 while the majority of scores achieved after permutation is less than 0.05.

Further results are summarized in the “tables” tab of screen 9 and provide a much more detailed overview. Especially the “Merged table” combines all results of the DMR search and permutation tests. The most important columns:

1. Hyperlink provides a UCSC [11] link that will show the selected DMR in the UCSC genome browser.
2. The number of CpG contains the number of differentially-/hyper-/hypo-methylated CpGs within the DMR depending on the previously selected test statistics.
3. The number of DMRs shows the amount of DMRs with the same size found in the non-permuted data.

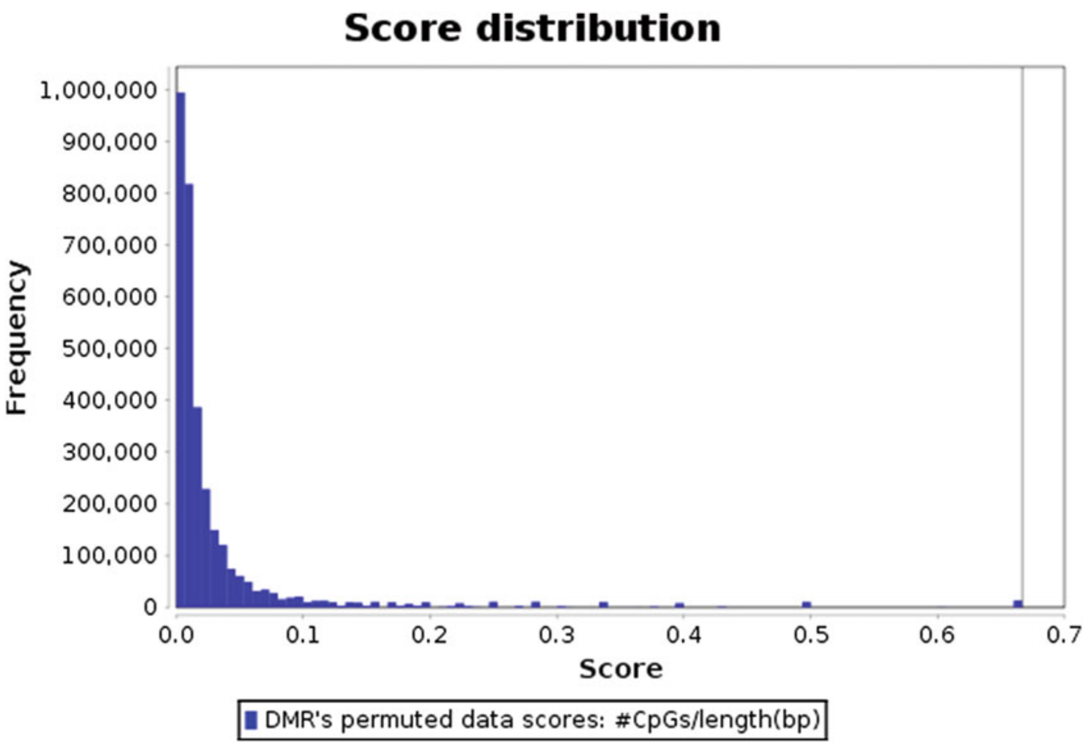


Fig. 8 Histogram showing the distribution of DMR scores of all permutations (blue bars). The vertical line indicates the score achieved by the best DMR in non-permuted data

4. Average DMRs: The number of DMRs in all permutations that have at least the same length.
5. The P-value indicates the probability to find at least the same number of DMRs with at least the same length over all permutations. This is probably the most important column and the final result one should look at.
6. In the next column the log-ratio provides the ratio between column three and four. It thereby assesses the quality of the calculated DMRs compared to permuted data. High values indicate that in average we found significantly less DMRs in the permuted data of the same length.
7. As previously mentioned the score is the ratio between differentially methylated CpG and the length of the selected region.

Acknowledgements

Jan Baumbach and Tobias Frisch are grateful for financial support from the VILLUM foundation (Young Investigator Grant nr. 13154).

References

1. Almeida D, Skov I, Silva A, Vandin F, Tan Q, Röttger R, Baumbach J (2016) Efficient detection of differentially methylated regions using dimmer. Bioinformatics 33(4):549–551
2. Aryee MJ, Jaffe AE, Corrada-Bravo H, Ladd-Acosta C, Feinberg AP, Hansen KD, Irizarry RA (2014) Minfi: a flexible and comprehensive bioconductor package for the analysis of infinium DNA methylation microarrays. Bioinformatics 30(10):1363–1369
3. Bock C (2012) Analysing and interpreting DNA methylation data. Nat Rev Genet 13(10):705
4. Frigola J, Song J, Stirzaker C, Hinshelwood RA, Peinado MA, Clark SJ (2006) Epigenetic remodeling in colorectal cancer results in coordinate gene suppression across an entire chromosome band. Nat Genet 38(5):540
5. Gardiner-Garden M, Frommer M (1987) CpG islands in vertebrate genomes. J Mol Biol 196(2):261–282
6. Hastie T, Tibshirani R, Friedman J (2003) The elements of statistical learning, corrected edn. Springer, Berlin
7. Houseman EA, Accomando WP, Koestler DC, Christensen BC, Marsit CJ, Nelson HH, Wiencke JK, Kelsey KT (2012) DNA methylation arrays as surrogate measures of cell mixture distribution. BMC Bioinf 13(1):86
8. Jaffe AE, Irizarry RA (2014) Accounting for cellular heterogeneity is critical in epigenome-wide association studies. Genome Biol 15(2):R31
9. Ji L, Sasaki T, Sun X, Ma P, Lewis ZA, Schmitz RJ (2014) Methylated DNA is over-represented in whole-genome bisulfite sequencing data. Front Genet 5:341
10. Karlić R, Chung HR, Lasserre J, Vlahoviček K, Vingron M (2010) Histone modification levels are predictive for gene expression. Proc Natl Acad Sci 107(7):2926–2931
11. Karolchik D, Baertsch R, Diekhans M, Furey TS, Hinrichs A, Lu Y, Roskin KM, Schwartz M, Sugnet CW, Thomas DJ et al (2003) The UCSC genome browser database. Nucleic Acids Res 31(1):51–54
12. Oracle (2014) Java 8. http://www.oracle.com/technetwork/java/javase/overview/java8-2100321.html. Accessed 06 Nov 2017
13. Plongthongkum N, Diep DH, Zhang K (2014) Advances in the profiling of DNA modifications: cytosine methylation and beyond. Nat Rev Genet 15(10):647
14. Rakyan VK, Down TA, Balding DJ, Beck S (2011) Epigenome-wide association studies for common human diseases. Nat Rev Genet 12(8):529
15. Wilhelm-Benartzi CS, Koestler DC, Karagas MR, Flanagan JM, Christensen BC, Kelsey KT, Marsit CJ, Houseman EA, Brown R (2013) Review of processing and analysis methods for DNA methylation array data. Br J Cancer 109(6):1394

Chapter 6

Implementing a Transcription Factor Interaction Prediction System Using the GenoMetric Query Language

Stefano Perna, Arif Canakoglu, Pietro Pinoli, Stefano Ceri, and Limsoon Wong

Abstract

Novel technologies and growing interest have resulted in a large increase in the amount of data available for genomics and transcriptomics studies, both in terms of volume and contents. Biology is relying more and more on computational methods to process, investigate, and extract knowledge from this huge amount of data. In this work, we present the TICA web server (available at http://www.gmql.eu/tica/), a fast and compact tool developed to support data-driven knowledge discovery in the realm of transcription factor interaction prediction. TICA leverages both the GenoMetric Query Language, a novel query tool (based on the Apache Hadoop and Spark technologies) specialized in the integration and management of heterogeneous, large genomic datasets, and a statistical method for robust detection of co-locations across interval-based data, in order to infer physically interacting transcription factors. Notably, TICA allows investigators to upload and analyze their own ChIP-seq experiments datasets, comparing them both against ENCODE data or between themselves, achieving computation time which increases linearly with respect to dataset size and density. Using ENCODE data from three well-studied cell lines as reference, we show that TICA predictions are supported by existing biological knowledge, making the web server a reliable and efficient tool for interaction screening and data-driven hypothesis generation.

Key words Transcription factor interaction, Gene regulation, Genomic computing, Biostatistics, ChIP-seq analysis, Data integration

1 Motivation

Gene expression in prokaryotes and eukaryotes determines almost every internal and external behavior of the cell(s), from reaction to stimuli all the way to cell development and death. To modulate gene expression, cells have evolved different mechanisms. One of the most well known and studied is the activity of Transcription Factors (TFs): these proteins possess highly specific DNA-binding domains that they use to latch onto specific parts of the DNA. Once attached, TFs can enhance or repress RNA polymerase access to the DNA area encoding for a particular gene, thereby reducing or enhancing the amount of its expression. This is one of the most basic forms of regulation and is widely used across all species in the natural world; thus, it is of high interest for researchers to understand the role of each transcription factor in the regulatory machinery.

Transcription factors are known to implement their regulatory mechanisms in coordination, acting as functional groups. Ways to discover TF complexes include in vivo experiments, observation of live cells, and testing potential interactors in vitro; however, given the intrinsic combinatorial nature of the problem, these approaches are unlikely to be complete or even feasible over the whole spectrum of TF-TF interactions. In the context of gene regulation, computational biology has become a powerful hypothesis generation tool, rooted in mathematical interpretation of experimental data: by screening unlikely interactions, the investigator can then focus resources on verifying the most interesting candidate interactors using more traditional methods.

In this chapter, we present the *TICA* (Transcriptional Interaction and Coregulation Analyser) web server, a convenient tool for analyzing chromatin immunoprecipitation and sequencing dataset targeting TF binding locations and predicting TF-TF interaction. The TICA web server leverages two powerful assets:

- the expressive power of *GenoMetric Query Language* (GMQL) [7], a novel high-level declarative language for seamless integration, management, and querying of heterogeneous genomic datasets;
- a *statistical classifier* which predicts colocation between interval-based data on a single reference system by exploiting the structural and positional information given by the intervals themselves.

We developed TICA in the context of the TF-TF interaction prediction problem (hence the name), and therefore its model is tailored to the needs of this biological context. The TICA web server, developed in the *Django* framework, is available for both data exploration of ENCODE narrowpeaks on *Homo sapiens* cell lines and for analysis of novel biological datasets, provided by biological investigators.

This chapter is structured as follows: in Subheading 2 we describe the web server, the main workflow and resulting output. In Subheading 3, we provide an overview of the implementation strategies we used to develop the web server and underlying algorithm, and discuss the advantages of using GenoMetric Query Language queries. In Subheading 4, we analyze the performance of the web server, in particular we describe datasets provided in the initial deployment and how the prediction algorithm scales with increasing amounts data provided by the user. Finally, in Subheading 5, we highlight the most interesting aspects of the web service in terms of performance, accuracy, and acceptable data formats.

2 TICA Web Server

We have developed and deployed a web server (and related web application), with which investigators can use the TICA framework to predict TF-TF interaction on ChIP-seq datasets on a set of model cell lines from *Homo sapiens*. The web server can be accessed at: http://www.gmql.eu/tica/. The web implementation can be employed in three ways:

1. users can investigate the latest version of ENCODE ChIP-Seq data available to search for evidence regarding interaction hypotheses;
2. they can upload their own TF ChIP-seq datasets to the database and analyze all possible interactor couples therein; or
3. they can upload their datasets and compare them with the ENCODE datasets, searching for potential interaction phenomena.

2.1 Workflow

Users connecting to the server see the welcome page reported in Fig. 1. They are not required to create an account or authenticate in any way in order to use the web server: data uploaded is stored in a temporary folder (with a session ID for tracking during analysis), and subsequently discarded. In the welcome page, the user is prompted to select the context cell line: this determines the *p*-values for statistical tests (due to different null distributions) and the list of ENCODE TFs available for comparisons.

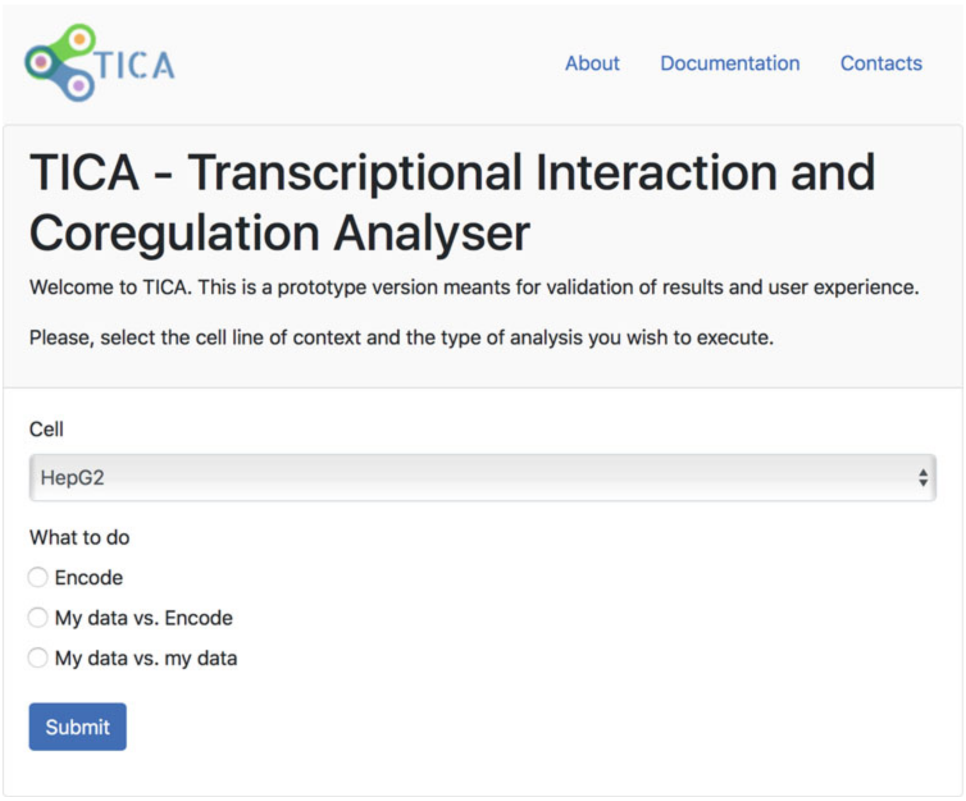
The workflow in the cases 1, 2, and 3 above is identical, except for the upload procedure required to submit, transform, and filter user-provided datasets (*see* Subheading 3.1). Experimental data have to be uploaded via a single zip file containing one folder for each TF, which must be named as the TF itself. Each sample will be assigned to the TF inferred by its folder, regardless of the actual filename; single files should be in ENCODE bed narrowpeak format.[1]

If the user selects “ENCODE” in the main page, they will be immediately redirected to parameter selection.

2.2 Parameters

After uploading data (if required) users have to specify the parameters for the analysis using the parameter input page (Fig. 2). A user can tune most of the TICA classifier parameters to suit biological assumptions or experimental conditions (cf. Table 2): among other choices, the user can restrict the analysis to a sublist of the TFs to be compared, define mindist couples maximum distance (from preselected values: 1100, 2200, 5500 bp), declare which test conditions

[1]The schema for ENCODE narrowpeak data files is defined at https://genome.ucsc.edu/FAQ/FAQformat.html#format12.



TICA - Transcriptional Interaction and Coregulation Analyser

Welcome to TICA. This is a prototype version meants for validation of results and user experience.

Please, select the cell line of context and the type of analysis you wish to execute.

Cell: HepG2

What to do

- ◯ Encode
- ◯ My data vs. Encode
- ◯ My data vs. my data

Fig. 1 Screenshot of TICA web application main page. Through the drop-down menu, the users can decide the context cell line among those available; users can also select whether they want to upload data or use ENCODE data

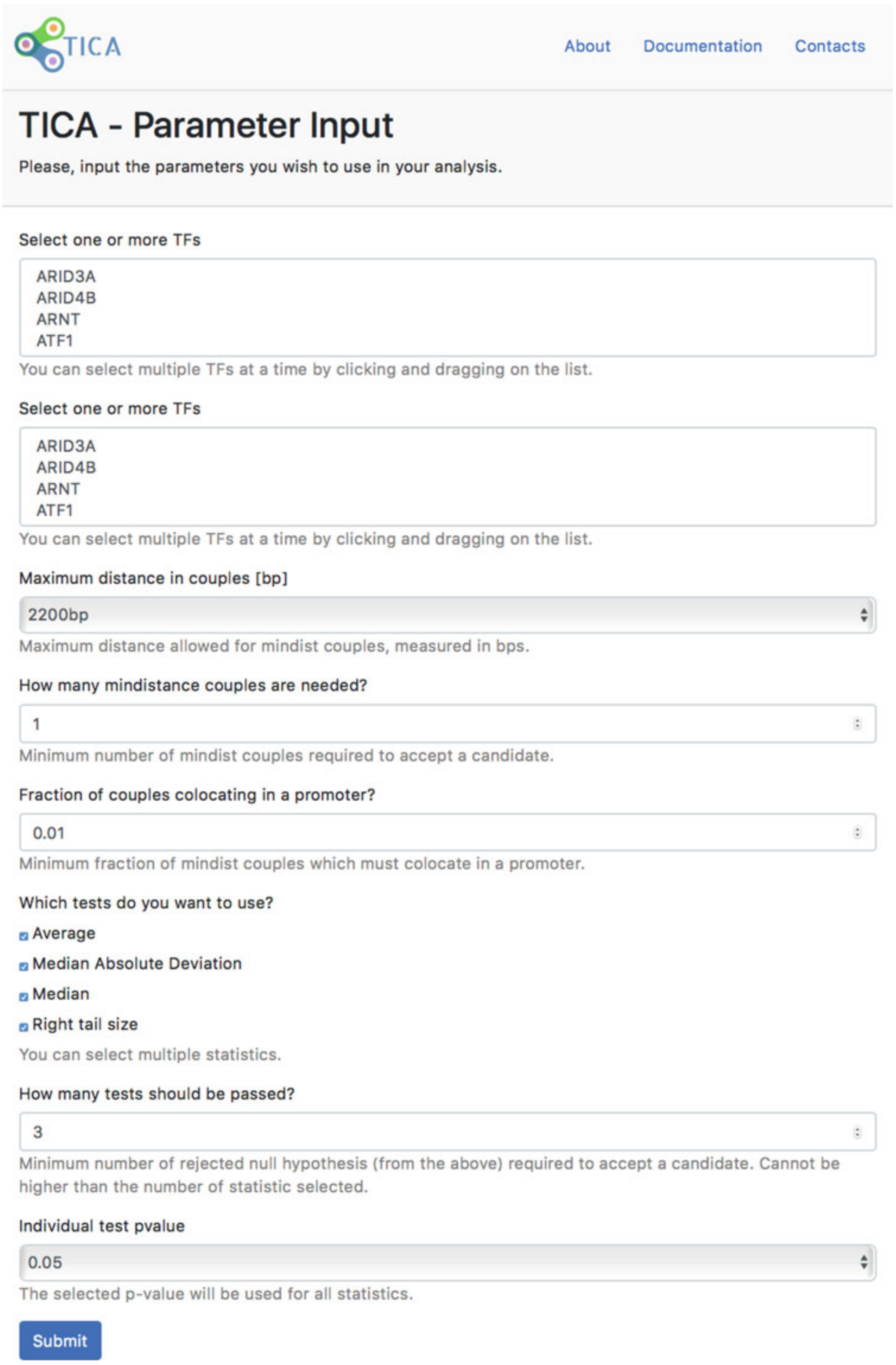
have to be used (by ticking or unticking the corresponding test names), and state global significance level required and minimum number of test conditions to be satisfied (for additional details on the TICA classification algorithm, *see* Subheading 3.2). Default values are provided, matching specifications in Table 2.

2.3 Output

Results are presented to the user through a table and a heatmap (*see* Fig. 3): the heatmap shows the number of test conditions satisfied, with −1 represents TF-TF pairs that do not meet the biological information screening criteria (*see* Subheading 3.2). Details on each feature extracted from observed mindistance couple distributions are given in a separate table, on the same page. Results can be exported as a .csv file using the “Export to CSV” link (also in Fig. 3).

2.4 Deployment

All mindistance couples and related distances for the default cell lines in ENCODE data are precomputed and stored in a PostgreSQL database. These tables are only refreshed during major data updates; when user-provided data is uploaded in the system,



TICA - Parameter Input

Please, input the parameters you wish to use in your analysis.

Select one or more TFs

ARID3A
ARID4B
ARNT
ATF1

You can select multiple TFs at a time by clicking and dragging on the list.

Select one or more TFs

ARID3A
ARID4B
ARNT
ATF1

You can select multiple TFs at a time by clicking and dragging on the list.

Maximum distance in couples [bp]

2200bp

Maximum distance allowed for mindist couples, measured in bps.

How many mindistance couples are needed?

1

Minimum number of mindist couples required to accept a candidate.

Fraction of couples colocating in a promoter?

0.01

Minimum fraction of mindist couples which must colocate in a promoter.

Which tests do you want to use?

- [✓] Average
- [✓] Median Absolute Deviation
- [✓] Median
- [✓] Right tail size

You can select multiple statistics.

How many tests should be passed?

3

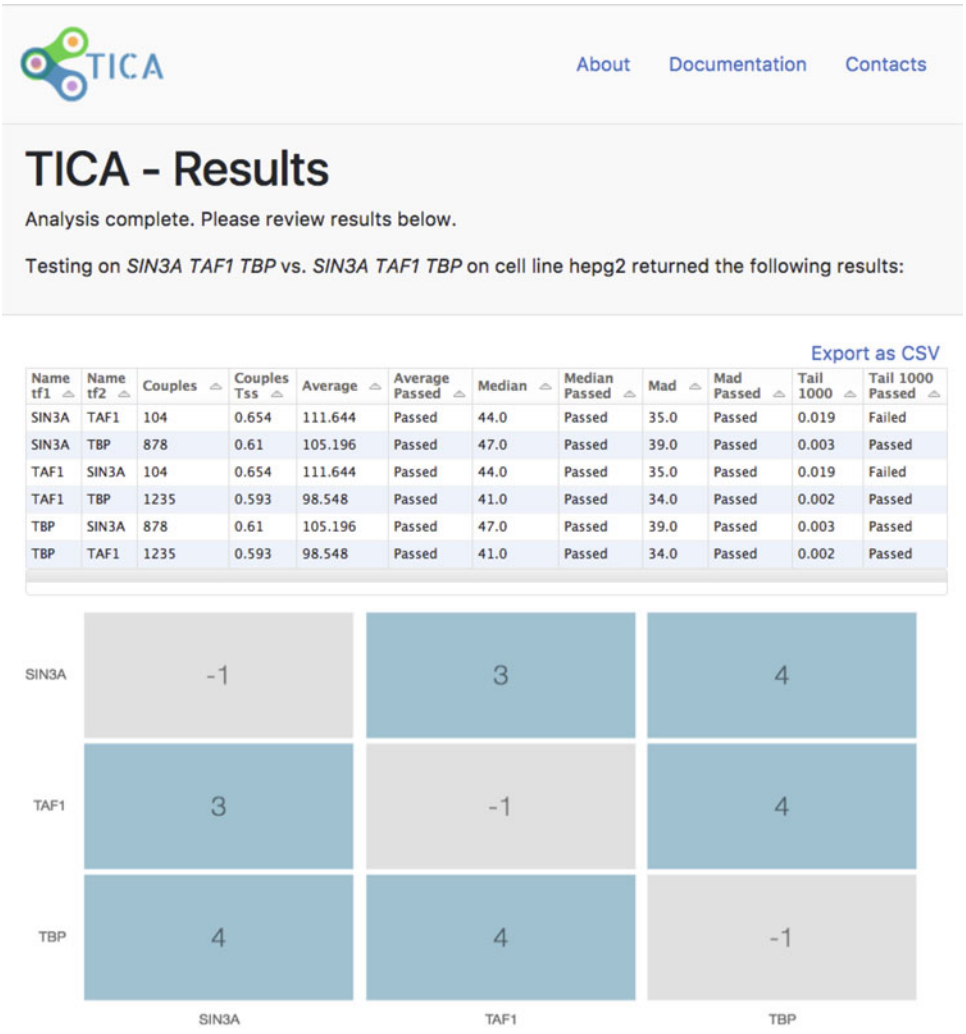
Minimum number of rejected null hypothesis (from the above) required to accept a candidate. Cannot be higher than the number of statistic selected.

Individual test pvalue

0.05

The selected p-value will be used for all statistics.

Fig. 2 Screenshot of TICA parameter input page



TICA - Results

Analysis complete. Please review results below.

Testing on *SIN3A TAF1 TBP* vs. *SIN3A TAF1 TBP* on cell line hepg2 returned the following results:

Name tf1	Name tf2	Couples	Couples Tss	Average	Average Passed	Median	Median Passed	Mad	Mad Passed	Tail 1000	Tail 1000 Passed
SIN3A	TAF1	104	0.654	111.644	Passed	44.0	Passed	35.0	Passed	0.019	Failed
SIN3A	TBP	878	0.61	105.196	Passed	47.0	Passed	39.0	Passed	0.003	Passed
TAF1	SIN3A	104	0.654	111.644	Passed	44.0	Passed	35.0	Passed	0.019	Failed
TAF1	TBP	1235	0.593	98.548	Passed	41.0	Passed	34.0	Passed	0.002	Passed
TBP	SIN3A	878	0.61	105.196	Passed	47.0	Passed	39.0	Passed	0.003	Passed
TBP	TAF1	1235	0.593	98.548	Passed	41.0	Passed	34.0	Passed	0.002	Passed

	SIN3A	TAF1	TBP
SIN3A	-1	3	4
TAF1	3	-1	4
TBP	4	4	-1

Fig. 3 Screenshot of TICA results page, after submitting a query on cell line GM12878. Middle table reports all features from statistical tests and deterministic filters. Blue squares in the heatmap denote higher number of tests passed

only minimal distance couple distance distributions between TFs provided are computed on the fly. The server was developed using the Django v1.11.7 framework (http://djangoproject.com); queries are implemented inside the Django framework using the Python API for GMQL, PyGMQL [9].

3 Implementation

The back-end supporting TICA is made of two conceptual blocks:

- a data preprocessing step, which takes either ENCODE or user-provided narrowpeaks and removes noisy binding sites and inactive transcription start sites, according to the context cell line (described in Subheading 3.1) and is implemented using GMQL;
- the prediction algorithm, a statistical procedure that compares candidate TF-TF pairs against null distributions from random pairs in the same cell line, with respect to a set of statistical aggregators (Subheading 3.2).

3.1 Data Preprocessing

We implement the preprocessing step of TICA by taking advantage of *GenoMetric Query Language* (GMQL), a high-level, declarative query language which supports data extraction as well as many standard data-driven computations required by tertiary data analysis [7]. We use mostly ChIP-seq datasets extracted from ENCODE, but GMQL supports an integrated repository with datasets extracted from ENCODE, TCGA, Epigenomic Roadmap, GDC, and GEO; integration of heterogeneous datasets is supported by the GDM data model [8]. In GDM, a dataset includes several samples; each sample is a pair of regions and metadata. For instance, in the case of a sample resulting from a ChIP-seq technology, regions describe the peaks of expressions (their start, stop, peak positions and score; region samples are similar to tracks that can be seen on a genome browser); metadata describe additional attributes of each sample, for instance the specific experiment name and tissue.

GMQL queries are written as sequence of statements operating on abstract variables, each representing a genomic dataset; it is a high-level language whose conditions apply both to regions and to metadata. GMQL implements most of the standard relational algebra operations [2], such as SELECT, PROJECT, GROUP, ORDER, UNION, DIFFERENCE; it also supports domain-specific operations, such as genometric JOIN, MAP, and COVER, whose semantics and implementation are defined in [5].[2]

GMQL is particularly powerful as a data extraction language, due to its implicit iteration over multiple samples of a dataset and its very compact and readable query specification. The language is also highly effective when integrating data coming from vastly different data sources, as the standardization to GDM allows for direct comparison between regions (represented by the same coordinates, such as chromosome, start and stop) while preserving all information ascribed to a particular data format (such as peak calling p-values from ChIP-seq experimental data, or rpkm values

[2]The full description of GMQL language for the latest version (2.1 at the time of writing) can be found at http://www.bioinformatics.deib.polimi.it/geco/?try.

from RNA-seq). GMQL seamlessly combines these attributes using commands such as PROJECT and MAP, supporting and streamlining data analysis pipelines.

As an example of the above, we show the queries which are used for extracting TF binding sites (TFBSes) and transcription start sites (TSSes), relative to a given cell line, from the repository (Listing 6.1). The TFBS filtering query (lines 1 through 6, same Listing) is also performed on user-provided narrowpeaks.

```
1  # extracts 1-base exact TF peaks and produces one sample for each TF
2  TFS = SELECT(experiment_type == 'ChIP-seq' AND cell == 'target_cell') ENCODE_NARROWPEAK;
3  TF_PEAKS = PROJECT(region_update:left AS start + peak, right AS start + peak +1) TFS;
4  TF_PEAK = COVER(1,ANY;groupby: tf_name) TF_PEAKS;
5
6  # extracts TFBSes by looking at enclosing windows with enough TF signal, i.e. enough peaks falling in a window of 1000 bases
7  WINDOW = PROJECT(region_update: start AS start - 1000, stop AS stop + 1000) TF_PEAK;
8  MAPPED_WINDOW = MAP(joinby: tf_name) WINDOW TF_PEAK;
9  TF_EXTRACTED = SELECT(region: count >= w) MAPPED_WINDOW;
10
11 # extract histone marks -- H3K9ac and H3K4me3 are found in promoter areas of actively transcribed TSSes. Similar queries are written for histones H3K4me1 (enhancers) and H3K36me3 (exons) - here omitted
12 HMS = SELECT((histone_name == 'H3K9ac' OR histone_name == 'H3K4me3') AND cell == 'target_cell') ENCODE_BROADPEAK;
13 HM = COVER(1,ANY) HMS;
14
15 # filter TSS with enough overlap with histone marks
16 TSS = SELECT(annotation_type == 'TSS') ENCODE_BED_ANNOTATION;
17 PROMOTER = PROJECT(region_update: start as start - 2000, stop as stop + 200) TSS;
18 MAPPED_PROM = MAP() PROMOTER HM;
19 TSS_FILTERED = SELECT(region: count >= h) MAPPED_PROM;
20
21 # further filters TSS with enough overlap with TF-PEAKS -- from arbitrary TF peaks
22 MERGED_PEAKS = MERGE() TF_PEAKS
23 MAPPED_TSS = MAP() TSS_FILTERED MERGED_PEAKS
24 TSS_EXTRACTED = SELECT(region: count >= k) MAPPED_TSS;
```

Listing 6.1 GMQL query used to filter TF binding sites and TSSes used by the method (summary)

- *Lines 2–4:* the TFS variable includes all the relevant TF samples extracted from ENCODE narrowpeak datasets.[3] The PROJECT operation is used to reduce the size of ChIP-seq regions to a single base pair. The COVER(1,ANY) operation is used to combine replicates from different transcription factors, keeping all regions from all samples and merging any two or more regions which overlap. The *groupby* option limits the merging to samples that share the same *tf_name* metadata attribute, i.e. contain experiment data on the same transcription factor. The result includes one sample for each distinct TF, with regions corresponding to a single base pair where the peak is located.
- *Lines 7–9:* Candidate TFs for the method are selected. A window of 1000 base pairs is constructed around each peak, and TFs associated with windows enclosing a counter of peaks over a threshold (w) are extracted. The PROJECT operation builds the WINDOW, the MAP operation counts the number of peaks included in each window, and the final SELECTion extracts the TFs.

According to the method, TSSes are extracted based on three progressively applied conditions: overlap with histone marks of promoters, of exons, and of enhancers; we only explain how to select TSSes by using histone marks of promoters, as the second and third extractions are very similar.

- *Lines 12–13:* Histone marks are selected. Extraction is done by means of a SELECTion; replicates are then combined using the COVER, keeping all regions from all samples and merging any two or more regions which overlap. Eventually, each HM sample includes all the regions of a given (set of) histone modifications present in ENCODE.
- *Lines 16–19:* TSSes are filtered. Promoter regions are built, and overlapping histone modification regions are counted; a TSS is selected if it is supported by a sufficient number of overlaps (one for each histone mark in the relevant regions). As promoter regions, we take standardized extensions of transcription start sites; these are built using a PROJECT, which takes TSSes and modifies their start and stop positions by extending them 2000 pairs upstream and 200 pairs downstream.[4] Then, the MAP operation counts the number of overlapping regions and the final selection filters the TSSes.

[3]ENCODE narrowpeaks are also given for ChIP-seqs targeting histone modifications. We remove them from the dataset by means of NOT clauses—omitted for brevity.

[4]These are nominal values for promoter and exon length, chosen for our experiments. Different investigators can use their own values for regulatory regions extension, depending on their biological assumptions.

- *Lines 22–24:* Finally, TSSes to be used in the method are extracted. In addition to overlaps with histone modifications, we also require TSSes to be supported by a sufficient number of TF peaks. The MERGE operation puts all the peaks of different transcription factors into a single sample, then the MAP counts how many peaks overlap with promoter regions for TSS as defined above; the final SELECT extracts the TSSes.

3.2 Interaction Prediction Method

After TF binding site data has been filtered and reduced to 1 bp length by means of the GMQL queries, TICA investigates colocation between two sets of transcription binding sites in a statistically robust way. It does that by performing a significance test based on the null hypothesis that two random TFs (named *candidate interactors*) are not found in close position to one another (according to suitable aggregation functions, as below).

Briefly, the main concept behind TICA is the *minimal distance couple* (or *mindist couple* for short), a pair of intervals which are found to be the closest to one another according to the given coordinate system, and are not located too far apart. Minimal distance couples for a given pair of transcription factors (represented by the positions of their binding sites) induce a distance distribution via the genomic distance function, which is used to generate a set of observations related to that particular pair of TFs. TICA uses both standard (average, median) and novel (median absolute deviation, distribution right tail size) statistical aggregators of the distances as features to feed a statistical classifier (Fig. 4). The output of the classifier is whether the null hypothesis above is rejected for a certain TF-TF pair.

TICA builds null distributions for each feature by randomly sampling pairs of TFs from those available in ENCODE phase 2 and 3 datasets (narrowPeak format) in a given cell line. Data comes from chromatin immunoprecipitation and sequencing experiments on three major context cells: HepG2, GM12878, K562. For each cell line, we also extract the TSSes which are more likely to be actively transcribed, based on available histone marks (*see* Subheading 4.1) and TF binding information, which we use to impose additional restriction on the candidate interactors: TICA rejects a candidate pair if the ratio of couples which colocate in the same promoter is too low, with respect to the total size of the distribution. This is done to make sure that results have biological relevance as indicators of potential coregulatory behavior, which is linked with physical interactions [3].

We calculate *p*-values of null distributions and TFBS colocation in promoters using a Python script (v3.6). In particular, mindistance couples are computed first with respect to one of the TF (meaning, for each of its binding sites, the algorithm finds the ones for the potential partner which are closest and not above the distance threshold), then with respect to the other. The two results

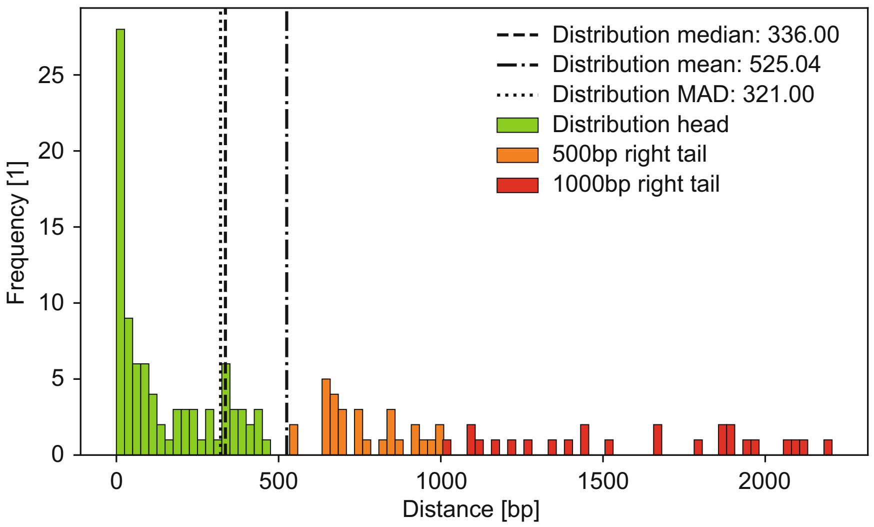


Fig. 4 Distance distribution inferred from minimal distance couples of transcription factors CTCF and JUN in cell link HepG2. Vertical lines denote statistical aggregators used in TICA tests (mean, median, and median absolute deviation). Two dimensions for the right tail are given: long (distance greater than 500 bp, orange) and short (distance greater than 1000 bp, red). Right tail size in this case is approximately 15% of the total

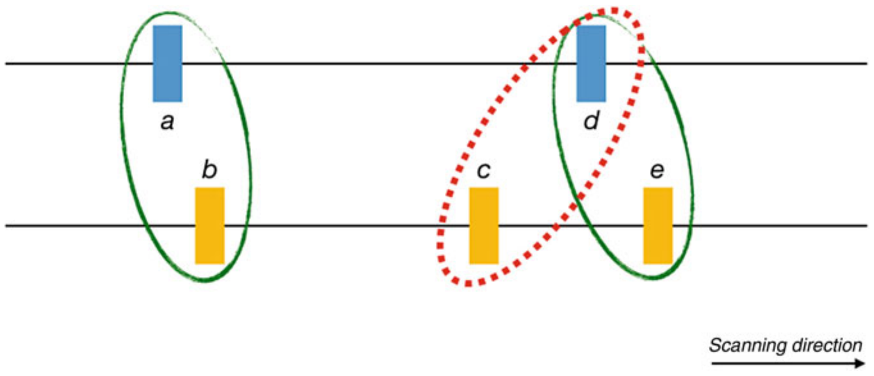


Fig. 5 Example computation of mindistance couples, highlighting possible ambiguities. Two TF track snippets are given (blue and orange). Proceeding as per the scanning direction, if blue is chosen as anchor (and orange as experiment), the minimal distance couples are correctly identified as (a,b) and (d,e) (note that d is closer to e than to c). However, if roles are inverted, three couples will be found instead: (b,a), (c,d), (e,d). Intersecting results guarantee consistency with the model

are then intersected, yielding the final mindist couple list: this is done to avoid scenarios where one binding site is the closest with respect to a target, but the reverse is not true (Fig. 5).

3.3 Data Format

TICA can in principle work with any kind of genomic regions, due to the fact that data is managed by the flexible GDM model via GMQL. However, it is reasonable to assume that the required maximum displacement between candidates will be small (in other words, we expect regions to be very close to each other with respect to the linear dimension of the universe set): this is due to the fact that physical interaction between TFs happens at molecule scale, where distances are in the order of 1–10 nucleotide base pairs [4] (compare with the average size of a human chromosome, 1.2×10^8 base pairs).

Data from ChIP-seq experiments is given in variable size, usually in the range of 10^1 (point-source information or TSS locations) to 10^3 base pairs (histone modifications, genes), making certain fine-grained analysis much more difficult. We overcome this by using ENCODE narrowpeak regions, which contain the position of the highest confidence point-source for each region (as offset from the starting point): we represent each binding site using only this high-confidence, 1 base pair-long peak in order to make statistics on small values of distance meaningful.

4 Performance

4.1 Materials

We test and validate our model using data from ENCODE phase 2 and 3 ChIP-seq experiments in narrowpeak format, currently available in GMQL public repositories. Our chosen model organism was *Homo sapiens*. We use the following data in our experiments:

- *Context cell lines:* three cell lines were selected due to data availability and quality: *HepG2* (liver carcinoma), *K562* (myelogenous leukemia) and *GM12878* (healthy lymphoblastoids);
- *TF binding locations:* data representing transcription factor binding points (TFBSes) in narrowPeak format [6], due to higher peak precision and presence of point-source location information for each region;
- *Histone marks:* the following marks have been chosen for highlighting actively transcribed TSS (*see* Subheading 3): H3K36me3 (exons), H3K9ac and H3K4me3 (promoters), H3K4me1 (enhancers). Data from ENCODE phase 2 and 3 repository, limited to cell lines mentioned above. Data format chosen is ENCODE broadPeak [6];
- *Transcription start sites:* data also from ENCODE phase 2 experiments, in standard bed format. TSS are described in terms of the first exon base only (regions are 1 bp in length).

  Data quantities are listed in the Table 1.

4.2 Parameter Settings

Parameter chosen for GMQL queries and TICA algorithm during performance evaluation are reported in Table 2. The choice of parameters is driven by the following biological considerations:

- standardized regulatory region length is a common assumption when working with gene expression regulation;

Table 1
Data volume used in pipeline experiments, listed by cell

Cell line	TF number	File number	Data size (Gb)	Data size (Millions regions)	Actively transcribed TSSes number
HepG2	200	1085	13.16	181	25097
GM12878	148	794	8.66	121	31660
K562	288	2057	23.19	322	32356

TSS numbers refer to sample size after GMQL filtering

Table 2
Parameter setting for TF-TF interaction prediction pipeline

	Parameter	Value
	Exon length	200 bp
Genomic dimensions[a]	Promoter length	2000 bp
	Enhancer length	100 kbp
	Clustering value *k*	3
Data filters	TFBS scanning window size	1000 bp
	Min. number of TFBS in active promoters	50
Metric constraints	Mindist couple max distance	2200 bp
	Number of points in nulls	$\geq$10,000
Tests and thresholds	Right-tail threshold	1000
	Test *p*-value	0.2
	Required number of rejected null hypotheses	3
	Minimum number of mindist couples	1
	Minimum fraction of mindist couples colocating in a promoter	0.01

[a]Extending TSS according to their strand

- TFBS window of accumulation is chosen so that it covers most of a standard promoter size without overextending;
- mindist couple max distance is one promoter length plus one exon (assumed size of promoter area)
- the minimum number of TFBSes in active promoter is chosen as the first quartile of the overall distribution of the counts of TFBSes in promoters in HepG2 (taken as preferred modelling environment).

Experiments and performance evaluation have been performed on the GeCo server at DEIB, Politecnico of Milano. The TICA web server is hosted on a Dell PowerEdge R730xd server with 2 Intel Xeon E5-2660 v4 processors and 384 GB of RAM.

4.3 Performance Assessment

Performance estimation for the web server can be divided into two blocks:

- computation time needed to (re)generate the database from ENCODE data and/or to analyze novel data;
- accuracy of predictions.

In the context of this work, we focus mostly on evaluation of the actual computation performance (i.e., time consumed) as opposed to discussing algorithm accuracy. Future works will be targeted towards the correctness of the method.

4.3.1 Null Distribution Generation from ENCODE

Execution times for the full pipeline on ENCODE data are listed in Table 3. Cell lines and data volumes correspond to those reported in Subheading 4.1. The pipeline has been split into four major parts:

- *TFBS query:* corresponding to lines 2 through 9 of Listing 6.1;
- *TSS query:* corresponding to lines 12 through 24 of the same;
- *TSS map:* the mapping of each binding site to all TSS in the promoter of which it binds, used to determine whether a mindist couples binds to shared promoter;
- *Mindist couples:* where the mindistance couples are computed by TICA.

Computation times reported in Table 3 refer the full analysis of the entire ENCODE cell line they refer to, which can involve thousands of millions of regions at a time (in the case of K562, ca. 3×10^8 regions are analyzed—cf. Table 1). In typical use cases, the computation times are faster by two to three orders of magnitude (cf. next paragraph).

Table 3
Tabulation of execution times for TICA pipeline steps on the three context cell lines

Cell line	TFBS query	TSS query	TSS map	Mindist couples
HepG2	108	194	21	120
GM12878	77	138	15.5	60
K562	204	407	46	376.5

Input data is taken directly from ENCODE (*see* Table 1). Time measured in minutes

4.3.2 Analysis of Novel Data

As a simulation of typical levels of workload, we generate synthetic data in narrowpeak format with variable levels of data volume. Two scaling factors were considered:

- number of transcription factors (each with a given number of regions): this influences the amount of candidates and therefore the number of times each step must be executed;
- sample size (in number of regions per sample, for a fixed amount of TFs): influences the amount of data filtered by TFBS queries, the mapping times, and the number of comparisons during mindist couples' distance distribution creation.

Note that each TF contains only one sample: giving more for each TF would not influence the computation times in a tangible manner (the COVER operation would collapse them to a single one).

We time the execution of the full pipeline on seven different scenarios, using HepG2 as context cell line: results are reported in Table 4. The datasets are built as follows:

- we first consider a baseline scenario where the user provides data for 20 TFs, each containing 5000 regions of 100 bp length—we estimate this to be a typical data size for user-submitted datasets;
- moving on the TF number scale, we submit one small (10 TFs), one medium (100 TFs), and one large (1000 TFs) dataset. Each dataset contains one sample per TF, and all samples contain 1000 regions (lines);
- moving on region-per-sample number scale, we submit three other datasets: small (10^3 regions), medium (10^4 regions), and large (10^5 regions). Each dataset contains 50 TFs and one sample per TF as before.

Table 4
Tabulation of execution times for TICA pipeline steps on synthetic datasets

Cell line	TFBS query	TSS map	Mindist couples	Total
Baseline	34	12	3	0.8′
TF-small	11	5	0.5	0.5′
TF-medium	35	52	23	2′
TF-large	219	525	802	26′
SAMPLE-small	13	28	7	1′
SAMPLE-medium	111	33	23	3′
SAMPLE-large	613	41	38	12′

Context cell line chosen is HepG2. Time measured in seconds except for total, which is converted to minutes for clarity

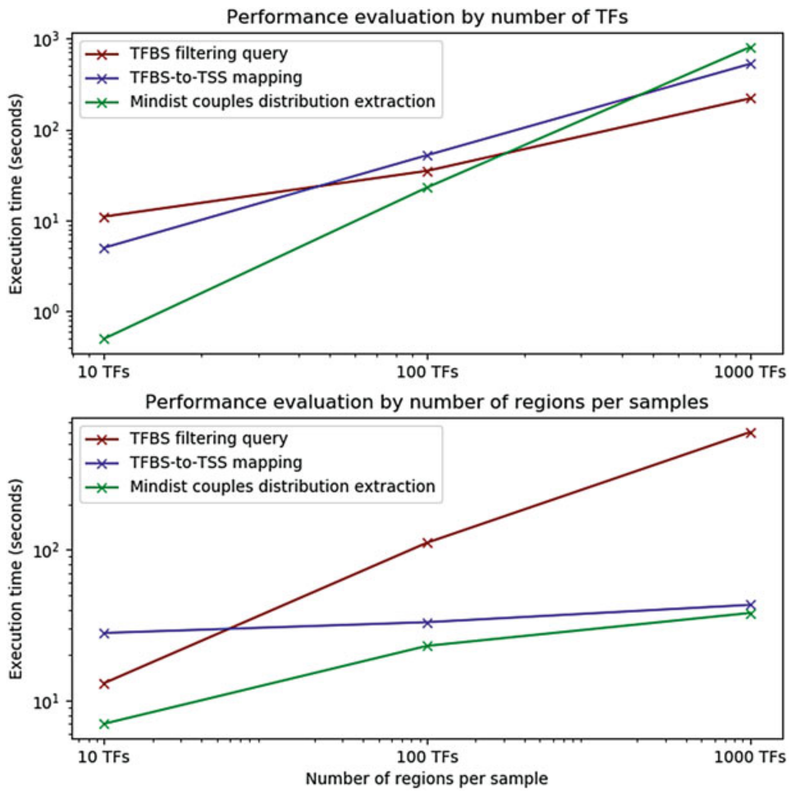


Fig. 6 Loglog scale graph of execution time for TICA on ENCODE datasets. Each line corresponds to one of the three algorithm steps timed as per Table 4. *Upper:* scaling with respect to the number of TF in a datasets, with fixed number of regions per sample; *lower:* scaling with respect to the number of region in a sample, with fixed number of TFs (and hence samples)

Note that each level (small, medium, large) increases the raw amount of data by a factor of 10, hence the increase in time is linear rather than exponential. To visualize this, we provide loglog plot of the scaling curves for TF- and sample size-scaling in Fig. 6. Note that TSS query filter time has not been timed in this scenario, as TSSes are not recomputed when user data is uploaded.

Baseline scenario is successfully computed in approx. 1 min, which is also the expected time for a typical user-provided dataset.

4.3.3 *Accuracy*

Briefly, we compare TICA predictions against existing biological knowledge, represented by two databases: CORUM [10], a collection of experimentally verified mammalian protein complexes, and BioGRID [11], which reports functional interactions between proteins based on both high-throughput datasets and individual focused studies. We consider an interaction to be supported by evidence if its two components are mentioned in a complex (CORUM) *or* as a protein–protein interaction (PPI, in BioGRID).

Table 5
Tabulation of quality measures for TICA predictions, with respect to the union of CORUM and BioGRID databases

Cell line	Recall	Specificity	Geometric mean performance	Enrichment
K562	0.297	0.848	0.502	1.95

Data from ENCODE cell line K562

The quality metrics that we use are *recall* (fraction of interactions correctly as positives out of all interaction supported by evidence), *specificity* (fraction of intersection not identified as positives out of all interactions which are not supported by evidence), and *geometric mean performance* (square root of the product between recall and specificity [1]). Results are tabulated in Table 5 for the largest cell line, K562.

A caveat is that not all TF-TF interactions correspond to complexes or PPIs (e.g., antagonistic TF-TF interactions), and not all complexes and PPIs correspond to TF-TF interactions. Nonetheless, co-operative TF-TF interactions are expected to be enriched in complexes and PPIs. This enrichment can be computed as recall over 1 minus specificity, which evaluates to 1.95 in our specific example. That is, a TF-TF pair that is predicted by TICA to interact is twice as likely to be found in a complex or as a PPI than a pair that is predicted not to interact.

5 Discussion

In this work, we introduce the TICA web server, a convenient tool for biologists to analyze ChIP-seq data on TF bindings for TF-TF interaction prediction. TICA leverages on GMQL, a novel language for data management, integration and querying of large, heterogeneous genomic datasets. Through the TICA web server, one can easily appreciate the expressive power and ease of use of the GMQL query language.

The TICA web server is a compact tool which nonetheless allows for fast analysis of entire cell lines from ENCODE ChIP-seq experiments: once data is generated (typically only after a major ENCODE release), running the prediction algorithm on repository data is computed in a short execution time. Updating the repositories with novel data has also very reasonable time requirements, considering that a repository's update rarely occurs (the cell line with the most data available, K562, takes about 16 h from start to finish on the server specified in Subheading 4.1).

TICA scales very well with increasing data size provided by the user: as shown in Fig. 6, it exhibits a linear or close to linear increase with respect to both the number of regions available in each samples and the number of TFs (samples) in the user provided datasets. This gives us confidence in saying that TICA can be used as a component of larger pipelines in the investigation of TF-TF interactions.

When cross-checked with popular protein–protein interaction (PPI) and protein complex databases, TICA shows very good specificity ($\geq$ 80%) while maintaining acceptable recall (circa 30%), considering that these reference datasets are currently incomplete. Given these quality measures, TICA can be used both as an effective screening tool in preparation for wet-lab experiments and as direct computational tool for investigating the interaction between novel transcription factors and/or experiments in specific conditions, such as disease or different cell lines.

Thanks to the expressive power of GMQL, the user is not required to pre-process data or convert it to any particular schema: peaks called in the standard narrowpeak format are sufficient to perform analysis, and are reduced to their point-source form directly by the query tool. Also, the TICA web server supports a high level of customization, allowing investigator to tune almost every parameter of the prediction algorithm without any loss of performance with respect to what has been mentioned above. In conclusion, we suggest the TICA web server as a compact, reliable, and efficient tool for tackling the TF-TF interaction prediction problem.

Acknowledgements

This work was supported by the ERC Advanced Grant *GeCo* (Data-Driven Genomic Computing) (Grant No. 693174) awarded to Prof. Stefano Ceri. We would like to thank members of the GeCo project for helpful insights. Prof. Limsoon Wong was supported in part by a Kwan-Im-Thong-Hood-Cho-Temple chair professorship.

References

1. Batuwita R, Palade V (2012) Adjusted geometric-mean: a novel performance measure for imbalanced bioinformatics datasets learning. J Bioinform Comput Biol 10(04):1250003
2. Codd EF (1970) A relational model of data for large shared data banks. Commun ACM 13(6):377–387
3. Geisel N, Gerland U (2011) Physical limits on cooperative protein-DNA binding and the kinetics of combinatorial transcription regulation. Biophys J 101(7):1569–1579
4. Jankowski A, Szczurek E, Jauch R, Tiuryn J, Prabhakar S (2013) Comprehensive prediction in 78 human cell lines reveals rigidity and compactness of transcription factor dimers. Genome Res 23(8):1307–1318
5. Kaitoua A, Pinoli P, Bertoni M, Ceri S (2017) Framework for supporting genomic operations. IEEE Trans Comput 66(3): 443–457
6. Landt SG, Marinov GK, Kundaje A, Kheradpour P, Pauli F, Batzoglou S, Bernstein BE, Bickel P, Brown JB, Cayting P et al (2012) Chip-seq guidelines and practices of the encode and modencode consortia. Genome Res 22(9):1813–1831

7. Masseroli M, Pinoli P, Venco F, Kaitoua A, Jalili V, Palluzzi F, Muller H, Ceri S (2015) Genometric query language: a novel approach to large-scale genomic data management. Bioinformatics 31(12):1881–1888
8. Masseroli M, Kaitoua A, Pinoli P, Ceri S (2016) Modeling and interoperability of heterogeneous genomic big data for integrative processing and querying. Methods 111:3–11
9. Nanni L (2017) A python data analysis library for genomics and its application to biology. Master's thesis, Politecnico di Milano - DEIB. Available at https://www.politesi.polimi.it/handle/10589/135989-
10. Ruepp A, Waegele B, Lechner M, Brauner B, Dunger-Kaltenbach I, Fobo G, Frishman G, Montrone C, Mewes HW (2009) Corum: the comprehensive resource of mammalian protein complexes—2009. Nucleic Acids Res 38(suppl_1):D497–D501
11. Stark C, Breitkreutz BJ, Reguly T, Boucher L, Breitkreutz A, Tyers M (2006) Biogrid: a general repository for interaction datasets. Nucleic Acids Res 34(suppl_1):D535–D539

Chapter 7

Multiple Testing Tool to Detect Combinatorial Effects in Biology

Aika Terada and Koji Tsuda

Abstract

Detecting combinatorial effects is important to various research areas, including biology, genomics, and medical sciences. However, this task was not only computationally nontrivial but also extremely difficult to achieve because of the necessity of a multiple testing procedure; hence few methods can comprehensively analyze high-order combinations. Recently, Limitless Arity Multiple-testing Procedure (LAMP) was introduced, allowing us to enumerate statistically significant combinations from a given dataset. This chapter provides instructions for LAMP using simple examples of combinatorial transcription factor regulation discovery and visualization of the results. This chapter also introduces LAMPLINK, which is extended software of LAMP. LAMPLINK can handle genetic dataset to detect statistically significant interactions among multiple SNPs from a genome-wide association study (GWAS) dataset.

Key words LAMP, Statistically significant combination, Multiple testing correction, Transcription factors, GWAS, Epistasis

1 Introduction

Understanding collaborative effects, such as combinatorial regulations by different transcription factors (TFs) [1, 2] and epistatic interactions among multiple loci [3, 4], is essential to make advancements in biology and medical sciences. However, few combinations have been discovered because of the difficulty in overcoming two problems simultaneously: statistical assessment and computational complexity. When we detect statistically significant factors from given multiple factors, a multiple testing correction must be applied [5]. One of the most widely used methods is Bonferroni correction [6], which multiplies the raw P-value by the number of tests. The Bonferroni correction theoretically controls the family-wise error rate (FWER) that is the probability that at least one false discovery happens in multiple tests. If it is applied to find the significant combinations from all possible combinations of up to k factors, the number of tested combinations increases exponentially to k, and the discovery of higher-order combinations is extremely unlikely. Moreover, even with the use of a supercomputer, it is difficult to enumerate the possible patterns.

Recently, statistical pattern mining methods such as Limitless Arity Multiple-testing Procedure (LAMP) [7] and its subsequent studies [8–10] have been developed to overcome these problems. These methods allow us to enumerate statistically significant combinations that are associated with an outcome variable. In this chapter, we first describe instructions for the use of LAMP, with application to combinatorial regulatory element discovery. Then, we introduce its extended software, LAMPLINK [11], that can be used for genome-wide association study (GWAS).

LAMP is available from http://a-terada.github.io/lamp/ and is written in Python, except for an external data mining software that is written in C [12] and runs on Linux, Mac, and Windows. LAMPLINK is available at http://a-terada.github.io/lamplink/ and is written in C++ and also runs on Linux, Mac, and Windows.

2 LAMP

LAMP is a software for multiple testing correction to detect combinatorial effect. For example, when provided with the relationships between transcription factors (TFs) and genes and gene expression levels, LAMP lists statistically significant combinations of TFs. LAMP also can be used for the following analyses: (1) combinations of single-nucleotide polymorphisms (SNPs) associated with a phenotype, (2) combinations of miRNAs that regulate gene expression, and (3) combinations of histone acetylations and methylations that influence gene transcription. The LAMP algorithm calibrates the FWER below a given threshold. Applying this method to analysis instead of using the Bonferroni correction gives the following advantages: (1) listing of significant combinations without an arity limit, (2) an increased sensitivity in comparison with the Bonferroni correction, and (3) a fast calculation speed.

Here, to provide an example of how LAMP may be applied, we detect combinations of TFs that regulate gene expression profiles. Application of LAMP to GWAS analysis using the LAMPLINK software is presented in Subheading 3.

2.1 Usage

LAMP runs by using the following command:

```
$ python lamp.py -p [p-value-procedure] \
  [item-file] [value-file] [significance-level] > [output-file]
```

LAMP requires four arguments and at least one option. The `[item-file]` and `[value-file]` indicate the input files. The result is saved to the `[output-file]`. The input and output file formats are described in Subheadings 2.2 and 2.3, respectively. The

Table 1
LAMP options

Option	Description
-p {“fisher”, “chi”, “u_test”}	Select the P-value calculation procedure. “Fisher” (Fisher’s exact test), “chi” (chi-squared test), or “u_test” (Mann-Whitney U test) are available.
--alternative = {“greater,” “less,” “two.sided”}	Indicate which alternative hypothesis is used. Select “greater,” “less,” or “two.sided.” The default setting is “greater.”
--max_comb = [integer]	Set the maximum arity of the tested combinations. The default setting is no limit.
–e [log-file]	Change the filename to save the running progress. The default setting is lamp_log_[date]_[time].txt.

Bolded letters indicate the required option

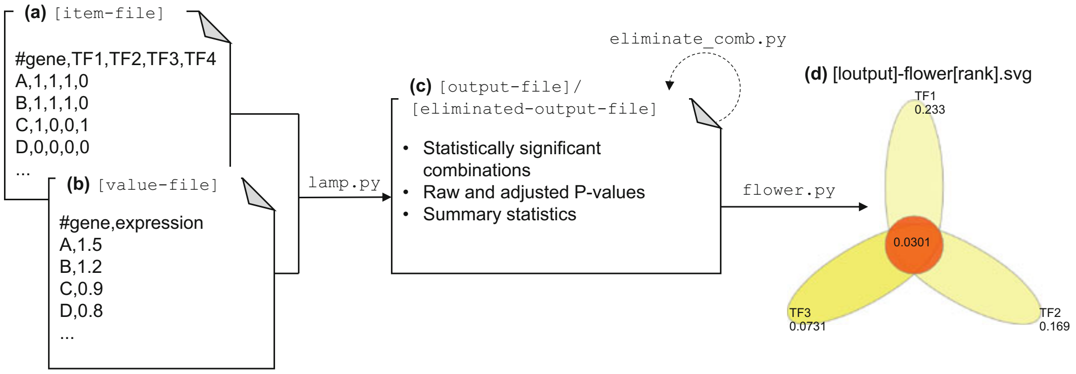


Fig. 1 Workflow to detect statistically significant combinations using LAMP. (**a**) and (**b**) Examples of [item-file] and [value-file] files, respectively. (**c**) Result file. The [output-file] is generated by lamp.py. When we run eliminate_comb.py for post-processing, the result appears in [eliminated-output-file] with the identical format to [output-file]. (**d**) Flower diagram representation generated using flower.py. The central circle represents the combination of TFs 1, 2, and 3. Each petal corresponds to a single TF. The color represents the P-values, and statistically significant ones are shown in red

option `-p` selects the P-value calculation procedure. All LAMP options are listed in Table 1.

2.2 Input File Formats

LAMP receives two input files: `[item-file]` and `[value-file]`.

The `[item-file]` provides the associations between TFs and their target genes. This file should be in the CSV format. An example of this file is presented in Fig. 1a. The first line provides the names of the TFs. Subsequent lines include the associations between each of the genes and TFs. The first column gives the gene name. The remaining columns indicate whether the TF targets that gene. If the gene is targeted by the TF, the column value is “1.” If it is not, then the column value is “0.” For example, gene A is the target gene of TF1, TF2, and TF3. When [item-file] contains M genes and N TFs, it consists of M + 1 lines and N + 1 columns.

The [value-file] provides the gene expression levels. This file is in the CSV format and has two columns: the gene name and the expression level. When LAMP derives the P-value using the Mann-Whitney U test, the expression levels are any real values. When Fisher's exact test or chi-squared test is used, the expression levels are either 1 if the gene is upregulated or 0 if it is not. An example of the content in this sample file is shown in Fig. 1b.

The genes listed in the [item-file] and [value-file] should be identical.

2.3 Output Format

LAMP first prints out the analysis settings, including the input file names, the P-value calculation method applied, and the numbers of TFs and the genes in the input data. LAMP then presents the adjusted significance level, the correction factor, and the number of significant combinations detected. The detected combinations are then presented, with each line containing the following seven columns:

- Rank: The rank ordered by the P-value.
- Raw P-value: The P-value calculated using the P-value procedure.
- Adjusted P-value: The adjusted P-value. The value of each combination contained in the LAMP output is smaller than the given significance level. Therefore, any combinations presented are statistically significant.
- Combination: The significant combination of TFs. The TFs are delimited by commas.
- Arity: The number of elements in the combination.
- # of target rows: The number of target genes of the TF combination.
- # of positives in the targets/z-score: The value used to compute the P-value. When Fisher's exact test was selected for the calculation, this value represents the number of genes that are both targeted and upregulated. When the Mann-Whitney U test was selected, this value is the z-score.

The last line presents information about the calculation time as follows: time (sec.): `Correction factor [float], P-value [float], and Total [float]`. The correction factor, P-value, and Total indicate the running time to compute the correction factor, to calculate the P-value of the combinations, and the total time required to perform these processes, respectively.

2.4 Post-processing

LAMP shows all significant combinations. However, for some applications, the raw result may be redundant because the results contain two very similar but slightly different combinations. To support the interpretation of such results, we prepared the script "eliminate_comb.py." This script selects the combination that is the most significant across all subsets of the combination. For example, suppose that the combinations A and B are both significant, and A has a smaller P-value than that of B. When A includes B or B includes A, the significance of B would be due to A. By using eliminate_comb.py, we remove combinations such as B from the results of LAMP. When the P-values are equal between A and B, the smaller combination is eliminated.

We perform this post-processing using the following command:

```
$ python eliminate_comb.py [output-file] \
  > [eliminated-output-file]
```

This command requires two filenames: `[output-file]` and `[eliminated-output-file]`. The former one is the result filename from LAMP. The later one is the filename to output the result after the elimination procedure.

This script prints "# Non-redundant combinations" in the first line of the `[lamp-output-file]`. The other lines are same as the `[lamp-output-file]`, except that redundant combinations are removed.

2.5 Flower Diagrams

LAMP includes a code for visualizing the detected combinations as flower diagrams, as presented in Fig. 1d. We generate the flower diagrams using the following command:

```
$ python flower.py [lamp-output-file].
```

The `[lamp-output-file]` is the result file of LAMP.

The flower diagrams are saved in SVG format files named `[lamp-output-file]-flower[rank].svg`. `[rank]` corresponds to the "Rank" column in `[lamp-output-file]`.

2.6 Demonstrations

Here, we present two small demonstrations of running LAMP. The first demonstration is a situation in which gene expression levels are given as numerical values. The other demonstration is a situation in which gene expression levels are represented in binary, such as upregulated/unregulated or presence/absence in a clustering result.

We use the following sample files:

- sample_item.csv: An example file of `[item-file]`.
- sample_expression_value.csv: A sample file of `[value-file]`. The gene expression levels are numerical values.
- sample_expression_over1.csv: A sample file of `[value-file]`. The gene expression levels are categorized into 1 or 0.

This dataset contains 15 genes and 4 TFs. All files are available from the LAMP website (http://a-terada.github.io/lamp/).

2.6.1 Demonstration 1: Mann-Whitney U Test

Here, we demonstrate LAMP using two files: sample_item.csv and sample_expression_value.csv.

The following command finds all of the significant combinations from the files with a significance level ≤ 0.05:

```
$ python lamp.py -p u_test \
 sample_item.csv sample_expression_value.csv 0.05 \
  > sample_u_test_result.txt
```

When LAMP finishes processing, the result is saved to sample_u_test_result.txt, which is presented in Fig. 2a.

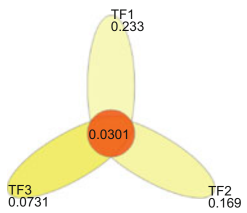
The metadata is presented in each of the lines starting with "#," including the following information:

- Line 1: The LAMP version used.
- Lines 2–5: The analysis settings.
- Line 6: The number of columns and rows in the `[item-file]` (sample_item.csv).
- Line 7: The adjusted significance level and the correction factor. When the raw P-value ≤ 0.01 (calculated by $0.05/5$), the combination is considered as statistically significant.
- Line 8: The number of significant combinations. We found one significant combination in this analysis.

The remaining lines present the results, including the following:

(a)

```
# LAMP ver. 2.0.0
# item-file: sample_item.csv
# value-file: sample_expression_value.csv
# significance-level: 0.05
# P-value computing procedure: u_test (greater)
# # of tested elements: 4, # of samples: 15
# Adjusted significance level: 0.01, Correction factor: 5 (# of target rows >= 3)
# # of significant combinations: 1
Rank     Raw p-value     Adjusted p-value     Combination     Arity     # of target rows     z-score
1        0.0060241       0.030121             TF1,TF2,TF3     3         5                    2.5107
Time (sec.): Computing correction factor 0.038, Enumerating significant combinations 0.000, Total 0.038
```



(b)

```
# LAMP ver. 2.0.0
# item-file: sample_item.csv
# value-file: sample_expression_over1.csv
# significance-level: 0.05
# P-value computing procedure: fisher (greater)
# # of tested elements: 4, # of samples: 15, # of positive samples: 7
# Adjusted significance level: 0.01, Correction factor: 5 (# of target rows >= 5)
# # of significant combinations: 1
Rank     Raw p-value     Adjusted p-value     Combination     Arity     # of target rows     # of positives in the targets
1        0.006993        0.034965             TF1,TF2,TF3     3         5                    5
Time (sec.): Computing correction factor 0.011, Enumerating significant combinations 0.000, Total 0.011
```

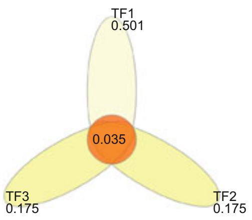


Fig. 2 Examples of LAMP results. (**a**) and (**b**) Results when Mann-Whitney U test and Fisher's exact test are used as the statistical tests, respectively. The left figure is the output of lamp.py for each test, and the right one is the corresponding flower diagram representation

- The ternary combination of TFs that includes TF1, TF2, and TF3 is significant. The raw P-value is 0.00602, and the adjusted P-value is 0.0301.
- The running time is 0.038 s in total.

The flower diagram in Fig. 2a shows the adjusted P-values for the significant combination. This diagram can be drawn in the SVG format file sample_u_test_result.txt-flower1.svg using the following command:

```
$ python flower.py sample_u_test_result.txt
```

2.6.2 Demonstration 2: Fisher's Exact Test

To demonstrate LAMP using Fisher's exact test, we use the two sample files: sample_item.csv and sample_expression_over1.csv. By changing the file "sample_expression_value.csv" to "sample_expression_over1.csv" and by changing "`-p u_test`" to "`-p fisher`" in the demonstration of Mann-Whitney U test, we can perform LAMP using Fisher's exact test. The following command finds all significant combinations from the files with a significance level ≤ 0.05:

```
$ python lamp.py -p fisher sample_item.csv \
  sample_expression_over1.csv 0.05 > sample_fisher_result.txt
```

When LAMP finishes processing, the result is saved to sample_fisher_result.txt as presented in Fig. 2b. The eight lines which start with "#" are similar to those from the Mann-Whitney U test result file in Fig. 2a. This file presents the results, including the following:

- The raw P-values of statistically significant combinations. In this case, combinations P-values ≤ 0.01 (calculated by $0.05/5$) are considered as statistically significant.
- There is one significant combination. The ternary combination of TFs that includes TF1, TF2, and TF3 is significant. The raw P-value is 0.00699, and the adjusted P-value is 0.0350.
- The running time is 0.011 s in total.

The diagram in Fig. 2b shows the adjusted P-values that are relevant to the statistically significant combination. This diagram can be drawn in sample_fisher_result.txt-flower1.svg using the following command:

```
$ python flower.py sample_fisher_result.txt
```

3 LAMPLINK

LAMPLINK [11] is a version of LAMP that is specific for analyzing genetics data and can detect statistically significant epistatic interactions of two or more SNPs from GWAS data. This software can be used in the same way as the widely used GWAS analysis

Table 2
LAMPLINK options

(a) Options to detect statistically significant SNP combinations	
Option	Description
--lamp	Detect combinatorial effect using LAMP
--file (or --bfile) [filename]	Input filename without the extension
--model-rec (or model-rec)	Select genetic model from dominant exclusive model (model-dom) or recessive exclusive model (model-rec)
--out [filename]	Output filename (default is "lamplink")
--fisher	Use Fisher's exact test as the statistical significance test (default is the chi-squared test)
--utest	Use Mann-Whitney U test as the statistical significance test (default is the chi-squared test)
--alternative {"greater," "less," "two.sided"}	Select which alternative hypothesis is used from "greater," "less," or "two.sided" (default is two.sided)
--ci [float]	Output confidence interval for CMH odds ratios
--sglev [float]	Set statistical significance level used in LAMP (default is 0.05)
--upper [float]	Set maximum MAF value (default is 0.1)
(b) Options to eliminate redundant SNP combinations	
Option	Description
--lamp-ld-remove	Eliminate SNP combinations in LD
--file (or --bfile) [filename]	Input filename without the extension
--comb < filename>	Filename generated by the —`lamp` option without the extension
--out [filename]	Output filename (default is "lamplink")
--lamp-r2	Set the threshold for the r^2 (default is 0.8)

Bolded letters indicate the required options

software, PLINK version 1.07 [13]. In addition to the functions of PLINK, LAMPLINK can detect epistatic interactions with LAMP. We can apply LAMPLINK to a PLINK analysis pipeline simply by replacing `plink` with `lamplink` and adding the `--lamp` option.

3.1 Usage

The use of LAMPLINK is identical to that of PLINK (http://zzz.bwh.harvard.edu/plink/), except that there are LAMPLINK-specific options. Here, we explain the LAMPLINK-specific options presented in Table 2.

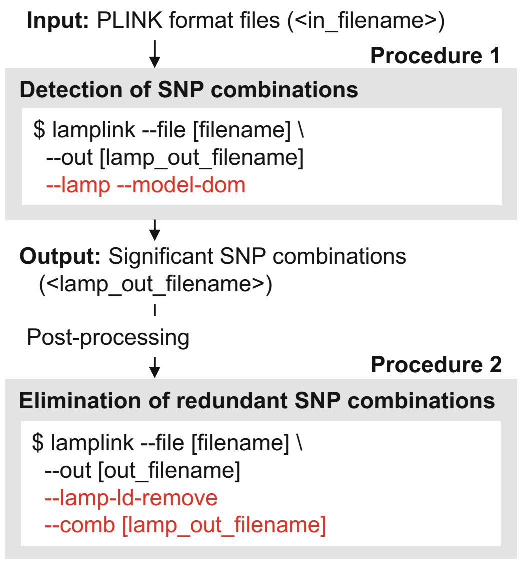

Input: PLINK format files (<in_filename>)
Procedure 1
Detection of SNP combinations
$ lamplink --file [filename] \
--out [lamp_out_filename]
--lamp --model-dom
Output: Significant SNP combinations (<lamp_out_filename>)
Post-processing
Procedure 2
Elimination of redundant SNP combinations
$ lamplink --file [filename] \
--out [out_filename]
--lamp-ld-remove
--comb [lamp_out_filename]

Fig. 3 Workflow to detect statistically significant SNP combinations using LAMPLINK

The `--lamp` option with `--model-dom` (or `--model-rec`) can be used to enumerate statistically significant SNP combinations (Procedure 1 in Fig. 3). Table 2(a) represents the relevant options. The input and output filenames are specified by the `--file` (or `--bfile` for binary format) and `--out` options, respectively. When `--model-dom` is used, LAMPLINK detects statistically significant combinations of SNPs according to a dominant exclusive model, while `--model-rec` uses a recessive exclusive model. The details of these two genetic models are described by Terada et al. [11]. LAMPLINK results are exported to the output files, `[lamp_out_filename].lamp` and `[lamp_out_filename].lamplink`. The output file formats are described in Subheading 3.3.

Like LAMP, LAMPLINK provides a set of options to eliminate redundant SNP combinations with the use of linkage disequilibrium (LD) region estimation. LAMPLINK may detect combinations of SNPs that are in the same LD region in Procedure 1 in Fig. 3. Because such combinations are redundant and can hinder the interpretation of SNP-phenotype associations, LAMPLINK provides the options, which are presented in Table 2(b), for filtering out these uninformative combinations (Procedure 2 in Fig. 3). Using `--lamp-od-remove` option eliminates SNP combinations whose members have an r^2 that is higher than the user-specified threshold, assuming that these SNP combinations are in the same LD region. If all the r^2 scores computed for the SNP pairs in each chromosome are higher than the threshold, then that SNP combination is removed. The results are saved to two files, `[out_filename].lamp` and `[lamp_out_filename].lamplink`.

3.2 Input File Formats

Input files should be set with the `--file` or `--bfile` option. For inputs using `--file [filename]`, LAMPLINK requires two files: `[filename].ped` and `[filename].map`. For inputs using `--bfile`, `[filename].bed`, `[filename].bim`, and `[filename].fam` files are required. For detailed descriptions of the required formats, please refer to the PLINK v1.07 instructions (http://zzz.bwh.harvard.edu/plink/).

3.3 Output File Formats

LAMPLINK reports its results in two output files, `[lamp_out_filename].lamp` and `[lamp_out_filename].lamplink`. The former file reports all SNP combinations that are statistically significantly associated with the phenotype. The latter file reports detailed information about each SNP in a format similar to the result generated by PLINK for association analysis.

The SNP combinations that are detected by LAMPLINK are summarized in the output file `[lamp_out_filename].lamp`. Each line represents an SNP combination and consists of four columns:

- COMBID: The combination ID corresponding to the COMBID in the [lamplink].lamplink file.
- Raw_P: The P-value of the SNP combination.
- Adjusted_P: The adjusted P-value calculated by LAMP.
- COMB: The SNPs that are members of the combination.

The `[lamp_out_filename].lamplink` file presents detailed information about each of the SNPs in the input dataset. Each line contains the following columns, irrespective of the statistical significance test applied:

- CHR: The chromosome number.
- SNP: The SNP name.
- A1 and A2: The names of the minor and major alleles.
- Test: The genetic model selected (DOM, dominant exclusive model; REC, recessive exclusive model). This output depends on the input option.
- AFF: The numbers of case individuals that have A1 and A2 alleles, respectively.
- UNAFF: The numbers of control individuals that have A1 and A2 alleles, respectively.
- P: The P-value of the SNP.
- OR: The odds ratio.
- COMB[ID]: Presence or absence of the SNP in the combination COMBID (presence: 1, absence: 0). When x combinations are detected, then x columns are generated.

When the chi-squared test is applied, this file contains the following additional columns:

- CHSQ: chi-squared score
- DF: degrees of freedom

When the `--ci` *x* option is used, the `[lamplink].lamplink` file contains the following additional columns:

- Lx: lower bound of *x*% confidence interval for the odds ratio
- Ux: upper bound of *x*% confidence interval for the odds ratio

3.4 Demonstrations

Here, we present a case-control analysis using LAMPLINK. The example dataset consists of two input files, as shown in Fig. 4a, b. The first one is lamplink_sample.map, which is an annotation file for each SNP locus being investigated. The other is lamplink_sample.ped, which provides sample information including phenotype and genotype.

(a) example.map

```
8          rs2631899   0          21169654
8          rs7817762   0          21164864
8          rs1841195   0          21190603
8          rs1564125   0          32080296
```

(b) example.ped

```
A   A      0    0    1    2    1 2    2 1    2 1    1 1
B   B      0    0    1    2    2 1    2 1    2 1    1 1
C   C      0    0    1    1    2 1    1 1    1 1    2 1
D   D      0    0    2    1    1 1    1 1    1 1    1 1
…
O   O      0    0    2    1    1 1    1 1    1 1    1 1
```

(c) example.lamp

```
COMBID    Raw_P       Adjusted_P  COMB
 COMB1    0.006993    0.034965    rs7817762,rs2631899,rs1841195
```

(d) example.lamplink

```
CHR   SNP         A1   A2   TEST   AFF   UNAFF   P           OR     COMB1
 8    rs7817762   2    1    DOM    5/2   1/7     0.0405594   17.5   1
 8    rs2631899   2    1    DOM    5/2   2/6     0.131935    7.5    1
 8    rs1841195   2    1    DOM    5/2   1/7     0.0405594   17.5   1
 8    rs1564125   2    1    DOM    2/5   4/4     0.608392    0.4    0
```

Fig. 4 Example files to conduct case-control analysis using LAMPLINK. (**a**) and (**b**) Input files that are given by the `--file` option. (**c**) and (**d**) Output files that are generated by LAMPLINK. The example.lamp file represents statistically significant combinations, and the example.lamplink file presents the details of the SNPs contained in the given dataset

The following command enumerates the statistically significant SNP combinations using Fisher's exact test. The significance level is set to 0.05.

```
$ ./lamplink --file ./example/lamplink_sample --lamp \
  model-dom   --sglev 0.05   --upper 0.5 \
  --out example --fisher
```

The LAMPLINK results are presented in example.lamp and example.lamplink output files. These are shown in Fig. 4c, d. Figure 4c indicates that one combination of SNPs (rs7817762, rs2631899, and rs1841195) is significantly associated with the phenotype after multiple testing correction. The raw and adjusted P-values are 0.006993 and 0.034965, respectively. Figure 4d presents detailed information about each of the SNPs in the dataset. Column P indicates the P-value for each SNP. If the value in COMB1 is 1, then this SNP is a member of the COMB1 SNP combination presented in Fig. 4c.

References

1. Baudry A, Heim MA, Dubreucq B et al (2004) TT2, TT8, and TTG1 synergistically specify the expression of BANYULS and proanthocyanidin biosynthesis in `Arabidopsis thaliana`. Plant J 39:366–380
2. Schlesinger J, Schueler M, Grunert M et al (2011) The cardiac transcription network modulated by Gata4, Mef2a, Nkx2.5, Srf, histone modifications, and microRNAs. PLoS Genet 7:e1001313
3. Carlborg O, Haley CS (2004) Epistasis: too often neglected in complex trait studies? Nat Rev Genet 5:618–625
4. Phillips PC (2008) Epistasis–the essential role of gene interactions in the structure and evolution of genetic systems. Nat Rev Genet 9:855–867
5. Noble WS (2009) How does multiple testing correction work? Nat Biotechnol 27:1135–1137
6. Bonferroni CE (1936) Teoria statistica delle classi e calcolo delle probabilità. Pubbl del R Ist Super di Sci Econ e Commer di Firenze 8:3–62
7. Terada A, Okada-hatakeyama M, Tsuda K, Sese J (2013) Statistical significance of combinatorial regulations. Proc Natl Acad Sci U S A 110:12996–13001
8. Terada A, Tsuda K, Sese J (2013) Fast Westfall-Young permutation procedure for combinatorial regulation discovery. In: 2013 IEEE International Conference on Bioinformatics and Biomedicine. pp 153–158
9. Sugiyama M, López FL, Kasenburg N, Borgwardt KM (2015) Significant subgraph mining with multiple testing correction. In: 2015 SIAM International Conference on Data Mining. pp 37–45
10. Llinares-López F, Sugiyama M, Papaxanthos L, Borgwardt K (2015) Fast and memory-efficient significant pattern mining via permutation testing. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp 725–734
11. Terada A, Yamada R, Tsuda K, Sese J (2016) LAMPLINK: detection of statistically significant SNP combinations from GWAS data. Bioinformatics 32:3513–3515
12. Uno T, Asai T, Uchida Y, Arimura H (2003) LCM: an efficient algorithm for enumerating frequent closed item sets. In: Workshop on Frequent Itemset Mining Implementations (FIMI'03)
13. Purcell S, Neale B, Todd-Brown K et al (2007) PLINK: a tool set for whole-genome association and population-based linkage analyses. Am J Hum Genet 81:559–575

Chapter 8

SiBIC: A Tool for Generating a Network of Biclusters Captured by Maximal Frequent Itemset Mining

Kei-ichiro Takahashi, David A. duVerle, Sohiya Yotsukura, Ichigaku Takigawa, and Hiroshi Mamitsuka

Abstract

Biclustering extracts coexpressed genes under certain experimental conditions, providing more precise insight into the genetic behaviors than one-dimensional clustering. For understanding the biological features of genes in a single bicluster, visualizations such as heatmaps or parallel coordinate plots and tools for enrichment analysis are widely used. However, simultaneously handling many biclusters still remains a challenge. Thus, we developed a web service named SiBIC, which, using maximal frequent itemset mining, exhaustively discovers significant biclusters, which turn into networks of overlapping biclusters, where nodes are gene sets and edges show their overlaps in the detected biclusters. SiBIC provides a graphical user interface for manipulating a gene set network, where users can find target gene sets based on the enriched network. This chapter provides a user guide/instruction of SiBIC with background of having developed this software. SiBIC is available at http://utrecht.kuicr.kyoto-u.ac.jp:8080/sibic/faces/index.jsp.

Key words Gene expression, Biclustering, Frequent itemset mining, Gene set network, Gene enrichment analysis

1 Introduction

Gene expression matrix (“genes” × “experimental conditions”) can be clustered by either of the two sides [1, 2], while expression patterns can usually be grouped with only part of rows or columns (neither the entire rows nor columns). This leads to the idea of biclusters, which consist of subgroups of rows and subgroups of columns.

In general, from an expression matrix, biclustering algorithms produce many biclusters [3, 4], causing a serious issue of visualizing them. To solve this issue, biclusters are visualized in many ways, such as heatmaps and parallel coordinate plots [5–8]. However they have limitation on scalability, particularly for many overlapping biclusters.

Another type of visualization is graph, which is more promising on handling many biclusters. Furby [9] displays overlapping biclusters as a graph, where a node corresponds to a heatmap (a bicluster itself) and edges correspond to rows and columns shared by two heatmaps. BicOverlapper [10] visualizes overlapping biclusters by a graph, in which each node represents a gene or a condition and edges are grouped by one or more biclusters. We developed SiBIC [11] by defining a weighted graph called as a *gene set network* on overlapping biclusters, where each node is a gene set derived from overlapping biclusters and each edge corresponds to the difference between two nodes. A gene set network removes duplications of genes, which share experimental conditions. This makes gene set networks more compact than Furby. Similarly, a gene set network is more compact (and scalable) than a network by BicOverlapper, because each node of gene set networks is a gene set, while each node of BicOverlapper is a single gene or a single condition. SiBIC also provides a GUI application for visualizing and manipulating gene set networks, to allow enrichment analysis in a more flexible manner than using only one bicluster.

In SiBIC, a bicluster is defined as genes that are coexpressed under each experimental condition. Figure 1b shows an example

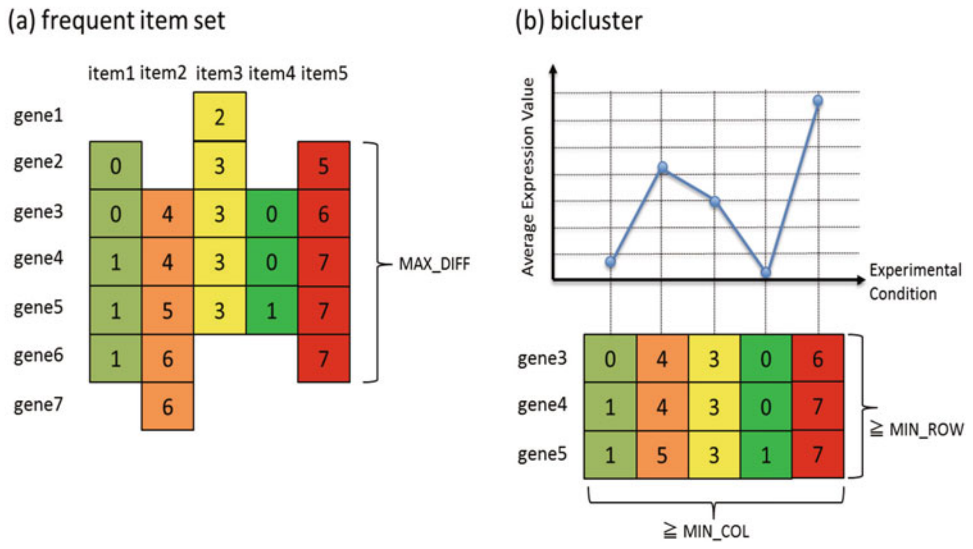


Fig. 1 (**a**) In SiBIC, genes having similar expression values within MAX_DIFF are dealt as one item. Note that items can have different numbers of genes. The figure assumes that each item is taken from different experimental conditions and the five items (item1 to item5) have the common genes (gene3 to gene5). The minimum number of common genes are specified by parameter MIN_ROW (in the bottom of the right figure (**b**)). SiBIC captures such items as a frequent itemset and enumerates all possible frequent itemsets from a given expression dataset. (**b**) The frequent itemset in (**a**) can be seen as a bicluster consisting of the common three genes and the five experimental conditions. This bicluster shows a coexpressed pattern in the top. MIN_ROW and MIN_COL are the parameters which specify the size of biclusters: minimum number of genes and minimum number of conditions, respectively

of such biclusters, where values are similar in each column. This bicluster reveals genes which express similarly under certain experimental conditions. To exhaustively find this type of biclusters from a given expression dataset, SiBIC employs frequent itemset mining (FIM) [12]. SiBIC regards every set of genes sharing similar expression values as one item, and frequent itemsets enumerated by FIM as biclusters. SiBIC generates biclusters from FIM and then a gene set network from biclusters [11]. This book chapter provides a comprehensive user instruction of SiBIC.

2 Materials

The input of SiBIC is gene expression data, which is a matrix, in which rows are genes and columns are experimental conditions. We describe the format of input files in Subheading 3.2.1. Note that SiBIC exhaustively enumerates all possible significant biclusters and so is not necessarily designed for dealing with large-scale expression datasets. We recommend that a subset of interest should be extracted from the original gene expression dataset.

3 Methods

3.1 Overview of SiBIC

Our method consists of roughly four steps: (1) enumerating all possible biclusters as frequent itemsets and assigning p-values to them, (2) merging the overlapping biclusters, removing their redundancy, (3) generating gene set networks from merged biclusters, and (4) analyzing gene functions by using the generated gene set networks. Figure 2(1) to (4) show a schematic flow of the above (1) to (4), respectively.

3.1.1 Enumerating Biclusters

Our approach produces multiple, overlapping biclusters by frequent itemset mining (FIM). SiBIC first aggregates genes with similar values into items per experimental condition, and then FIM (MAFIA [13]) is run on the database of all items. Figure 1a shows an explanatory example of a frequent itemset, which can be seen as a bicluster, as shown in Fig. 1b.

SiBIC computes empirical p-values to rank generated biclusters in terms of how significantly row vectors in biclusters are correlated. For each bicluster (with N genes and M experimental conditions), 500,000 matrices of the same size are randomly generated out of the input gene expression matrix. To generate an empirical distribution for a bicluster, SiBIC computes the following test statistic T over each random matrix:

$$T = \frac{1}{N}\sum_{i=1}^{N} \mathrm{corr}(g_i, \bar{g}), \text{ where } \bar{g} = \frac{1}{N}\sum_{i=1}^{N} g_i \qquad (1)$$

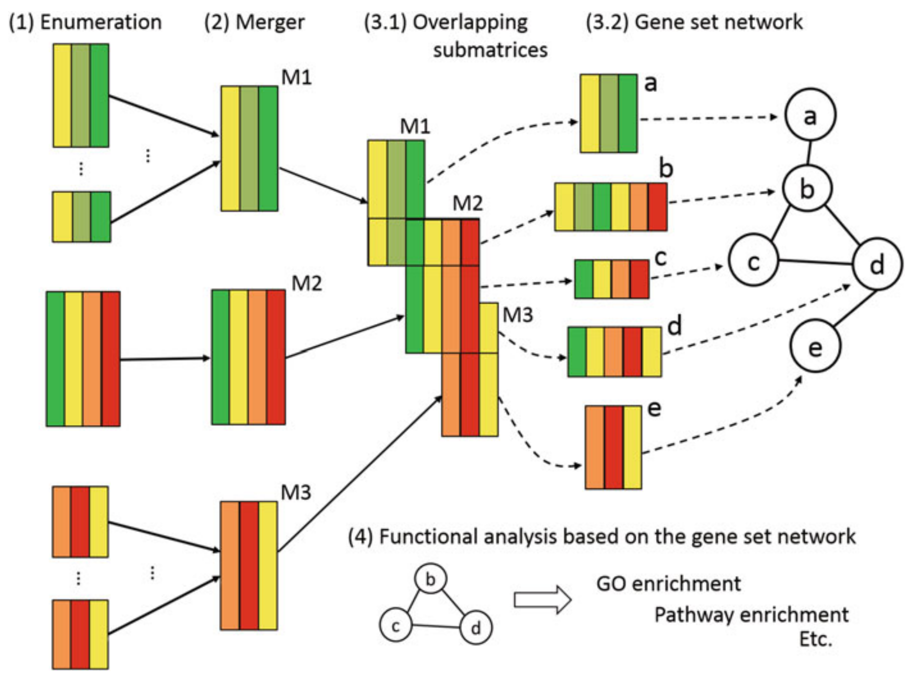


Fig. 2 A schematic flow of SiBIC: (1) Enumerating all biclusters by maximal frequent itemset mining, where each rectangle represents a maximal frequent itemset (i.e., a bicluster). Colors stand for similarities in expression values. (2) Merging overlapping biclusters with exactly the same conditions if they keep statistical significance. (3.1) and (3.2) Generating gene set networks from overlapping biclusters. Each node in the network indicates a newly redefined bicluster based on the overlapping submatrices and nodes from each bicluster form a complete subgraph. (4) Analyzing gene functions by using the obtained network

where g_i is an M-dimensional row vector and $corr(\cdot, \cdot)$ is Pearson correlation coefficient. SiBIC then calculates the T score of the bicluster to give its empirical p-value on the distribution.

3.1.2 Merging Biclusters

Maximal FIM enumerates all possible biclusters of the largest frequent itemsets, while they can be redundant in the sense that they can be still heavily overlapped with each other. SiBIC merges biclusters that have exactly the same experimental conditions, as long as the significance as computed in Eq. (1) is kept.

3.1.3 Gene Set Networks

To visualize the biclusters, we use *gene set networks*, each being a weighted graph, where a node corresponds to a coexpressed gene set and an edge indicates the difference of experimental conditions between two nodes. SiBIC generates gene set networks from overlapping biclusters as follows: (1) All genes in overlapping biclusters are first divided into disjoint gene sets so that respective sets are shared by the same biclusters. (2) SiBIC then treats each set as a node, and an edge connects two nodes if both nodes are taken from the same bicluster, where the weight of an edge is the number of biclusters containing the genes of the two nodes of the

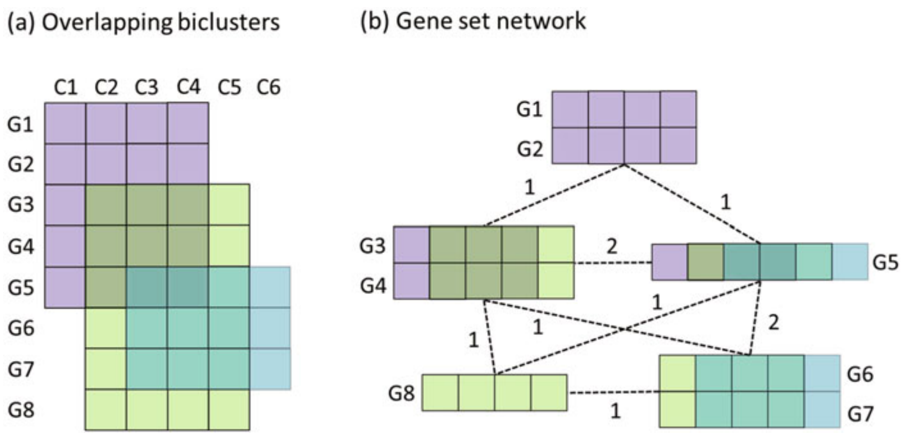


Fig. 3 Construction of a gene set network: (**a**) Three biclusters are overlapping with each other, for eight genes (G1 to G8) and six conditions (C1 to C6). (**b**) The three overlapping biclusters in (**a**) are converted into a graph with five nodes, according to the overlapping submatrices. Each node consists of genes shared by the same biclusters, resulting in a newly redefined bicluster with relevant conditions. If two nodes are from the same bicluster, they are connected by an edge weighted by the number of biclusters containing the genes of both nodes

edge. Figure 3 shows an explanatory example of building a gene set network (b) from overlapping biclusters (a). From Fig. 3b, we can easily see that each node represents a newly redefined bicluster with an expression pattern.

A gene set network has the following three properties:

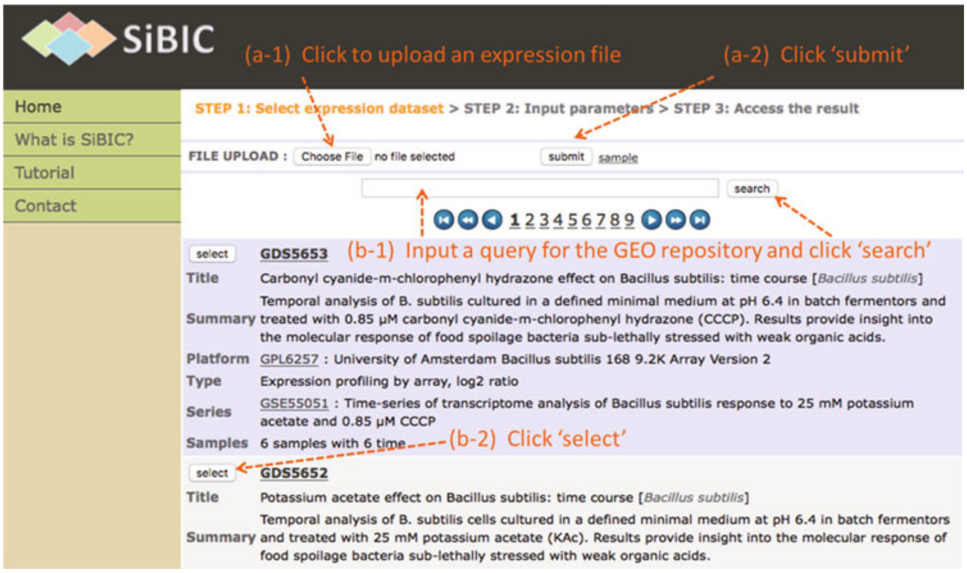
1. Reversibility: a gene set network exhaustively keeps overlapping bicluster information, by which the original biclusters can be reproduced from a gene set network.
2. Compactness: each node is a gene set (a bicluster), by which a gene set network is more compact than usual gene networks.
3. Interpretability: an edge corresponds to the difference between two node of this edge, by which nodes can be interpreted as coexpressed gene units.

3.2 Web Service

In this section, we describe the usage of SiBIC (*see* **Note 1**), which has two steps: (1) selecting an expression dataset, (2) inputting parameters. In the first step, an expression dataset is uploaded or a SOFT file is selected from the GEO repository [14], which will be described in Subheading 3.2.1. In the second step, Subheading 3.2.2 describes parameters details. Subheadings 3.2.3 and 3.2.4 explain how to analyze and interpret biclusters and gene set networks.

3.2.1 Expression Dataset

The top page of SiBIC provides the following two interfaces to send an expression dataset to the server:


SiBIC
(a-1) Click to upload an expression file
(a-2) Click 'submit'
Home
What is SiBIC?
Tutorial
Contact
STEP 1: Select expression dataset > STEP 2: Input parameters > STEP 3: Access the result
FILE UPLOAD : Choose File no file selected
submit sample
search
1 2 3 4 5 6 7 8 9
(b-1) Input a query for the GEO repository and click 'search'
(b-2) Click 'select'
select

GDS5653

Title	Carbonyl cyanide-m-chlorophenyl hydrazone effect on Bacillus subtilis: time course [*Bacillus subtilis*]
Summary	Temporal analysis of B. subtilis cultured in a defined minimal medium at pH 6.4 in batch fermentors and treated with 0.85 μM carbonyl cyanide-m-chlorophenyl hydrazone (CCCP). Results provide insight into the molecular response of food spoilage bacteria sub-lethally stressed with weak organic acids.
Platform	GPL6257 : University of Amsterdam Bacillus subtilis 168 9.2K Array Version 2
Type	Expression profiling by array, log2 ratio
Series	GSE55051 : Time-series of transcriptome analysis of Bacillus subtilis response to 25 mM potassium acetate and 0.85 μM CCCP
Samples	6 samples with 6 time

GDS5652

Title	Potassium acetate effect on Bacillus subtilis: time course [*Bacillus subtilis*]
Summary	Temporal analysis of B. subtilis cells cultured in a defined minimal medium at pH 6.4 in batch fermentors and treated with 25 mM potassium acetate (KAc). Results provide insight into the molecular response of food spoilage bacteria sub-lethally stressed with weak organic acids.

Fig. 4 The top page of SiBIC: a user can upload a local expression file or select a SOFT file from the GEO repository as the input of SiBIC

File uploading interface (Fig. 4a):

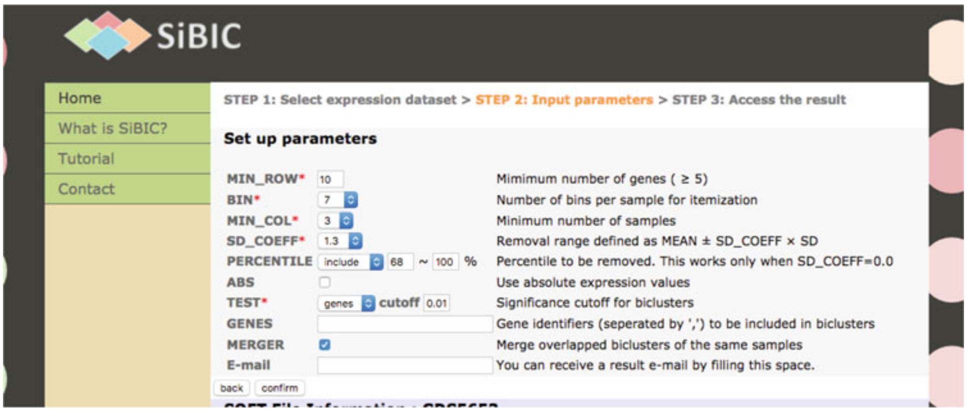
A local plain text file with the extension .txt or .text can be uploaded by clicking the "Choose File" button to pick up a local file and then clicking the "submit" button. The file must be a tab-delimited file, where the maximum file size is 15 MB and the file format must have sample identifiers in the first line and a gene identifier followed by expression values in the following lines. Lines starting with a symbol "#" or "!" are ignored. Note that no identifier can start with a symbol "#" or "!". Missing expression values can be entered as null or na.

GEO dataset search interface (Fig. 4b):

A query in the text field can be typed and the "search" button can be clicked to list the matched SOFT files from the GEO database [14]. Then the "select" button can be clicked to obtain the corresponding SOFT file. Query syntax should follow ESearch format [15], where a space should be replaced by a plus sign "+" if required in a query term (*see* **Note 2**).

3.2.2 Parameters

Submitting the input expression dataset directs to the parameter setting page. Figure 5 shows a screenshot of the page, where MIN_ROW, MIN_COL, and MAX_DIFF are the main three parameters, which define biclusters. MIN_ROW and MIN_COL specify the minimum size of biclusters, and MAX_DIFF gives the maximum range of values in each column. It is not easy for users to decide MAX_DIFF, because a proper range depends upon experimental conditions. Thus, instead of MAX_DIFF, SiBIC has



STEP 1: Select expression dataset > STEP 2: Input parameters > STEP 3: Access the result

Set up parameters

MIN_ROW*	10	Mimimum number of genes (≥ 5)
BIN*	7	Number of bins per sample for itemization
MIN_COL*	3	Minimum number of samples
SD_COEFF*	1.3	Removal range defined as MEAN ± SD_COEFF × SD
PERCENTILE	include 68 ~ 100 %	Percentile to be removed. This works only when SD_COEFF=0.0
ABS	☐	Use absolute expression values
TEST*	genes cutoff 0.01	Significance cutoff for biclusters
GENES		Gene identifiers (seperated by ',') to be included in biclusters
MERGER	☑	Merge overlapped biclusters of the same samples
E-mail		You can receive a result e-mail by filling this space.

Fig. 5 The parameter setting page of SiBIC: After selecting a dataset, SiBIC leads the user to this page, where the user can specify a set of parameters such as MIN_ROW, BIN and MIN_COL

BIN as a parameter to easily compute MAX_DIFF. Below are all parameters to be specified in the parameter input interface of SiBIC.

1. MIN_ROW: specifies the minimum number of genes in biclusters. Computation time becomes heavier as MIN_ROW is smaller, because the number of frequent itemsets (i.e., biclusters) is larger. An integer of 5 or larger is the possible input. The default value is 10.
2. BIN: defines MAX_DIFF, which is (MAX-MIN)/BIN, where MAX and MIN are the maximum and minimum expression values per experimental condition, respectively. SiBIC handles the combination of (1) a gene set with values within MAX_DIFF and (2) an experimental condition as an item for maximal FIM. Note that assuming that some Gaussian distribution over expression values, the number of genes in one item is larger as the MAX_DIFF is wider, particularly including the mean expression value. To remove outliers, SiBIC internally modifies the distribution's tails, by replacing all expression values falling outside of $(\hat{\mu} - 3\hat{\sigma}, \hat{\mu} + 3\hat{\sigma})$ with $\mu \pm 3\hat{\sigma}$, by which MAX = $\hat{\mu} + 3\hat{\sigma}$, MIN = $\hat{\mu} - 3\hat{\sigma}$, and MAX_DIFF = $6\hat{\sigma}$ / BIN. Note that smaller BIN allows larger biclusters with less similarity in expression values and more experimental conditions. The default value is 7.
3. MIN_COL: specifies the minimum number of experimental conditions in biclusters. After running maximal FIM and before computing p-values, biclusters with a smaller number of experimental conditions than MIN_COL are filtered out. The default value is 3.
4. SD_COEFF: is used to remove expression values (for each column) with only little changes and biologically insignificant. Genes with expression values within SD_COEFF×SD

(SD:standard deviation) are removed. Alternatively, a user can set SD_COEFF=0.0 and specify a range of interest by PERCENTILE.

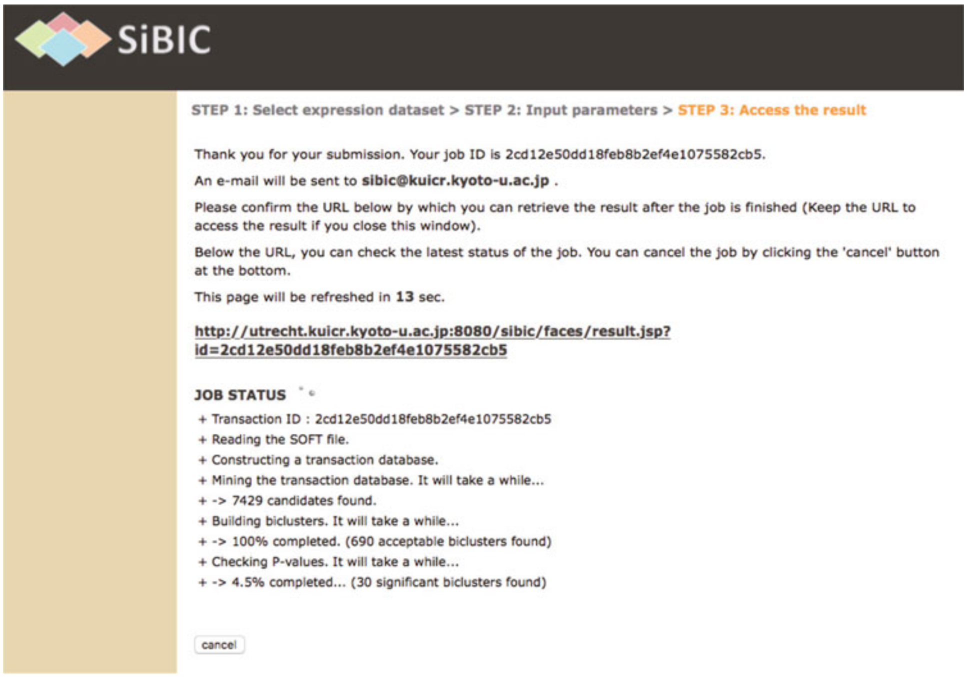
5. ABS: is a boolean parameter to treat expression values as absolute values. The default value is "false" (unchecked in the check button).
6. TEST: is a parameter to choose a method for computing *p*-values, out of three choices:

   "genes": *p*-values are computed in terms of how much genes in a bicluster are correlated each other by Eq. (1).
   "conds": *p*-values are computed in terms of how much experimental conditions in a bicluster are correlated each other.
   "both": both genes and conditions are used.

   When TEST is set, the cutoff value for *p*-values can also be specified. The default setting is "genes" with the cut-off value of 0.01.
7. GENES: is a parameter to specify genes of interest. The input is identifiers (separated by ",") of genes, which should be included in biclusters. If this parameter is used, biclusters without the specified genes are ignored, making the computation far faster.
8. MERGER: is a boolean parameter to skip the merging process. The default value is "true" (checked in the check button).

Another possible input is an email address to receive a notification e-mail immediately after the result is obtained. After inputting all parameters and clicking "confirm", "run" can be clicked if there are no problems on the input parameter values; otherwise click "back" to go back to the parameter input interface to input parameter values again. By clicking "run" in the confirmation page, SiBIC moves to the running status page as shown in Fig. 6, where the status of computation (*see* **Note 3**) can be shown. In the running status page, the "cancel" button to cancel the current job can be clicked. The URL to the page showing results, provided on the running status page, is recommended to be saved as a bookmark. Note that this URL is the only way to access the result, if the email address has not been given in the parameter setting page.

Practically several different parameter sets are recommended to be tried to find a good balance to obtain favorable biclusters without spending much computation time. Computation time directly depends on the number of biclusters to be generated, which further depends on two factors: (1) the size of items (the number of genes in each item) and (2) the number of items (the size of an expression matrix). We mention a couple of points regarding them below.



STEP 1: Select expression dataset > STEP 2: Input parameters > STEP 3: Access the result

Thank you for your submission. Your job ID is 2cd12e50dd18feb8b2ef4e1075582cb5.

An e-mail will be sent to **sibic@kuicr.kyoto-u.ac.jp** .

Please confirm the URL below by which you can retrieve the result after the job is finished (Keep the URL to access the result if you close this window).

Below the URL, you can check the latest status of the job. You can cancel the job by clicking the 'cancel' button at the bottom.

This page will be refreshed in **13** sec.

http://utrecht.kuicr.kyoto-u.ac.jp:8080/sibic/faces/result.jsp?id=2cd12e50dd18feb8b2ef4e1075582cb5

JOB STATUS

```
+ Transaction ID : 2cd12e50dd18feb8b2ef4e1075582cb5
+ Reading the SOFT file.
+ Constructing a transaction database.
+ Mining the transaction database. It will take a while...
+ -> 7429 candidates found.
+ Building biclusters. It will take a while...
+ -> 100% completed. (690 acceptable biclusters found)
+ Checking P-values. It will take a while...
+ -> 4.5% completed... (30 significant biclusters found)
```

Fig. 6 The running status page of SiBIC: In this page, the user can check the running status and cancel the current job

1. Size of items: Simply smaller BIN (the larger size of items) can generate a larger number of biclusters, taking longer computation time. Also with a larger number of genes in one item (which results in being less similar each other), candidate biclusters can be less statistically significant, despite longer computation time (since the number of experimental conditions of biclusters can be larger). Thus several values of MIN_ROW and BIN should be tried, where a larger value of MIN_ROW results in less computation time, and a larger value of BIN can also reduce the computation time by making the size of items smaller and expression values more similar. The progress of the running job can be checked through the running status page (Fig. 6). If the progress is very slow, the job can be canceled to try another parameter set (*see* **Note 4**).
2. Number of items: SiBIC produces $O(MN)$ items for M genes and N experimental conditions. The search space of FIM is larger due to a larger number of items, meaning that a larger expression dataset is heavier in computation. A simple way of making an expression dataset smaller is to remove genes by making SD_COEFF larger (or PERCENTILE). Also another way is to specify particular genes of interest by using GENES, by which SiBIC focuses on only biclusters with those genes. Furthermore, if the expression dataset has replicates of

experimental conditions or genes, they should be removed (by taking the average over the replicated, etc.) to make the input file smaller (*see* **Note 5**).

3.2.3 Biclusters

The result page shown in Fig. 7a can be accessed through the bookmark link or the link in the notification email mentioned in Subheading 3.2.2. The result page has two parts: “Bicluster Information” and “Gene Set Networks for Overlapping Biclusters.” Below we explain “Bicluster Information” and “Gene Set Networks for Overlapping Biclusters” will be explained in the next section.

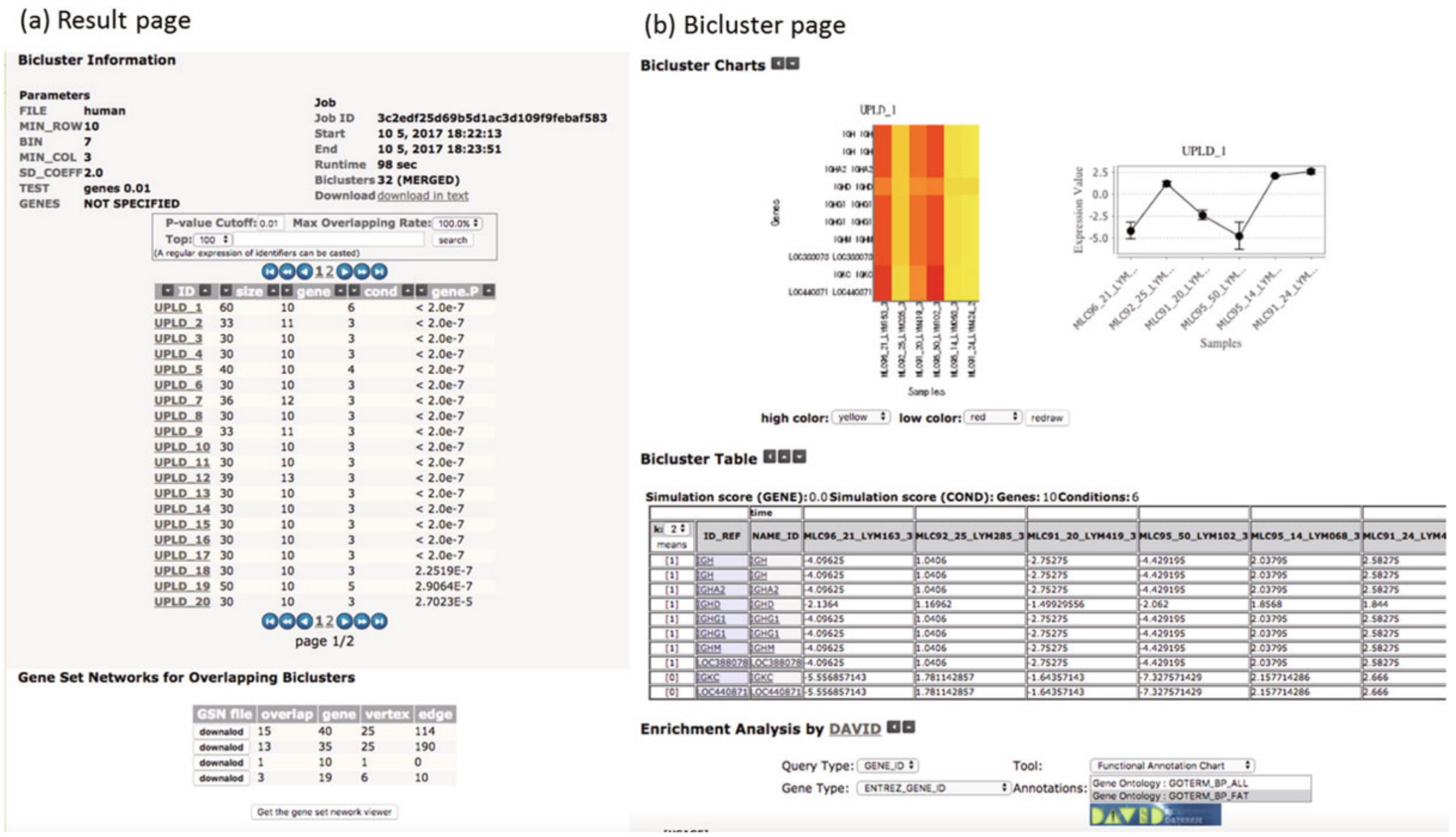
“Bicluster Information” shows a table of biclusters with their sizes, heights, widths and *p*-values, where biclusters can be shown by clicking a triangle icon right beside the column titles. Also, the size of biclusters can be adjusted by the “Top” dropbox or by searching biclusters with genes specified in the query box right above the table. An individual page for the bicluster shown in Fig. 7b can be accessed by clicking each ID in the table. The individual page contains a heatmap, a line chart, and a numerical table of the bicluster. Moreover, SiBIC provides an interface for enrichment analysis by using DAVID (Database for Annotation, Visualization and Integrated Discovery) [16] on the bottom of the page. Enrichment analysis can be done for the bicluster, through this interface, as follows:

1. Select a proper “Query Type” out of ID_REFF, NAME_ID, ACC, and GENE_ID each of which corresponds to a column of the bicluster table above.
2. Select a proper “Gene Type” defined in DAVID such as ENTREZ_GENE_ID, which must be a whole background containing “Query Type.”
3. Select a “Tool” such as “Functional Annotation Chart” for annotation analysis.
4. Select “Annotations” such as GOTERM_BP_FAT and KEGG_PATHWAY, and then click the DAVID logo. If “Query Type” and “Gene Type” are a correct combination, then SiBIC opens a DAVID page to show the result.

3.2.4 Gene Set Network Viewer

“Gene Set Networks for Overlapping Biclusters” shows a table of gene set networks. In this table, the “GNS file” column allows to download `.gns` files to run with the GUI application to be described below. The “overlap” column shows the number of overlapping biclusters from which a gene set network is made, the “genes” column shows the number of genes in the network, the “vertex” column shows the number of nodes, and the “edge” column shows the number of edges.

Figure 8 shows the GUI application for displaying gene set networks which enables to conduct enrichment analysis in a more



(a) Result page

Bicluster Information

Parameters	
FILE	human
MIN_ROW	10
BIN	7
MIN_COL	3
SD_COEFF	2.0
TEST	genes 0.01
GENES	NOT SPECIFIED

Job	
Job ID	3c2edf25d69b5d1ac3d109f9febaf583
Start	10 5, 2017 18:22:13
End	10 5, 2017 18:23:51
Runtime	98 sec
Biclusters	32 (MERGED)
Download	download in text

P-value Cutoff: 0.01 Max Overlapping Rate: 100.0%
Top: 100
(A regular expression of identifiers can be casted)

ID	size	gene	cond	gene.P
UPLD_1	60	10	6	< 2.0e-7
UPLD_2	33	11	3	< 2.0e-7
UPLD_3	30	10	3	< 2.0e-7
UPLD_4	30	10	3	< 2.0e-7
UPLD_5	40	10	4	< 2.0e-7
UPLD_6	30	10	3	< 2.0e-7
UPLD_7	36	12	3	< 2.0e-7
UPLD_8	30	10	3	< 2.0e-7
UPLD_9	33	11	3	< 2.0e-7
UPLD_10	30	10	3	< 2.0e-7
UPLD_11	30	10	3	< 2.0e-7
UPLD_12	39	13	3	< 2.0e-7
UPLD_13	30	10	3	< 2.0e-7
UPLD_14	30	10	3	< 2.0e-7
UPLD_15	30	10	3	< 2.0e-7
UPLD_16	30	10	3	< 2.0e-7
UPLD_17	30	10	3	< 2.0e-7
UPLD_18	30	10	3	2.2519E-7
UPLD_19	50	10	5	2.9064E-7
UPLD_20	30	10	3	2.7023E-5

page 1/2

Gene Set Networks for Overlapping Biclusters

GSN file	overlap	gene	vertex	edge
downalod	15	40	25	114
downalod	13	35	25	190
downalod	1	10	1	0
downalod	3	19	6	10

(b) Bicluster page

Bicluster Table

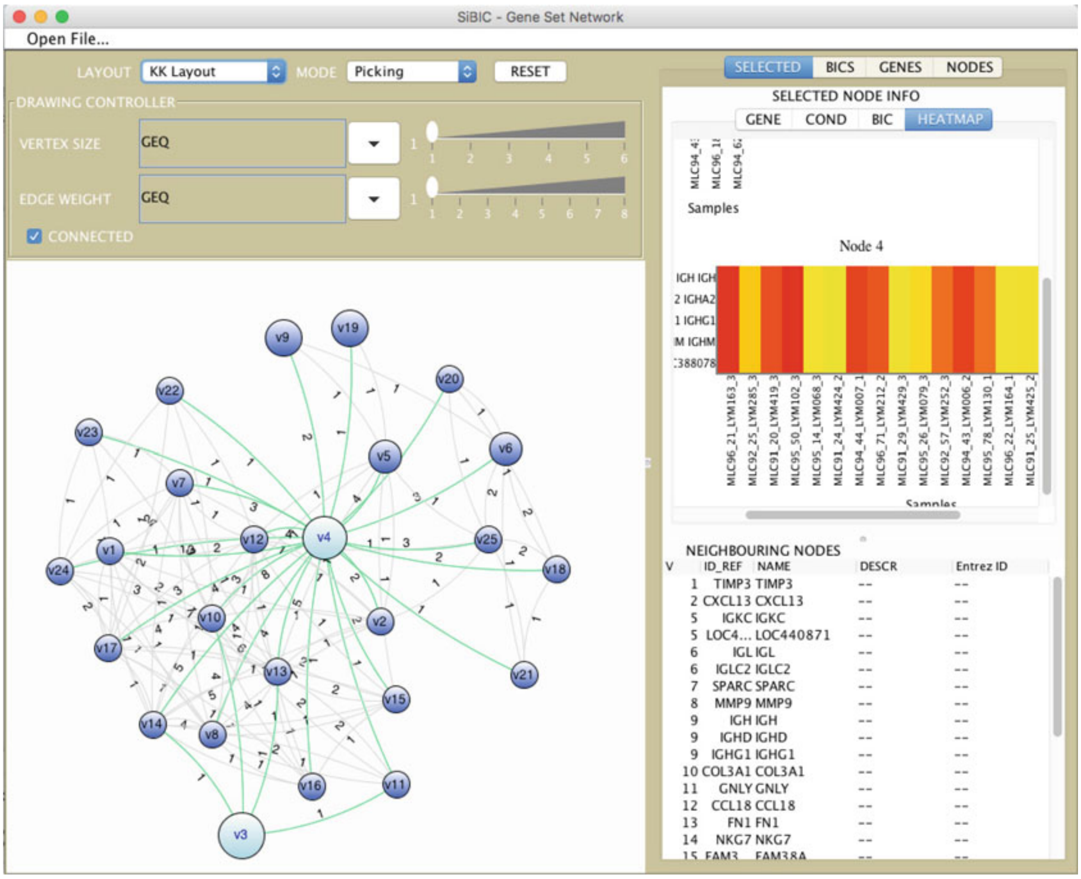
Simulation score (GENE): 0.0 **Simulation score (COND): Genes:** 10 **Conditions:** 6

		time						
k: 2 means	ID_REF	NAME_ID	MLC96_21_LYM163_3	MLC92_25_LYM285_3	MLC91_20_LYM419_3	MLC95_50_LYM102_3	MLC95_14_LYM068_3	MLC91_24_LYM4
[1]	IGH	IGH	-4.09625	1.0406	-2.75275	-4.429195	2.03795	2.58275
[1]	IGH	IGH	-4.09625	1.0406	-2.75275	-4.429195	2.03795	2.58275
[1]	IGHA2	IGHA2	-4.09625	1.0406	-2.75275	-4.429195	2.03795	2.58275
[1]	IGHD	IGHD	-2.1364	1.16962	-1.49929556	-2.062	1.8568	1.844
[1]	IGHG1	IGHG1	-4.09625	1.0406	-2.75275	-4.429195	2.03795	2.58275
[1]	IGHG1	IGHG1	-4.09625	1.0406	-2.75275	-4.429195	2.03795	2.58275
[1]	IGHM	IGHM	-4.09625	1.0406	-2.75275	-4.429195	2.03795	2.58275
[1]	LOC388078	LOC388078	-4.09625	1.0406	-2.75275	-4.429195	2.03795	2.58275
[0]	IGKC	IGKC	-5.556857143	1.781142857	-1.64357143	-7.327571429	2.157714286	2.666
[0]	LOC440871	LOC440871	-5.556857143	1.781142857	-1.64357143	-7.327571429	2.157714286	2.666

Enrichment Analysis by DAVID

Query Type: GENE_ID Tool: Functional Annotation Chart
Gene Type: ENTREZ_GENE_ID Annotations: Gene Ontology : GOTERM_BP_ALL; Gene Ontology : GOTERM_BP_FAT

Fig. 7 The result pages: (**a**) The result page consists of a table of biclusters and a table of gene set networks. (**b**) The individual page of a bicluster shows a heatmap, a line chart, and a numerical table. This page also has an interface of DAVID at the bottom



NEIGHBOURING NODES

V	ID_REF	NAME	DESCR	Entrez ID
1	TIMP3	TIMP3	--	--
2	CXCL13	CXCL13	--	--
5	IGKC	IGKC	--	--
5	LOC4...	LOC440871	--	--
6	IGL	IGL	--	--
6	IGLC2	IGLC2	--	--
7	SPARC	SPARC	--	--
8	MMP9	MMP9	--	--
9	IGH	IGH	--	--
9	IGHD	IGHD	--	--
9	IGHG1	IGHG1	--	--
10	COL3A1	COL3A1	--	--
11	GNLY	GNLY	--	--
12	CCL18	CCL18	--	--
13	FN1	FN1	--	--
14	NKG7	NKG7	--	--
15	FAM3	FAM38A	--	--

Fig. 8 Gene set network viewer has the left and right panes. The left pane shows a gene set network, and the right pane shows information on selected nodes in the network

flexible manner than using one bicluster only. For example, genes in the node with the maximum degree and its neighboring nodes or genes in the most significant bicluster can be selected. The application is available from the "Get gene set network viewer" button on the result page (right under the table of gene set networks) in Fig. 7a. The GUI is developed using Java 8 Swing and JUNG (Java Universal Network/Graph Framework) library [17]. The application can be launched by the following command

```
java –jar sibic_gsn_app.jar
```

or by right-clicking the jar file to open a dialog box, which will show a menu item for launching the application. The input of the application is a GNS (Gene Set Network) file, which is a binary file containing information about a gene set network. GNS files can be downloaded by clicking "download" in the "GNS file" column of the gene set networks' table in the result page. JRE (Java Runtime Environment) 1.8 or later must be installed on the local machine before launching the application.

As shown in Fig. 8, the GUI has left and right panes:

Left pane: has a drawing controller on the top and a network viewer on the bottom. In the top, the network "LAYOUT" can be selected out of four types: "Circle," "KK," "FR," and "ISOM." The default layout is "Circle," which positions nodes equally spaced on a regular circle, while "KK," "FR," and "ISOM" use the Kamada–Kawai algorithm [18], the Fruchterman–Reingold force-directed algorithm [19], and a layout algorithm based on Meyer's self-organizing graph methods [20], respectively. The diameter of nodes indicates the number of genes.

The "MODE" of the network viewer can be further selected, where "Picking" allows to pick and drag the nodes of interest, while "Transforming" enables to drag the whole network. Both modes allow to zoom in or out the network view by scrolling with the mouse wheel. Subnetworks can be filtered out by adjusting "VERTEX SIZE" or "EDGE WEIGHT."

By unchecking the "CONNECTED" box above, the network viewer can show unconnected networks. Clicking a vertex under the "Picking" mode updates the information in the right pane.

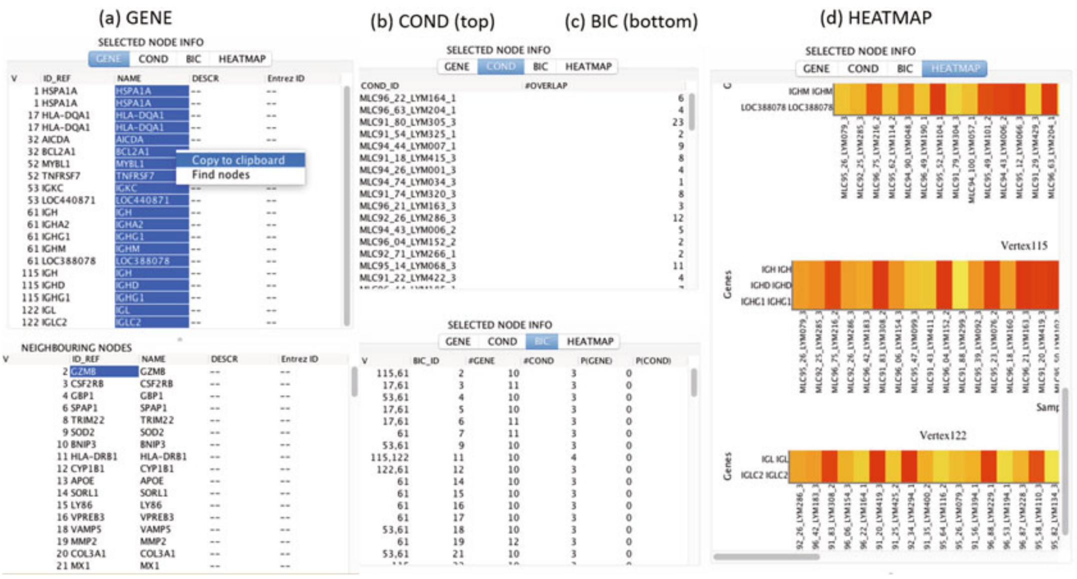
A helpful function of this side is that multiple nodes are clickable at the same time by dragging the mouse to make a rectangle so that it encompasses the multiple nodes, and clicking one of the selected nodes.

Right pane: has four tabs, "SELECTED," "BICS," "GENES," and "NODES," to display various information of the gene set network in the left pane.

"SELECTED" shows information on the selected nodes in the network, with "SELECTED NODES INFO" in the top and "NEIGHBORING NODES" in the bottom. "SELECTED NODES INFO" further consists of four subtabs, "GENE," "COND," "BIC" and "HEATMAP." "GENE" (Fig. 9a) shows a table of genes in the selected nodes, where data can be copied to the clipboard by dragging target cells and clicking the cells. "COND" (Fig. 9b) and "BIC" (Fig. 9c) show a table of experimental conditions and a table of biclusters, respectively. "HEATMAP" (Fig. 9d) shows a list of heatmaps for the respective nodes. These heatmaps are not those of merged biclusters but of Fig. 3b. Finally "NEIGHBORING NODES" displays a list of neighboring nodes of the selected nodes.

"BICS," "GENES," and "NODES" provide the information on the entire network in the left pane. "BICS" shows a table of all biclusters of the network. "GENES" shows a table of all genes in all nodes in the network. "NODES" shows a table of features of all nodes, such as the degree and weighted degree.

By using the left and right panes, genes of interest can be picked up to check, following the basic flow of manipulating a gene set network:



(a) GENE — SELECTED NODE INFO

V	ID_REF	NAME	DESCR	Entrez ID
1	HSPA1A	HSPA1A	--	--
1	HSPA1A	HSPA1A	--	--
17	HLA-DQA1	HLA-DQA1	--	--
17	HLA-DQA1	HLA-DQA1	--	--
32	AICDA	AICDA	--	--
32	BCL2A1	BCL2A1		
52	MYBL1	MYBL1		
52	TNFRSF7	TNFRSF7		
53	IGKC	IGKC	--	--
53	LOC440871	LOC440871	--	--
61	IGH	IGH	--	--
61	IGHA2	IGHA2	--	--
61	IGHG1	IGHG1	--	--
61	IGHM	IGHM	--	--
61	LOC388078	LOC388078	--	--
115	IGH	IGH	--	--
115	IGHD	IGHD	--	--
115	IGHG1	IGHG1	--	--
122	IGL	IGL	--	--
122	IGLC2	IGLC2	--	--

NEIGHBOURING NODES

V	ID_REF	NAME	DESCR	Entrez ID
2	GZMB	GZMB	--	--
3	CSF2RB	CSF2RB	--	--
4	GBP1	GBP1	--	--
6	SPAP1	SPAP1	--	--
8	TRIM22	TRIM22	--	--
9	SOD2	SOD2	--	--
10	BNIP3	BNIP3	--	--
11	HLA-DRB1	HLA-DRB1	--	--
12	CYP1B1	CYP1B1	--	--
13	APOE	APOE	--	--
14	SORL1	SORL1	--	--
15	LY86	LY86	--	--
16	VPREB3	VPREB3	--	--
18	VAMP5	VAMP5	--	--
19	MMP2	MMP2	--	--
20	COL3A1	COL3A1	--	--
21	MX1	MX1	--	--

(b) COND — SELECTED NODE INFO

COND_ID	#OVERLAP
MLC96_22_LYM164_1	6
MLC96_63_LYM204_1	4
MLC91_80_LYM305_3	23
MLC91_54_LYM325_1	2
MLC94_44_LYM007_1	9
MLC91_18_LYM415_3	8
MLC94_26_LYM001_3	4
MLC94_74_LYM034_3	1
MLC91_74_LYM320_3	8
MLC96_21_LYM163_3	3
MLC92_26_LYM286_3	12
MLC94_43_LYM006_2	5
MLC96_04_LYM152_2	2
MLC92_71_LYM266_1	2
MLC95_14_LYM068_3	11
MLC91_22_LYM422_3	4

(c) BIC — SELECTED NODE INFO

V	BIC_ID	#GENE	#COND	P(GENE)	P(COND)
115,61	2	10	3	0	
17,61	3	11	3	0	
53,61	4	10	3	0	
17,61	5	10	3	0	
17,61	6	11	3	0	
61	7	11	3	0	
53,61	9	10	3	0	
115,122	11	10	4	0	
122,61	12	10	3	0	
61	14	10	3	0	
61	15	10	3	0	
61	16	10	3	0	
61	17	10	3	0	
53,61	18	10	3	0	
61	19	12	3	0	
53,61	21	10	3	0	

Fig. 9 By clicking "SELECTED" in the right pane of Fig. 8, (**a**) GENE, (**b**) COND, (**c**) BIC, and (**d**) HEATMAP are displayed in the pane. GENE and COND show the list of genes and conditions in the selected nodes, respectively. BIC shows the information on biclusters related to the selected nodes. HEATMAP shows a list of heatmaps of the selected nodes

1. Click node(s) in the network viewer (the left pane in Fig. 8).
2. Find genes of interest in the information tables (the right pane in Fig. 8).
3. Copy the genes into your clipboard by right-click and run a third-party tool to perform functional analysis.

Here we describe two detailed scenarios for functional analysis:

1. Finding particular genes in the most significant bicluster in a gene set network
   - (a) Click any node in the network in the left pane, to activate the right pane.
   - (b) Click "BICS" in the right pane and then click "P(GENE)" (*p*-values) to sort the table.
   - (c) Click a cell in the row of the most significant bicluster.
   - (d) Open a popup by right-click and select "Find nodes."
   - (e) Click the nodes highlighted in aqua blue in the network viewer.
   - (f) Click "SELECTED" in the right pane to see the information such as gene names and heatmaps.
   - (g) Copy the target genes in "GENE" to the clipboard by right-click for functional analysis.

2. Finding particular genes in the node with the maximum weighed degree and its neighboring nodes
   (a) Click any node in the network viewer.
   (b) Click "NODES" and then click "W.DEG" (weighted degree) to sort the table.
   (c) Click a cell in the row of the node with the maximum weighted degree.
   (d) Open a popup by right-click and select "Find nodes."
   (e) Click the nodes highlighted in aqua blue in the network viewer.
   (f) Click "SELECTED" in the right pane to see the information such as gene names and heatmaps.
   (g) Copy the target genes from "GENE" and "NEIGHBORING NODES" to the clipboard by right-click for functional analysis.

4 Notes

1. SiBIC is available at http://utrecht.kuicr.kyoto-u.ac.jp:8080/sibic/faces/index.jsp. For a quick start, SiBIC has a tutorial page at http://utrecht. kuicr.kyoto-u.ac.jp:8080/sibic/faces/howto.jsp. We note that SiBIC cannot handle concurrent multiple jobs from a single user. The user must run only one job at a time.
2. In an ESearch query, users can specify a term with a "field tag" such as [orgn] for "organism" or [n_samples] for "the number of samples." Note that if no field tag is specified for a term, the term is directed to all fields. Below are some query examples.
   - saccharomyces+cerevisiae[orgn]+AND+4:7[n_samples]
   - log+ratio[vtyp]+OR+log2+ratio[vtyp]+OR+log10+ratio[vtyp]
   - cancer[title]+AND+(homo+sapiens[orng]+OR+mus+musculus[orgn])
3. SiBIC was a single-server platform in [11]. Currently, the architecture of SiBIC is the combination of a web server with a high-end fast server. Thus the server response has been much improved. SiBIC generates a five times larger number of random matrices than [11] to compute p-values, by concurrent and distributed computing, which makes p-values more precise. However, MAFIA [13] does not support concurrent computing, by which the computation time of FIM has not

been improved so drastically. Hence, the running status page should be checked to see if FIM takes an extraordinary large computation time.

4. In more detail, if FIM takes more than 1 min, the number of bicluster candidates generated is already intractable for computing *p*-values. So the job should be canceled and another parameter set to generate a smaller dataset should be tried.
5. For a GEO dataset, SiBIC can provide a "unified expression matrix" (.uem) file at the bottom of the parameter input interface, which is made by simply taking the average over replicates and can be uploaded as the input in the top page of SiBIC.

Acknowledgements

Part of this research has been supported by MEXT KAKENHI #16H02868 and #17H01783, ACCEL (JPMJAC1503) and PRESTO of JST, FiDiPro of Tekes and AIPSE programme, Academy of Finland.

References

1. Jiang D, Tang C, Zhang A (2004) Cluster analysis for gene expression data: a survey. IEEE Trans Knowl Data Eng 16(11):1370–1386
2. A Ben-Dor, Shamir R, Yakhini Z (1999) Clustering gene expression patterns. J Comput Biol 6(3–4):281–297
3. Madeira SC, Oliveira AL (2004) Biclustering algorithms for biological data analysis: a survey. IEEE Trans Comput Biol Bioinf 1(1):24–45
4. Saber HB, Elloumi M (2015) DNA microarray data analysis: a new survey on biclustering. Int J Comput Biol 4(1):21–37
5. Barkow S, Bleuler S, Prelic A, Zimmermann P, Zitzler E (2006) BicAT: biclustering analysis toolbox. Bioinformatics 22:1282–1283
6. Cheng KO, Law NF, Siu WC, Lau TH (2007) BiVisu: software tool for bicluster detection and visualization. Bioinformatics 23(17):2342–2344
7. Grothaus GA, Mufti A, Murali TM (2006) Automatic layout and visualization of biclusters. Algorithms Mol Biol 1:15
8. Heinrich J, Seifert R, Burch M, Weiskopf D (2011) Bicluster viewer: a visualization tool for analyzing gene expression data. In: Advances in visual computing, pp 641–652. Springer, Berlin
9. Streit M, Gratzl S, Gillhofer M, Mayr A, Mitterecker A, Hochreiter S (2014) Furby: fuzzy force-directed bicluster visualization. BMC Bioinf 15(Suppl 6):4
10. Santamaria R, Theron R, Quintales L (2008) BicOverlapper: a tool for bicluster visualization. Bioinformatics 24(9):1212–1213
11. Takahashi K, Takigawa I, Mamitsuka H (2013) SiBIC: a web server for generating gene set networks based on biclusters obtained by maximal frequent itemset mining. PLoS One 8(12):e82890
12. Han J, Cheng H, Xin D, Yan X (2007) Frequent pattern mining: current status and future directions. Data Min Knowl Disc 15:55–86
13. Burdick D, Calimlim M, Flannick J, Gehrke J, Yiu T (2005) MAFIA: a maximal frequent itemset algorithm. IEEE Trans Knowl Data Eng 17:1490–1504
14. Edgar R, Domrachev M, Lash AE (2002) Gene expression omnibus: NCBI gene expression and hybridization array data repository. Nucleic Acids Res 30:207–210
15. Sayers E, Wheeler D (2004) Building customized data pipelines using the entrez programming utilities (eUtils). NCBI

16. Huang DW, Sherman BT, Lempicki RA (2009) Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. Nat Protoc 4:44–57
17. Madadhain J, Fisher D, Smyth P, White S, Boey Y (2005) Analysis and visualization of network data using JUNG. J Stat Soft 10:1–35
18. Kamada T, Kawai S (1989) An algorithm for drawing general undirected graphs. Inf Process Lett 31:7–15
19. Fruchterman TMJ, Reingold EM (1991) Graph drawing by force-directed placement. Softw-Pract Exp 21:1129–1164
20. Meyer B (1998) Self-organizing graphs-a neural network perspective of graph layout. In: Graph drawing symposium, August 1998

Chapter 9

Computing and Visualizing Gene Function Similarity and Coherence with NaviGO

Ziyun Ding, Qing Wei, and Daisuke Kihara

Abstract

Gene ontology (GO) is a controlled vocabulary of gene functions across all species, which is widely used for functional analyses of individual genes and large-scale proteomic studies. NaviGO is a webserver for visualizing and quantifying the relationship and similarity of GO annotations. Here, we walk through functionality of the NaviGO webserver (http://kiharalab.org/web/navigo/) using an example input and explain what can be learned from analysis results. NaviGO has four main functions, accessed from each page of the webserver: "GO Parents," "GO Set", "GO Enrichment", and "Protein Set." For a given list of GO terms, the "GO Parents" tab visualizes the hierarchical relationship of GO terms, and the "GO Set" tab calculates six functional similarity and association scores and presents results in a network and a multidimensional scaling plot. For a set of proteins and their associated GO terms, the "GO Enrichment" tab calculates protein GO functional enrichment, while the "Protein Set" tab calculates functional association between proteins. The NaviGO source code can be also downloaded and used locally or integrated into other software pipelines.

Key words NaviGO, Gene ontology, Functional similarity, Visualization, Quantification, Function enrichment analysis, GO association score, Protein functional association score, Proteomic analysis

1 Introduction

The gene ontology (GO) is a widely used vocabulary for representing gene functions across all species [1, 2]. It is maintained and updated by the Gene Ontology Consortium. Currently, GO terms are classified into three categories: biological process (BP), which describes pathway information of gene products such as cellular physiological process or signal transduction; molecular function (MF), which describes molecular level activities such as enzymatic activity; and cellular component (CC), which describes cellular localization of gene products. Currently, there are over 46,000 GO terms, which are organized in a hierarchical structure, a directed acyclic graph (DAG). GO is very useful, particularly

Hiroshi Mamitsuka (ed.), *Data Mining for Systems Biology: Methods and Protocols*, Methods in Molecular Biology, vol. 1807, https://doi.org/10.1007/978-1-4939-8561-6_9, © Springer Science+Business Media, LLC, part of Springer Nature 2018

for computational analysis of gene functions; however, the volume of the vocabulary and the complicated relationships often makes analysis cumbersome.

NaviGO was developed to facilitate easy handling of GO terms, particularly for quantifying and visualizing relationships between GO terms [3]. NaviGO has four main functions: "GO Parents", "GO Set", "GO Enrichment", and "Protein Set." "GO Parents" maps and visualizes the hierarchical relationship of GO terms in an interactive fashion, and "GO Set" calculates six functional similarity and association scores and provides two visualization tools, a network and a multidimensional scaling visualization. For a list of proteins and associated GO terms, "GO Enrichment" performs GO enrichment analysis, while "Protein Set" identifies functionally related proteins. Compared with other related online tools [4, 5], NaviGO server has several advantages: first, it provides multiple similarity scores, which not only compare GO terms in the same GO category but also across GO categories. NaviGO provides biologists an intuitive and interactive tool to visualize parental relationships between GO terms. NaviGO is also integrated into the popular gene function prediction webservers PFP [6, 7] and ESG [8, 9].

2 Materials

NaviGO can be freely accessed at http://kiharalab.org/web/navigo/. It is a web application and does not require any platform other than a web browser. The source codes of NaviGO and GO Visualizer, a tool for visualizing the hierarchy of GO terms, can be downloaded from GitHub at https://github.com/kiharalab/NaviGO and https://github.com/kiharalab/GOVisualizer under the terms of the GNU Lesser General Public License Ver. 2.1.

In order to use NaviGO, users need to provide a set of GO terms or a set of UniProt IDs of proteins and associated GO terms to be analyzed. These can be retrieved from the Gene Ontology Consortium website (http://www.geneontology.org/) [1] or from the UniProt database (http://www.uniprot.org/) [10], respectively.

3 Methods

3.1 Overview of NaviGO

The NaviGO server has four main functions (Fig. 1). Either a set of GO terms or a set of proteins (with their GO terms) can be analyzed. For a set of GO terms, the "GO Parents" tab visualizes input GO terms in the GO DAG, and the "GO Set" tab calculates the functional similarity and association scores and visualizes them. On the other hand, for a list of proteins with their GO terms, the

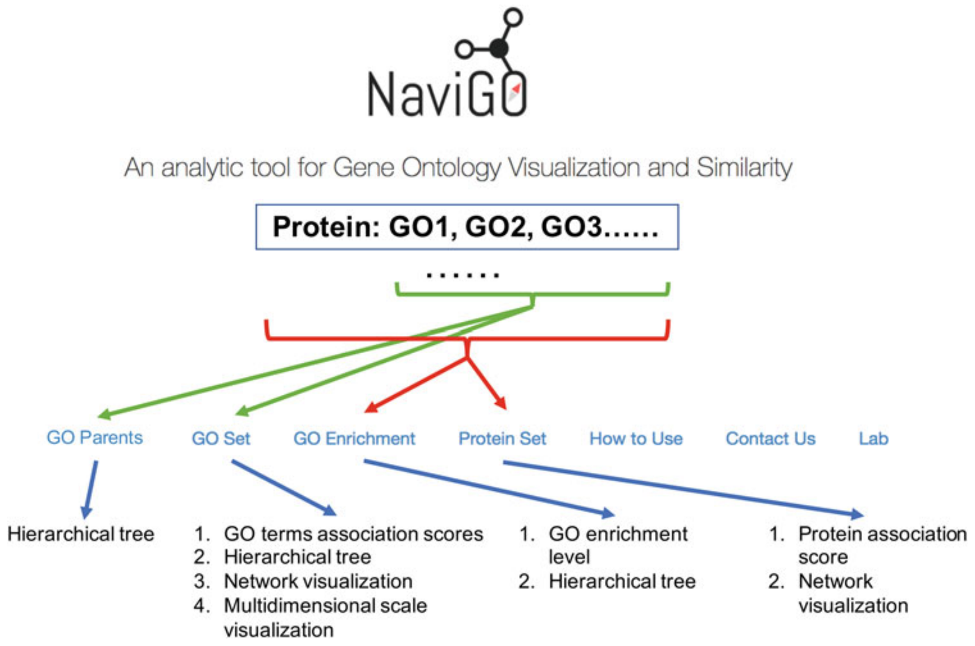


Fig. 1 Overview of NaviGO functionality. Input to be analyzed can be either a set of GO terms or a set of proteins

"GO Enrichment" tab performs GO enrichment analysis, and the "Protein Set" tab calculates functional similarity and association scores between proteins (Fig. 1).

Throughout this tutorial, we use the following six proteins, which are involved in the light signaling pathway, as examples: phytochrome A (PHYA, UniProt ID: P14712), phytochrome B (PHYB, UniProt ID: P14713), phytochrome D (PHYD, UniProt ID: P42497), phytochrome-interacting factor 3 (PIF3, UniProt ID: O80536), pseudo-response regulator 7 (PRR7, UniProt ID: A0A1P8BCB0), and histone deacetylase 15 (HDA15, UniProt ID: Q8GXJ1) (Table 1). PHYA, PHYB, and PHYD are from the phytochrome family and mainly function as red and far-red photoreceptors. They have been experimentally verified to interact with each other [11]. Interaction of the transcription factor PIF3 with the phytochrome family causes phosphorylation and degradation of phytochrome [12, 13]. Interaction of PIF3 with HDA15, a transcriptional repressor, represses the chlorophyll biosynthesis and the photosynthesis [14]. PRR7 is one of the key components of molecular clock in *Arabidopsis* and involved in the phytochrome-mediated red light signal transduction pathway [15]. PRR7 interacts with phytochrome and PIF to regulate the red light signal transduction. Known physical interactions of the six proteins are summarized in Fig. 2.

Table 1
List of the six example proteins and their associated GO terms

Protein name	UniProt ID	CC	MF	BP
PHYA	P14712	GO:0005737, GO:0016604, GO:0016607, GO:0005634	GO:0031516, GO:0042802, GO:0003729, GO:0000155, GO:0042803, GO:0004672, GO:0009883	GO:0009584, GO:0009630, GO:0017148, GO:0009640, GO:0009638, GO:0018298, GO:0017006, GO:0010161, GO:0006355, GO:0046685, GO:0010201, GO:0010218, GO:0010203, GO:0006351
PHYB	P14713	GO:0005829, GO:0016604, GO:0016607, GO:0005634	GO:0031516, GO:0042802, GO:0000155, GO:1990841, GO:0042803, GO:0031517, GO:0009883, GO:0043565	GO:0009687, GO:0006325, GO:0010617, GO:0009584, GO:0009649, GO:0009630, GO:0009867, GO:0045892, GO:0009640, GO:0015979, GO:0009638, GO:0018298, GO:0017012, GO:0010161, GO:0031347, GO:2000028, GO:0010029, GO:0009409, GO:0010218, GO:0010244, GO:0010202, GO:0009266, GO:0010374, GO:0006351, GO:0010148
PRR7	A0A1P8BCB0	GO:0005634	NA	GO:0000160
PIF3	O80536	GO:0005634	GO:0003677, GO:0042802, GO:0046983, GO:0003700	GO:0009704, GO:0009740, GO:0031539, GO:0010017, GO:0009585, GO:0006355, GO:0009639, GO:0006351
PHYD	P42497	GO:0005634	GO:0042802, GO:0000155, GO:0009881, GO:0042803	GO:0018298, GO:0017006, GO:0009585, GO:0006355, GO:0006351
HDA15	Q8GXJ1	GO:0005634	GO:0046872, GO:0032041	GO:0006355, GO:0006351

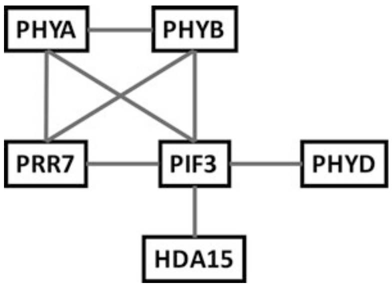


Fig. 2 The interaction relationship of the six example proteins. These six proteins are involved in the light signaling pathway

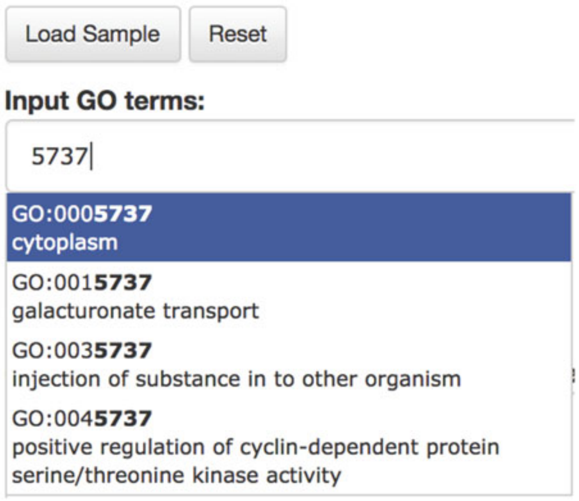
3.2 Quantification and Visualization of GO Term Association and Similarity

The "GO Set" tab computes six GO term similarity and association scores for all the pairs of input GO terms. The scores are Resnik's, Lin's, relevance similarity score (RSS), the Interaction Association Score (IAS), the PubMed Association Score (PAS), and the Co-occurrence Association Score (CAS). The first three scores quantify similarity of a GO term pair of the same category. They are calculated based on the frequencies of two GO terms in the gene annotation database and their location in the GO DAG [16–18]. Among the three scores, RSS not only considers the relative depth of the common ancestor between the two GO terms but also considers how rare the query GO terms are to identify the common ancestor. The last three semantic-based functional similarity scores were developed by our group. IAS quantifies the probability that two GO terms appear in physically interacting protein pairs [19]. PAS and CAS quantify the frequency with which two GO terms appear in the same PubMed abstract and in a single gene annotation, respectively [20].

To use the "GO Set" tab, please follow the steps described below:

1. Enter your input in the box. The input format of the "GO set" tab is a list of GO terms separated by comma. Users can upload a formatted file or type in the GO term ID. As a GO term ID is being typed, NaviGO will automatically recognize the GO term with the number and show candidates in a pull-down list. Thus, users can choose one from the list. For example, "GO:0005737" can be retrieved after typing "5737" and clicking the first GO term in the pull-down list (Fig. 3).
2. To empty inputs, click the "Reset button" located above the input box. To delete a single GO term in the input box, click the "X" sign at the GO term.
3. Clicking the "Submit" button below the input box will start the analysis and show a result page when done.

At the top of the results page, query GO term scores are listed in colors that indicate categories: BP terms are in red, MF in blue,



Input GO terms:

5737

GO:0005737
cytoplasm

GO:0015737
galacturonate transport

GO:0035737
injection of substance in to other organism

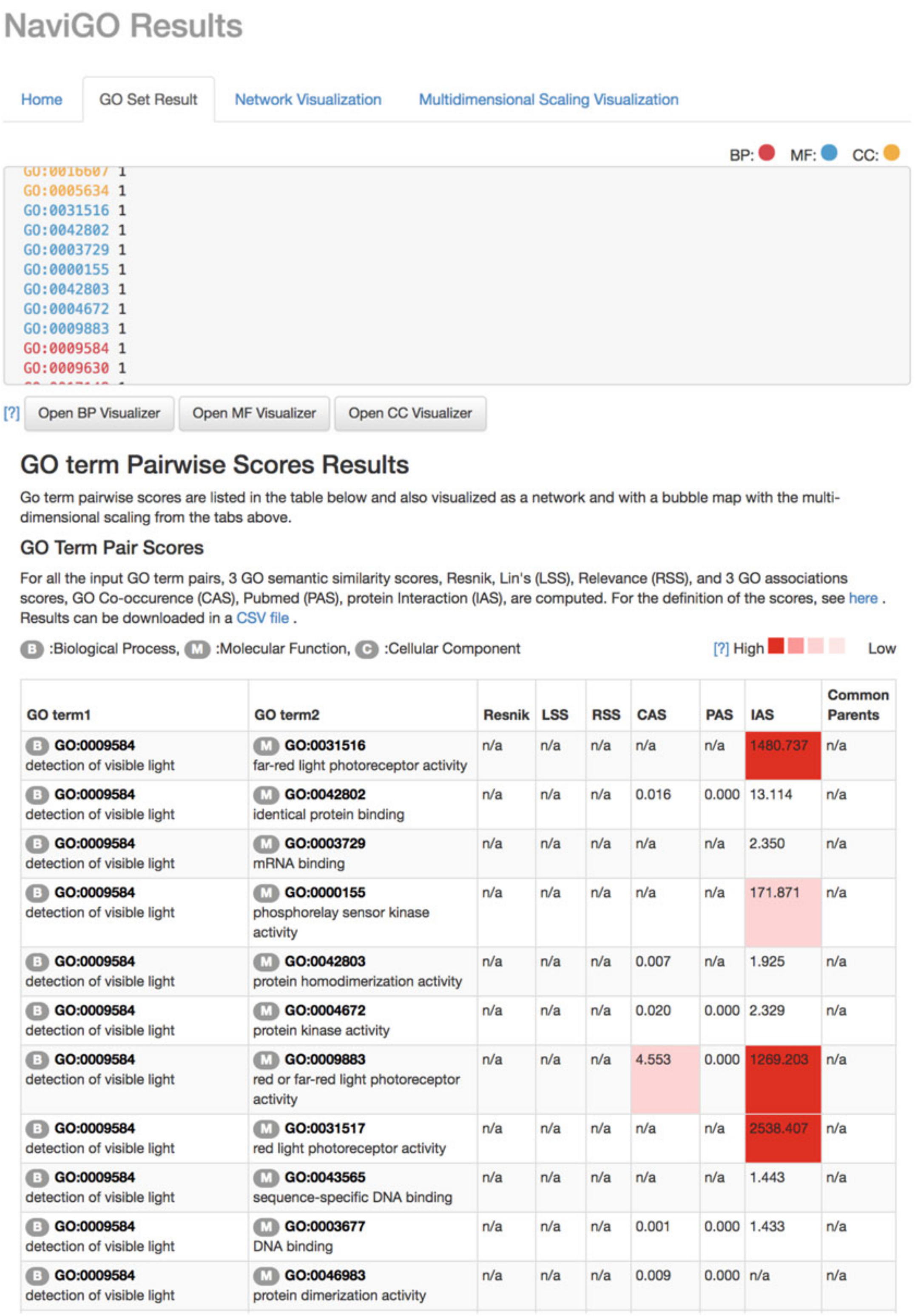
GO:0045737
positive regulation of cyclin-dependent protein serine/threonine kinase activity

Fig. 3 Example of inputting GO terms

and CC in yellow (Fig. 4, top). The numbers on the right side of GO terms are the counts of each GO term in the input. Clicking the BP/MF/CC Visualizer button below the query GO term list will open a new page that shows the GO terms of the category in the GO DAG (Fig. 5). The color legends of GO terms are listed on the right side, and colors of GO term relationships are shown in the left upper corner of the page. The query GO terms are shown in a larger font in the DAG. Clicking a GO term in a graph will expand links to all the children GO terms. In the example shown in Fig. 5, seven molecular function GO terms are mapped (Fig. 5). We can see that the GO terms locate in two branches, photoreceptor activity and protein-binding activity.

Pairwise GO term scores are calculated and listed in the table below the input GO term list (Fig. 4, bottom). GO pairs in the table can be sorted by a score by clicking the title of the score column. If the members of a pair of GO terms do not belong to the same category or the score of the pair is not available, "n/a" is shown. The significance level of scores in each column is indicated in a color scale, from light pink to red as the significance level increases. Clicking the "+" in the "common parents" column expands the list of all common parents of the GO pair in the GO DAG. Clicking a GO term will take users to the AmiGO website, which provides more detailed information of the term.

In Fig. 4, a part of the result page for the example input in Table 1 is shown. IAS of the BP term GO:0009584 (related to detection of visible light) and the MF term GO:00031516 (far-red light photoreceptor activity) is highlighted in red, because the score 1480.737 is within the top 1% of scores relative to the score background distribution. Both GO terms annotate the



NaviGO Results

Home | GO Set Result | Network Visualization | Multidimensional Scaling Visualization

BP: ● MF: ● CC: ●

```
GO:0016607 1
GO:0005634 1
GO:0031516 1
GO:0042802 1
GO:0003729 1
GO:0000155 1
GO:0042803 1
GO:0004672 1
GO:0009883 1
GO:0009584 1
GO:0009630 1
```

[?] Open BP Visualizer | Open MF Visualizer | Open CC Visualizer

GO term Pairwise Scores Results

Go term pairwise scores are listed in the table below and also visualized as a network and with a bubble map with the multi-dimensional scaling from the tabs above.

GO Term Pair Scores

For all the input GO term pairs, 3 GO semantic similarity scores, Resnik, Lin's (LSS), Relevance (RSS), and 3 GO associations scores, GO Co-occurence (CAS), Pubmed (PAS), protein Interaction (IAS), are computed. For the definition of the scores, see here . Results can be downloaded in a CSV file .

B :Biological Process, M :Molecular Function, C :Cellular Component

[?] High 🟥 ▨ ▨ ▨ Low

GO term1	GO term2	Resnik	LSS	RSS	CAS	PAS	IAS	Common Parents
B **GO:0009584** detection of visible light	M **GO:0031516** far-red light photoreceptor activity	n/a	n/a	n/a	n/a	n/a	1480.737	n/a
B **GO:0009584** detection of visible light	M **GO:0042802** identical protein binding	n/a	n/a	n/a	0.016	0.000	13.114	n/a
B **GO:0009584** detection of visible light	M **GO:0003729** mRNA binding	n/a	n/a	n/a	n/a	n/a	2.350	n/a
B **GO:0009584** detection of visible light	M **GO:0000155** phosphorelay sensor kinase activity	n/a	n/a	n/a	n/a	n/a	171.871	n/a
B **GO:0009584** detection of visible light	M **GO:0042803** protein homodimerization activity	n/a	n/a	n/a	0.007	n/a	1.925	n/a
B **GO:0009584** detection of visible light	M **GO:0004672** protein kinase activity	n/a	n/a	n/a	0.020	0.000	2.329	n/a
B **GO:0009584** detection of visible light	M **GO:0009883** red or far-red light photoreceptor activity	n/a	n/a	n/a	4.553	0.000	1269.203	n/a
B **GO:0009584** detection of visible light	M **GO:0031517** red light photoreceptor activity	n/a	n/a	n/a	n/a	n/a	2538.407	n/a
B **GO:0009584** detection of visible light	M **GO:0043565** sequence-specific DNA binding	n/a	n/a	n/a	n/a	n/a	1.443	n/a
B **GO:0009584** detection of visible light	M **GO:0003677** DNA binding	n/a	n/a	n/a	0.001	0.000	1.433	n/a
B **GO:0009584** detection of visible light	M **GO:0046983** protein dimerization activity	n/a	n/a	n/a	0.009	0.000	n/a	n/a

Fig. 4 GO Set result page

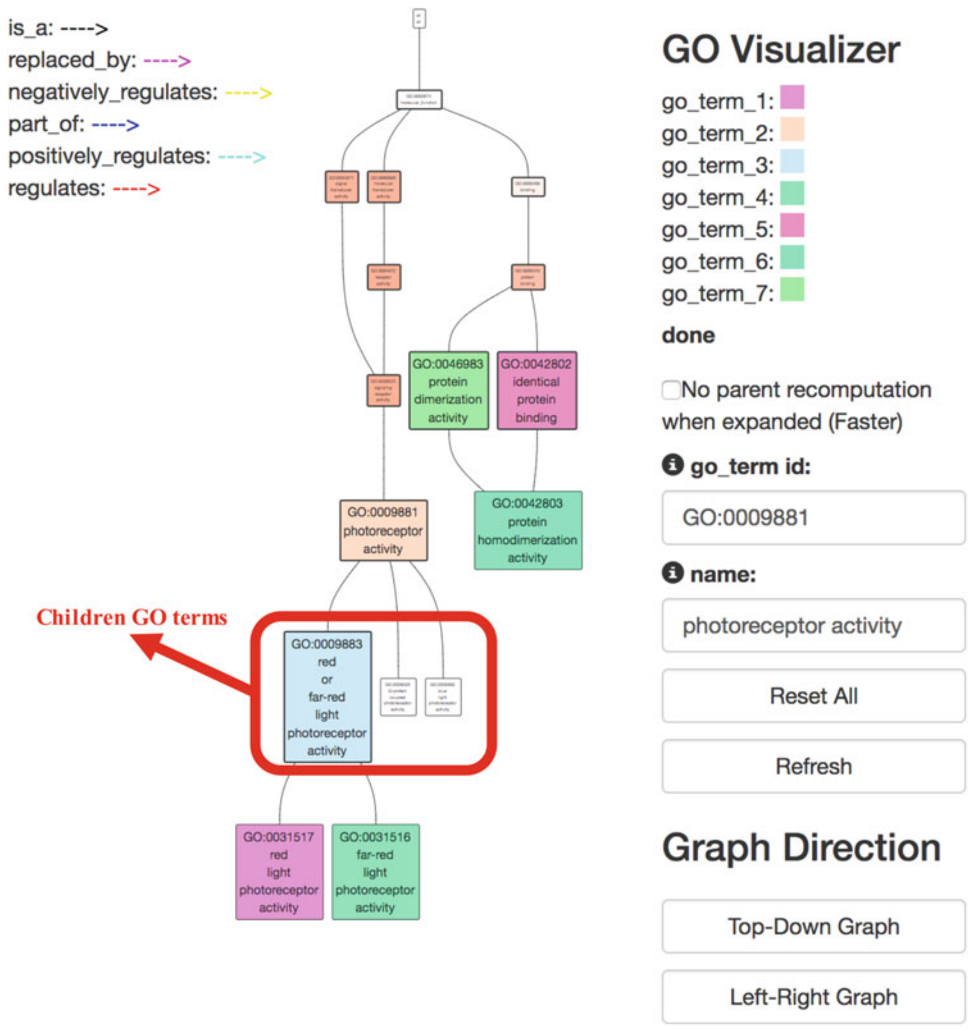

is_a: ---->
replaced_by: ---->
negatively_regulates: ---->
part_of: ---->
positively_regulates: ---->
regulates: ---->
Children GO terms
GO:0046983 protein dimerization activity
GO:0042802 identical protein binding
GO:0009881 photoreceptor activity
GO:0042803 protein homodimerization activity
GO:0009883 red or far-red light photoreceptor activity
GO:0031517 red light photoreceptor activity
GO:0031516 far-red light photoreceptor activity
GO Visualizer
go_term_1:
go_term_2:
go_term_3:
go_term_4:
go_term_5:
go_term_6:
go_term_7:
done
No parent recomputation when expanded (Faster)
go_term id:
GO:0009881
name:
photoreceptor activity
Reset All
Refresh
Graph Direction
Top-Down Graph
Left-Right Graph

Fig. 5 Hierarchical graph representation using GO Visualizer. Seven molecular function GO terms are visualized here, GO:0031517, GO:0009881, GO:0009883, GO:0031516, GO:0042802, GO:0042803, and GO:0046983. Clicking a GO term expands edges to all the children GO terms

phytochrome proteins, which are known to form heterodimers [11], so it is reasonable that the two GO terms annotating these interacting proteins have a very high IAS.

The results table can be also downloaded in a comma separated data file (a CSV format file) by clicking the "CSV file" button.

NaviGO provides two types of visualizations for GO pairwise score results. One is a network visualization available under the "Network Visualization" tab (Fig. 6). In the network, functionally related GO terms are connected by edges. The score cutoff value to define edges can be controlled by sliding the bar or by typing the value in a text box. The scores to visualize can be chosen at the upper left corner of the page. In the example in Fig. 6, the

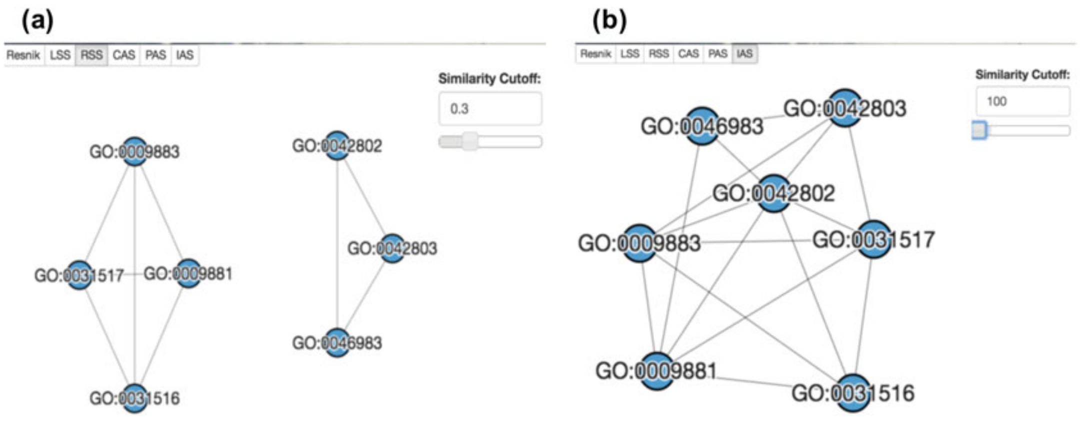


Fig. 6 The network of functional association score of seven GO terms: GO:0031517, GO:0009881, GO:0009883, GO:0031516, GO:0042802, GO:0042803, and GO:0046983. (**a**) Network using RSS with a cutoff value of 0.3. (**b**) Network using IAS with a cutoff value of 100

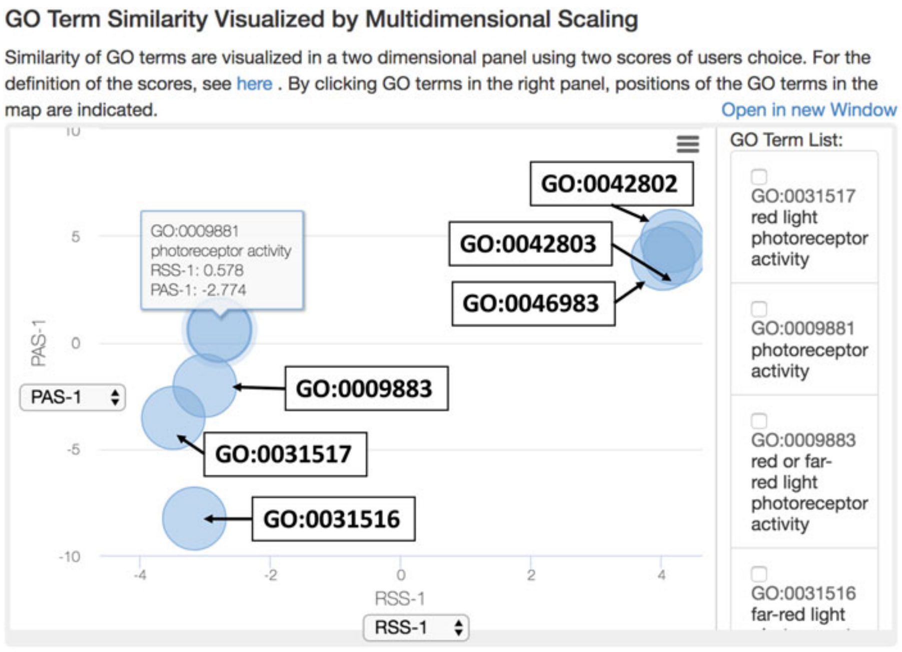
same set of GO terms as Fig. 5 was used. In the network with RSS (Fig. 6, left), the GO terms were clustered into two groups, which is consistent with the two branches in the GO hierarchy shown in Fig. 5. Using IAS, all GO terms are connected (Fig. 6b). This is also reasonable because these terms are associated with light signaling proteins, and they are known to interact with each other as shown in Fig. 2.

The second visualization is available at the "Multidimensional Scaling Visualization" tab. In this two-dimensional (2D) graph, GO terms are classified and mapped onto a 2D space with two scores selected by users. Placing the cursor over a GO term will show the normalized functional score of the GO term. In the example in Fig. 7, the x-axis is RSS and the y-axis is PAS. The GO terms are largely classified in two groups, which are again consistent with the results in Figs. 5 and 6.

3.3 GO Enrichment Analysis

The goal of GO enrichment analysis is to find if any GO term appears more frequently in a set of proteins than would be expected from the background frequency of the term in the genome. The significance of protein is quantified by a *p-value*. A *p-value* of a GO term for a protein set is calculated by considering the number of proteins in the set, the number of proteins annotated with the GO term in the genome, and the total number of proteins in the organism. The smaller the *p-value* is, the more significant the GO term is.

The input format of the "GO Enrichment" tab is a list of UniProt IDs associated with GO terms, e.g., "A0A1P8BCB0 GO:0005634, GO:0000160". NaviGO automatically identifies the organism based on the UniProt ID of the first protein in the input.

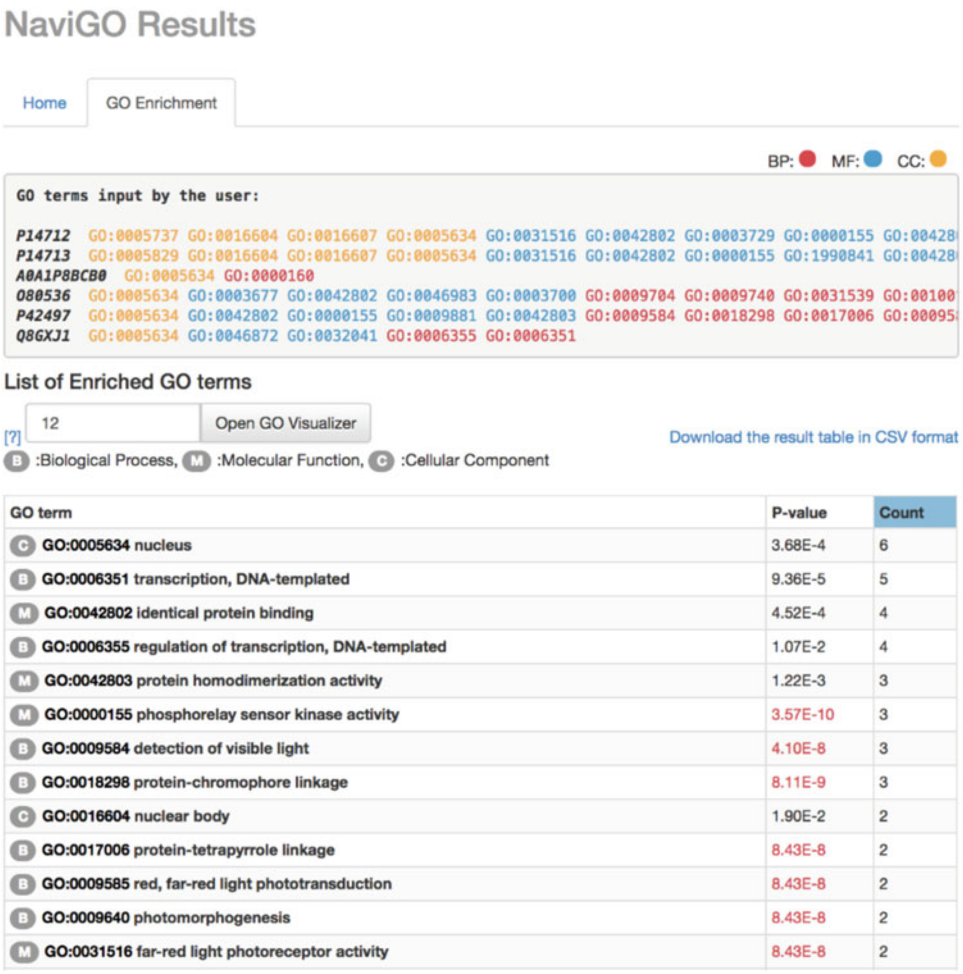

GO Term Similarity Visualized by Multidimensional Scaling

Similarity of GO terms are visualized in a two dimensional panel using two scores of users choice. For the definition of the scores, see here . By clicking GO terms in the right panel, positions of the GO terms in the map are indicated.

Fig. 7 The multidimensional scaling visualization of seven GO terms: GO:0031517, GO:0009881, GO:0009883, GO:0031516, GO:0042802, GO:0042803, and GO:0046983

In the results page, GO terms of input proteins are sorted by their p-value (Fig. 8). The significant p-value (below 0.00005 or top 30) GO terms are highlighted in red. The total number of significantly enriched GO terms is counted in the box on the left from "Open GO Visualizer". The third column shows the number of input proteins that have the GO term. In the example shown in Fig. 8, the most enriched GO term among the proteins from *Arabidopsis* is GO:0000155 *phosphorelay sensor kinase activity* with a p-value of 3.57E-10 and GO:0018298 *protein-chromophore linkage* with a p-value of 8.11E-9.

Enriched GO terms with p-value above 0.00005 (or the top 30 GO terms) can be mapped to the GO hierarchy by clicking "open GO Visualizer". The enriched GO terms are shown in a larger font and colored based on p-value from red to yellow indicating most to least significance. The figure can be downloaded by clicking "Download Figure Here". In the example in Fig. 9, the significantly enriched GO terms are involved in the signal receptor activity such as GO: 0000155, "phosphorelay sensor kinase activity" with p-value of 3.57e-10, and GO:0009883, "red or far-red light photoreceptor" with *p-value* of 8.43e-8. Also, GO terms identified as enriched are involve in red or far-red light signaling pathway such as GO:0010161, "red light signaling pathway" with *p-value* of 8.43e-8, and detection of light stimulus such as GO:0009854, "detection of visible light" with *p-value* of 4.10e-8.



```
GO terms input by the user:

P14712   GO:0005737 GO:0016604 GO:0016607 GO:0005634 GO:0031516 GO:0042802 GO:0003729 GO:0000155 GO:00428
P14713   GO:0005829 GO:0016604 GO:0016607 GO:0005634 GO:0031516 GO:0042802 GO:0000155 GO:1990841 GO:00428
A0A1P8BCB0   GO:0005634 GO:0000160
O80536   GO:0005634 GO:0003677 GO:0042802 GO:0046983 GO:0003700 GO:0009704 GO:0009740 GO:0031539 GO:00100
P42497   GO:0005634 GO:0042802 GO:0000155 GO:0009881 GO:0042803 GO:0009584 GO:0018298 GO:0017006 GO:00095
Q8GXJ1   GO:0005634 GO:0046872 GO:0032041 GO:0006355 GO:0006351
```

List of Enriched GO terms

[?]
12
Open GO Visualizer
Download the result table in CSV format
B :Biological Process, M :Molecular Function, C :Cellular Component

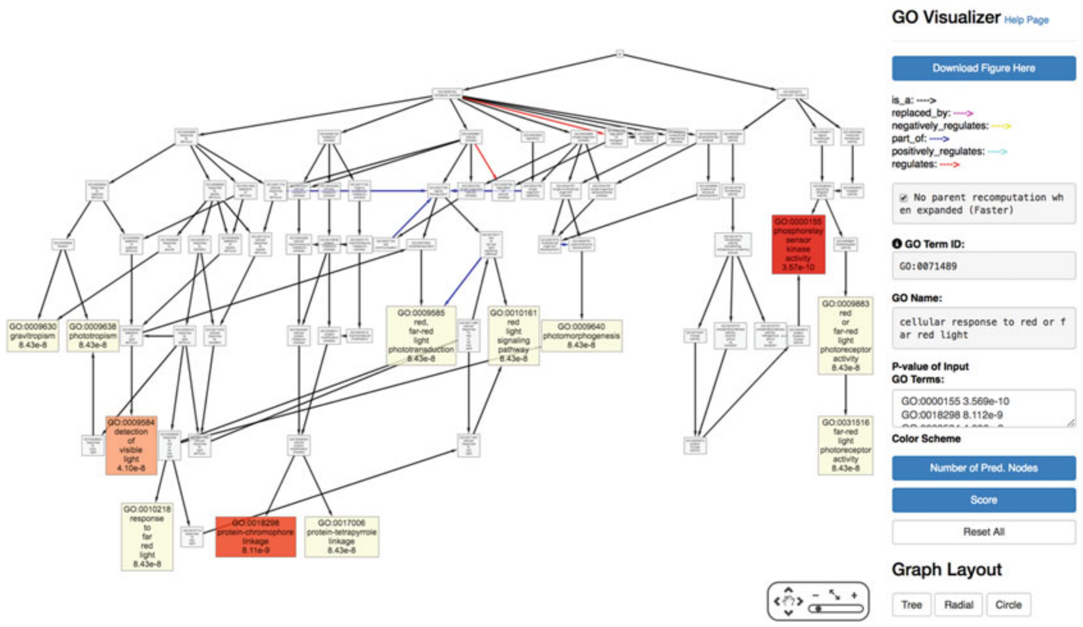
GO term	P-value	Count
C GO:0005634 nucleus	3.68E-4	6
B GO:0006351 transcription, DNA-templated	9.36E-5	5
M GO:0042802 identical protein binding	4.52E-4	4
B GO:0006355 regulation of transcription, DNA-templated	1.07E-2	4
M GO:0042803 protein homodimerization activity	1.22E-3	3
M GO:0000155 phosphorelay sensor kinase activity	3.57E-10	3
B GO:0009584 detection of visible light	4.10E-8	3
B GO:0018298 protein-chromophore linkage	8.11E-9	3
C GO:0016604 nuclear body	1.90E-2	2
B GO:0017006 protein-tetrapyrrole linkage	8.43E-8	2
B GO:0009585 red, far-red light phototransduction	8.43E-8	2
B GO:0009640 photomorphogenesis	8.43E-8	2
M GO:0031516 far-red light photoreceptor activity	8.43E-8	2

Fig. 8 The GO enrichment analysis result of six proteins: PHYA (P14712), PHYB (P14713), PRR7 (A0A1P8BCB0), PIF3 (O80536), PHYD (P42497), and HDA15 (Q8GXJ1)

3.4 Quantifying Functional Association of Proteins

This function identifies protein pairs in a query protein set that have functional relevance. Using a GO pair score, functional relevance of a protein pair is evaluated by the funSim score [21], which is in essence the average GO pair scores of GO annotations of the two proteins (*see* **Note 1**). Eight different GO pair scores are used in NaviGO (Fig. 10): “MF”, “BP”, and “CC” use RSS of the particular GO category, “BP + MF” is the funSim score using BP and MP, while “All” is the funSim using MF, BP, and CC. “PAS”, “CAS”, and “IAS” use the corresponding functional association scores to compute funSim.

For example, when studying whether proteins exist in the same cellular component, it would be interesting to check the CC funSim score. When studying whether proteins are involved in the same pathway or biological process, users would want to check “BP”, “MF”, or “BP + MF” columns.


GO Visualizer Help Page
Download Figure Here
is_a: --->
replaced_by: --->
negatively_regulates: --->
part_of: --->
positively_regulates: --->
regulates: --->
GO:0000155 phosphorelay sensor kinase activity 3.57e-10
GO:0009630 gravitropism 8.43e-8
GO:0009638 phototropism 8.43e-8
GO:0009585 red, far-red light phototransduction 8.43e-8
GO:0010161 red light signaling pathway 8.43e-8
GO:0009640 photomorphogenesis 8.43e-8
GO:0009883 red or far-red light photoreceptor activity 8.43e-8
GO:0031516 far-red light photoreceptor activity 8.43e-8
GO:0009584 detection of visible light 4.10e-8
GO:0010218 response to far red light 8.43e-8
GO:0018298 protein-chromophore linkage 8.11e-9
GO:0017006 protein-tetrapyrrole linkage 8.43e-8
Color Scheme
Number of Pred. Nodes
Score
Reset All
Graph Layout
Tree
Radial
Circle

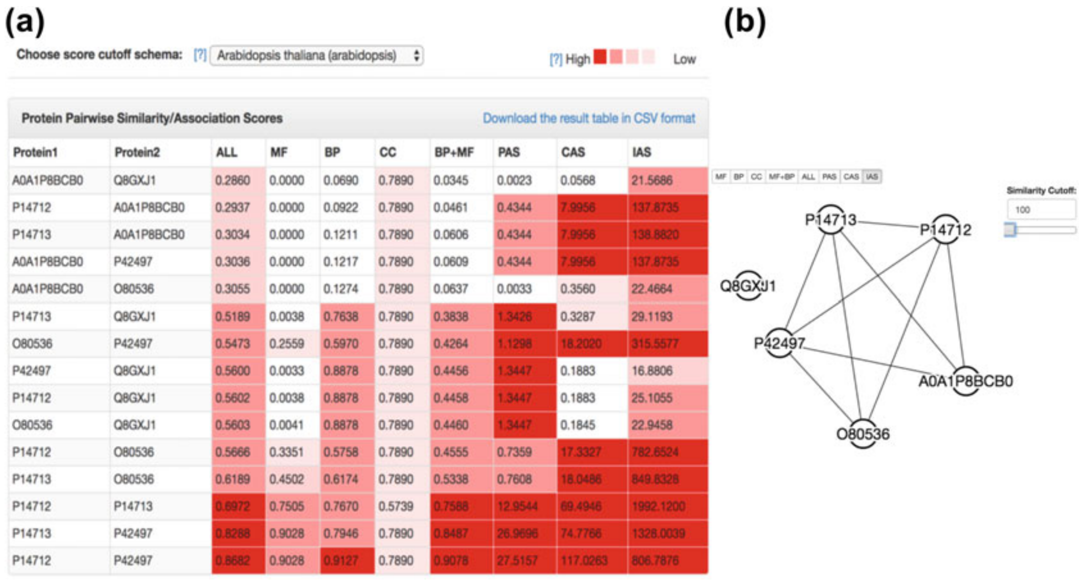
☑ No parent recomputation when expanded (Faster)

GO Term ID: GO:0071489

GO Name: cellular response to red or far red light

P-value of Input GO Terms:
GO:0000155 3.569e-10
GO:0018298 8.112e-9

Fig. 9 Visualization of 12 significantly enriched GO terms



Choose score cutoff schema: [?] Arabidopsis thaliana (arabidopsis)

Protein Pairwise Similarity/Association Scores

Download the result table in CSV format

Protein1	Protein2	ALL	MF	BP	CC	BP+MF	PAS	CAS	IAS
A0A1P8BCB0	Q8GXJ1	0.2860	0.0000	0.0690	0.7890	0.0345	0.0023	0.0568	21.5686
P14712	A0A1P8BCB0	0.2937	0.0000	0.0922	0.7890	0.0461	0.4344	7.9956	137.8735
P14713	A0A1P8BCB0	0.3034	0.0000	0.1211	0.7890	0.0606	0.4344	7.9956	138.8820
A0A1P8BCB0	P42497	0.3036	0.0000	0.1217	0.7890	0.0609	0.4344	7.9956	137.8735
A0A1P8BCB0	O80536	0.3055	0.0000	0.1274	0.7890	0.0637	0.0033	0.3560	22.4664
P14713	Q8GXJ1	0.5189	0.0038	0.7638	0.7890	0.3838	1.3426	0.3287	29.1193
O80536	P42497	0.5473	0.2559	0.5970	0.7890	0.4264	1.1298	18.2020	315.5577
P42497	Q8GXJ1	0.5600	0.0033	0.8878	0.7890	0.4456	1.3447	0.1883	16.8806
P14712	Q8GXJ1	0.5602	0.0038	0.8878	0.7890	0.4458	1.3447	0.1883	25.1055
O80536	Q8GXJ1	0.5603	0.0041	0.8878	0.7890	0.4460	1.3447	0.1845	22.9458
P14712	O80536	0.5666	0.3351	0.5758	0.7890	0.4555	0.7359	17.3327	782.6524
P14713	O80536	0.6189	0.4502	0.6174	0.7890	0.5338	0.7608	18.0486	849.8328
P14712	P14713	0.6972	0.7505	0.7670	0.5739	0.7588	12.9544	69.4946	1992.1200
P14713	P42497	0.8288	0.9028	0.7946	0.7890	0.8487	26.9696	74.7766	1328.0039
P14712	P42497	0.8682	0.9028	0.9127	0.7890	0.9078	27.5157	117.0263	806.7876

Fig. 10 Example of protein set analysis. (**a**) Results of pairwise protein association scores. (**b**) The protein association network of six proteins PHYA (P14712), PHYB (P14713), PRR7 (A0A1P8BCB0), PIF3 (O80536), PHYD (P42497), and HDA15 (Q8GXJ1) with a cutoff value of 100

The input data is a list of proteins and their GO annotations, the same as described in the GO enrichment analysis. In the results table (Fig. 10), the significance levels of scores are shown in color scale (red to pink for high to low). Since the significant cutoff is defined by the score distribution of a particular organism, there is a pull-down menu above the table to select the reference organism. In this example of six proteins, they all have the same RSS score of CC (Fig. 10a), reflecting that all proteins are located in the nucleus. PHYA (P14712) and PHYB (P14713) have a significantly high IAS of 1992.12, because they physically interact with each other [22].

Sometimes, it is difficult to see the functional association between proteins by looking at the score numbers in the output table. NaviGO provides a network visualization, which is available at "Open in new Window" (Fig. 10b). In the network, proteins are connected if their association scores are above a cutoff value. In the example, only HDA15 is not connected in the association network with IAS cutoff value set to 100. This is consistent with the STRING database [23], where only HDA15 has low binding scores with all the other proteins in this network.

3.5 Downloading Source Codes

The entire source code of NaviGO is available for academic use on GitHub (https://github.com/kiharalab/NaviGO). The code for GO Visualizer is also separately available on GitHub (https://github.com/kiharalab/GOVisualizer). GO Visualizer is the tool integrated in NaviGO that performs real-time rendering of GO DAGs in an interactive way. The code is licensed under the terms of the GNU Lesser General Public License Ver. 2.1. Users can download the package for local use of the software or to integrate it into other software pipelines, for example, a pipeline for proteomic mass spectrometry data with protein function analysis.

To set up the GO Visualizer locally, follow the steps below:

1. Install the software dependencies. The GO Visualizer requires Ruby, Gem, Sinatra, and MySQL. Ruby and Gem are two programming languages which have been installed in most of computers. Sinatra is a free and open-source software web application library. MySQL is one of the most popular open-source relational database management system. The general MySQL installer is available at https://dev.mysql.com/downloads.
2. Download the GO database from the GO Consortium. The size of the GO database is around 12 MB. In a Linux terminal, the GO database can be downloaded by running the following:

```
$ wget http://archive.geneontology.org/latest-full/
         go_monthly-termdb-data.gz
```

3. After downloading a GO database file, the Linux command to uncompress the file is:

```
$ gunzip go_monthly-termdb-data.gz
```

4. The GO database can be created with the following Linux command, where the "database_name" is the name users want to assign for the database, and "db_login" and "db_password" are the username and password defined by users, respectively:

```
$ echo ‘‘create database  database_name’’ |
        mysql --user=db_login --password=db_password
$ mysql --user=db_login --password=db_password
        database_name < go_monthly-termdb-data
```

5. To run GO Visualizer, use the following terminal commands:

```
$ git https://github.com/kiharalab/GOVisualizer.git
$ cd GOVisualizer
$ mv config_template.rb config.rb
```

6. Users need to change the settings in config.rb according to the local MySQL settings using following terminal command:

```
$ ruby server.rb
```

To set up the NaviGO webserver locally, follow the steps below:

1. In order to implement the NaviGO package on the local machine, Perl, Python 2.7, Python 3.4, and MySQL are needed.
2. The pre-calculated pairwise GO IAS, PAS, and CAS scores are also needed to compute protein functional association scores and can be downloaded with the following commands:

```
$ wget http://kiharalab.org/web/navigo/data/PAS.txt
$ wget http://kiharalab.org/web/navigo/data/CAS.txt
$ wget http://kiharalab.org/web/navigo/data/
        BIOGRID-3.2.107_GOpair_IASscores.txt
```

3. To install GO Visualizer, run:

```
$ git https://github.com/kiharalab/NaviGO.git
```

4. Users need to change the directory to “NaviGO/AutoUpdate”:

```
$ cd NaviGO/AutoUpdate
```

5. If there is no temporary folder, users need to make one named “tmp” to save the updated GO database:

```
$ mkdir tmp
```

6. Users need to run the “update.pl” script to generate the database NaviGO uses with the following command. After running the script, there should be a folder named like GO_YYYYMM, for example, GO_201701, and all the files should be inside this folder.

```
$ perl update.pl
```

7. Before running NaviGO, users need to set the paths correctly in “config_template.pl” and then rename the script “config_template.pl” to “config.pl”:

```
$ mv config_template.pl config.pl
```

8. To run the protein set function on NaviGO, users need to create their own folder under the "job" folder with the following commands, where "your_job_name" is the customized job name defined by user:

```
$ cd job
$ mkdir your_job_name
$ cd your_job_name
```

9. Users need to create the input file. The required format of input file can be found on our server NaviGO. For a given list of GO terms, the format should be "GO:xxxxxxx, GO:xxxxxxx. . . . . . ." For a given list of UniProt ID and their associated GO terms, the format should be "UniProt_ID: GO:xxxxxxx, GO:xxxxxxx. . . . . . ." Currently, NaviGO supports the required format and also the CAFA format (https://github.com/idoerg/cafa-format-check). NaviGO also provides a file checker under the "format_check" folder. The "cafa_go_format_checker.py" script will convert the CAFA format to our required format or check the file you provided. Users can run by the following command, where "file" is the CAFA-formatted file provided by the user, and "input_file" is the required format file converted by the script:

```
$ python ../../format_check/cafa_go_format_checker.py file >
  input_file
```

10. Then, you can run NaviGO by the following command:

```
$ perl ../../run.pl input_file
```

11. To run GO set, users can type the following commands, where "path_to_your_go_file" is the input file provided by users:

```
$ cd job
$ mkdir your_job_name
$ cd your_job_name
$ cp path_to_your_go_file ./input_file
$ perl execute.pl
```

12. To run enrichment analysis, users can type the following commands, where "organism_id" is the organism ID defined by UniProt database:

```
$ cd job
$ mkdir your_job_name
$ cd your_job_name
$ python3.4 ../../Enrich/enrich.py -f input_file -o
         organism_id
```

4 Notes

1. Functional relevance of a protein pair is quantified with the *funSim* score of a particular GO pair score [18]. The *funSim* score of a GO pair score for a protein pair is defined as, in

short, the average of the best combination of GO term pairs that annotate the two proteins. For a pair of protein, as shown in Fig. 10, NaviGO provides *funSim* scores for MF, BP, CC, BP + MF, all (i.e., BP + MF + CC), PAS, CAS, and IAS. For the first five scores, RSS of GO term pairs is used. From mathematical standpoint, the *funSim* score of a protein pair is defined as follows [18]:

First, RSS of two GO terms *c*1 and *c*2 is computed as

$$\mathrm{sim}\,(c1, c2) = \max_{c \in \mathrm{Ancestor}(c1,c2)} \left(\frac{2\log\left(p(c)\right)}{\log p(c1) + \log p(c2)} \cdot \left(1 - p(c)\right) \right) \tag{1}$$

where *c* represents a set of their common ancestors and $p(c)$ is defined as the fraction of proteins in the GOA database annotated with GO term *c*.

Then, the *funSim* score of a GO category, $\mathrm{GOscore}_{\mathrm{GOcategory}}(X, Y)$, is computed by averaging the *sim* scores between GO annotations of two proteins, *X* and *Y*, in the given category as

$$\begin{aligned}\mathrm{GOscore}_{\mathrm{GOcategory}}\,(X, Y) &= \max\left\{\left(\frac{1}{A_x}\sum_{i=1}^{A_x}\max_{1\le j\le A_y} sim\,(Pxi, Pyj)\right),\right.\\ &\qquad\left.\left(\frac{1}{A_y}\sum_{j=1}^{A_y}\max_{1\le i\le A_x} sim\,(Pxi, Pyj)\right)\right\}\end{aligned} \tag{2}$$

where A_x and A_y are the number of annotations for proteins *X* and *Y*, respectively, in that category, and *Pxi* is ith annotation for protein X, and *Pyj* is jth annotation for protein Y.

For computing the *funSim* score for three categories,

$$\begin{aligned}funSim\,(X, Y) &= \frac{1}{3}\left(\left(\mathrm{GOscore}_{BP}\,(X, Y)\right)^2 + \left(\mathrm{GOscore}_{MF}\,(X, Y)\right)^2\right.\\ &\quad\left. + \left(\mathrm{GOscore}_{CC}\,(X, Y)\right)^2\right)\end{aligned} \tag{3}$$

This is used for “All” in a NaviGO result page and for “BP + MF” is the average is computed for BP and MF.

Acknowledgments

We thank Charles Christoffer for proofreading the manuscript. This work was partly supported by the National Institute of General Medical Sciences of the NIH (R01GM123055) and the National Science Foundation (DBI1262189, IOS1127027, DMS1614777).

References

1. Consortium GO (2013) Gene ontology annotations and resources. Nucleic Acids Res 41(D1):D530–D535
2. Ashburner M, Ball C, Blake J, Botstein D, Butler H, Cherry J, Davis A, Dolinski K, Dwight S, Eppig J (2000) Gene ontology: tool for the unification of biology. The gene ontology consortium. Nat Genet 25(1):25–29. https://doi.org/10.1038/75556
3. Wei Q, Khan IK, Ding Z, Yerneni S, Kihara D (2017) NaviGO: interactive tool for visualization and functional similarity and coherence analysis with gene ontology. BMC Bioinformatics 18(1):177. https://doi.org/10.1186/s12859-017-1600-5
4. Carbon S, Ireland A, Mungall CJ, Shu S, Marshall B, Lewis S (2009) AmiGO: online access to ontology and annotation data. Bioinformatics 25(2):288–289. https://doi.org/10.1093/bioinformatics/btn615
5. Binns D, Dimmer E, Huntley R, Barrell D, O'donovan C, Apweiler R (2009) QuickGO: a web-based tool for gene ontology searching. Bioinformatics 25(22):3045–3046
6. Hawkins T, Luban S, Kihara D (2006) Enhanced automated function prediction using distantly related sequences and contextual association by PFP. Protein Sci 15(6):1550–1556. https://doi.org/10.1110/ps.062153506
7. Hawkins T, Chitale M, Luban S, Kihara D (2009) PFP: automated prediction of gene ontology functional annotations with confidence scores using protein sequence data. Proteins 74(3):566–582. https://doi.org/10.1002/prot.22172
8. Chitale M, Hawkins T, Park C, Kihara D (2009) ESG: extended similarity group method for automated protein function prediction. Bioinformatics 25(14):1739–1745. https://doi.org/10.1093/bioinformatics/btp309
9. Khan IK, Qing W, Kihara D (2015) PFP/ESG: automated protein function prediction servers enhanced with gene ontology visualization tool. Bioinformatics 31(2):271–272. https://doi.org/10.1093/bioinformatics/btu646
10. Pundir S, Martin MJ, O'Donovan C (2017) UniProt protein knowledgebase. Methods Mol Biol 1558:41–55. https://doi.org/10.1007/978-1-4939-6783-4_2
11. Dieterle M, Bauer D, Büche C, Krenz M, Schäfer E, Kretsch T (2005) A new type of mutation in phytochrome A causes enhanced light sensitivity and alters the degradation and subcellular partitioning of the photoreceptor. Plant J 41(1):146–161
12. Nito K, Wong CC, Yates JR, Chory J (2013) Tyrosine phosphorylation regulates the activity of phytochrome photoreceptors. Cell Rep 3(6):1970–1979
13. Al-Sady B, Ni W, Kircher S, Schäfer E, Quail PH (2006) Photoactivated phytochrome induces rapid PIF3 phosphorylation prior to proteasome-mediated degradation. Mol Cell 23(3):439–446
14. Liu X, Chen C-Y, Wang K-C, Luo M, Tai R, Yuan L, Zhao M, Yang S, Tian G, Cui Y (2013) PHYTOCHROME INTERACTING FACTOR3 associates with the histone deacetylase HDA15 in repression of chlorophyll biosynthesis and photosynthesis in etiolated Arabidopsis seedlings. Plant Cell 25(4):1258–1273
15. Ito S, Nakamichi N, Nakamura Y, Niwa Y, Kato T, Murakami M, Kita M, Mizoguchi T, Niinuma K, Yamashino T (2007) Genetic linkages between circadian clock-associated components and phytochrome-dependent red light signal transduction in `Arabidopsis thaliana`. Plant Cell Physiol 48(7):971–983
16. Resnik P (1995) Using information content to evaluate semantic similarity in a taxonomy. arXiv preprint cmp-lg/9511007
17. Lin D (1998) An information-theoretic definition of similarity. In: ICML, vol 1998. Citeseer, pp 296–304

18. Schlicker A, Domingues F, Rahnenführer J, Lengauer T (2006) A new measure for functional similarity of gene products based on gene ontology. BMC Bioinformatics 7:302. https://doi.org/10.1186/1471-2105-7-302
19. Yerneni S, Khan I, Wei Q, Kihara D (2015) IAS: interaction specific GO term associations for predicting protein-protein interaction networks. IEEE/ACM Trans Comput Biol Bioinform. https://doi.org/10.1109/TCBB.2015.2476809
20. Chitale M, Palakodety S, Kihara D (2011) Quantification of protein group coherence and pathway assignment using functional association. BMC Bioinformatics 12(1):373
21. Hawkins T, Chitale M, Kihara D (2010) Functional enrichment analyses and construction of functional similarity networks with high confidence function prediction by PFP. Bmc Bioinformatics 11(1):265
22. Clack T, Shokry A, Moffet M, Liu P, Faul M, Sharrock RA (2009) Obligate heterodimerization of Arabidopsis phytochromes C and E and interaction with the PIF3 basic helix-loop-helix transcription factor. Plant Cell 21(3):786–799
23. Szklarczyk D, Morris JH, Cook H, Kuhn M, Wyder S, Simonovic M, Santos A, Doncheva NT, Roth A, Bork P, Jensen LJ, von Mering C (2017) The STRING database in 2017: quality-controlled protein-protein association networks, made broadly accessible. Nucleic Acids Res 45(D1):D362–D368. https://doi.org/10.1093/nar/gkw937

Chapter 10

Analyzing Glycan-Binding Profiles Using Weighted Multiple Alignment of Trees

Kiyoko F. Aoki-Kinoshita

Abstract

This chapter describes the Multiple Carbohydrate Alignment with Weights (MCAW) tool, which is available as a part of the RINGS (Resource for INformatics of Glycomes at Soka) website. It implements a combination of KCaM (Aoki, Yamaguchi, Ueda, et al., Nucl Acids Res 32:W267–W272, 2004), a pairwise glycan alignment algorithm, and ClustalW (Thompson, Higgins, Gibson, Nucleic Acids Res 22:4673–80, 1994), a weighted multiple protein sequence alignment algorithm. This tool computes the multiple glycan alignment by first computing a guide tree to determine the order by which to progressively add glycans to the multiple alignment. The dynamic programming algorithm results in a glycan profile of the alignment glycans, containing "monosaccharide positions" indicating the ratio of monosaccharides and their glycosidic bonds that are aligned at the corresponding position. This tool has been used to analyze databases of glycan array experimental data, incorporating weights to reflect the biological significance of certain glycans over others. As a result, it has been shown that the alignments obtained are biologically relevant, matching the results as found in the literature.

Key words Glycans, Glycobiology, Complex carbohydrates, Alignments, Glycan recognition

1 Introduction

Tree alignment is known to be an NP-hard problem [1], and methods such as combinatorial optimization and iterative methods are thus used [2]. In this chapter, the web-based tool called the Multiple Carbohydrate Alignment with Weights tool [3], or MCAW, is introduced, which is a combination of KCaM [4], a pairwise glycan alignment algorithm, and ClustalW [5], a weighted multiple protein sequence alignment algorithm. This tool allows anyone to use a web browser to perform multiple alignments of glycans. It is most useful for analyzing glycan array data, which produces large amounts of glycan-binding data for a particular analyte, such as a virus or glycan-binding protein. The list produced from this experiment contains each glycan and their binding affinity values. Major databases containing such experimental data are now

Hiroshi Mamitsuka (ed.), *Data Mining for Systems Biology: Methods and Protocols*, Methods in Molecular Biology, vol. 1807, https://doi.org/10.1007/978-1-4939-8561-6_10, © Springer Science+Business Media, LLC, part of Springer Nature 2018

available for analysis. Therefore, in this chapter, a brief background to glycobiology including these databases is provided, followed by an introduction of the multiple tree alignment algorithm and the actual MCAW web tool which implements this algorithm.

2 Materials

2.1 Brief Background of Glycobiology

Glycobiology is the study of complex carbohydrates, often called *glycans*, which are chains of monosaccharides that often form branched structures [6]. Thus in computer science terms, glycans can be represented as tree structures, where the nodes represent monosaccharides and edges represent glycosidic bonds. Because glycosidic bonds can be formed between any of several hydroxyl groups, or carbon atoms, on monosaccharides, a single monosaccharide may be bonded (via its first or second carbon) with several other monosaccharides via different carbon atoms, usually those with carbon numbers 2, 3, 4, or 6. Figure 1 is an illustration of a disaccharide structure called cellobiose. Two galactose residues are linked in an alpha 1–4 configuration. In this case, the "parent" would be considered the glucose on the right, and the "child" would be the glucose on the left, because glycan structures are usually drawn from right to left. Note that the child residue usually bonds to its parent residue through its carbon 1 atom.

2.2 Brief Background of Glycan Representations

In order to use the MCAW tool, we next describe the text format called KCF, which is one of the formats used to describe glycan structures. IUPAC format is also accepted, and it is described below.

KEGG Chemical Function format, or KCF, was first described for use in the KEGG GLYCAN database [7]. It uses a graph model to describe glycan structures, where nodes represent monosaccharides and edges represent glycosidic bonds. An example of the

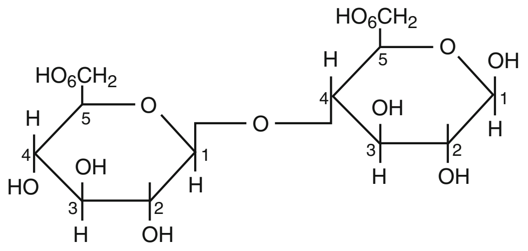


Fig. 1 An illustration of a disaccharide structure, where two glucoses are connected via a glycosidic linkage in alpha 1–4 configuration. In this case, the "parent" would be considered the glucose on the right, and the "child" would be the glucose on the left

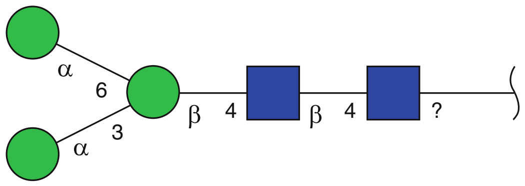


Fig. 2 The core structure common to all *N*-linked glycans

tri-mannose *N*,*N*′-diacetyl chitobiose core structure of *N*-glycans, depicted in Fig. 2, is described in KCF format as follows:

```
ENTRY       G00311                      Glycan
NODE        5
            1   GlcNAc      15     0
            2   GlcNAc       5     0
            3   Man         -5     0
            4   Man        -15     5
            5   Man        -15    -5
EDGE        4
            1     2:b1     1:4
            2     3:b1     2:4
            3     4:a1     3:6
            4     5:a1     3:3
///
```

The KCF file format must contain the NODE and EDGE sections. The ENTRY line is used to denote the name or ID (G00311 in this example) of the represented structure, if any. The NODE section starts with a number indicating the number of residues (monosaccharides and substituents, such as sulfate) being represented. The same number of rows follows this line, each containing the residue number, residue name, and x- and y-coordinates indicating the location at which to draw the residue. The residue name can be any text, although it is recommended that a standard (IUPAC) abbreviated name such as “GlcNAc” or “Neu5Ac” is used for compatibility with other formats. However, if used solely within MCAW, the only requirement would be that the same text is used to represent the same monosaccharide. Similarly, the EDGE section follows with a number indicating the number of bonds in the structure, which will usually be one less than the number of residues. The same number of rows follows, with each row containing the following information: bond number, node numbers of residues being bound, anomeric configuration, and hydroxyl groups in the bond, if any. For example, the following row:

```
3      4:a1     3:6
```

represents the third bond, which indicates that node number 4 (Man at position −15, 5) is bound to node number 3 (Man at position −5, 0) in an alpha 1–3 configuration. If the detailed bond information is unknown, then it is omitted, and if no information is known, the colon is omitted as well.

In order to represent an alignment of glycans, another similar format called PKCF (ProfileKCF) was also defined. The following extensions have been made to KCF:

- The ENTRY field contains an enumerated list, delimited by hyphens, of the names of the glycans being aligned. For example:

```
ENTRY    G1-G2-G3     GlycanProfile
```

- The names of monosaccharides corresponding to each glycan aligned are listed together in each row in the NODE section. For example, given the glycan profile of three glycans G1, G2, and G3, as listed above, the following is an example of the second NODE aligning a mannose with two glucoses, where the first Man is contained in glycan G1, and the two other glucoses are from G2 and G3, respectively:

```
2    1=Man 2=Glc 3=Glc   -8   0
```

  Note that gaps are represented as hyphens (−) and ends, which are nodes that go off the end of the glycan being aligned, are represented as zeros (0).

- The EDGE section contains rows of EDGE information for each glycan, resulting in $g * e$ rows, where g is the number of glycans aligned, and e is the number of EDGEs in the glycan profile. For example, the following three rows would represent the first EDGE in the profile continued from above:

```
1       2-1:b1    1-1:4
1       2-2:b1    1-2:4
1       2-3    1-3
```

  Note that each NODE is referenced by the glycan number, hyphen, and NODE number. Thus 2–1 is NODE 1 in glycan 2. Note the omission of any bond information in the third row.

The International Union of Pure and Applied Chemistry (IUPAC) has published a recommendation for representing carbohydrates as a standardized representation [8]. In this recommendation, a list of symbols for representing monosaccharides is proposed, such as Man for mannose, and Glc for glucose. For chains of monosaccharides, IUPAC provides an extended form, condensed form, and short form as recommendations. Because MCAW supports the condensed form, it is briefly described here.

The IUPAC condensed form omits the ring form unless it is not pyranose, and the anomer is written in parentheses with the carbon numbers of the glycosidic bond. For example, the disaccharide in Fig. 1 would be represented as Glc(α1–4)Glc.

Square brackets are used to indicate branches. Thus the glycan in Fig. 2 would be represented as Man(α1–3)[Man(α1–6)]Man(β1–4)GlcNAc(β1–4)GlcNAc.

The RINGS website [9] provides a convenient drawing tool whereby users can draw a glycan structure and immediately obtain its KCF format. RINGS also provides several format conversion utilities such that glycan data in a particular format can be converted to KCF format. Conversely, there are tools that can translate from KCF to other formats, including IUPAC, and image files. As shown later, the MCAW tool also provides a useful tool so that IUPAC can be used as input, and its image is shown before actually being input into MCAW for execution, so that the user can confirm its accuracy.

2.3 Glycan Databases

2.3.1 KEGG GLYCAN Database

The KEGG GLYCAN database is available at http://www.genome.jp/kegg/glycan/ and is a database of glycan structures, accumulated from the original CarbBank database [10]. It has since been refined and updated with structures from the literature. All the data is freely available from the web and can also be accessed via an application programming interface (API), which is described at http://www.kegg.jp/kegg/rest/keggapi.html.

2.3.2 Consortium for Functional Glycomics (CFG)

The CFG glycan structure database was originally developed as a part of the bioinformatics core of the Consortium for Functional Glycomics. In addition to their own database of glycan-binding proteins and glycosyltransferases, they had initially accumulated *N*- and *O*-linked glycan data from the CarbBank database [10] as well as the glycan structure data from Glycominds, Ltd. Since then, they have added their own synthesized and characterized data from their glycan array library as well as from their glycan profiling data.

All of the glycan profiling data, glycan array data, and knockout mouse data generated by the CFG are also available as data resources over the web. The glycan array data consist of binding affinity information for various glycans with different glycan-binding proteins (GBP), viruses, bacteria, etc. Each data set focuses on a particular GBP or other binder and lists the binding affinity for each glycan structure on the array. Glycan arrays have developed over the years, and the latest version contains over 600 glycan structures [11].

The data provided by the CFG for each glycan-binding experiment is provided as an Excel spreadsheet, with glycans represented in IUPAC format. The relative fluorescence unit (RFU) reflects the binding affinity of the glycan to the analyte.

2.3.3 Lectin Frontier Database: LfDB

In addition to CFG, the National Institute of Advanced Industrial Science and Technology (AIST) has provided their lectin array experimental data available as the Lectin frontier Database, available at http://acgg.asia/db/lfdb/ [12]. AIST has long performed lectin array experiments using frontal affinity chromatography (FAC) and has thus accumulated a large amount of glycan affinity data for a variety of lectins. Currently, it contains 311 lectin information, for which 240 have been analyzed using FAC.

Although the data is not downloadable, a list of the glycans and their dissociation values are provided, from which binding affinity values can be computed. In order to obtain the data for use in MCAW, the user would have to hand-draw each glycan using the DrawRINGS tool in RINGS to obtain KCF formatted data or generate their own list of IUPAC strings for each glycan.

3 Methods

3.1 MCAW Dynamic Programming Algorithm

MCAW performs the following steps to compute a weighted, multiple tree alignment:

1. Compute a distance matrix of distance scores between all pairs of glycans in the input.
2. Create a weighted guide tree based on this distance matrix.
3. According to the guide tree, progressively align the glycans, adding each to the growing multiple alignment profile.

The pairwise alignments are computed according to the KCaM global alignment algorithm in **step 1**. In **step 3**, MCAW implements a dynamic programming algorithm that attempts to align two profiles (multiple alignments) of glycans. A single glycan is the minimum size of a glycan profile. When a profile contains two or more glycans, monosaccharides are aligned to one another, forming what is called a *position*, consisting of the monosaccharides of each glycan aligned in the given profile.

MCAW locally aligns pairs of positions by maximizing the sum of the scores of its children positions. A scoring system is also incorporated so that scores for matching monosaccharide, anomers, and the carbon numbers on either end of the glycosidic linkage can be adjusted. For two glycan profiles A and B, each containing $|A|$ and $|B|$ glycans, respectively, the dynamic programming algorithm is as follows:

$$Q[u,0]=0$$
$$Q[0,v]=0$$
$$Q[u,v]=\max\begin{cases}\max\limits_{v_i\in \mathrm{sons}(v)}\{Q[u,v_i]+d(v)\}\\ \max\limits_{u_i\in \mathrm{sons}(u)}\{Q[u_i,v]+d(u)\}\\ \frac{1}{|A||B|}\left(\sum\limits_{n=1}^{|A|}\sum\limits_{m=1}^{|B|} w(u,v)\,a_n b_m\right)\\ +\max\limits_{\psi\in\mathcal{M}(u,v)}\left\{\sum\limits_{u_i\in \mathrm{sons}(u)} \mathrm{Q}\left[u_i,\psi\left(u_i\right)\right]\right\}\\ 0\end{cases}$$

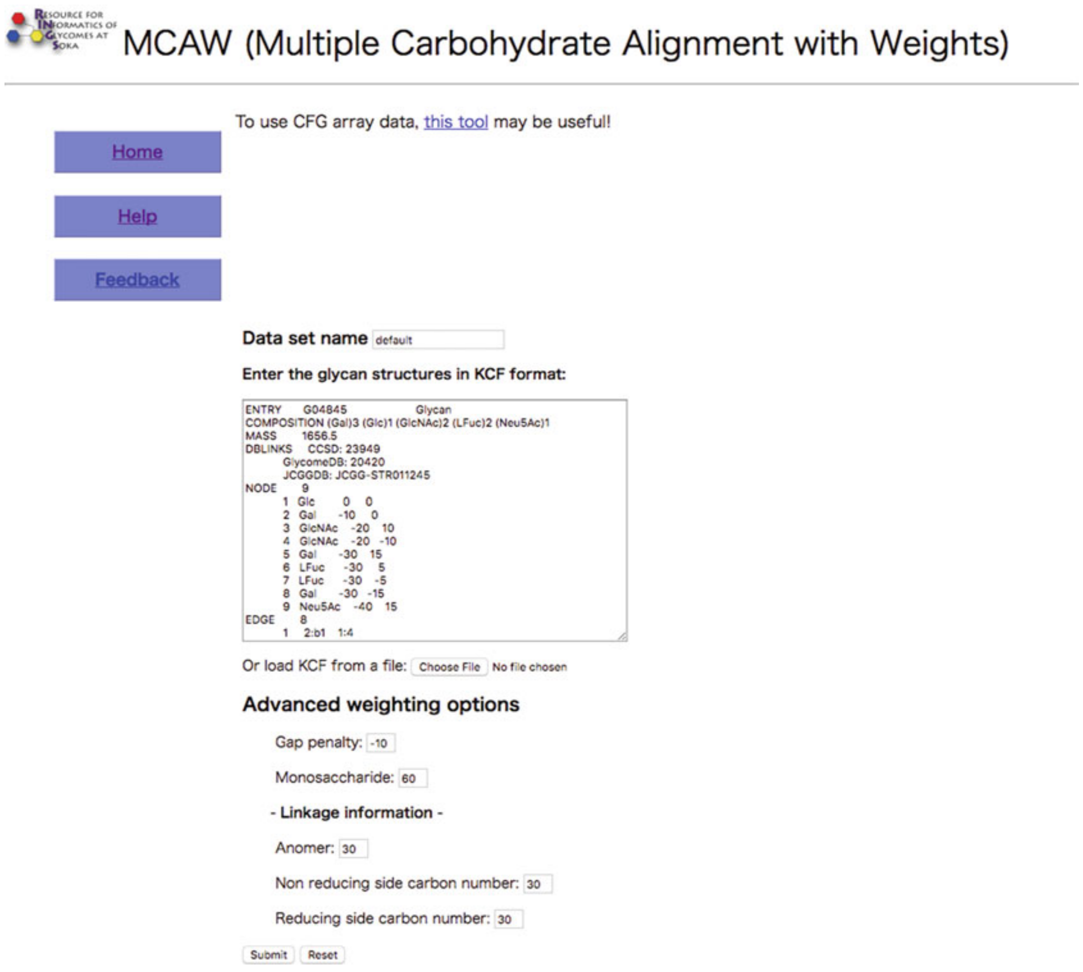
In this equation, u is a position in A and v is a position in B, $Q[u, v]$ computes the alignment score of the subtree profiles rooted at u and v, sons(z) refer to the children of position z, $d(z)$ refers to the gap penalty of deleting position z, a_i (resp. b_j) is the weight of the ith (resp. jth) glycan in profile a (resp. b), and $\mathcal{M}(u, v)$ is the mapping of sons(u) with sons(v). $w(u, v)$ refers to the score of matching nodes u and v, consisting of parameters that weight the matching of monosaccharides, anomers, and carbon numbers of the glycosidic bond linking the nodes in question to their parent nodes.

3.2 MCAW Tool

The MCAW tool is available in the RINGS resource at http://www.rings.t.soka.ac.jp [9]. By registering as a user, any uploaded data is automatically saved in the user data space, which is password protected. Therefore, all executions of any programs on RINGS can be recorded and managed by the user. This allows users to safely manage their data online without having to download the input and results onto their own computer.

The input to the MCAW tool is a list of glycan structures in either KCF or IUPAC condensed format. Using the latter, integral values indicating weights (e.g., glycan-binding affinity) can be specified (*See* **Note 1**). A screenshot of the MCAW tool where default values have been supplied is illustrated in Fig. 3. Given these input, the tool then computes the multiple alignment and draws the aligned profile as an image, as shown in Fig. 4. In this figure, it is apparent that positions 6 through 8 are aligned 100% with a GlcNAc at the root of this subtree profile, whereas the other two terminal branches also contain similar patterns, but with lower alignment values. Thus, the visual representation of highly binding glycans to a particular analyte makes it easier to ascertain its binding determinants. Note that the resulting alignment can be downloaded in PKCF format from the link on the upper right. *See* **Note 2** for issues regarding the output.

For convenience to biologists, a glycan conversion tool was developed such that the glycan array data of the CFG, for example, can be applied directly to MCAW. The link to this tool is provided at the top of the input screen of MCAW (Fig. 3). Users can input the IUPAC condensed-formatted glycans to be used as input, along with weights to the left separated by a tab, to indicate the strength of binding. The weakest would be given a value of 1, and others would be multiples of this value. In most cases, the user would select only the strongly binding glycan structures for analysis. Conversely, it may be possible to select the weakly binding ones to compare those subtrees that must *not* be in a glycan for binding to occur.


MCAW (Multiple Carbohydrate Alignment with Weights)

To use CFG array data, this tool may be useful!

Data set name: default

Enter the glycan structures in KCF format:

```
ENTRY     G04845              Glycan
COMPOSITION (Gal)3 (Glc)1 (GlcNAc)2 (LFuc)2 (Neu5Ac)1
MASS      1656.5
DBLINKS    CCSD: 23949
        GlycomeDB: 20420
        JCGGDB: JCGG-STR011245
NODE      9
        1  Glc      0    0
        2  Gal     -10   0
        3  GlcNAc  -20   10
        4  GlcNAc  -20  -10
        5  Gal     -30   15
        6  LFuc    -30   5
        7  LFuc    -30  -5
        8  Gal     -30  -15
        9  Neu5Ac  -40   15
EDGE      8
        1   2:b1   1:4
```

Or load KCF from a file: No file chosen

Advanced weighting options

Gap penalty: -10

Monosaccharide: 60

- Linkage information -

Anomer: 30

Non reducing side carbon number: 30

Reducing side carbon number: 30

Fig. 3 The MCAW tool where default values have been provided

4 Notes

1. It may be convenient to organize the input data in a spreadsheet software with weights and IUPAC formatted strings in adjacent columns. Users can then copy-and-paste the data into the glycan conversion tool of MCAW. Note that when logged in, the input data is stored on RINGS in KCF format, with each KCF represented repeatedly in the input data according to its weight.
2. The results of the MCAW tool may be empty. This occurs when either a structure of just a single monosaccharide or some error in the input KCF format has been entered. Single monosaccharides cannot be aligned with MCAW at the time of this writing, so they should not be included in the input.

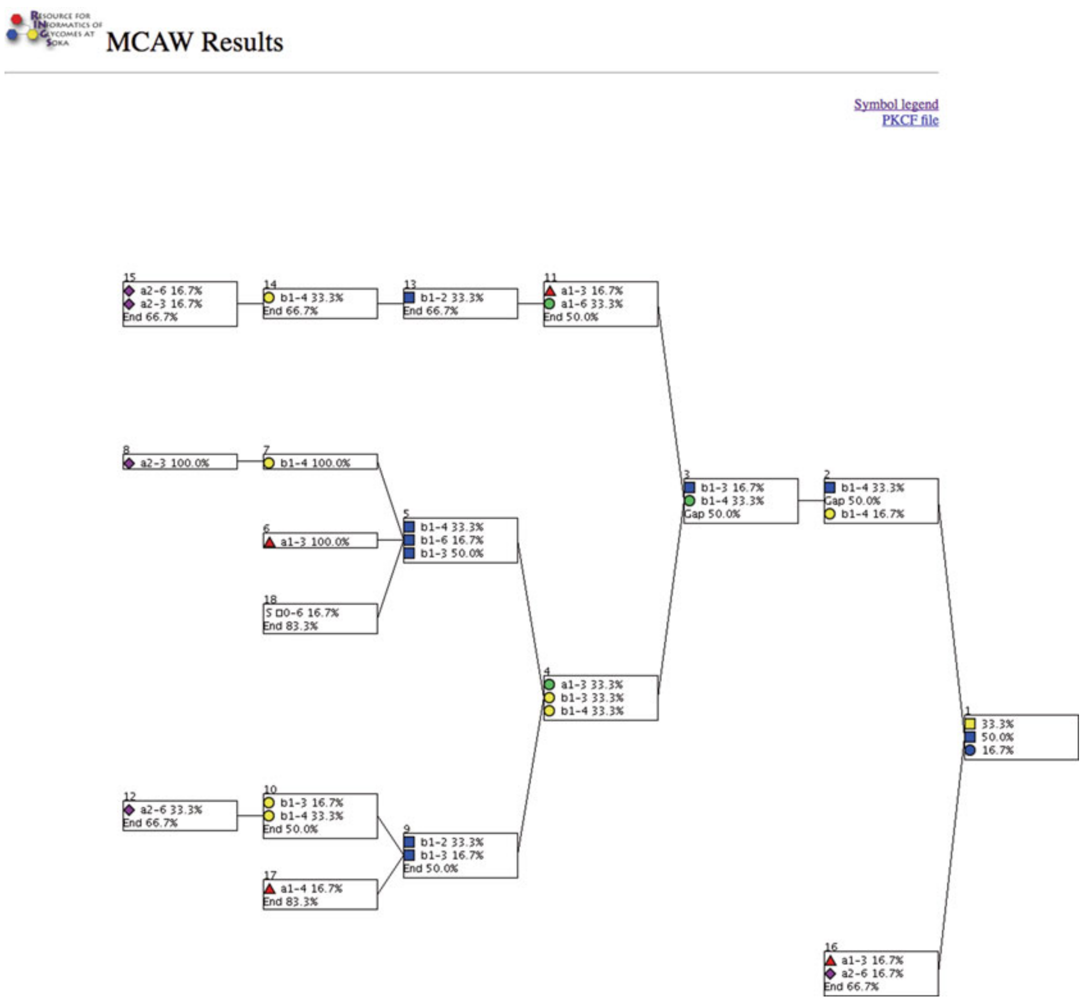

Resource for Informatics of Glycomes at Soka
MCAW Results
Symbol legend
PKCF file
15 a2-6 16.7% a2-3 16.7% End 66.7%
14 b1-4 33.3% End 66.7%
13 b1-2 33.3% End 66.7%
11 a1-3 16.7% a1-6 33.3% End 50.0%
8 a2-3 100.0%
7 b1-4 100.0%
6 a1-3 100.0%
5 b1-4 33.3% b1-6 16.7% b1-3 50.0%
18 S 0-6 16.7% End 83.3%
3 b1-3 16.7% b1-4 33.3% Gap 50.0%
2 b1-4 33.3% Gap 50.0% b1-4 16.7%
4 a1-3 33.3% b1-3 33.3% b1-4 33.3%
1 33.3% 50.0% 16.7%
12 a2-6 33.3% End 66.7%
10 b1-3 16.7% b1-4 33.3% End 50.0%
9 b1-2 33.3% b1-3 16.7% End 50.0%
17 a1-4 16.7% End 83.3%
16 a1-3 16.7% a2-6 16.7% End 66.7%

Fig. 4 A screenshot of the results of the MCAW tool, displaying the multiple alignment as a figure

References

1. Elias I (2006) Settling the intractability of multiple alignment. J Comput Biol 13:1323–1339. https://doi.org/10.1089/cmb.2006.13.1323
2. Batzoglou S (2005) The many faces of sequence alignment. Brief Bioinform 6:6–22. https://doi.org/10.1093/bib/6.1.6
3. Hosoda M, Akune Y, Aoki-Kinoshita KF (2017) Development and application of an algorithm to compute weighted multiple glycan alignments. Bioinformatics 33:btw827. https://doi.org/10.1093/bioinformatics/btw827
4. Aoki KF, Yamaguchi A, Ueda N et al (2004) {KCaM} ({KEGG Carbohydrate Matcher}): a software tool for analyzing the structures of carbohydrate sugar chains. Nucl Acids Res 32:W267–W272
5. Thompson JD, Higgins DG, Gibson TJ (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res 22:4673–4680
6. Varki A, Cummings RD, Esko JD et al (2017) Essentials of glycobiology, 3rd edn. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY
7. Hashimoto K, Goto S, Kawano S et al (2006) KEGG as a glycome informatics

resource. Glycobiology 16:63R–70R. https://doi.org/10.1093/glycob/cwj010

8. McNaught AD (1996) Nomenclature of carbohydrates (IUPAC recommendations 1996). Pure Appl Chem 68:1919–2008. https://doi.org/10.1351/pac199668101919
9. Akune Y, Hosoda M, Kaiya S et al (2010) The RINGS resource for glycome informatics analysis and data mining on the web. OMICS 14:475–486. https://doi.org/10.1089/omi.2009.0129
10. Doubet S, Albersheim P (1992) CarbBank. Glycobiology 2:505
11. Raman R, Venkataraman M, Ramakrishnan S et al (2006) Advancing glycomics: implementation strategies at the {C}onsortium for {F}unctional {G}lycomics. Glycobiology 16:82R–90R
12. Hirabayashi J, Tateno H, Shikanai T et al (2015) The lectin frontier database (LfDB), and data generation based on frontal affinity chromatography. Molecules 20:951–973. https://doi.org/10.3390/molecules20010951

Chapter 11

Analysis of Fluxomic Experiments with Principal Metabolic Flux Mode Analysis

Sahely Bhadra and Juho Rousu

Abstract

In the analysis of metabolism, two distinct and complementary approaches are frequently used: Principal component analysis (PCA) and stoichiometric flux analysis. PCA is able to capture the main modes of variability in a set of experiments and does not make many prior assumptions about the data, but does not inherently take into account the flux mode structure of metabolism. Stoichiometric flux analysis methods, such as Flux Balance Analysis (FBA) and Elementary Mode Analysis, on the other hand, are able to capture the metabolic flux modes, however, they are primarily designed for the analysis of single samples at a time, and assume the stoichiometric steady state of the metabolic network.

We will discuss a new methodology for the analysis of metabolism, called Principal Metabolic Flux Mode Analysis (PMFA), which marries the PCA and stoichiometric flux analysis approaches in an elegant regularized optimization framework. In short, the method incorporates a variance maximization objective form PCA coupled with a stoichiometric regularizer, which penalizes projections that are far from any flux modes of the network. For interpretability, we also discuss a sparse variant of PMFA that favors flux modes that contain a small number of reactions. PMFA has several benefits: (1) it can be applied to large metabolic network in efficient way as PMFA does not enumerate elementary modes, (2) the method is more robust to the steady-state violations than competing approaches, and (3) can compactly capture the variation in the data by a few factors. This chapter will describe the detailed steps how to do the above task on experimental data from fluxomic and gene expression measurements.

Key words Principal component analysis, Metabolic flux analysis, Sparsity

1 Introduction

In the context of transcriptomics and fluxomics, Principal Component Analysis (PCA) has been widely applied [1, 2], where a principal component (PC) identifies linear combinations of genes or enzymatic reactions whose activity changes explain a maximal fraction of variance within the set of samples under analysis. The main goals of PCA in fluxomic data analysis are (1) to identify which parts of the metabolism retain the main variability in flux data and (2) to relate them to the samples, i.e, behavior of the organism for particular experimental condition.

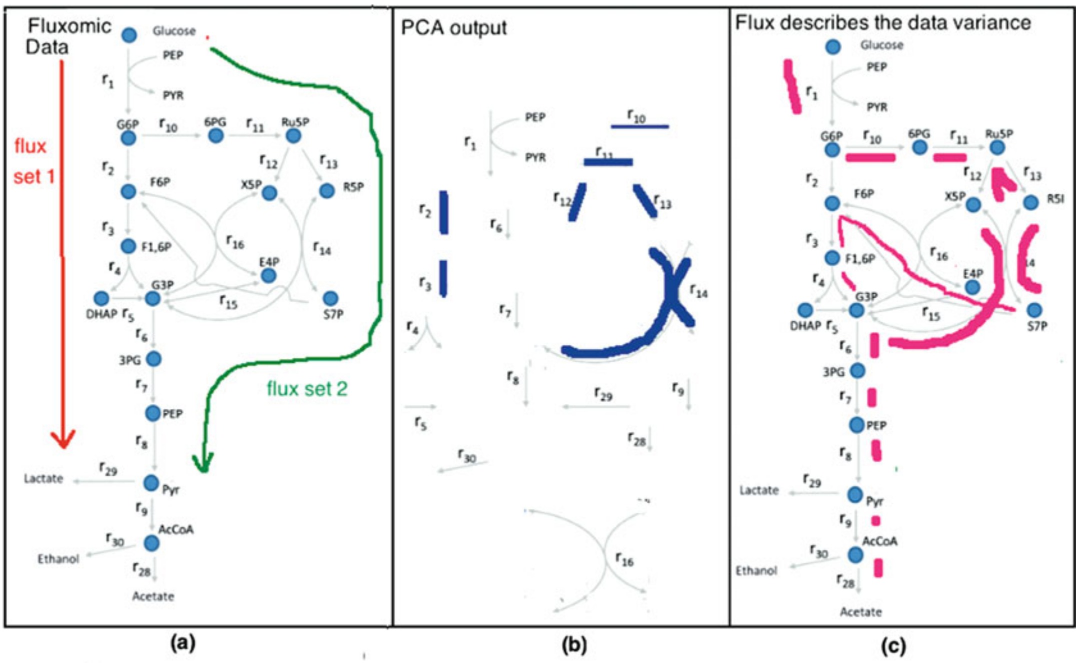


Fig. 1 While doing differential analysis of data given by (**a**), PCA considers reaction independently and extracts the reaction which are important to describe sample variance (as shown in (**b**)). PMFA can be used for differential analysis of fluxomic data to extract interpretable pathways which are responsible for maximum sample variance (as shown in (**c**))

However, in the context of analyzing metabolic networks, PCA has a few limitations [3] as depicted in Fig. 1b: PCA considers reactions independently without considering any other structure or relationship among reactions, including stoichiometric relations implied by metabolic pathways. PCA simply extracts a set of reactions that are important to describe sample variance. Moreover, the principal components output by PCA are known to be generally dense, thus including most of the variables, which precludes their interpretation of pathways of any kind.

This chapter discusses a method called Principal Metabolic Flux Mode Analysis (PMFA) [4], which aims to rectify the deficiencies of the PCA approach. PMFA finds metabolic flux modes that explain the variance in experiments consisting of fluxomic or gene expression data collected from heterogeneous environmental conditions, without requiring a fixed set of predefined pathways to be given. The method can be seen as a cross between Principal Component Analysis (PCA) and stoichiometrix flux analysis: It combines the variance maximization objective of PCA coupled with a stoichiometric regularizer, which penalizes projections that are far from any flux modes of the network.

The benefit of the approach for modelling and biological interpretation is that the sample variance captured by PMFA can be expressed in terms of metabolic pathways or flux modes (Fig. 1c). Let us first briefly review the PCA and Flux Balance Analysis methods, which are frequently used to analyze data arising from metabolic systems, before describing PMFA.

1.1 Principal Component Analysis

Principal component analysis (PCA) is one of the most frequently applied statistical methods in systems biology [1, 2, 5]. PCA is used to reduce the dimensionality of the data while retaining most of the variation in the data-set [6]. This reduction is done by identifying directions, i.e. linear combination of variables, called principal components, along which the variation in the data is maximal. By using a few such components, each sample can be represented by relatively few variables compared to thousands of features. It also helps us to distinguish between biologically relevant variables and noise.

We assume $\mathbf{X} \in \mathbb{R}^{N \times N_r}$ be the data matrix of N_r reactions in N samples, with each entry corresponding to the flux, i.e. the rate of the reaction, through a particular reaction in a particular experiment. We assume throughout the paper that all variables have been centered to have zero empirical mean. The empirical covariance matrix is then given by $\Sigma = \frac{1}{N}\mathbf{X}^T\mathbf{X}$. Denoting $\Sigma_1 = \Sigma$, the 1st principal component (PC) $\mathbf{w}_1$ can be found by solving

$$\mathbf{w}_1 = \underset{w \in \mathbb{R}^{N_r}}{\arg\max}\, \mathbf{w}^T \Sigma_1 \mathbf{w}, \quad \text{s.t. } \|\mathbf{w}\|_2 = 1 \tag{1}$$

Above, $\|\mathbf{w}\|_2 = \sqrt{\mathbf{w}^T\mathbf{w}}$ is the l_2 norm of the vector $\mathbf{w}$. The second PC can be found by applying Eq. (1) on updated covariance matrix using deflation as $\Sigma_2 = (1 - \mathbf{w}_1\mathbf{w}_1^T)\Sigma_1(1 - \mathbf{w}_1\mathbf{w}_1^T)$ [7].

The weights, also called the loadings, of the principal component $\mathbf{w} \in \mathbb{R}^{N_r}$ can be interpreted as the importance of reactions in explaining the variance in fluxomic data. The principal components are generally dense, containing most of the reactions of the metabolic network. Sparse PCA [8] aims to increase the interpretability of PCA by finding principal components that have a small number of non-zero weights through solving the following optimization problem

$$\max_{\mathbf{w}} \mathbf{w}^T \Sigma \mathbf{w} - \gamma \|\mathbf{w}\|_1, \quad \text{s.t. } \|\mathbf{w}\|_2 = 1 \tag{2}$$

where γ is a user defined hyper-parameter which controls the degree of sparsity on PC. However, the principal components extracted by neither method represent metabolic flux modes, and will not in general adhere to the thermodynamic constraints on reaction directions.

1.2 Flux Balance Analysis (FBA)

Flux balance analysis (FBA) [9] is a mathematical method for simulating metabolism in genome-scale reconstructions of metabolic networks. FBA is designed be used to find a flux distribution, in a stoichiometrix steady state, that maximizes a given objective (e.g., growth).

The metabolic balance of the metabolic system is described using the exchange stoichiometric matrix $\mathbf{S} \in \mathbb{R}^{N_m \times N_r}$ [10] which contains transport reactions for inflow of nutrients and output flow of products, but does not contain any external metabolites (as they cannot be balanced). Rows of this matrix represent the N_m internal metabolites, columns present the N_r metabolic reactions including transport reactions, and each element $\mathbf{S}_{m,r}$ shows participation of the mth metabolite in the rth reaction: $\mathbf{S}_{m,r} = 1$ (or -1) indicates that reaction r produces (or consumes) the metabolite m. The value $\mathbf{S}_{m,r} = 0$ indicates metabolite m is not involved in the reaction r. For a flux vector $\mathbf{w}$, $\mathbf{Sw}$ gives the change of metabolic concentration for all metabolites. The metabolic steady-state is assured by imposing a constraint $\mathbf{Sw} = 0$.

FBA solves the following optimization problem

$$\max_{\mathbf{w}} c^T \mathbf{w} \ \text{ s.t. } \mathbf{Sw} = 0 \text{ and } l \leq \mathbf{w} \leq u, \tag{3}$$

that calls for a finding a combination of reaction rates ($\mathbf{w}$) that adhere to stoichiometric steady state as well as upper (u) and lower bounds (l), and maximize the objective given by the combination of coefficients c and the reaction rates $\mathbf{w}$. Typically, the objective is taken as maximization of biomass production, and in this case c is equal to a row in the stoichiometric matrix corresponding to biomass production.

Simulations performed using FBA are computationally inexpensive and can calculate steady-state metabolic fluxes for large models (over 2000 reactions) in a few seconds on modern personal computers. However, as the experimental data is not directly represented in the optimization problem (3), FBA cannot be efficiently used to understand the variability between samples.

1.3 Principal Metabolic Flux Mode Analysis (PMFA)

Here we describe the Principal Metabolic Flux Mode Analysis (PMFA) approach, that combines the PCA and stoichiometric modelling views of metabolism. It finds metabolic flux modes that explain the variance in gene expression or fluxomic data collected from heterogeneous environmental conditions without requiring a fixed set of predefined pathways to be given. Here each principal component, called *principal metabolic flux mode* (PMF), is found by selecting a set of reactions which represents a metabolic flux mode which is approximately in steady state and explains most of the data variability. In addition, we present a sparse variant, called Sparse Principal Metabolic Flux Mode analysis (SPMFA), to further help the interpretation of the principal components.

To obtain meaningful solutions of steady state flux distributions as PC loading one can impose two additional constraints in PCA formulation:

1. the weights associated with irreversible reactions should always be positive, i.e., $w_{\text{ir}} \geq 0$, where ir is an index of an irreversible reaction.

(continued)

2. System is in a steady state, where the internal metabolite concentrations do not change, i.e. the metabolite producing and consuming fluxes cancel each other out: $\mathbf{Sw} = 0$.

Considering (1) and (2) the modified optimization problem for doing PCA with structural constraint is as follows:

$$\begin{aligned} \max_{\mathbf{w}} \quad & \mathbf{w}^T \Sigma \mathbf{w} \\ \text{s.t.} \quad & \mathbf{Sw} = 0 \text{ (stoichiometric steady state)} \\ & \mathbf{w}_{\text{ir}} \geq 0 \text{ (irreversible reactions can have only positive flux)} \\ & \|\mathbf{w}\|_2 = 1 \end{aligned} \tag{4}$$

The constraint $\|\mathbf{w}\|_2 = 1$ restricts the spurious scaling up of the weights in the solution. Here, $\mathbf{Sw} = 0$ is a hard constraint and in practice imposes too much restriction, due to noise in the data, or when the data does not actually arise from steady-state conditions, e.g. given transients or perturbations of the fluxes during the experiment. Numerically, the steady state constraint amounts to a set of linear equations of size $N_M \times N_R$ which makes the problem (Eq. (4)) also computationally hard to solve. Hence instead of considering this hard constraint on the PC loadings we introduce a soft constraint which penalizes the deviation from the steady state. Our aim is to find a flux which optimizes a combination of (1) maximal explained sample variance $\mathbf{w}^T \Sigma \mathbf{w}$ and (2) minimal deviation from a steady-state condition, expressed in the l_2 norm: $\|\mathbf{Sw} - 0\|_2^2 = \|\mathbf{Sw}\|_2^2$. This entails solving the following optimization problem:

$$\boxed{\begin{aligned} \max_{\mathbf{w}} \quad & \mathbf{w}^T \Sigma \mathbf{w} - \lambda \|\mathbf{Sw}\|_2^2 \\ \text{s.t.} \quad & \mathbf{w}_{\text{ir}} \geq 0 \\ & \|\mathbf{w}\|_2 = 1 \end{aligned}} \tag{5}$$

Here λ imposes the degree of hardness of the steady-state constraint. For $\lambda = 0$ Eq. (5) produces loadings similar to PCA with the exception of the reaction directionality constraint. The model will be henceforth denoted as PMFA$^{(l_2)}$. If desirable, we can make our model to disregard reaction directionality simply by dropping the inequality constraints $w_{\text{ir}} > 0$. By dropping the directionality constraint PMFA gives fluxes corresponding to a metabolic network where all reactions are reversible.

The l_2 norm on $\mathbf{Sw}$ in Eq. (5) has the tendency to penalize large steady state deviations in individual metabolites, at the cost of favoring small deviations in many metabolites. This is probably the desired behavior in case the data comes from conditions where there is no subsystems that is considerably farther from steady state than other parts of the system. In order to capture the opposite scenario, where a small subset of metabolites have large deviation from steady state, one can use l_1 norm regularizer on $\mathbf{Sw}$. The l_1 norm regularizer $\|\mathbf{Sw}\|_1$ in Eq. (5) puts the emphasis of pushing most of the steady-state deviations to zero, whilst allowing a few outliers, metabolites that markedly deviate from steady state. Using l_1 regularizer and a trade-off parameter λ we get to solve the following optimization problem:

$$\boxed{\begin{array}{ll} \max\limits_{\mathbf{w}} & \mathbf{w}^T \Sigma \mathbf{w} - \lambda \|\mathbf{Sw}\|_1 \\ \text{s.t.} & \mathbf{w}_{\text{ir}} \geq 0 \\ & \|\mathbf{w}\|_2 = 1 \end{array}} \qquad (6)$$

Here λ imposes the degree of hardness of the steady-state constraint. Similarly to Eq. (5) for $\lambda = 0$ Eq. (6) also produces loadings similar to PCA with selective non-negative constraint. The model will be henceforth denoted as PMFA$^{(l_1)}$. Note that the solution of PMFA$^{(l_2)}$ is more stable than the solution of PMFA$^{(l_1)}$.

1.3.1 Sparse Principal Metabolic Flux Mode Analysis

The above formulation of PCA with stoichiometric constraint still suffers from the fact that each principal component is a linear combination of all possible reaction activities, thus it is often difficult to interpret the results. This problem can be avoided by a variant of PMFA, the sparse principal metabolic flux mode analysis (SPMFA) using an l_1 regularizer [11] on $\mathbf{w}$ to produce modified principal components with sparse loadings.

$$\boxed{\begin{array}{ll} \max\limits_{\mathbf{w}} & \mathbf{w}^T \Sigma \mathbf{w} - \lambda \|\mathbf{Sw}\|_* \\ \text{s.t.} & \mathbf{w}_{\text{ir}} \geq 0 \\ & \|\mathbf{w}\|_1 = C \end{array}} \qquad (7)$$

where $\|\cdot\|_*$ can be any of the l_2 and l_1 norm and C is a user defined hyper-parameter which controls the degree of sparsity in principal metabolic flux (PMF) loadings. Similarly to PMFA, sparse PMFA can also be made consider all reaction reversible by dropping the directionality constraints $w_{\text{ir}} \geq 0$.

1.3.2 Analysis of Metabolic Subsystems

One can apply our method to study differential flux modes only in a subsystem of metabolic network (e.g., central carbon metabolism, redox subsystem, lipid metabolism) by restricting the covariance matrix in objective function to the fluxes in the subsystem, while keeping the stoichiometric regularizer the same as before. Similarly, when some flux measurements are missing, one can change the covariance matrix in the objective function to exclude the fluxes that are missing.

For example, to study the variation within the redox subsystem, let $\mathbf{X}_{rdx}$ contain the columns of $\mathbf{X}$ corresponding to reactions containing redox co-factors, and let $\mathbf{w}_{rdx}$ represent the corresponding part of $\mathbf{w}$. We will consider $\Sigma_{rdx} = \frac{1}{N}\mathbf{X}_{rdx}^T\mathbf{X}_{rdx}$ for finding variance maximizing directions. Hence need to solve

$$\begin{aligned} \max_{w} \quad & \mathbf{w}_{rdx}^T \Sigma_{rdx} \mathbf{w}_{rdx} - \lambda \|\mathbf{S}\mathbf{w}\|_* \\ \text{s.t.} \quad & \mathbf{w}_{\text{ir}} \geq 0 \text{ and } \|\mathbf{w}\|_2 = 1 \end{aligned} \tag{8}$$

Similarly we can also apply SPMFA on metabolic subsystem.

2 Materials

We demonstrate the PMFA methods through two datasets: a simulation case study on *Pichia pastoris* metabolic network, and an experimental study on *Saccharomyces cerevisiae* metabolic network. The details of the datasets are given in the following.

2.1 Datasets

2.1.1 Saccharomyces cerevisiae *Experimental Case Study*

We use the metabolic network for *Saccharomyces cerevisiae* proposed by Hayakawa et al. [12] and 13C isotopic tracer based fluxome data used in [12–14] to demonstrate the methods. The network describes the central cytosolic and mitochondrial metabolism of *S. cerevisiae*, comprising glycolysis, the pentose phosphate pathway, anaplerotic carboxylation, fermentative pathways, the TCA cycle, malic enzyme and anabolic reactions from intermediary metabolites into anabolism [14].

The network contains 42 compounds (30 of which are internal metabolites, which can be balanced for growth) and 47 reactions of which 39 are intracellular. The objective in this case study is to evaluate the performance of PMFA equation (5) on fluxome data and compare it with PEMA and PCA. For PEMA we have used 1182 EMs provided by Stosch et al. [14].

This dataset is available at https://github.com/aalto- ics-kepaco/PMFA/tree/master/Data/SaccharomycesFluxomicData.mat. Table 1 describes its elements.

2.2 Scripts

Matlab software for PMFA and SPMFA (Table 1) are available at https://github.com/aalto-ics-kepaco/PMFA. Both PMFA and SPMFA can be applied on fluxomic and transcriptomic data (Table 2).

Table 1
Description of *Saccharomyces cerevisiae* fluxomic data

Matrix name	Size	Description
StoichiometricMatrix	42×47 double	Stoichiometric information matrix for all reactions
rxnE	47×7 double	Fluxomic data
metNames	42×1 cell	Name of metabolites
rxnNames	47×1 cell	Name of reaction
ExternalmetaboliteID	1×12 double	ID of extra cellular metabolites
EMs	47×1182 double	Elementary modes
L	47×1 double	Lower bound for reaction flux
		($L_r = 0$ for irreversible and $L_r = -1$ for reversible reactions)

Table 2
List of scripts required for using PMFA

Script name	Description
CentralizedExpression.m	To centralized expression/fluxomic data
PCA.m	To find principal components (PC) of a expression/fluxomic data
SPCA.m	To find sparse PC of a expression/fluxomic data
PMFA_L2.m	To find PMF by minimizing squared norm of steady-state deviations of intracellular metabolites
PMFA_L1.m	To find PMF by minimizing l_1 norm of steady-state deviations of intracellular metabolites
SPMFA_L2.m	To find sparse PMF by minimizing squared norm of steady-state deviations of intracellular metabolites
SPMFA_L1.m	To find sparse PMF by minimizing l_1 norm of steady-state deviations of intracellular metabolites
Deflation.m	To deflate a covariance matrix of the variability explained by a PMF

3 Finding Principal Flux Modes

3.1 Data Centralization

PCA, SPCA, PMFA, and SPMFA aim at explaining the main variability in data using a few PCs.

If the original data have non-zero mean, typically the first principal component is heavily biased towards the sample mean, and fails to capture any variability between the sample.

Hence before applying any of the methods, we need to centralize the expression and fluxomic data.

function Ec= CentralizedExpression (Einput,axis)

- Input:
  - Einput: Expression/fluxomic matrix
  - axis: Centralization should be done according to this axis
- output:
  - Ec: Centralized expression/fluxomic matrix

Example in Matlab for centralizing fluxomic matrix of *Saccharomyces cerevisiae* such that for every reaction the sample mean of the expression/flux is zero.

```
>> load('../Data/SaccharomycesFluxomicData.mat');
>> Ec= CentralizedExpression(saccharomyces.rxnE,2);
>> mean(Ec,2)       % this will produce a zero vector
```

3.2 Principal Component Analysis

Principal component analysis (PCA) as given by Eq. (1) and Sparse PCA corresponding to Eq. (2) are implemented in *PCA.m* and *SPCA.m*

function W = PCA(E,num)

- Input:
  - E: Expression/fluxomic matrix
  - num: The number of PCs to be extracted
- Output:
  - W: Each column of this matrix represents PC loadings

Example in Matlab for finding the first 3 PC loadings for *Saccharomyces cerevisiae* fluxomic data:

```
>> load('../Data/SaccharomycesFluxomicData.mat')
>> W = PCA(saccharomyces.rxnE,3)
```

function W = SPCA(Einput,gamma, num)

- Input:
  - Einput: expression/fluxomic matrix
  - gamma: User-defined parameter which indicates the degree of required sparsity in PC loadings. It corresponds to γ in Eq. (2)
  - num: The number of PCs to be extracted
- Output:
  - W: Columns of this matrix represent sparse PC loadings

Example in Matlab for finding the first 3 sparse PC loadings for *Saccharomyces cerevisiae* fluxomic data:

```
>> load('../Data/SaccharomycesFluxomicData.mat')
>> W = SPCA(saccharomyces.rxnE,1,3)
```

3.3 Finding Principal Metabolic Fluxes with PMFA

Principal Flux Mode Analysis as described in Subheading 1.3 is solved by the following scripts. The script *PMFA_L2.m* solves PMFA$^{(l_2)}$ with l_2 regularization on the stoichiometric constraint equation (5) while *PMFA_L1.m* solves PMFA$^{(l_1)}$ with l_1 regularization on stoichiometric constraint equation (6).

Both scripts can be used to also find principal flux modes with respect to a subsystem of metabolic network as described in Subheading 1.3.2. Both take reaction expression/fluxomic matrix corresponding to the defined subsystems along with a list of indices of these reactions in the stoichiometric matrix of the whole system. For the steady state constraint both methods use the **exchange stoichiometric matrix** that contains all reactions (intracellular and transport reactions) in the whole metabolic network but only inter-cellular metabolites as this allows consumption and production of extra-cellular metabolites through the principal flux modes.

function [W,TotalrunTime] = PMFA_L2(Einput, S,lambda,L,U,num,ID)

- Input:
  - Einput: The expression/flux data for reactions in the defined subsystem. The size of this matrix is *number of reactions in subsystem* $\times$ *number of samples*
  - S: Exchange stoichiometric matrix, containing all reactions in whole metabolic network but only inter-cellular metabolites. The size of this matrix is *number of metabolites* $\times$ *number of reactions*

- lambda: User-defined regularization parameter which indicates the degree of penalization of steady-state violations in the PMF loadings. It corresponds to λ in Eq. (5)
- L: Vector containing lower bounds for fluxes in the reactions
- U: Vector containing upper bounds for fluxes in the reactions (Default = vector of all ones)
- num: How many principal flux modes are to be computed (Default = 1)
- ID: If we consider the analysis of a subsystem, then ID contains list of index of target reactions in stoichiometrix matrix (Default = index of all reactions in the metabolic network)

• Output:

- W: Columns of this matrix represent the PMF loadings
- TotalrunTime: Total CPU time taken by PFMA

Example in Matlab for finding the first 3 PMF loadings for *Saccharomyces cerevisiae* fluxomic data when $\lambda = 1$:

```
>> load('../Data/SaccharomycesFluxomicData.mat')

% to find Stoichiometric matrix with all reactions
% in whole  metabolic network but with only
% intercellular metabolites.

>> M = size(saccharomyces.StoichiometricMatrix,1)
>> IDin = setdiff([1:1:M],saccharomyces.Externalmeta
         boliteID)
>> S = saccharomyces.StoichiometricMatrix(IDin,:)

% Find the first 3 PMF loadings when λ = 1

>> [W,TotalTime] = PMFA_L2(saccharomyces.rxnE, ...
    S,1,saccharomyces.L,saccharomyces.U,3)

 % Find the first 3 rev−PMF loadings when lambda = 1
 % Here we set the lower bound for all reactions at
 % negative one

 >> L = −1*ones(N,1)
 >> [W,TotalTime] = PMFA_L2(saccharomyces.rxnE, ...
    S,1,L,saccharomyces.U,3)
```

For this data set optimum value for λ is 5. For $\lambda = 5$ PMF loadings for this data are available at https://github.com/aalto-ics-kepaco/PMFA/tree/master/SuplementaryResult/PMFsaccharoResultandAnalysis.

Table 3
Comparing variance captured and changes of intra-cellular metabolites by PMFA$^{(l_2)}$ for optimum λ and by PCA

	PMFA$^{(l_2)}$ $\lambda = 5$				**PCA**		**PMFA$^{(l_2)}$** $\lambda = 7$		
Principal components	PMF1	PMF2	PMF3	PC1	PC2	PC3	PMF1	PMF2	PMF3
Fraction of sample variance	0.94	0.95	0.96	0.97	0.99	1.00	0.71	0.72	0.72
Metabolites changes ($\|Sw\|_2^2$)	0.27	0.28	0.05	0.28	0.38	1.05	0.09	0.01	0.00

The total percentage of variance captured by up to 1st, 2nd, and 3rd PMFs are 94.11, 94.99, and 95.76. The ℓ_2-norm of steady-state deviations in intracellular metabolites of the PMF are 0.27, 0.28 and 0.05. Table 3 shows the comparison of optimal PFM with PCA. With increase of λ value the resultant PMFs captured lesser variance but, on the other hand, they are very close to steady state fluxes.

function [W,TotalrunTime]= PMFA_L1(Einput,S, lambda,L, U,num,ID)

- Input:
  - Einput: Expression/fluxomic data for reactions in the defined subsystem. The size of this matrix is *number of reactions in subsystem* × *number of samples*
  - S: Stoichiometric matrix with all reactions in whole metabolic network but with only intracellular metabolites. The size of this matrix is *number of metabolites* × *number of reactions*
  - lambda: User-defined regularization parameter which indicates the degree of penalization of steady-state violations in the PMF loadings. It is corresponding to λ in Eq. (6).
  - L: Vector containing lower bounds of fluxes in the reactions
  - U: Vector containing upper bounds of fluxes in the reactions (Default = vector of all ones)
  - num: How many principal flux modes are to be computed (Default = 1)
  - ID: If we consider the analysis of a subsystem, then ID contains list of index of target reactions in stoichiometrix matrix (Default = index of all reactions in the metabolic network).
- Output:
  - W: Columns of this matrix represent the PMF loadings
  - TotalrunTime: Total cpu time taken by PFMA.

Example in Matlab for finding the first 3 PMF loadings for *Saccharomyces cerevisiae* fluxomic data when $\lambda = 1$:

```
>> load('../Data/SaccharomycesFluxomicData.mat')

% Find Stoichiometric matrix with all reactions
% in whole  metabolic network but with only
% intercellular metabolites.

>> M = size(saccharomyces.StoichiometricMatrix,1)
>> IDin = setdiff([1:1:M],saccharomyces.
           ExternalmetaboliteID)
>> S = saccharomyces.StoichiometricMatrix(IDin,:)

% Find the first 3 PMF loadings when λ = 1

>> [W,TotalTime] = PMFA_L1(saccharomyces.rxnE, ...
    S,1,saccharomyces.L,saccharomyces.U,3)

 % Find the first 3 rev−PMF loadings when lambda = 1
 % Here we set the lower bound for all reactions at
 % negative one

 >> L = −1*ones(N,1)
 >> [W,TotalTime] = PMFA_L1(saccharomyces.rxnE, ...
    S,1,L,saccharomyces.U,3)
```

3.4 Finding Sparse Principal Metabolic Fluxes with SPMFA

Sparse Principal Flux Mode Analysis as described in Subheading 1.3.1 is solved by the following scripts. The script *SPMFA_L2.m* solves SPMFA$^{(l_2)}$ with l_2 regularization on the stoichiometric constraint while *SPMFA_L1.m* solves SPMFA$^{(l_1)}$ with l_1 regularization on stoichiometric constraint.

Similarly to PMFA, SPMFA can also find differential flux modes only in a subsystem of metabolic network as described in Subheading 1.3.2.

function [W,TotalrunTime]= SPMFA_L2(Einput,S, lambda,C, L,U,num,ID)

- Input:
  - Einput: The expression/fluxomic data for reactions in the defined subsystem. The size of this matrix is *number of reactions in subsystem* $\times$ *number of samples*.
  - S: Stoichiometric matrix with all reactions in whole metabolic network but with only intracellular metabolites. The size of this matrix is *number of metabolites* $\times$ *number of reactions*
  - lambda: User-defined regularization parameter which indicates the degree of penalization of steady-state violations in the PMF loadings. It is corresponding to λ in Eq. (7).

  - C: The parameter controlling the sparsity; PMFs are more sparse for smaller C.
  - L: Vector containing lower bounds of fluxes in the reactions
  - U: Vector containing upper bounds of fluxes in the reactions (Default = vector of all ones)
  - num: How many principal flux modes are to be computed (Default = 1)
  - ID: If we consider the analysis of a subsystem, then ID contains list of index of target reactions in stoichiometrix matrix (Default = index of all reactions in metabolic network).

- Output:
  - W: Columns of this matrix represent the sparse PMF loadings
  - TotalrunTime: Total CPU time taken by PFMA

Example in Matlab for finding the first 3 PMF loadings for *Saccharomyces cerevisiae* fluxomic data when $\lambda = 1$:

```
>> load('../Data/SaccharomycesFluxomicData.mat')

% to find Stoichiometric matrix with all reactions
% in whole  metabolic network but with only
% intercellular metabolites.

>> M = size(saccharomyces.StoichiometricMatrix,1)
>> IDin = setdiff([1:1:M],saccharomyces.
           ExternalmetaboliteID)
>> S = saccharomyces.StoichiometricMatrix(IDin,:)

% Find the first 3 PMF loadings when λ = 1 and C = 3

>> [W,TotalTime] = SPMFA_L2(saccharomyces.rxnE, ...
    S,1,3,saccharomyces.L,saccharomyces.U,3)

 % Find the first 3 rev−SPMF loadings when λ = 1
 % Here we set the lower bound for all reactions at
 % negative one

 >> L = −1*ones(N,1)
 >> [W,TotalTime] = SPMFA_L2(saccharomyces.rxnE, ...
    S,1,3,L,saccharomyces.U,3)
```

function [W,TotalrunTime]= SPMFA_L1(Einput,S, lambda,C, L,U,num,ID)

- Input:
  - Einput: The expression/fluxomic data for reactions in the defined subsystem. The size of this matrix is *number of reactions in subsystem* × *number of samples.*
  - S: Stoichiometric matrix with all reactions in whole metabolic network but with only intracellular metabolites. The size of this matrix is *number of metabolites* × *number of reactions*
  - lambda: User-defined regularization parameter which indicates the degree of penalization of steady-state violations in the PMF loadings. It is corresponding to λ in Eq. (7).
  - C: The parameter controlling the sparsity; PMFs are more sparse for smaller C.
  - L: Vector containing lower bounds of fluxes in the reactions
  - U: Vector containing upper bounds of fluxes in the reactions (Default = vector of all ones)
  - num: How many principal flux modes are to be computed (Default = 1)
  - ID: If we consider the analysis of a subsystem, then ID contains list of index of target reactions in stoichiometrix matrix (Default = index of all reactions in metabolic network).
- Output:
  - W: Columns of this matrix represent the sparse PMF loadings
  - TotalrunTime: Total cpu time taken by PFMA

Example in Matlab for finding the first 3 sparse PMF loadings for *Saccharomyces cerevisiae* fluxomic data when $\lambda = 1$ and $C = 1$:

```
>> load('../Data/SaccharomycesFluxomicData.mat')

% Find Stoichiometric matrix with all reactions
% in whole  metabolic network but with only
% intercellular metabolites.

>> M = size(saccharomyces.StoichiometricMatrix,1)
>> IDin = setdiff([1:1:M],saccharomyces.
          ExternalmetaboliteID)
>> S = saccharomyces.StoichiometricMatrix(IDin,:)
```

(continued)

```
% Find the first 3 PMF loadings when λ = 1

>> [W,TotalTime] = SPMFA_L1(saccharomyces.rxnE, ...
    S,1,3,saccharomyces.L,saccharomyces.U,3)

 % Find the first 3 rev–PMF loadings when lambda = 1
 % Here we set the lower bound for all reactions at
 % negative one

 >> L = −1*ones(N,1)
 >> [W,TotalTime] = SPMFA_L1(saccharomyces.rxnE, ...
    S,1,3,L,saccharomyces.U,3)
```

3.5 Deflating the Covariance Matrix

To obtain a *multi-factor* PMFA model, i.e. a model containing several PMFs jointly representing the data, we follow a approach similar to some PCA algorithms, namely the deflation of the covariance matrix. However, due to additional stoichiometric constraint here we deal with a sequence of non-orthogonal vectors, $[\mathbf{w}_1, \ldots, \mathbf{w}_d]$ hence we must take care to distinguish between the variance explained by a vector and the additional variance explained, given all previous vectors. We have used orthogonal projection for deflating the data matrix [7]. This also maintains positive definiteness of covariance. For every iteration $d+1$ we first transfer already found principal flux modes $W \in \mathbb{R}^{N_R \times d}$ to a set of orthogonal vectors, $\{q_1, \ldots, q_d\}$.

$$q_d = \frac{(I - Q_{d-1}Q_{d-1}^T)\mathbf{w}_d}{\|(I - Q_{d-1}Q_{d-1}^T)\mathbf{w}_d\|} \tag{9}$$

where $q_1 = \mathbf{w}_1$, and $q_1, \ldots, q_d$ form the columns of Q_d. $q_1, \ldots, q_d$ form an orthonormal basis for the space spanned by $\mathbf{w}_1, \ldots, \mathbf{w}_d$. Then the Schur complement deflation of covariance matrix is done by

$$\Sigma_{d+1} = \Sigma_d - \frac{\Sigma_d q_d q_d^T \Sigma_d}{q_d^T \Sigma_d q_d} \tag{10}$$

function [Covdef,Q] = Deflation(Cov,W)

- Input:
  - Cov: Covariance of Expression/fluxomic data for reactions in the defined subsystem. The size of this matrix is *number of reactions in subsystem* × *number of reactions in subsystem.*
  - W: Columns of this matrix represent the sparse PMF loadings

- Output:
  - Covdef: Deflated Covariance matrix.
  - Wn: Orthogonal transformation of PMFs. It is Q in Eq. (9)

Example in Matlab for finding the first 2 PMF loadings for *Saccharomyces cerevisiae* fluxomic data using deflation of expression matrix:

```
>> load('../Data/SaccharomycesFluxomicData.mat')

% Find Stoichiometric matrix with all reactions
% in whole  metabolic network but with only
% intercellular metabolites.

>> M = size(saccharomyces.StoichiometricMatrix,1)
>> IDin = setdiff([1:1:M],saccharomyces.
            ExternalmetaboliteID)
>> S = saccharomyces.StoichiometricMatrix(IDin,:)

% Find the first  PMF loadings when λ = 1

>> [W,TotalTime] = PMFA_L2(saccharomyces.rxnE, ...
    S,1,saccharomyces.L,saccharomyces.U,1)

% Data centralization

>> E=CentralizedExpression(saccharomyces.rxnE,2);

% covariance

>> CovE=E*E';

% Find Covariance matrix deflated by the first PMF

>>[Covdef,Q] = Deflation(Cov,W)
```

3.6 Computing the Total Variance Captured by PMFs

To find the total sample variance explained by first few PMFs, we first transfer already found principal flux modes $W \in \mathbb{R}^{N_R \times d}$ to a set of orthogonal vectors, $\{q_1, \ldots, q_d\}$ using Eq. (10). Then we sum up the variance captured by $\{q_1, \ldots, q_d\}$. The script *varianceCap.m* calculate total cumulative variance captured by up to kth PFMs.

function [v] = varianceCap(E,W)

- Input:
  - E: The expression/fluxomic data for reactions in the defined subsystem. The size of this matrix is *number of reactions in subsystem* × *number of samples.*
  - W: Columns of this matrix represent the PMF loadings

- Output:
  - v: A vector where the *k*th element shows the total fraction of sample variance captured by all PMF upto the *k*th PMF together.

Example in Matlab for finding the variance captured by first 3 PMF loadings for *Pichia pastoris* simulation data:

```
>> load('../Data/SaccharomycesFluxomicData.mat')

% Find Stoichiometric matrix with all reactions
% in whole  metabolic network but with only
% intercellular metabolites.

>> M = size(saccharomyces.StoichiometricMatrix,1)
>> IDin = setdiff([1:1:M],saccharomyces.
          ExternalmetaboliteID)
>> S = saccharomyces.StoichiometricMatrix(IDin,:)


% Find the first 3  PMF loadings when λ = 1

>> [W,TotalTime] = PMFA_L2(saccharomyces.rxnE, ...
    S,1,saccharomyces.L,saccharomyces.U,3)

% Find total variance captured by PMFs

>> [v] = varianceCap(saccharomyces.rxnE,W)
```

4 Further Guidelines

4.1 Directionality Constraints in PMFA and SPMFA

The benefit of the directionality constraint is that the results are interpretable as flux modes with thermodynamically correct reaction directions. The directionality constraint also has been observed to increase the stability of PMFA. However, insisting on interpretability of flux modes with correct directionality may lose some power of explaining the variance. Hence dropping the directionality constraints may sometimes give further insight on the main sources of variation.

4.2 Finding Mean Flux Modes

PMFA is similar in philosophy with the differential expression analyses where genes that vary between experiments are of interest. PMFA is not very well suitable for the analysis of a single sample at a time. If one uses the method for technical or biological replicates, the resulting flux modes will mostly capture the pattern in the noise. Also, the method is not designed to capture the main active flux modes but to capture fluxes that explain differences between different samples. However, it is easy to modify the

PMFA objective so that it finds the average flux mode in a set of experiments, essentially replacing the covariance with the mean.

4.3 Analysis of Non-linear Trajectories

PMFs are good for explaining the main linear directions of variance, interpretable as pathways, in the samples but are not expected to fully explain complex nonlinear trajectories, e.g. time course data.

4.4 Finding the Optimal Models

The objective function is non-convex equation (5), and can be interpreted as difference of two differentiable convex functions. This type of optimization problem is known as Difference of Convex functions (DC) program. We have used the convex-concave procedure (CPP), a local heuristic that utilizes the tools of convex optimization to find local optima of difference of convex functions (DC) programming problems [15]. Using the CCP method we solved Eq. (5) by solving following convex approximation (quadratic program) in each iteration t:

$$\begin{aligned} \mathbf{w}^{t+1} = \arg\min_{\mathbf{w}} \ & \frac{\lambda}{2}\|\mathbf{S}\mathbf{w}^{T}\|_q - \mathbf{w}^{t^T}\Sigma_E\mathbf{w} \\ \text{s.t.} \ & \mathbf{w}_{\text{ir}} \geq 0 \end{aligned} \tag{11}$$

followed by projecting $\mathbf{w}^{t+1}$ on $\|w\|_p = C$. The norms $p, q \in \{1, 2\}$ are chosen according to the desired model.

To find a good local optimum, we repeat the above optimization with different random starting points, and take the best local minimum as the solution. In our experiments we used 100 repetitions (Rep=100).

4.5 Estimating Optimal Values for User-Defined Parameters

The performance of SPCA, PMFA, and SPMFA depends on the value of used defined parameters, namely the regularization parameters λ for PMFA and SPMFA, and the level of sparsity C for SPMFA, and γ for SPCA. One should carefully choose those parameters to find correct differential fluxes.

With $\gamma = 0$, SPCA corresponds to normal PCA. With the increase of γ we increase the sparseness in PC loadings and hence increase the interpretability of it but decrease the amount of sample variance described by the PC. Hence too high value in γ is not good. Similarly, the parameter C controls the sparsity in SPMFA. Here with decrease of C the sparsity in loading increases.

The deviation from the steady-state in PMFA and SPMFA is controlled by the regularization parameter $\lambda \geq 0$: high values of λ give low deviation from steady-state and vice versa. In particular on the fluxomic datasets, relatively heavy regularization can be applied without decrease of variance explained (cf. Fig. 2). By change of the regularization parameter λ, the statistics of PMFA exhibit a continuous transition from fully steady state flux modes ($\|\mathbf{S}\mathbf{w}\|_2^2 = 0$) to the PCA augmented with reaction directionality constraints.

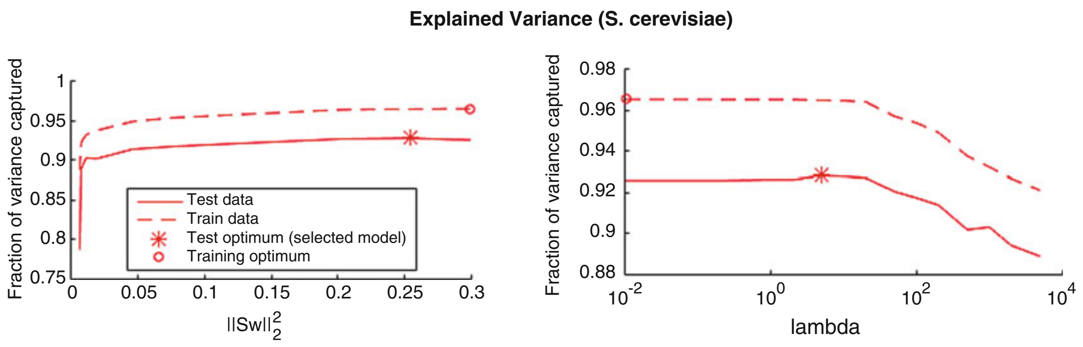


Fig. 2 Figure shows the variance captured by the first PMF on training and test data for various values of λ and also corresponding $\|Sw\|_2$. Optimum λ is chosen to be 5

The optimum levels of the Stoichiometric constraints can be set by cross-validation maximizing the *fraction of sample variance captured* on test samples

$$\text{Fraction of variance} = \frac{\mathbf{w}^T \Sigma \mathbf{w}}{\text{Trace}(\Sigma)},$$

which is a classic measure used with PCA and related approaches. Above, $\mathbf{w}$ is the PC computed from the training data, and Σ is the co-variance matrix of the test sample. Leave-One-Out (LOO) cross-validation can be used on smaller data-sets and less time-intensive techniques, such as fivefold cross-validation on larger datasets.

For *Saccharomyces cerevisiae* fluxomic data we have selected optimum parameter using LOO cross validation. Figure 2 shows the variance captured by the first PMF on training and test data for various values of λ and also corresponding $\|Sw\|_2$. Optimum λ is chosen to be 5.

4.6 PMFA on Expression Data

To analyze gene expression data with PMFA and SPMFA, one needs to map the gene expression to the corresponding biochemical reactions. One can transfer the expression matrix from gene to reaction-wise with help of gene rules defined in metabolic network [16, 17]. Gene rules are Boolean rules that determine the effect of the expression of regulatory genes on the activity of reactions in the metabolic network.

5 Conclusion

In this chapter we have demonstrated the analysis of fluxomic data with Principal Metabolic Flux Mode Analysis, PMFA [4]. Through the combination of stoichiometric flux analysis and principal component analysis, the PMFA finds flux modes that explain most of the variation in fluxes in a set of samples. Unlike most stoichiometric modeling methods, PMFA is not tied to the steady-state assumption, but can automatically adapt—by the change of a single regularization parameter—to deviations from the stoichiometric steady-state, whether they are due to measurement errors, biological variation, or other causes. PMFA can also be applied time course and gene expression data. On the other hand, SPMFA that allows us to discover flux modes with a small fraction of reactions activated, thus could be interpreted as pathways. Thus, SPMFA is effective in the analysis large metabolic networks.

References

1. Barrett CL, Herrgard MJ, Palsson B (2009) Decomposing complex reaction networks using random sampling, principal component analysis and basis rotation. BMC Syst Biol 3(1):30
2. Yao F, Coquery J, Lê Cao K-A (2012) Independent principal component analysis for biologically meaningful dimension reduction of large biological data sets. BMC Bioinf 13(1):1
3. Folch-Fortuny A, Marques R, Isidro IA, Oliveira R, Ferrer A (2016) Principal elementary mode analysis. Mol BioSyst 12(3):737–746
4. Bhadra S, Blomberg P, Castillo S, Rousu J (2017) Principal metabolic flux mode analysis. bioRxiv, p. 163055
5. Ma S, Dai Y (2011) Principal component analysis based methods in bioinformatics studies. Brief Bioinform 12(6):714–722
6. Shlens J (2014) A tutorial on principal component analysis. Preprint, arXiv:1404.1100
7. Mackey LW (2009) Deflation methods for sparse PCA. In: Advances in neural information processing systems, pp. 1017–1024
8. Zou H, Hastie T, Tibshirani R (2006) Sparse principal component analysis. J Comput Graph Stat 15(2):265–286
9. Orth JD, Thiele I, Palsson BØ (2010) What is flux balance analysis? Nat Biotechnol 28(3):245–248
10. Raman K, Chandra, N (2009) Flux balance analysis of biological systems: applications and challenges. Brief Bioinform 10(4):435–449
11. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Ser B Methodol **58**(1):267–288
12. Hayakawa K, Kajihata S, Matsuda F, Shimizu H (2015) 13 c-metabolic flux analysis in s-adenosyl-l-methionine production by Saccharomyces cerevisiae. J Biosci Bioeng 120(5):532–538
13. Frick O, Wittmann C (2005) Characterization of the metabolic shift between oxidative and fermentative growth in Saccharomyces cerevisiae by comparative 13 c flux analysis. Microb Cell Factories 4(1):1
14. von Stosch M, de Azevedo CR, Luis M, de Azevedo SF, Oliveira R (2016) A principal components method constrained by elementary flux modes: analysis of flux data sets. BMC Bioinf 17(1):200
15. Lipp T, Boyd S (2016) Variations and extension of the convex–concave procedure. Optim Eng 17(2):263–287
16. Herrgård MJ, Lee B-S, Portnoy V, Palsson BØ (2006) Integrated analysis of regulatory and metabolic networks reveals novel regulatory mechanisms in Saccharomyces cerevisiae. Genome Res 16(5):627–635
17. Jensen PA, Lutz KA, Papin JA (2011) Tiger: toolbox for integrating genome-scale metabolic models, expression data, and transcriptional regulatory networks. BMC Syst Biol 5(1):147

Chapter 12

Analyzing Tandem Mass Spectra Using the DRIP Toolkit: Training, Searching, and Post-Processing

John T. Halloran

Abstract

Tandem mass spectrometry (MS/MS) is a high-throughput technology used to identify the proteins present in a complex, biological sample. Critical to MS/MS is the ability to accurately identify the peptide responsible for producing each observed spectrum. Recently, a dynamic Bayesian network (DBN) approach was shown to achieve state-of-the-art accuracy for this peptide identification problem. Modeling the stochastic process by which a peptide produces an MS/MS spectrum, this DBN for Rapid Identification of Peptides (DRIP) uses probabilistic inference to efficiently determine the most probable alignment between a peptide and an observed spectrum. DRIP's dynamic alignment strategy improves upon standard "static" alignment strategies, which rely on fixed quantization of the temporal axis of MS/MS data, in several significant ways. In particular, DRIP allows learning non-linear shifts of the temporal axis and, owing to the generative nature of the model, accurate feature extraction for substantially improved discriminative analysis (i.e., Percolator post-processing), all of which are supported in the DRIP Toolkit (DTK). Herein we describe how DTK may be used to significantly improve MS/MS identification accuracy, as well as DTK's interactive features for fine-grained analysis, including on the fly inference and plotting attributes.

Key words DRIP, Tandem mass spectrometry, Shotgun proteomics, Dynamic Bayesian networks, DBNs, Graphical models

1 Introduction

The most widely used method to identify the proteins present in a complex biological sample consists of liquid chromatography followed by tandem mass spectrometry (MS/MS), commonly called *shotgun proteomics*. With the original complex sample as input, a typical shotgun proteomics experiment begins by first digesting the proteins into peptides using a digesting agent (such as trypsin). Each peptide is then loaded into a mass spectrometer via liquid chromatography and undergoes two rounds of mass spectrometry. The first round of mass spectrometry (called the *MS1* phase) measures the precursor mass and charge of the intact peptide. In the second round of mass spectrometry (called the

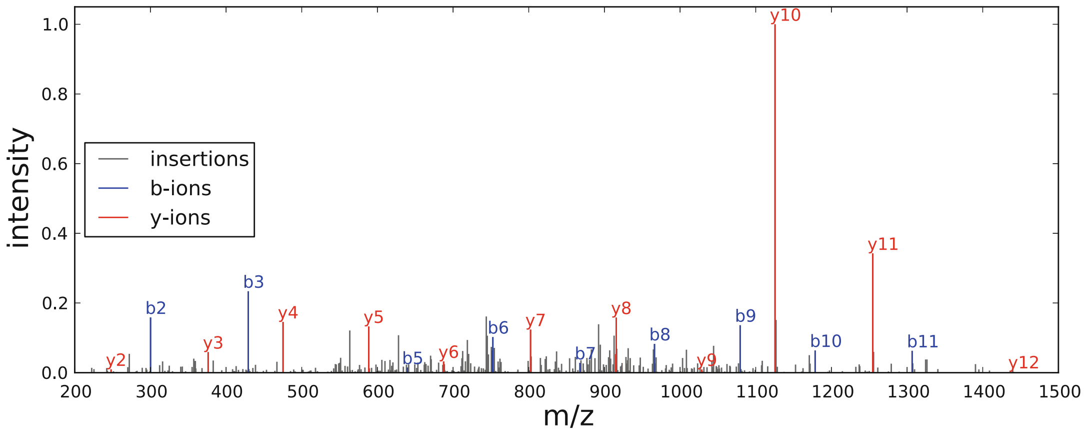


Fig. 1 Example tandem mass spectrum, the generating peptide of which is LWEPLLDVLVQTK with precursor charge 2+. The b- and y-ion peaks of the theoretical spectrum are colored blue and red, respectively. The remaining spurious peaks are called insertions, colored in gray

MS2 phase), the intact peptide is ionized and broken down into its constituent fragment ions, resulting in a spectrum where the x-axis measures *mass-to-charge* (abbreviated as *m/z*) and the y-axis measures relative abundance at an m/z value (i.e., intensity). An example observed spectrum is illustrated in Fig. 1. At the output of this entire process is a collection of spectra typically numbering in the tens-to-hundreds of thousands, where each spectrum in the collection is representative of a peptide from the original complex sample. A key challenge in shotgun proteomics is thus accurately identifying each peptide responsible for generating each observed MS/MS spectrum.

The most accurate methods identify MS/MS spectra by searching a database of peptides. Candidate peptides from the database are mapped to a spectrum of idealized fragment ions (called the *theoretical spectrum*, comprised of suffix and prefix ions called *b*- and *y-ions*, respectively) and compared to an observed spectrum using a scoring function. The pair consisting of a candidate peptide and observed spectrum is typically referred to as a *peptide-spectrum match* (PSM). Many scoring functions have been proposed, all of which discretize the temporal axis in order to compare the theoretical and observed spectra to compute a score [2, 4, 5, 8, 10, 14, 15]. Using dynamic Bayesian networks (DBNs), a DBN for Rapid Identification of Peptides (DRIP) takes an alternative approach, whereby quantization of observed spectrum measurements is avoided altogether.

DBNs are probabilistic models of temporal data for which highly efficient algorithms allow tractable inference. For a given candidate peptide and observed spectrum, the temporal sequence modeled in DRIP is the observed spectrum, where observed peaks are scored in their natural resolution using Gaussians centered at theoretical peak locations. Each time instance, referred to as a *frame*, contains an observed peak and several random variables dictating the current theoretical peak being accessed (used to score the frame's observed peak measurements). DRIP explicitly models two prevalent phenomena as random variables in each frame: the presence of a spurious observed peak (called an *insertion*) and the absence of theoretical peaks (called *deletions*). Defining an *alignment* as a scoring of observed peaks by a sequence of theoretical peaks or, equivalently, as the corresponding sequences of insertions and deletions, DRIP thus models all possible alignments for a given PSM. DBN inference is performed to efficiently calculate the most-probable such alignment, also known as DRIP's *Viterbi path*. The log-likelihood of the Viterbi path (called the *Viterbi score*) is then used to score peptides during a DRIP database search. For further, extensive details of DRIP, including a general overview of DBNs, readers are directed to [7].

The generative nature of DRIP's scoring function affords several important advantages over other existing approaches. Owing to the use of Gaussians to score m/z measurements in their natural resolution rather than relying on quantization, the expected positions of theoretical peaks (i.e., a theoretical peak's corresponding DRIP Gaussian mean) may be learned given high-quality training data. In cases where there is low precision when collecting m/z measurements, such as when searching low-resolution MS2 spectra, the DRIP learned parameters are much more accurate than scoring peptides using evenly spaced theoretical peak locations (as is done with scoring functions relying on quantization) [6]. Owing to DRIP's dynamic alignment strategy, a PSM's Viterbi path returns an abundance of information detailing how a peptide aligns with an observed spectrum. By extracting information from DRIP's output Viterbi path, we are able to derive features which significantly increase recalibration accuracy for post-processors [7] such as Percolator [9].

The previously discussed DRIP utilizations for improved MS/MS analysis (i.e., pre-search training, database search, and post-search feature extraction) are readily available using the open source DRIP Toolkit (DTK). DTK is written in Python 2.7 around a highly efficient DBN inference engine, the Graphical Models Toolkit (GMTK) [1]. In this work, we provide a practical guide to effectively using DTK for each stage of MS/MS analysis.

2 Methods

DTK is comprised of major modules for pre-search learning of parameters (using `dripTrain`), database search with peptide's ranked by their DRIP Viterbi score (using `dripSearch`), and extracting features of a search algorithm's output PSMs for

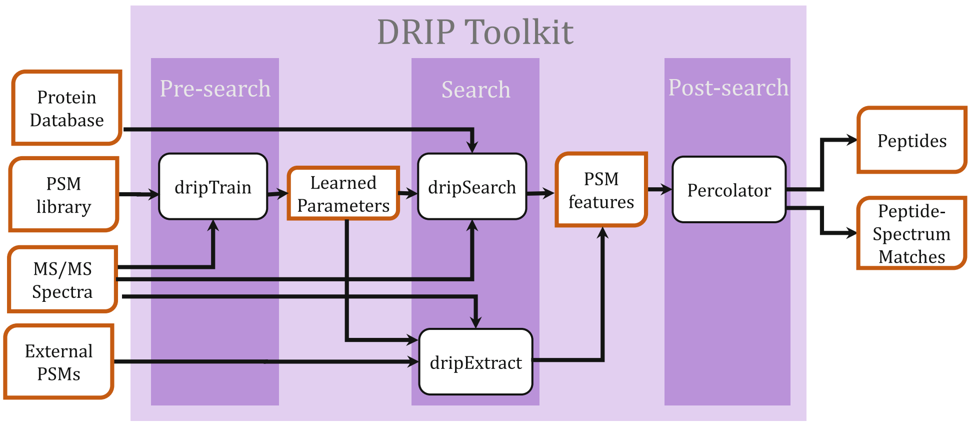


Fig. 2 Flowchart illustrating the inputs and outputs of the major DTK MS/MS search modules

Percolator post-processing (using `dripExtract`). Figure 2 illustrates the inputs and outputs of these major modules. Further modules include `dripDigest`, a protein-digestion Python library used prior to running a search, and `dtk`, a Python library for plotting and interactive analysis using the Python shell (including instantiating a DRIP PSM object and calculating its Viterbi path in real time). For each module, a comprehensive list of input and output parameters is accessible via the command-line or in the respective online man page (*see* **Note 1**).

2.1 Installation

DTK is written in Python and requires the following: g++, GMTK, Python 2.7, `argparse` and `numpy` Python packages, and `SWIG`. A POSIX environment is required by GMTK, so Windows users will additionally need to install Cygwin. From here on, we assume commands are run on a POSIX command-line using a Bash shell (denoted by a preceding `$`), unless otherwise specified.

Once all prerequisite software is installed, download and unzip the latest DTK release from: jthalloran.bitbucket.io/dripToolkit. Navigate to the unzipped directory and run the following:

```
$ cd pyFiles/pfile
$ swig -c++ -python libpfile.i
$ CC=g++ python setup.py build_ext -i
```

This builds and links the `pFile` library for use in Python (*see* **Note 2**), necessary to efficiently represent MS/MS data for GMTK. To test that the package was correctly linked, the following should run without error:

```
$ ./test.py
```

If DTK's plotting capabilities are desired (described in Subheading 2.5), `matplotlib` and/or Lorikeet must be installed.

2.2 Pre-search Parameter Learning Using dripTrain

Given a high-quality set of PSMs, the `dripTrain` module learns DRIP Gaussian parameters using the Expectation-Maximization (EM) algorithm [3]. The learned DRIP Gaussian means correspond to the expected m/z locations of theoretical peaks and the learned DRIP Gaussian variance corresponds to the uncertainty along the intensity axis. The output parameter files are then available for use during a database search (using `dripSearch`) or feature extraction (using `dripExtract`). These learned parameters are particularly useful for analyzing MS/MS data where there is imprecision in m/z measurements (i.e., low-resolution MS2 data where the fragment-mass tolerance is typically a Dalton or greater).

`dripTrain` requires two files: the library of PSMs, specified using parameter `--psm-library`, and the corresponding `.ms2` file containing the MS/MS spectra, specified using parameter `--spectra`. The file format for the PSM library should be tab-delimited with fields *Peptide*, *Scan*, and *Charge*. Note that the header listing the (case-sensitive) PSM fields is required, although the order these fields are listed does not matter. It is assumed the MS/MS spectra have unique scan identification numbers. Multiple peptides per spectrum may be specified in the PSM library. Variable and static modifications must be specified (discussed in detail in Subheading 2.3.2). The learned Gaussian means and variance will be written to files specified by `--output-mean-file` and `--output-covar-file`, respectively. As an example, the DRIP parameters used in [6] may be learned by running the following in the unzipped DTK directory:

```
$ ./dripTrain.py \
  --psm-library data/riptideTrainingData/
    strict-orbitrap.psm \
  --spectra data/riptideTrainingData/strict-
    orbitrap.ms2 \
  --output-mean-file dripLearned.means \
  --output-covar-file dripLearned.covars \
  --mods-spec 'C+57.0214'
```

where `data/riptideTrainingData` contains high-quality PSMs (included in the DTK repository) collected in [11], we've specified a static modification of Carbamidomethyl, and the learned Gaussian means and variances are written to `dripLearned.means` and `dripLearned.covars`, respectively. Specified static and variable modifications should match the data collection digest enzyme and search digestion parameters so that theoretical spectra are correctly computed. The output parameter files are written in GMTK's parameter specification and may be used in all other DTK modules.

2.3 DRIP Database Search

A DRIP database search proceeds in two stages: the protein database is digested and candidate peptide sequences produced, then MS/MS spectra are identified through the scoring of candidate peptides. We now describe completing each such stage using the modules `dripDigest` and `dripSearch`.

2.3.1 Pre-search Database Digestion Using dripDigest

Prior to a search, the protein database (in FASTA format) must first be digested using `dripDigest`. The resulting peptides are written to a compact binary format for quick and efficient reading from disk. `dripDigest` supports common digestion settings, such as static modifications (including N- and C-terminal mods), variable modifications (including N- and C-terminal mods), partial digests, full digests, missed cleavages, peptide mass and length restrictions, and decoy peptide generation. `dripDigest` also supports out-of-core processing in anticipation of variable modifications. If memory consumption is a concern, the total number of peptides held in memory may be specified using the option `--peptide-buffer`. Note that the value of `--peptide-buffer` is also used during the subsequent search (i.e., running `dripSearch`) when loading candidate peptides into memory. The default buffer value of 100,000 works well in practice and should accommodate most modern workstations with several gigabytes of memory. When running a target-decoy search, setting `--recalibrate` to true will produce a disjoint set of search decoys used to renormalize DRIP scores.

`dripDigest` and all other DTK modules utilize a command-line interface with fine-grained parameter options (*see* **Note 3**), as is commonly encountered in MS/MS software. As an example, the following specifies a digest with variable modifications and static TMT labeling for an example protein database included (`data/plasmo_Pfalciparum3D7_NCBI.fasta`) in the DTK repo:

```
$ ./dripDigest.py \
  --fasta data/plasmo_Pfalciparum3D7_NCBI.fasta \
  --digest-dir dripDigest-output \
  --custom-enzyme '[K]|[X]' \
  --min-length 7 \
  --mods-spec '3M+15.9949,K+229.16293' \
  --nterm-peptide-mods-spec 'X+229.16293' \
  --decoys True
```

The resulting binary files containing the digested peptides are written to directory `dripDigest-output`. If ASCII files are desired to easily view the digested target and decoy peptides, such files will be produced in the output directory by setting `--peptide-list` to true.

The `--custom-enzyme` follows a regular expression syntax and specifies cleaving at each K proceeded by any amino acid (signified by X). Common cleavages (e.g., trypsin, lys-c, glu-c, etc.) may also be specified by keyword using the parameter `--enzyme`. The minimum peptide length is specified so that all digested peptides of length shorter than seven are discarded. Modifications are specified using syntax $[n][U]\pm[\delta]$, where n is the maximum number of times the modification may appear in the peptide, U is the set of amino acids being modified, and δ is the mass offset. A static modification is specified when n is omitted, otherwise the modification is variable. In our example, we've specified up to three variable modifications of M (methionine) and a static modification of K (lysine). In the case of N-terminal and C-terminal modifications, the syntax closely follows that of a general modification with the caveat that when $n = 1$, a variable N- or C-terminal modification is specified and when n is omitted, an N- or C-terminal static modification is specified (as is the case in our example).

2.3.2 Spectra Identification Using dripSearch

Once the protein database has been digested, we may proceed to score candidate peptides and identify the spectra of an MS/MS dataset. `dripSearch` receives as input the directory of digested peptides output by `dripDigest`, the MS/MS dataset to be identified, and, optionally, the parameters learned using `dripTrain`. Currently, the MS/MS dataset must be in the `.ms2` file format. The precursor mass tolerance, used during data acquisition in the MS1 phase, is specified either in units of Daltons or parts-per-million (ppm) using the parameter `--precursor-window-type`. By default, the precursor mass window is specified in Daltons.

Before proceeding, we note that DRIP may be operated in two "modes." In the first mode, where `--high-res-ms2` is set to false, the learned parameters output by `dripTrain` are loaded and used to specify the theoretical peak centers and intensity variance for DRIP scoring. The current assumed granularity of the m/z axis in this first mode is approximately 1 Da, so that there are 2000 theoretical peak means (corresponding to the maximum measurable m/z value). In the second mode, where `--high-res-ms2` is set to true, the theoretical peak centers used by DRIP are set to the exact b- and y-ion values of each candidate peptide and the m/z variance (i.e., the anticipated uncertainty along the m/z axis) is set such that 99.9% of the m/z Gaussian mass lies within the value fragment-mass-tolerance specified by parameter `--high-res-gauss-dist` (*see* **Note 4**). These two modes reflect the accuracy of m/z measurements collected using high-resolution and low-resolution machines, and the specified value of `--high-res-ms2` should reflect the machine resolution used during the MS2 phase. We note that, owing to the use of the general-purpose DBN-inference engine GMTK, the underlying DRIP model may be easily tailored to suit specific needs without the need for rigorous recoding (*see* **Note 5**).

As an example, the following identifies `data/malariaTest.ms2` (included in the DTK repository), collected from high-resolution MS1 and MS2 scans of a *Plasmodium falciparum* sample [13], searching the output directory `dripDigest-output` produced by `dripDigest`:

```
$ ./dripSearch.py \
  --digest-dir 'dripDigest-output' \
  --spectra data/malariaTest.ms2 \
  --precursor-window-type 'ppm' \
  --precursor-window 50 \
  --high-res-ms2 true \
  --high-res-gauss-dist 0.05 \
  --output dripSearch-plasmodium-output
```

DRIP is being run in high-resolution MS2 mode. The search results will be written to the tab-delimited file `dripSearch-plasmodium-output.txt`. A narrow 50 ppm precursor mass windows is used, as well as a stringent fragment-mass-tolerance of 0.05 Da. Note that we refer to the value of `--high-res-gauss-dist` as a fragment-mass-tolerance due to the similarities between its meaning in DTK (i.e., most of an m/z Gaussian's mass is centered in this region) and the use of this term in all other search algorithms (i.e., the quantization bin-width). However, unlike other search algorithms, fragment-ion-matches may still occur for observed peaks further from a theoretical peak's Gaussian center than the fragment-mass-tolerance, as DRIP considers all possible scorings of observed peaks by theoretical peaks when computing the most-probable alignment (a further discussion of when this might occur may be found in [7]). Thus, the value specified by `--high-res-gauss-dist` is not a strict fragment-ion-match cutoff, but rather dictates the variance of theoretical Gaussians used to score m/z measurements relative to their centers.

Users with multithreaded systems may run `dripSearch` on multiple threads by specifying the value of `--num-threads`. For users with access to a high throughput compute (HTC) cluster, `dripSearch` also supports the use of general HTC job schedulers, such as HTCondor or Univa Grid Engine (*see* **Notes 6** and **7**). In cluster mode, `dripSearch` is run in a MapReduce fashion (illustrated in Fig. 3), where the spectra and candidate peptides to be scored are partitioned into N jobs, the jobs are submitted and run on the cluster, and the search results are copied back to the local directory and merged (*see* **Note 8**).

To run the previous search example on a cluster, first set the following environment variable:

```
$ export DRIPTOOLKIT=<directory>
```

where <*directory* > is the unzipped DTK directory containing `dripSearch.py`. Next, split the data and create the jobs to be submitted to the cluster by adding the `--num-cluster-jobs` option when calling `dripSearch`:

```
$ ./dripSearch.py \
  --digest-dir 'dripDigest-output' \
```

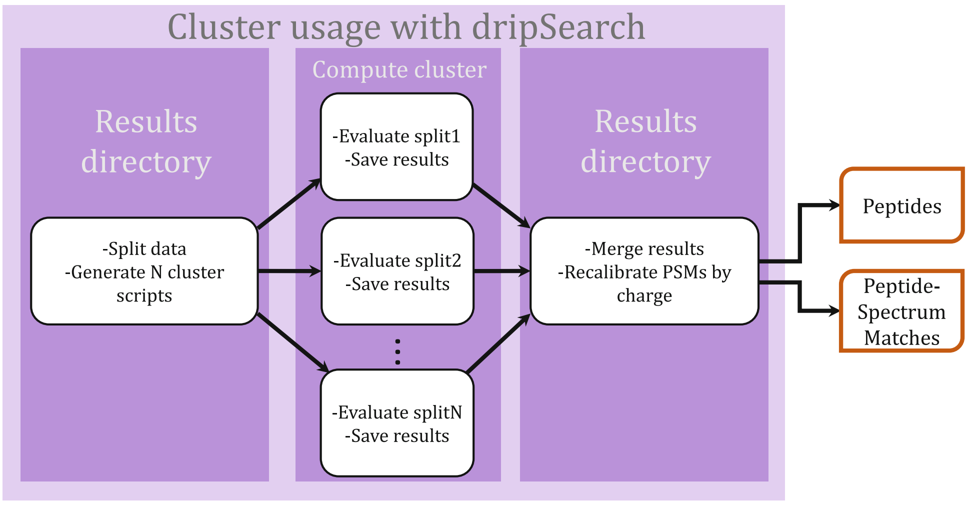


Fig. 3 Flowchart of dripSearch cluster usage

```
  --precursor-window-type 'ppm' \
  --precursor-window 50 \
  --high-res-ms2 true \
  --high-res-gauss-dist 0.05 \
  --spectra data/malariaTest.ms2 \
  --output dripSearch-plasmodium-output \
  --num-cluster-jobs 10
```

The resulting binary files for each job will be written to local directory `encode`. To accommodate different cluster environments, DRIP generates a Bash script (*see* **Note** 9) for easy submission to the cluster queue. A list of all generated Bash scripts (one per job) is written to `encode/clusterJobs.txt`. Upon completion, job results are copied over to local directory `log`. Once all jobs are completed, the results may be merged together by running:

```
$ ./dripSearch.py \
  --merge-cluster-results True \
  --output dripSearch-plasmodium-output
```

The final search results will be written to tab-delimited file `dripSearch-plasmodium-output.txt`, the fields of which are described in Table 1.

Note that, if running a multicharge target-decoy search, it is recommended `--recalibrate` be set to true. In this case, a new, disjoint set of decoys will be generated and used to recalibrate the discrepancy in DRIP score distributions for differently charged PSMs (higher charged PSMs tend to produce larger scores than unity or doubly charged PSMs). Although this increases search time, this step is critical to accurately rank and compare PSMs of different charge-states.

Table 1
Tab-delimited fields output by `dripSearch`

Field	Description
Kind	Target (t), decoy (d), or recalibration decoy (r) PSM
Scan	The identifying number of each scan
Score	The DRIP score for this PSM
Peptide	The peptide sequence
Obs_Inserts*	Inferred number of insertions (observed noise peaks)
Theo_Deletes*	Inferred number of deletions (missing theoretical peaks)
Obs_peaks_scored*	Inferred number of non-inserted peaks
Theo_peaks_used*	Inferred number of non-deleted theoretical peaks
Sum_obs_intensities*	Sum of the inferred non-inserted peak intensities
Sum_scored_mz_dist*	Sum of the absolute peak m/z distances from the DRIP Gaussian means used to score each non-inserted observed peak
Charge	Charge state of the PSM
Flanking_nterm	Amino acid preceding this peptide in the parent protein
Flanking_cterm	Amino acid following this peptide in the parent protein
Protein_id	String consisting of Kind followed by the number the protein appears in the target/decoy database
Var_mod_seq	String the characters of which denote the type of variable mod applied to each of a PSM's amino acids (only output if variable mods are passed into dripDigest). 0 denotes no variable mod, 1 denotes a variable mod (specified in mods-spec), 2 denotes an n-terminal variable mod (specified in nterm-peptide-mods-spec), and 3 denotes a c-terminal variable mod (specified in cterm-peptide-mods-spec)

Fields highlighted with * indicate features extracted using DRIP's calculated Viterbi path (described further in Subheading 2.4)

2.4 Feature Extraction Using dripExtract

DRIP's Viterbi path contains a large amount of information describing the manner in which a peptide aligns with an observed spectrum. This information may be used to extract highly detailed features, described in Table 1, for significantly improved target-decoy classification in Percolator [7]. DRIP feature extraction for arbitrary scoring algorithms is supported through the module `dripExtract`.

`dripExtract` requires as input a file of PSMs (in either tab-delimited or PIN file format) and the corresponding MS/MS dataset (in `.ms2` format). If the PSM file is a general tab-delimited file, the following fields must be present (descriptions are available in Table 1): Kind, Scan, Charge, Peptide, and Score. If a PIN file is supplied, DRIP features may be appended to the Percolator features already listed by setting the parameters `--write-pin` and `--append-to-pin` to true. The parameters learned using `dripTrain` may be used (if `--high-res-ms2` is set to false) by specifying the learned mean and covariance files using parameters `--learned-means` and `--learned-covariances`, respectively. If analyzing high-resolution MS2 spectra, `dripExtract` may run in high-res mode by setting `--high-res-ms2` to true and `--high-res-gauss-dist` to the desired fragment-mass-tolerance (as described in Subheading 2.3.2). Peptide modifications are supplied in the previously described DTK format (Subheading 2.3.1). As with all other modules, specified static and variable modifications should match the data collection digest enzyme and search digestion parameters. As with `dripSearch`, feature extraction may be run in multithreaded environments by specifying the desired number of threads with parameter `-num-threads`.

As an example, assuming the same utilized database and digestion settings as in Subheading 2.3.1, consider having run a Tide [12] XCorr search on `data/malariaTest.ms2` with the same search settings as described in Subheading 2.3.2 and `--pin-output True`, resulting in PIN file `crux-output.pin`. We may extract DRIP features for these PSMs by running:

```
$ ./dripDigest.py \
  --psm-file crux-output.pin \
  --spectra data/malariaTest.ms2 \
  --mods-spec '3M+15.9949,K+229.16293' \
  --nterm-peptide-mods-spec 'X+229.16293' \
  --high-res-ms2 true \
  --high-res-gauss-dist 0.05 \
  --num-threads 16 \
  --pin-output True \
  --append-to-pin True \
  --output dripExtracted-crux-output.pin
```

Note that parameters 3–4 match the peptide modification settings from Subheading 2.3.1 and 5–6 match the DRIP search setting from Subheading 2.3.2.

2.5 Fine-Grained Analysis Tools

DTK offers an importable Python library for detailed analysis. Using this library, a user may specify a peptide and a spectrum, instantiate a PSM, and calculate DRIP's Viterbi path in real time on the Python shell (*see* **Note 10**). A PSM's Viterbi path may then be plotted to view the maximal alignment between the theoretical and observed spectra. Alternatively, sets of PSMs may just as easily be specified and analyzed.

Once the Python library `dtk.py` (found in the unzipped DTK directory) is imported, spectra may then be loaded using function `load_spectra()` and PSMs may be specified using `psm()`. As an example, consider the MS/MS dataset included with DTK, `data/test.ms2`, for which we'd like to plot the Viterbi paths of the generating peptides. In the following, lines beginning with ⋙ denote commands entered in the Python interactive shell.

We begin by invoking the Python interactive shell and importing the library:

```
$ python
>>> import dtk
```

Next, we load the spectra into memory:

```
>>> spectra = 'data/test.ms2'
>>> s = dtk.load_spectra(spectra)
```

`dtk.load_spectra(spectra)` returns a dictionary of MS/MS spectra, the keys of which are spectrum scan numbers. Assuming parameters learned using `dripTrain` were written to `learned.means` and `learned.covars`, we may instantiate a DRIP PSM for the spectrum with scan number 6028 using:

```
>>> charge = 2
>>> scan = 6028
>>> peptide = 'TGPSPQPESQGSFYQR'
>>> highRes = False
>>> mods = 'C+57.0214'
>>> nterm_mods = ''
>>> cterm_mods = ''
>>> p = dtk.psm(peptide, s[scan], charge,
        highRes,
        'learned.means', 'learned.covars',
           mods, nterm_mods, cterm_mods)
```

Peptide modifications passed to `psm()` follow the normal DTK convention. At this point, the PSM's theoretical spectrum has been calculated, all data files have been produced, GMTK has been called, and the PSM's Viterbi path loaded into memory. All

Table 2
Attributes of DTK PSM object `p` instantiated using Python library `dtk.py`

Attributes	Description
`p.peptide`	Peptide string
`p.spectrum`	Observed spectrum, instance of spectrum object
`p.scan`	Scan ID number
`p.num_ions`	Number of unique *b*- and *y*-ions
`p.num_dels`	Number of deletions
`p.num_frames`	Number of observed peaks
`p.insertion_sequence`	Decoded sequence of Booleans denoting whether the *i*th peak in the observed spectrum is an insertion or not

information is accessible through the returned DRIP PSM object `p`, the attributes of which are listed in Table 2.

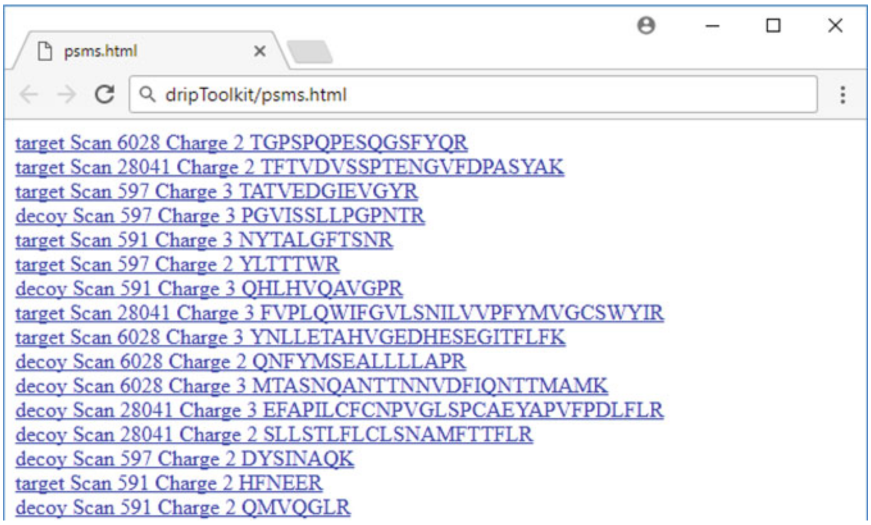
Once a DTK PSM has been instantiated, the PSM's most-probable sequences of insertions and deletions (i.e., the Viterbi path) may be easily plotted by using the function `plot_drip_viterbi()` (the `matplotlib` Python library must be installed to use this function). The following continues our earlier example, plotting the PSM's calculated Viterbi path:

```
>>> vitPlot = 'scan%d-charge%d-peptide%s.png' %
              (scan_number, charge, peptide)
>>> p.plot_drip_viterbi(vitPlot)
```

An example such DTK plot is illustrated in Fig. 1. A set of PSMs' Viterbi paths may also be plotted all at once using the function `plot_psms()` (*see* **Note 11**), given the output file of a DRIP search. For example, assuming `dripSearch` was run over `data/test.ms2` (an example such search is available in `test.sh` of the unzipped DTK directory), resulting in the output file of PSM identifications `dripSearch-test-output.txt`, we plot the Viterbi paths of these PSMs with the following:

```
>>> import dtk
>>> psms = 'dripSearch-test-output.txt'
>>> spectra = 'data/test.ms2'
>>> html_output = 'psms.html'
>>> dtk.plot_psms(psms, spectra, html_output)
```

The resulting Viterbi path plots are written to the local directory in PNG format. The output HTML file `psms.html` contains links to each plot (displayed in Fig. 4) so that they may be quickly and easily viewed in a web browser.



target Scan 6028 Charge 2 TGPSPQPESQGSFYQR
target Scan 28041 Charge 2 TFTVDVSSPTENGVFDPASYAK
target Scan 597 Charge 3 TATVEDGIEVGYR
decoy Scan 597 Charge 3 PGVISSLLPGPNTR
target Scan 591 Charge 3 NYTALGFTSNR
target Scan 597 Charge 2 YLTTTWR
decoy Scan 591 Charge 3 QHLHVQAVGPR
target Scan 28041 Charge 3 FVPLQWIFGVLSNILVVPFYMVGCSWYIR
target Scan 6028 Charge 3 YNLLETAHVGEDHESEGITFLFK
decoy Scan 6028 Charge 2 QNFYMSEALLLLAPR
decoy Scan 6028 Charge 3 MTASNQANTTNNVDFIQNTTMAMK
decoy Scan 28041 Charge 3 EFAPILCFCNPVGLSPCAEYAPVFPDLFLR
decoy Scan 28041 Charge 2 SLLSTLFLCLSNAMFTTFLR
decoy Scan 597 Charge 2 DYSINAQK
target Scan 591 Charge 2 HFNEER
decoy Scan 591 Charge 2 QMVQGLR

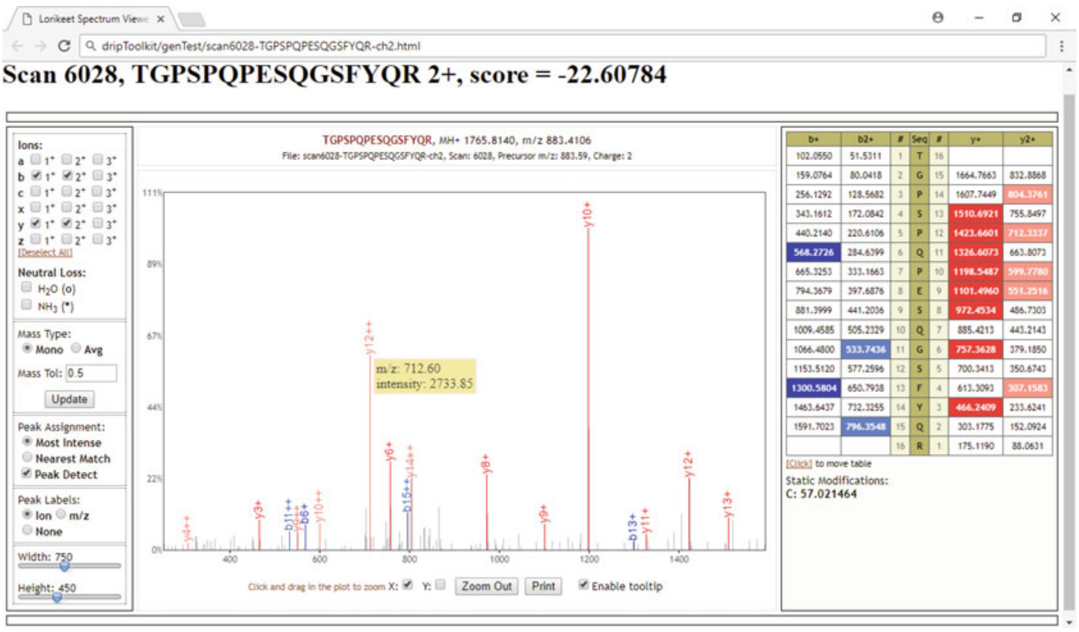
Fig. 4 Browser screenshot of DTK produced HTML file linking to plots of PSM Viterbi paths

2.5.1 Interactive Analysis Using Lorikeet

DTK supports interactive MS/MS analysis through the use of the Lorikeet Javascript plugin. After HTML files have been generated, users may select and view different fragmentation events for a given PSM, as well as inspect specific portions of the observed spectrum interactively within a browser (Fig. 5 displays a screenshot of such an HTML file opened in Google Chrome). To use this functionality, Lorikeet version 0.3.5 must be downloaded and unzipped in the main DTK directory (*see* **Note 12**).

Lorikeet file generation is supported through the `dtk` function `gen_lorikeet()`, which accepts as inputs an MS/MS dataset and a PSM file. The MS/MS data must be in the `.ms2` file format. For the PSM file, any general tab-delimited file may be used, provided that PSMs have fields denoting their spectrum scan number, charge, peptide sequence, and score. Users then provide which fields correspond to which PSM feature when calling `gen_lorikeet()`. Peptide modifications follow the standard DTK specification. For instance, assuming the earlier PSM file `dripSearch-test-output.txt` and dataset `data/test.ms2`, we would use DTK to generate the Lorikeet HTML files with the following:

```
>>> import dtk
>>> psmFile='dripSearch-test-output.txt'
>>> ms2="data/test.ms2"
>>> scanField = 'Scan'
>>> chargeField = 'Charge'
>>> peptideField = 'Peptide'
>>> scoreField = 'Score'
>>> mods = 'C+57.0214'
```


Lorikeet Spectrum Viewer
dripToolkit/genTest/scan6028-TGPSPQPESQGSFYQR-ch2.html
TGPSPQPESQGSFYQR, MH+ 1765.8140, m/z 883.4106
File: scan6028-TGPSPQPESQGSFYQR-ch2, Scan: 6028, Precursor m/z: 883.59, Charge: 2
m/z: 712.60
intensity: 2733.85
Click and drag in the plot to zoom X: ✓ Y:
Zoom Out
Print
Enable tooltip
Update

Scan 6028, TGPSPQPESQGSFYQR 2+, score = -22.60784

Ions: b 1+ ✓ 2+ ✓; y 1+ ✓ 2+ ✓

Neutral Loss: H₂O (o) NH₃ (*)

Mass Type: [✓ Mono] Avg

Mass Tol: 0.5

Peak Assignment: [✓ Most Intense] Nearest Match; Peak Detect ✓

Peak Labels: [✓ Ion] m/z None

Width: 750

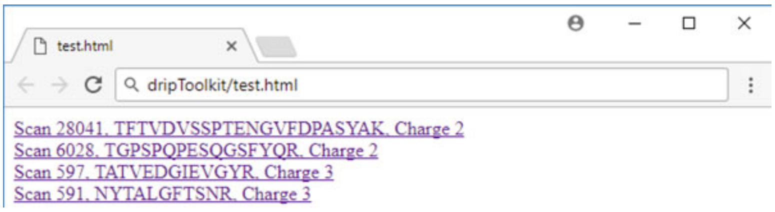
Height: 450

b+	b2+	#	Seq	#	y+	y2+
102.0550	51.5311	1	T	16		
159.0764	80.0418	2	G	15	1664.7663	832.8868
256.1292	128.5682	3	P	14	1607.7449	804.3761
343.1612	172.0842	4	S	13	1510.6921	755.8497
440.2140	220.6106	5	P	12	1423.6601	712.3337
568.2726	284.6399	6	Q	11	1326.6073	663.8073
665.3253	333.1663	7	P	10	1198.5487	599.7780
794.3679	397.6876	8	E	9	1101.4960	551.2516
881.3999	441.2036	9	S	8	972.4534	486.7303
1009.4585	505.2329	10	Q	7	885.4213	443.2143
1066.4800	533.7436	11	G	6	757.3628	379.1850
1153.5120	577.2596	12	S	5	700.3413	350.6743
1300.5804	650.7938	13	F	4	613.3093	307.1583
1463.6437	732.3255	14	Y	3	466.2409	233.6241
1591.7023	796.3548	15	Q	2	303.1775	152.0924
		16	R	1	175.1190	88.0631

[Click] to move table

Static Modifications:
C: 57.021464

Fig. 5 Screenshot of DTK produced HTML file, loaded in Google Chrome, for interactive analysis using the Lorikeet plugin



Scan 28041, TFTVDVSSPTENGVFDPASYAK, Charge 2
Scan 6028, TGPSPQPESQGSFYQR, Charge 2
Scan 597, TATVEDGIEVGYR, Charge 3
Scan 591, NYTALGFTSNR, Charge 3

Fig. 6 Browser screenshot of DTK produced HTML file linking to interactive PSM plots using Lorikeet

```
>>> nterm_mods = ''
>>> cterm_mods = ''
>>> dtk.gen_lorikeet(psmFile, ms2, 'htmFiles',
                     'test.html', mods,
                      nterm_mods, cterm_mods,
                      scanField, peptideField,
                      chargeField, scoreField)
```

The resulting HTML files are written to subdirectory `htmlFiles` and the HTML file `test.html` contains links to each individual file for easy navigation (displayed in Fig. 6).

3 Notes

1. For each DTK module, a list of parameters (accompanied by a short description and valid input type) may be viewed by running `--help`. For instance, to view a list of DTK training options, run `./dripTrain.py --help`. Alternatively, the following are main pages for each module describing both their usage and parameter options:
   - jthalloran.bitbucket.io/dripToolkit/dripTrain.html
   - jthalloran.bitbucket.io/dripToolkit/dripDigest.html
   - jthalloran.bitbucket.io/dripToolkit/dripSearch.html
   - jthalloran.bitbucket.io/dripToolkit/dripExtract.html
   - jthalloran.bitbucket.io/dripToolkit/dtk.html
2. If you are receiving error message:

   ```
   ImportError: No module named libpfile
   ```

   please remember to first build and link the `pFile` library, as described in Subheading 2.1.
3. As with all MS/MS software, the DTK function calls can be quite long. It is thus helpful to write and call these commands within a shell script (such as Bash). Due to the overlap in parameters between modules (for instance, specified peptide modifications), this also decreases erroneous calls and is an effective form of experiment recordkeeping for reproducibility/debugging.
4. Setting the `dripSearch` and `dripExtract` parameter `--precursor-filter` to true is very effective on high-resolution MS2 data. When true, this parameter tightens the filtering window around the observed precursor mass during observed spectrum preprocessing.
5. Thanks to the use GMTK, the underlying DRIP model may be easily altered by modifying the function `create_drip_structure()` in file `pyFiles/dripEncoding.py`. The qualitative effects of changes to the model may be easily viewed by using `dtk` to plot the resulting Viterbi paths for test spectra (this is especially useful for debugging).
6. Due to the heavy compute and I/O required, running a DRIP search is most effective using an HTC cluster with `dripSearch` in cluster mode. Feature extraction using `dripExtract` is much less computationally intensive and should complete quickly using only a multithreaded workstation.
7. For compute environments with a large amount of main memory, the maximum number of peptides loaded into main

memory during database search may be increased using parameter `--peptide-buffer` when calling `dripDigest`.

8. If running `dripSearch` in cluster mode, all spectra to be searched should be passed to `dripSearch` all at once. Upper and lower bounds on the m/z axis are computed and used to filter theoretical peaks outside of these bounds.
9. The `dripSearch` generated Bash scripts for cluster use may be easily customized for specific compute clusters by editing `pyFiles/cluster.py`.
10. The DRIP scores reported by different modules `dripSearch`, `dripExtract`, and `dtk` may differ. This is due to `dripExtract` and `dtk` not supporting `dripSearch`'s recalibration functionality (performed when `recalibrate` is set to true), although scores should agree if recalibration is not turned on and all data/parameters agree between the three modules.
11. The `dtk.py` source illustrates how to call the `dripExtract` API (i.e., `runDripExtract()`) if custom manipulation of DRIP Viterbi paths in Python is desired.
12. If using `Cygwin`, note that Windows unzip creates an extra parent directory (unlike standard Unix unzip). When using Lorikeet in this situation, please move the unzipped subdirectory into the DTK repo (otherwise the Lorikeet Javascript will not load in your browser).

References

1. Bilmes J, Zweig G (2002) The Graphical Models Toolkit: an open source software system for speech and time-series processing. In: Proceedings of the IEEE international conference on acoustics, speech, and signal processing
2. Craig R, Beavis RC (2004) Tandem: matching proteins with tandem mass spectra. Bioinformatics 20:1466–1467
3. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B 39:1–22
4. Eng JK, McCormack AL, Yates, JR III (1994) An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. J Am Soc Mass Spectrom 5:976–989
5. Geer LY, Markey SP, Kowalak JA, Wagner L, Xu M, Maynard DM, Yang X, Shi W, Bryant SH (2004) Open mass spectrometry search algorithm. J Proteome Res 3(5):958–964
6. Halloran JT, Bilmes JA, Noble WS (2014) Learning peptide-spectrum alignment models for tandem mass spectrometry. In: Uncertainty in artificial intelligence (UAI), Quebec City, QC. AUAI, Arlington
7. Halloran JT, Bilmes JA, Noble WS (2016) Dynamic Bayesian network for accurate detection of peptides from tandem mass spectra. J Proteome Res 15(8):2749–2759
8. Howbert JJ, Noble WS (2014) Computing exact p-values for a cross-correlation shotgun proteomics score function. Mol Cell Proteomics 13(9):2467–2479. mcp–O113
9. Kall L, Storey JD, MacCoss MJ, Noble WS (2008) Assigning significance to peptides identified by tandem mass spectrometry using decoy databases. J Proteome Res 7(1):29–34
10. Kim S, Mischerikow N, Bandeira N, Navarro JD, Wich L, Mohammed S, Heck AJR, Pevzner PA (2010) The generating function of CID, ETD, and CID/ETD pairs of tandem mass spectra: applications to database search. Mol Cell Proteomics 9(12):2840–2852
11. Klammer AA, Reynolds SM, Bilmes JA, MacCoss MJ, Noble WS (2008) Modeling peptide fragmentation with dynamic Bayesian net-

works for peptide identification. Bioinformatics 24(13):i348–356

12. McIlwain S, Tamura K, Kertesz-Farkas A, Grant CE, Diament B, Frewen B, Howbert JJ, Hoopmann MR, Käll L, Eng JK et al (2014) Crux: rapid open source protein tandem mass spectrometry analysis. J Proteome Res 13:4488–4491
13. Pease BN, Huttlin EL, Jedrychowski MP, Talevich E, Harmon J, Dillman T, Kannan N, Doerig C, Chakrabarti R, Gygi SP, Chakrabarti D (2013) Global analysis of protein expression and phosphorylation of three stages of Plasmodium falciparum intraerythrocytic development. J Proteome Res 12(9):4028–4045
14. Singh AP, Halloran J, Bilmes JA, Kirchoff K, Noble WS (2012) Spectrum identification using a dynamic Bayesian network model of tandem mass spectra. In: Uncertainty in artificial intelligence (UAI), Catalina Island. AUAI, Arlington
15. Wenger CD, Coon JJ (2013) A proteomics search algorithm specifically designed for high-resolution tandem mass spectra. J Proteome Res 12(3):1377–1386

Chapter 13

Sparse Modeling to Analyze Drug–Target Interaction Networks

Yoshihiro Yamanishi

Abstract

Most drugs produce their phenotypic effects by interacting with target proteins, and understanding the molecular features that underpin drug–target interactions is crucial when designing a novel drug. In this chapter, we introduce the protocols that have driven recent advances in sparse modeling methods for analyzing drug–target interaction networks within a chemogenomic framework. In this approach, the chemical structures of candidate drug compounds are correlated with the genomic sequences of the candidate target proteins. We demonstrate the use of sparse canonical correspondence analysis and sparsity-induced binary classifiers to extract the underlying molecular features that are most strongly involved in drug–target interactions. We focus on drug chemical substructures and protein domains. Workflows for applying these methods are presented, and an application is described in detail. We consider the characteristics of each method and suggest possible directions for future research.

Key words Drug–target interactions, Sparse modeling, Feature extraction, Chemogenomics, Chemical substructures, Protein domains

1 Introduction

Most drugs achieve their phenotypic effects by interacting with target proteins (drug–target interactions). It is therefore important to identify the full set of drug–target interactions, including not only primary targets but also off-targets. This also improves the understanding of polypharmacology. Determining potential drug–target interactions experimentally is time consuming and costly, creating a strong incentive to develop computational methods for predicting such interactions. Traditional computational techniques for predicting drug–target interactions or compound–protein interactions have been categorized into ligand-based and structure-based approach. Ligand-based approaches, which include quantitative structure activity relationship model, relate drug candidate compounds with the known ligands of a target protein and then apply statistical machine learning techniques to predict the binding patterns of the proteins [1, 2]. However, the performance of such approaches deteriorates when only a small number of appropriate ligands are present in the learning set. Structure-based approaches, which include docking simulations, are powerful, but they can only be applied when the 3D structure of the protein is available [3]. This limitation is particularly serious in the case of membrane proteins such as G protein-coupled receptors (GPCRs), because few of whose structures are known.

Chemogenomics is an emerging approach to exploring the relationship between the chemical space of bioactive compounds and the genomic space of all proteins. It allows chemical and biological information to be generalized across possible compound-protein pairs [4–6]. Genome-wide computational prediction of drug–target or compound–protein interactions has become an important research area in chemogenomics. Within the chemogenomic framework, a range of computational methods have been developed to predict drug–target interactions or compound–protein interactions [7–13]. While different methods apply different algorithms, there is a shared assumption that similar compounds will interact with similar proteins. Predictions are therefore predicated on the compound chemical structures, protein sequences or structures, and partially known interactions. However, the chemogenomic methods only output predictions and the predictive models are not interpretable. This makes it challenging to derive biological insights into the molecular mechanisms of the drug–target interactions.

An understanding of the molecular features underpinning the drug–target interaction is also crucial in the design of novel drugs. Chemical substructures (e.g., pharmacophores, functional groups) and protein functional sites (e.g., domains and motifs) tend to be conserved across a range of drug–target interactions. Thus, the molecular features of these interactions may be explained by associations between the chemical substructures of the drug and the functional sites of the protein. Recently, a range of methods have been developed to allow feature extraction to be used in the analysis of drug–target interactions. One proposed approach combined graph mining with sequence mining to identify drug substructures and protein subsequences that appear frequently in known interactions [14]. Sparse canonical correspondence analysis has been used to identify ensembles of chemical substructures and protein domains that are involved in drug–target interactions [15]. Sparsity-induced binary classifiers (e.g., L1-regularized logistic regression and L1-regularized support vector machine) have also been applied to extract informative pairings of chemical substructures and protein domains [16–18].

In this chapter, we discuss the protocols that have driven recent advances in sparse modeling within the chemogenomic framework. These relate the chemical structures of candidate drug compounds with the genomic sequences of candidate target proteins. Our focus is on the use of sparse canonical correspondence analysis and sparsity-induced binary classifiers to identify the underlying molecular features that are most strongly associated with drug–target interactions. We present workflows for applying these methods and discuss some of their applications. The characteristics of each method are considered and directions for future research are suggested.

2 Materials

2.1 Drug–Target Interactions

The drug–target interactions were taken from the DrugBank database [19]. The target human proteins belonged to a range of families and included enzymes, ion channels, G protein-coupled receptors (GPCRs), transporters, and nuclear receptors. The final dataset comprised 4809 interactions involving 1862 drugs and 1554 target proteins (Fig. 1).

2.2 Descriptors of Target Proteins

The amino acid sequences of the target proteins were taken from the UniProt database [20], and the associated protein domains from the PFAM database [21]. Moreover, PFAM domains whose proteins did not appear in the drug–target interaction set were eliminated, leaving 876 domains. Each protein was represented as a 876 dimensional binary vector, and the presence or absence of each PFAM domain was noted by a coding of 1 or 0, respectively. Other possible descriptors of target proteins are discussed in **Note 1**.

2.3 Descriptors of Drugs

The chemical structures of the drugs were encoded using a chemical fingerprint in the PubChem database [22]. We applied an 881 dimensional binary feature vector in which the presence

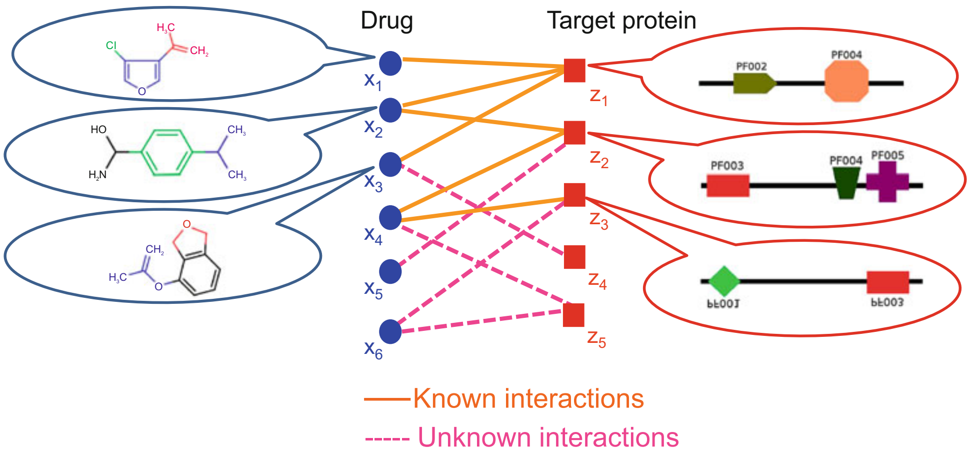


Fig. 1 An illustration of the problem of drug–target interaction prediction

or absence of each substructure was coded by 1 or 0. In most cases, drugs that appear in DrugBank are linked to compounds in PubChem. Substructures that were not involved in the drug–target interactions were eliminated, leaving 663 in the final set. Other possible descriptors of drugs are discussed in **Note 2**.

3 Methods

3.1 Problem Setting

1. The molecular features of a drug–target interaction can be characterized by associations between the chemical substructures of the drug and the protein domains. A key challenge is to extract those sets of drug chemical substructures and protein domains that appear together in known drug–target pairings, but not in other pairings.
2. Suppose that we have a set of n_x drugs with p substructure features, a set of n_z target proteins with q domain features, and information on drug–target interactions ($n_x \neq n_z$). Each drug can then be represented as a p-dimensional feature vector $\mathbf{x} = (x_1, \ldots, x_p)^T$, and each target protein can be represented as a q-dimensional feature vector $\mathbf{z} = (z_1, \ldots, z_q)^T$.

3.2 Sparse Canonical Correspondence Analysis (SCCA)

1. Canonical correspondence analysis (CCA) is an approach to the analysis of two heterogeneous data sets. The relationship between canonical correspondence analysis and canonical correlation analysis is discussed in **Note 3**. Two forms are used: ordinary CCA (OCCA) and sparse CCA (SCCA). In this section, we briefly introduce a feature extraction method based on SCCA [15].
2. Two linear combinations of drugs and proteins can be modeled as $u_i = \boldsymbol{\alpha}^T \mathbf{x}_i$ $(i = 1, 2, \ldots, n_x)$ and $v_j = \boldsymbol{\gamma}^T \mathbf{z}_j$ $(j = 1, 2, \ldots, n_z)$, respectively, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_p)^T$ and $\boldsymbol{\beta} = (\gamma_1, \ldots, \gamma_q)^T$ are weight vectors. The goal of OCCA is to estimate those weight vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ that maximize the following canonical correlation coefficient:

$$\mathrm{corr}(u, v) = \frac{\sum_{i,j} I(\mathbf{x}_i, \mathbf{z}_j) \boldsymbol{\alpha}^T \mathbf{x}_i \cdot \boldsymbol{\gamma}^T \mathbf{z}_j}{\sqrt{\sum_i d_{x_i} (\boldsymbol{\alpha}^T \mathbf{x}_i)^2} \sqrt{\sum_j d_{z_j} (\boldsymbol{\gamma}^T \mathbf{z}_j)^2}}, \tag{1}$$

where $I(\cdot, \cdot)$ is an indicator function which returns 1 if drug $\mathbf{x}_i$ and protein $\mathbf{z}_j$ interact and 0 otherwise, d_{x_i} (resp. d_{z_j}) is the degree of $\mathbf{x}_i$ (resp. $\mathbf{z}_j$), the degree is the number of interaction partners, $\sum_i u_i = 0$ (resp. $\sum_j v_j = 0$) is assumed, and u (resp. v) is denoted as the *canonical components* for $\mathbf{x}$ (resp. $\mathbf{z}$) [23].

3. In OCCA, the maximization problem can be written as follows:

$$\max\left\{\sum_{i,j} I(\mathbf{x}_i,\mathbf{z}_j)\boldsymbol{\alpha}^T\mathbf{x}_i\cdot\boldsymbol{\gamma}^T\mathbf{z}_j\right\}\quad\text{subject to}$$
$$\sum_i d_{x_i}(\boldsymbol{\alpha}^T\mathbf{x}_i)^2\le 1,\quad \sum_j d_{z_j}(\boldsymbol{\gamma}^T\mathbf{z}_j)^2\le 1. \tag{2}$$

   Here we define the $n_x \times n_z$ adjacency matrix A, in which element $(A)_{ij}$ is set to 1 (resp. 0) if drug $\mathbf{x}_i$ and protein $\mathbf{z}_j$ interact (resp. do not interact). Let X be the $n_x \times p$ matrix defined as $X = \left[\mathbf{x}_1,\ldots,\mathbf{x}_{n_x}\right]^T$, and let Z denote the $n_z \times q$ matrix defined as $Z = \left[\mathbf{z}_1,\ldots,\mathbf{z}_{n_z}\right]^T$, assuming the columns of X and Z to be centered and scaled.

4. The optimization problem can then be rewritten in matrix form:

$$\max\{\boldsymbol{\alpha}^T X^T A Z\boldsymbol{\gamma}\}\quad\text{subject to}$$
$$\boldsymbol{\alpha}^T X^T D_x X\boldsymbol{\alpha}\le 1,\quad \boldsymbol{\gamma}^T Z^T D_z Z\boldsymbol{\gamma}\le 1, \tag{3}$$

   where D_x and D_z are matrices whose diagonals, respectively, represent the degrees of the drugs and target proteins. We substitute identity matrices for $X^T D_x X$ and $Z^T D_z Z$, an operation that is often applied when the feature vectors are high dimensional [24, 25]. Thus, the OCCA optimization problem can then be rewritten as follows:

$$\max\{\boldsymbol{\alpha}^T X^T A Z\boldsymbol{\gamma}\}\quad\text{subject to}\quad ||\boldsymbol{\alpha}||_2^2\le 1,\quad ||\boldsymbol{\gamma}||_2^2\le 1, \tag{4}$$

   where $||\cdot||_2$ is the L_2 norm (the square root of the sum of squared values in the vector).

5. In OCCA, the weight vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are not unique if p is greater than n_x or q is greater than n_z. In addition, most elements in $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are nonzero and take highly variable values. To aid interpretation, it is desirable to limit the number of nonzero elements in $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$.

6. By adding L_1 penalty terms, sparsity can be induced on $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ in the optimization problem:

$$\max\{\boldsymbol{\alpha}^T X^T A Z\boldsymbol{\gamma}\}\quad\text{subject to}$$
$$||\boldsymbol{\alpha}||_2^2\le 1,\quad ||\boldsymbol{\gamma}||_2^2\le 1,\quad ||\boldsymbol{\alpha}||_1\le c_1\sqrt{p},\quad ||\boldsymbol{\gamma}||_1\le c_2\sqrt{q}, \tag{5}$$

   where $||\cdot||_1$ is the L_1 norm (the sum of absolute values in the vector), and c_1 and c_2 controlling the sparsity are restricted to ranges $0 < c_1 \le 1$ and $0 < c_2 \le 1$.

7. In SCCA, the optimization problem can be regarded as one of penalized matrix decomposition (PMD) [26], and applying the PMD algorithm to the matrix $Q = X^T AZ$ can yield solutions. The maximization criterion is expressed as $\rho = \boldsymbol{\alpha}^T Q \boldsymbol{\gamma}$ and referred to as the singular value.
8. To obtain multiple canonical components, the maximization of the above criterion iterated under certain constraints (deflation). The Q matrix provides the residuals obtained by subtracting the factors already found from the matrix. The kth weight vectors $\boldsymbol{\alpha}_k$ and $\boldsymbol{\gamma}_k$ for $k = 1, 2, \ldots, m$ are estimated recursively, by setting $Q^{(1)} \leftarrow Q$. Applying the PMD algorithm to $Q^{(k)}$ yields $\boldsymbol{\alpha}_k$, $\boldsymbol{\gamma}_k$, and d_k. We then set $Q^{(k+1)} \leftarrow Q^{(k)} - \rho_k \boldsymbol{\alpha}_k \boldsymbol{\gamma}_k^T$ and repeat the above steps for $k = 1, 2, \ldots, m$. Finally, m pairs of weight vectors $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m$ and $\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_m$ are obtained.
9. Substructures and domains that are highly weighted in the weight vectors of the same canonical component are considered significant in terms of drug–target interactions. The strength of the association between chemical substructures and protein domains can be derived as the product between the weight elements in $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ within each component.
10. If the weight elements in $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are biologically meaningful, it should be possible to generalize their properties. Given pairing of drug $\mathbf{x}$ with protein $\mathbf{z}$, the potential interaction of drug $\mathbf{x}$ and protein $\mathbf{z}$ can be predicted from chemical substructures present in $\mathbf{x}$ and protein domains present in $\mathbf{z}$ using weight vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$. For any given pairing of drug $\mathbf{x}$ and protein $\mathbf{z}$, a prediction score can then be derived:

$$s(\mathbf{x}, \mathbf{z}) = \sum_{k=1}^{m} u_k \rho_k v_k = \sum_{k=1}^{m} \mathbf{x}^T \boldsymbol{\alpha}_k \rho_k \boldsymbol{\gamma}_k^T \mathbf{z}, \qquad (6)$$

where m is the number of canonical components and ρ_k is the kth singular value. If the score $s(\mathbf{x}, \mathbf{z})$ is high, interactions are predicted to occur.

3.3 Sparsity-Induced Binary Classifiers (SIBCs)

1. An alternative approach to predicting drug–target interactions is the use of supervised binary classification. In this section we briefly review the use of sparsity-induced binary classifiers (SIBC) [16].
2. Given a collection of $n_x \times n_z$ drug-protein pairs $(\mathbf{x}_1, \mathbf{z}_1)$,$(\mathbf{x}_1, \mathbf{z}_2)$, ..., $(\mathbf{x}_{n_x}, \mathbf{z}_{n_z})$ that are known to interact or not to interact, a function $f(\mathbf{x}, \mathbf{z})$ is derived to predict whether drug $\mathbf{x}$ will interact with protein $\mathbf{z}$. The features that contributed to the correct prediction are also identified.

3. To apply existing binary classifiers, we represent the pairing of drug $\mathbf{x}$ and protein $\mathbf{z}$ as a feature vector $\Phi(\mathbf{x}, \mathbf{z})$, then estimate a linear function $f(\mathbf{x}, \mathbf{z}) = \mathbf{w}^{\mathrm{T}}\Phi(\mathbf{x}, \mathbf{z})$ whose sign can be used to predict whether or not the pairing of $\mathbf{x}$ and $\mathbf{z}$ will interact. The weight vector $\mathbf{w}$ is estimated using learning with interaction labels.
4. The drug-protein pairing is represented as a feature vector using the tensor product of two feature vectors. Each drug is represented as a p-dimensional feature vector, $\mathbf{x} = (x_1, x_2, \ldots, x_p)^{\mathrm{T}}$. Each protein is represented as a q-dimensional feature vector, $\mathbf{z} = (z_1, z_2, \ldots, z_q)^{\mathrm{T}}$. A feature vector of the drug-protein pair is defined by the tensor product of $\mathbf{x}$ and $\mathbf{z}$:

$$\begin{aligned}\Phi(\mathbf{x}, \mathbf{z}) &= \mathbf{x} \otimes \mathbf{z} \\ &= (x_1 z_1, x_1 z_2, \ldots, x_1 z_q, \ldots, x_p z_1, x_p z_2, \ldots, x_p z_q)^{\mathrm{T}},\end{aligned}$$

   where $\Phi(\mathbf{x}, \mathbf{z})$ is a $p \times q$ dimension binary vector, and the corresponding elements comprise all possible products of the elements in feature vectors $\mathbf{x}$ and $\mathbf{z}$. Since $\mathbf{x}$ and $\mathbf{z}$ encode for chemical substructures and protein domains, respectively, each element in $\Phi(\mathbf{x}, \mathbf{z})$ represents a paring of the two. Other possibilities for the design of $\Phi(\mathbf{x}, \mathbf{z})$ are discussed in **Note 4**.
5. We apply two binary linear classifiers: logistic regression (LOG) and linear support vector machine (SVM). Given a set of drug-protein pairs and labels $(\Phi(\mathbf{x}_i, \mathbf{z}_j), y_{ij})$, $y_{ij} \in \{+1, -1\}$, LOG and SVM are, respectively, formulated as unconstrained optimization problems:

$$\min_{\mathbf{w}} \sum_{i=1}^{n_x} \sum_{j=1}^{n_z} \log(1 + \exp(-y_{ij}\mathbf{w}^{\mathrm{T}}\Phi(\mathbf{x}_i, \mathbf{z}_j))), \tag{7}$$

   and

$$\min_{\mathbf{w}} \sum_{i=1}^{n_x} \sum_{j=1}^{n_z} \max\{1 - y_{ij}\mathbf{w}^{\mathrm{T}}\Phi(\mathbf{x}_i, \mathbf{z}_j), 0\}. \tag{8}$$

6. When learning to minimize an objective function, models typically include regularization to avoid overfitting. A common approach uses L_2-regularization. However, most elements in the weight vector are then nonzero, making interpretation from the learned weights difficult. L_2-regularized LOG and SVM are referred to as L2LOG and L2SVM, respectively.
7. An alternative approach uses L_1-regularization. This sets the weights of uninformative features to zero without loss of classification accuracy, simplifying interpretation from the learned

weights. To improve interpretability, we therefore use LOG and SVM with L_1-regularization. Optimization of the weight vector with L_1-regularization is performed as follows:

$$\min_{\mathbf{w}} ||\mathbf{w}||_1 + C \sum_{i=1}^{n_x} \sum_{j=1}^{n_z} \log(1 + \exp(-y_{ij} \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_i, \mathbf{z}_j))), \quad (9)$$

and

$$\min_{\mathbf{w}} ||\mathbf{w}||_1 + C \sum_{i=1}^{n_x} \sum_{j=1}^{n_z} \max\{1 - y_{ij} \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_i, \mathbf{z}_j), 0\}, \quad (10)$$

where $|| \cdot ||_1$ is the L_1 norm (the sum of absolute values in the vector) and C is a hyper-parameter. L_1-regularized LOG and SVM are referred to as L1LOG and L1SVM, respectively.

8. Highly weighted features in the weight vector **w** are assumed to have strong predictive power, and the corresponding combinations of substructures and domains to be promising candidates for drug–target interaction.

3.4 An Application

1. We applied the SCCA and SIBC methods to the drug–target interaction data. The relationship between SCCA and SIBC is discussed in **Note 5**.
2. In the application of SCCA, we compared with OCCA. We extracted 50 canonical components (CCs), each of which comprised a limited number of chemical substructures and protein domains. The resulting weight vectors for the drug substructures and protein domains were then examined. The detailed results can be found in the original paper [15].
3. Figures 2 and 3 show the index-plots of the weight vectors derived by applying OCCA and SCCA, respectively. Owing to space limitation, only the first 3 canonical components are shown. Almost all the elements in the OCCA weight

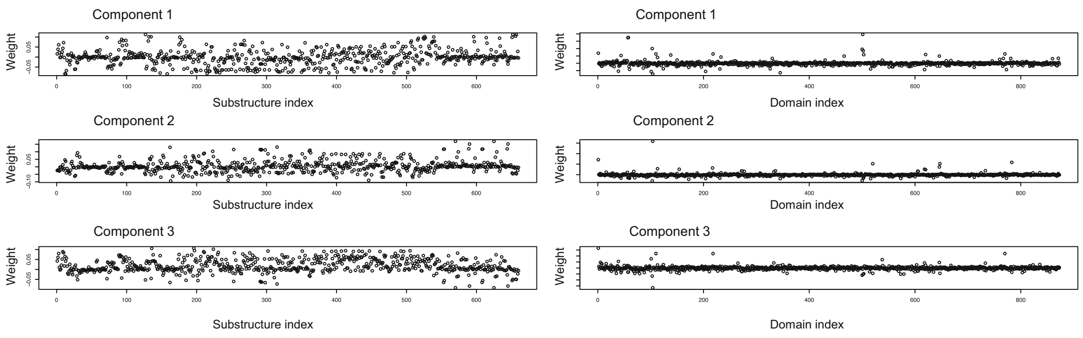


Fig. 2 Index-plot of weight vectors in OCCA for drug substructures (left) and protein domains (right)

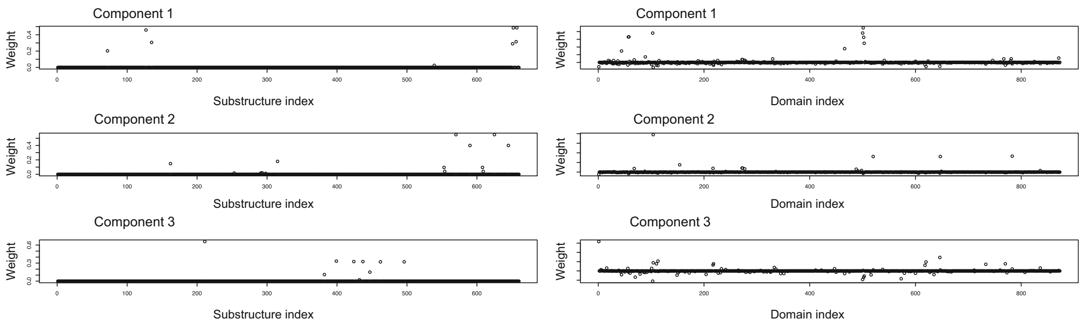


Fig. 3 Index-plot of weight vectors in SCCA for drug substructures (*left*) and protein domains (*right*)

vectors appeared to have nonzero values and a high degree of variation, whereas most of those in the SCCA weight vectors had zero values in each component. This suggested that SCCA selected a very small number of features. In practice, results are difficult to interpret when the number of weighted elements is very large, as was the case when using OCCA. The results suggested that SCCA was more selective when identifying drug substructures and protein domains.

4. In the application of SIBC, we tested L1LOG and L1SVM, and compared with L2LOG and L2SVM. In each case, the positively weighted features were extracted and the parameters (including regularization parameters, sparsity parameters, and number of components) were optimized by cross-validation. The relevance of connections between chemical substructures and protein domains was evaluated from the corresponding weightings in the classifier. The detailed results were reported in an earlier paper [16].

5. Figure 4 compares the number of features extracted by the four methods. L1LOG and L1SVM extracted a much more limited number of features than L2LOG and L2SVM. This was attributed to the sparsity that resulted from applying the L1 penalty rather than the L2 penalty, which reduced the number of features to be examined. This suggests that the L1-regularized classifier approach is preferred when applying biological interpretation to the analysis of extracted features.

4 Notes

1. Feature extraction methods can be applied when the drug molecules and target proteins are represented by high-dimensional descriptors (in this study, chemical substructures and protein domains). However, the performance depends heavily on the definition of the descriptors, and these methods are unable to extract features that are not listed in the

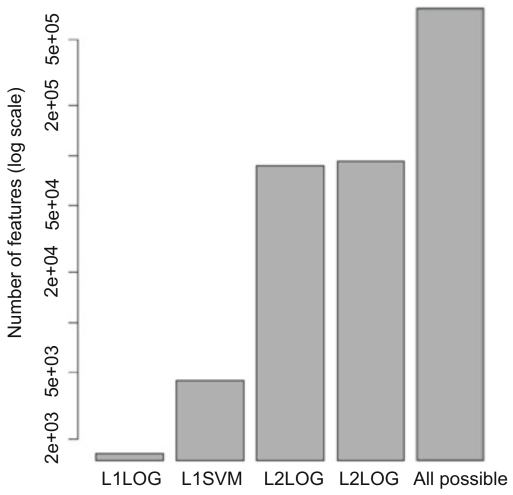


Fig. 4 Comparison of the number of extracted features between different methods

predefined descriptors. The creation of more appropriate descriptors may improve generalization. For example, it would be interesting to investigate the effect of incorporating prior information on pharmacophores [27] and biding pockets [28, 29].

2. A promising way of designing drug descriptors would be to use phenotypic data, such as known human side-effects [30, 31]. However, at present detailed side-effect information is available only from drug package inserts or clinical reports, so that this approach can be applied only to commercially marketed drugs. To address this, approaches have been developed that predict the phenotypic profile of a compound from the chemical structure [32, 33]. Another promising approach to the development of drug descriptors uses drug-induced gene expression profiles observed from chemical perturbations in human cell lines [34–38].

3. The SCCA approach presented in this chapter is a sparse version of canonical correspondence analysis, which is used to handle two different sets of data with their co-occurrence information on heterogeneous objects. The criterion for canonical *correspondence* analysis is similar to that of canonical *correlation* analysis [39], so the former can be considered a variant of the latter. There are two main differences between the two methods: (1) in canonical *correlation* analysis, the objects are the same in the two datasets, whereas in canonical *correspondence*

analysis, the objects are different, and (2) canonical correlation analysis cannot handle co-occurrence information about the heterogeneous objects.

4. When the feature vector $\Phi(\mathbf{x}, \mathbf{z})$ in SIBC is a sparse binary vector, the optimization problem can be solved with high efficiency [40]. The use of b-bit hashing techniques has been proposed as a way of transforming high-dimensional feature vectors into lower dimensions while preserving the essential information [17]. Combining L1SVM with b-bit hashing makes it possible to learn a predictive model from massive datasets comprising billions of sample points.

5. SCCA and SIBC are similar in that sparsity is induced into the predictive models. Since SCCA introduces sparsity into each canonical component, the extracted features (in this study, the chemical substructures and protein domains) differ from canonical component to canonical component. This is beneficial when the goal is to associate a set of chemical substructures with a set of protein domains in a modular manner. In contrast, SIBC induces sparsity into a pairwise statistical model. When many similar substructure–domain pairs are present, SIBC tends to extract a smaller number of representative substructure–domain pairs.

Acknowledgements

This work is supported by JST PRESTO Grant Number JPMJPR15D8.

References

1. Butina D, Segall M, Frankcombe K (2002) Predicting ADME properties in silico: methods and models. Drug Discov Today 7:S83–S88
2. Byvatov E, Fechner U, Sadowski J, Schneider G (2003) Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. J Chem Inf Comput Sci 43:1882–1889
3. Rarey M, Kramer B, Lengauer T, Klebe G (1996) A fast flexible docking method using an incremental construction algorithm. J Mol Biol 261:470–489
4. Kanehisa M, Goto S, Hattori M, Aoki-Kinoshita K, Itoh M, Kawashima S, Katayama T, Araki M, Hirakawa M (2006) From genomics to chemical genomics: new developments in KEGG. Nucleic Acids Res. 34:D354–357
5. Stockwell B (2000) Chemical genetics: ligand-based discovery of gene function. Nat Rev Genet 1:116–125
6. Dobson C (2004) Chemical space and biology. Nature 432:824–828
7. Erhan D, LÕheureux P-J, Yue SY, Bengio Y (2006) Collaborative filtering on a family of biological targets. J Chem Inf Model 46:626–635
8. Nagamine N, Sakakibara Y (2007) Statistical prediction of protein–chemical interactions based on chemical structure and mass spectrometry data. Bioinformatics 23:2004–2012
9. Yamanishi Y, Araki M, Gutteridge A, Honda W, Kanehisa M (2008) Prediction of drug-target interaction networks from the integration of chemical and genomic spaces. Bioinformatics 24:i232–i240

10. Faulon J, Misra M, Martin S, Sale K, Sapra R (2008) Genome scale enzyme–metabolite and drug–target interaction predictions using the signature molecular descriptor. Bioinformatics 24:225–233
11. Jacob L, Vert J-P (2008) Protein-ligand interaction prediction: an improved chemogenomics approach. Bioinformatics 24:2149–2156
12. Yamanishi Y (2009) Supervised bipartite graph inference. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) Advances in neural information processing systems, vol 21. MIT Press, Cambridge, pp 1841–1848
13. Bleakley K, Yamanishi Y (2009) Supervised prediction of drug-target interactions using bipartite local models. Bioinformatics 25:2397–2403
14. Takigawa I, Tsuda K, Mamitsuka H (2011) Mining significant substructure pairs for interpreting polypharmacology in drug-target network. PloS One 6:e16999
15. Yamanishi Y, Pauwels E, Saigo H, Stoven V (2011) Extracting sets of chemical substructures and protein domains governing drug-target interactions. J Chem Inf Model 51:1183–1194
16. Tabei Y, Pauwels E, Stoven V, Takemoto K, Yamanishi Y (2012) Identification of chemogenomic features from drug-target interaction networks using interpretable classifiers. Bioinformatics 28:i487–i494
17. Tabei Y, Yamanishi Y (2013) Scalable prediction of compound-protein interactions using minwise hashing. BMC Syst Biol 7:S3
18. Iwata H, Mizutani S, Tabei Y, Kotera M, Goto S, Yamanishi Y (2013) Inferring protein domains associated with drug side effects based on drug-target interaction network. BMC Syst Biol 7:S18
19. Wishart D, Knox C, Guo A, Shrivastava S, Hassanali M, Stothard P, Chang Z, Woolsey J (2006) Drugbank: a comprehensive resource for in silico drug discovery and exploration. Nucleic Acids Res 34:D668–D672
20. The Uniprot Consortium (2010) The universal protein resource (UniProt) in 2010. Nucleic Acids Res 38:D142–D148
21. Finn R, Tate J, Mistry J, Coggill P, Sammut J, Hotz H, Ceric G, Forslund K, Eddy S, Sonnhammer E, Bateman A (2008) The Pfam protein families database. Nucleic Acids Res 36:D281–D288
22. Wang Y, Xiao J, Suzek T, Zhang J, Wang J, Bryant S (2009) Pubchem: a public information system for analyzing bioactivities of small molecules. Nucleic Acids Res 37:D623–D633
23. Greenacre M (1984) Theory and applications of correspondence analysis. Academic Press, New York
24. Dudoit S, Fridlyand J, Speed T (2002) Comparison of discrimination methods for the classification of tumors using gene expression data. J Am Stat Assoc 97:77–87
25. Tibshirani R, Hastie T, Narasimhan B, Chu G (2003) Class prediction by nearest shrunken centroids, with applications to DNA microarrays. Stat Sci 18:104–117
26. Witten D, Tibshirani R, Hastie T (2009) A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. Biostatistics **10**:515–534
27. Mahe P, Ralaivola L, Stoven V, Vert J (2006) The pharmacophore kernel for virtual screening with support vector machines. J Chem Inf Model 46:2003–2014
28. Kratochwil N, Malherbe P, Lindemann L, Ebeling M, Hoener M, Muhlemann A, Porter R, Stahl M, Gerber P (2005) An automated system for the analysis of g protein-coupled receptor transmembrane binding pockets: Alignment, receptor-based pharmacophores, and their application. J Chem Inf Model 45:1324–1336
29. Jacob L, Hoffmann B, Stoven V, Vert J-P (2009) Virtual screening of GPCRs: an in silico chemogenomics approach. BMC Bioinf 9:363
30. Campillos M, Kuhn M, Gavin A, Jensen L, Bork P (2008) Drug target identification using side-effect similarity. Science 321(5886):263–266
31. Takarabe M, Kotera M, Nishimura Y, Goto S, Yamanishi Y (2012) Drug target prediction using adverse event report systems: a pharmacogenomic approach. Bioinformatics 28:i611–i618
32. Yamanishi Y, Kotera M, Kanehisa M, Goto S (2010) Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework. Bioinformatics 26:i246–i254
33. Atias N, Sharan R (2011) An algorithmic framework for predicting side-effects of drugs. J Comput Biol 18:207–218

34. Iorio F, Tagliaferri R, di Bernardo D (2009) Identifying network of drug mode of action by gene expression profiling. J Comput Biol 16:241–251
35. Iorio F, Bosotti R, Scacheri E, Belcastro V, Mithbaokar P, Ferriero, R, Murino L, Tagliaferri R, Brunetti-Pierri N, Isacchi A et al (2010) Discovery of drug mode of action and drug repositioning from transcriptional responses. Proc Natl Acad Sci 107:14621–14626
36. Wang K, Sun J, Zhou S, Wan C, Qin S, Li C, He L, Yang L (2013) Prediction of drug-target interactions for drug repositioning only based on genomic expression similarity. PLoS Comput Biol 9:e1003315
37. Hizukuri Y, Sawada R, Yamanishi Y (2015) Predicting target proteins for drug candidate compounds based on drug-induced gene expression data in a chemical structure-independent manner. BMC Med Genomics 8:1
38. Iwata M, Sawada R, Kotera M, Yamanishi Y (2017) Elucidating the modes of action of bioactive compounds by large-scale compound-induced transcriptomics: toward drug discovery and repositioning. Sci Rep 7:40164
39. Hotelling, H (1936) Relation between two sets of variates. Biometrika 28:322–277
40. Fan RE, Chang KW, Hsieh CJ, Wang X, Lin CJ (2008) LIBLINEAR: a library for large linear classification. J Mach Learn Res 9:1871–1874

Chapter 14

DrugE-Rank: Predicting Drug-Target Interactions by Learning to Rank

Jieyao Deng, Qingjun Yuan, Hiroshi Mamitsuka, and Shanfeng Zhu

Abstract

Identifying drug-target interactions is crucial for the success of drug discovery. Approaches based on machine learning for this problem can be divided into two types: feature-based and similarity-based methods. By utilizing the "Learning to rank" framework, we propose a new method, DrugE-Rank, to combine these two different types of methods for improving the prediction performance of new candidate drugs and targets. DrugE-Rank is available at http://datamining-iip.fudan.edu.cn/service/DrugE-Rank/.

Key words DrugE-Rank, Learning to rank, Drug discovery

1 Introduction

Identifying drug-target interactions is crucial for the success of drug discovery. It can facilitate the understanding of drug side effect [1–3], disease pathology, as well as the drug action mechanism. Compared with using biochemical experiments to identify drug-target interaction, computational approaches are more efficient and economical. Approaches based on machine learning for this problem can be divided into two types: feature-based and similarity-based methods [4–6]. By utilizing the 'Learning to rank' (LTR) [7, 8] framework, we propose a new method, DrugE-Rank [9], to combine these two different types of methods for improving the prediction performance of new candidate drugs and targets.

We are interested in the problem of predicting drug-target interactions for new drugs or new targets. This problem is especially challenging, due to three main reasons. Firstly, since there are no known interactions for new drug or target, the training of prediction models is difficult. Secondly, existing computational methods based on LTR do not consider the connections among different drugs or targets very well. Thirdly, the prediction of drug-target interaction is a challenging multi-label learning problem, where a new target (or drug) has multiple interacting drugs (or targets).

Compared with previous computational approaches, DrugE-Rank has multiple advantages. Firstly, by utilizing the LTR paradigm, DrugE-Rank can solve this multi-label learning problem naturally and provide the most powerful performance. Secondly, DrugE-Rank integrates diverse cutting-edge techniques in the framework of LTR, which include both similarity-based and feature-based methods. Thirdly, DrugE-Rank only considers the top drug (or target) candidates recommended by each component method, which can greatly reduce the computational burden.

2 Materials

The performance of DrugE-Rank was examined by using DrugBank [10], a manually annotated drug-target interaction database. We carried out three rounds of experiments: (1) cross validation over DrugBank data with FDA-approved drugs before March 2014, (2) independent test over DrugBank data with new targets and FDA-approved drugs after March 2014, and (3) independent test over FDA experimental drugs. The experimental results demonstrate that DrugE-Rank outperformed all competing methods, being statistically significant. The improvement is especially promising for new drugs. Finally, we train DrugE-Rank with DrugBank data by the end of 2015. It consists of 1324 human protein targets, 1242 FDA-approved drugs, and altogether 5484 known interactions.

3 Methods

Six cutting-edge similarity-based methods are used in DrugE-Rank as component methods in the LTR framework: bipartite local model with support vector classification (BLM-svc) [11], bipartite local model with support vector regression (BLMsvr), k-nearest neighbor (k-NN) [12], weighted nearest neighbor-based Gaussian Interaction Profile classifier (WNN-GIP) [13], Laplacian regularized least squares (LapRLS) [14], and network-based Laplacian regularized least squares(NetLapRLS). In addition, we extract drug features using RDKit (*see* **Note 1**) and target features from PROFEAT [15].

4 Usage

4.1 New Drug

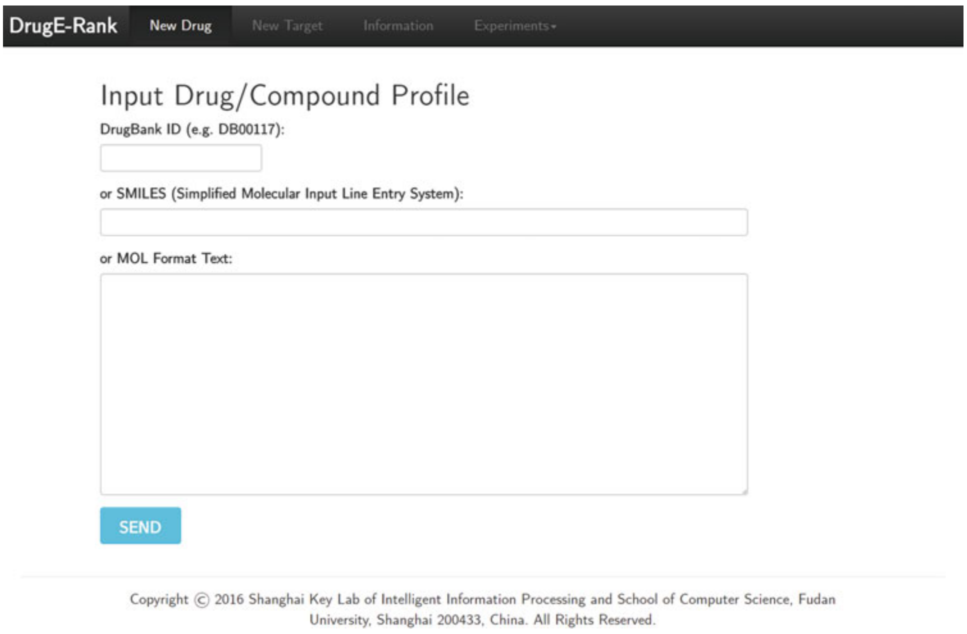
Given a new drug, DrugE-Rank returns the top 20 targets as the predicted result. The input interface is shown in Fig. 1.

1. Choose input format. You can input the drug profile by DrugBank ID, SMILES or MOL Format Text. An example of input is shown in Fig. 2.
2. Click the "Send" button. Click the button at the bottom of the page and your task will be in processing. The process takes about 10 min, and the server will return the top 20 predictions for each method (DrugE-Rank and six similarity-based methods). The result can help you to prioritize the most promising targets (Fig. 3).

4.2 New Target

Given a new target, DrugE-Rank returns the top 20 drugs as the predicted result. The input interface is shown in Fig 4.

1. Choose input format. You can input the target profile by UniProt ID or amino acid sequence (FASTA format). An example of input is shown in Fig. 5.
2. Click the "Send" button. Click the button at the bottom of the page and your task will be in processing. The process takes around 10 min, and the server will return the top 20 predictions for each method (DrugE-Rank and six similarity-based methods). The result may help you to prioritize the most promising drugs (Fig. 6).



Input Drug/Compound Profile

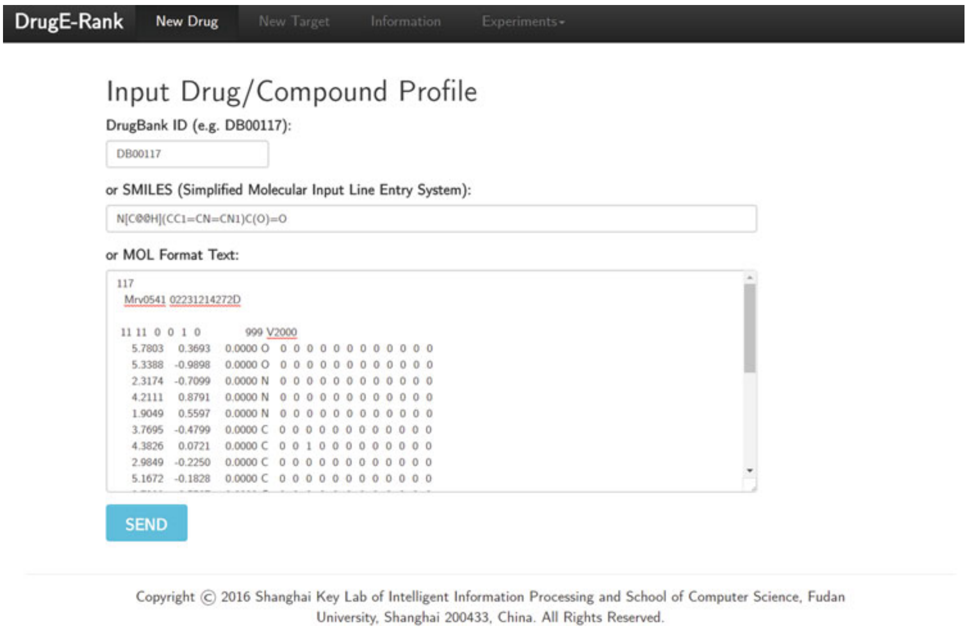
DrugBank ID (e.g. DB00117):

or SMILES (Simplified Molecular Input Line Entry System):

or MOL Format Text:

Copyright © 2016 Shanghai Key Lab of Intelligent Information Processing and School of Computer Science, Fudan University, Shanghai 200433, China. All Rights Reserved.

Fig. 1 Input interface for new drug



Input Drug/Compound Profile

DrugBank ID (e.g. DB00117):

DB00117

or SMILES (Simplified Molecular Input Line Entry System):

N[C@@H](CC1=CN=CN1)C(O)=O

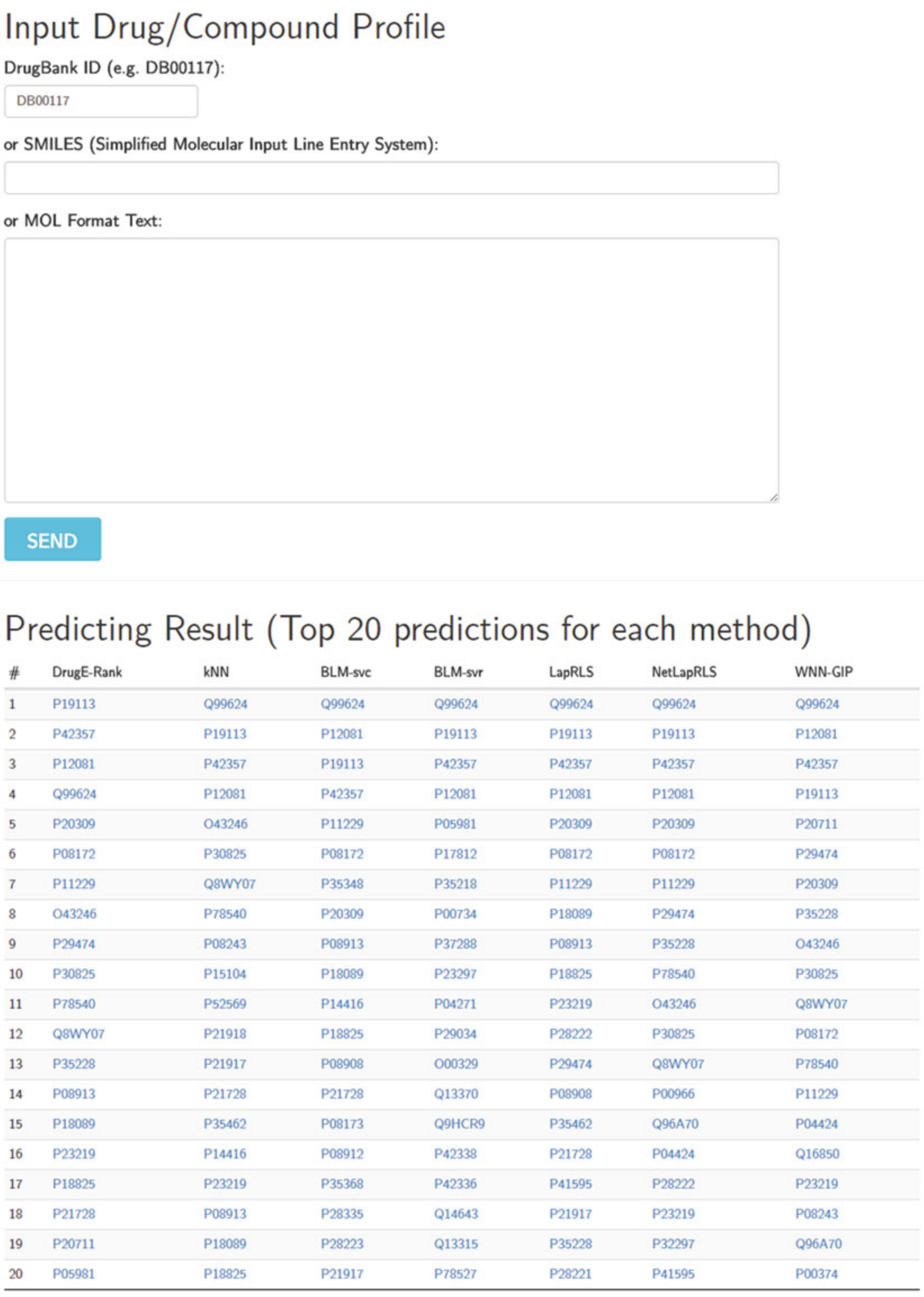
or MOL Format Text:

```
117
  Mrv0541 02231214272D

 11 11  0  0  1  0            999 V2000
    5.7803    0.3693    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
    5.3388   -0.9898    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
    2.3174   -0.7099    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0
    4.2111    0.8791    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0
    1.9049    0.5597    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0
    3.7695   -0.4799    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    4.3826    0.0721    0.0000 C   0  0  1  0  0  0  0  0  0  0  0  0
    2.9849   -0.2250    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    5.1672   -0.1828    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
```

Copyright © 2016 Shanghai Key Lab of Intelligent Information Processing and School of Computer Science, Fudan University, Shanghai 200433, China. All Rights Reserved.

Fig. 2 Input example for new drug



Input Drug/Compound Profile

DrugBank ID (e.g. DB00117):

DB00117

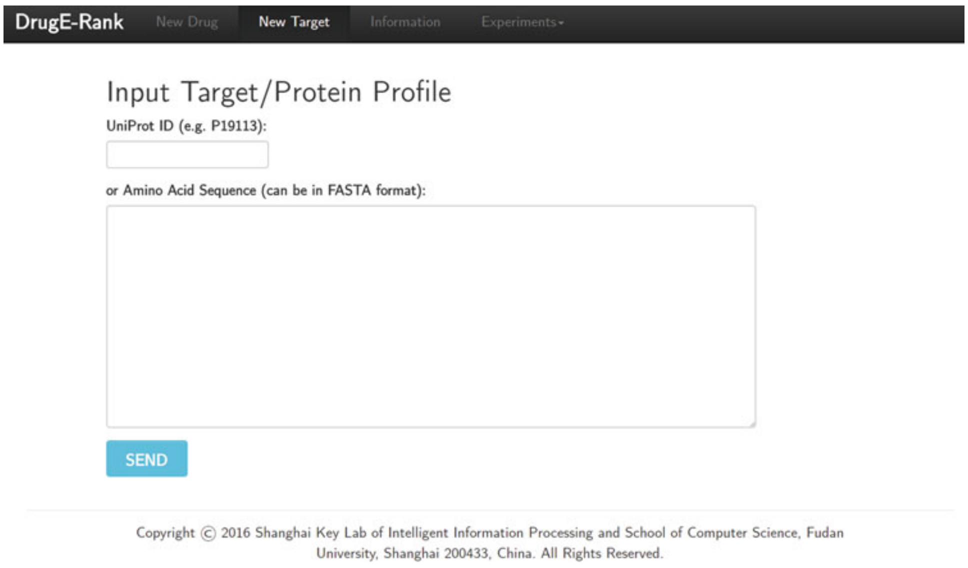
or SMILES (Simplified Molecular Input Line Entry System):

or MOL Format Text:

Predicting Result (Top 20 predictions for each method)

#	DrugE-Rank	kNN	BLM-svc	BLM-svr	LapRLS	NetLapRLS	WNN-GIP
1	P19113	Q99624	Q99624	Q99624	Q99624	Q99624	Q99624
2	P42357	P19113	P12081	P19113	P19113	P19113	P12081
3	P12081	P42357	P19113	P42357	P42357	P42357	P42357
4	Q99624	P12081	P42357	P12081	P12081	P12081	P19113
5	P20309	O43246	P11229	P05981	P20309	P20309	P20711
6	P08172	P30825	P08172	P17812	P08172	P08172	P29474
7	P11229	Q8WY07	P35348	P35218	P11229	P11229	P20309
8	O43246	P78540	P20309	P00734	P18089	P29474	P35228
9	P29474	P08243	P08913	P37288	P08913	P35228	O43246
10	P30825	P15104	P18089	P23297	P18825	P78540	P30825
11	P78540	P52569	P14416	P04271	P23219	O43246	Q8WY07
12	Q8WY07	P21918	P18825	P29034	P28222	P30825	P08172
13	P35228	P21917	P08908	O00329	P29474	Q8WY07	P78540
14	P08913	P21728	P21728	Q13370	P08908	P00966	P11229
15	P18089	P35462	P08173	Q9HCR9	P35462	Q96A70	P04424
16	P23219	P14416	P08912	P42338	P21728	P04424	Q16850
17	P18825	P23219	P35368	P42336	P41595	P28222	P23219
18	P21728	P08913	P28335	Q14643	P21917	P23219	P08243
19	P20711	P18089	P28223	Q13315	P35228	P32297	Q96A70
20	P05981	P18825	P21917	P78527	P28221	P41595	P00374

Fig. 3 Output example for new drug



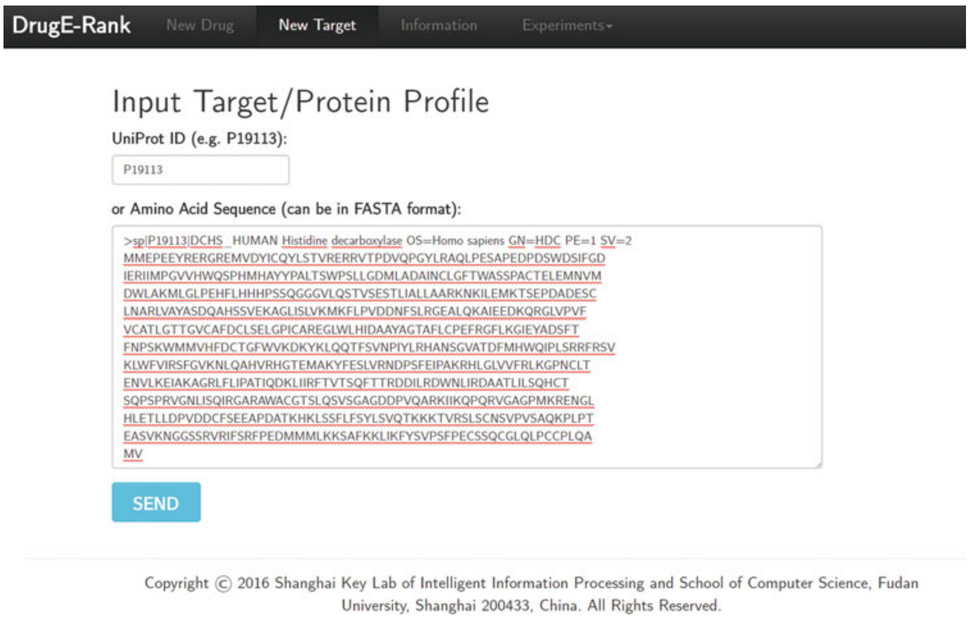
Input Target/Protein Profile

UniProt ID (e.g. P19113):

or Amino Acid Sequence (can be in FASTA format):

Copyright © 2016 Shanghai Key Lab of Intelligent Information Processing and School of Computer Science, Fudan University, Shanghai 200433, China. All Rights Reserved.

Fig. 4 Input interface for new target



Input Target/Protein Profile

UniProt ID (e.g. P19113):

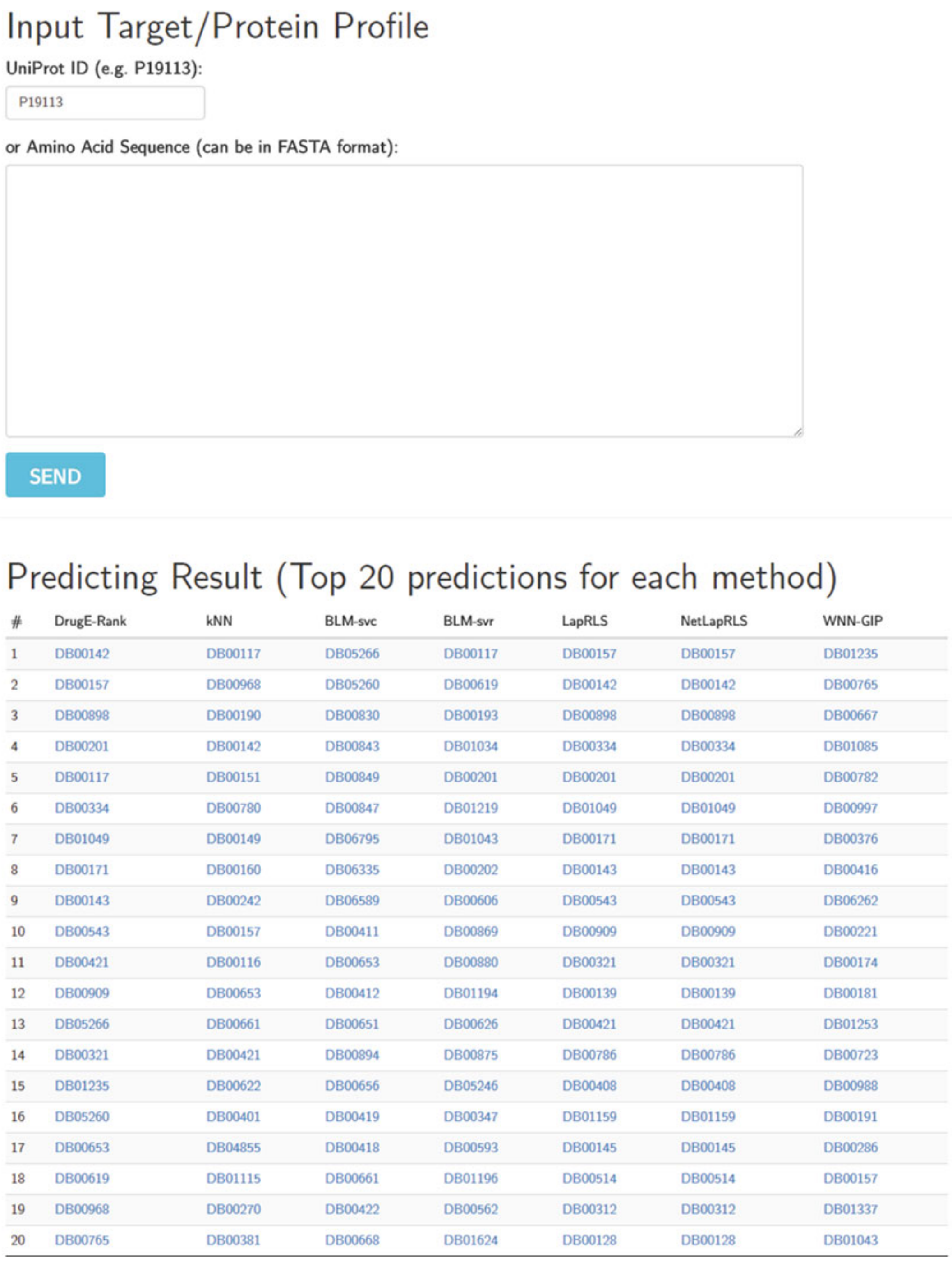
P19113

or Amino Acid Sequence (can be in FASTA format):

```
>sp|P19113|DCHS_HUMAN Histidine decarboxylase OS=Homo sapiens GN=HDC PE=1 SV=2
MMEPEEYRERGREMVDYICQYLSTVRERRVTPDVQPGYLRAQLPESAPEDPDSWDSIFGD
IERIIMPGVVHWQSPHMHAYYPALTSWPSLLGDMLADAINCLGFTWASSPACTELEMNVM
DWLAKMLGLPEHFLHHHPSSQGGGVLQSTVSESTLIALLAARKNKILEMKTSEPDADESC
LNARLVAYASDQAHSSVEKAGLISLVKMKFLPVDDNFSLRGEALQKAIEEDKQRGLVPVF
VCATLGTTGVCAFDCLSELGPICAREGLWLHIDAAYAGTAFLCPEFRGFLKGIEYADSFT
FNPSKWMMVHFDCTGFWVKDKYKLQQTFSVNPIYLRHANSGVATDFMHWQIPLSRRFRSV
KLWFVIRSFGVKNLQAHVRHGTEMAKYFESLVRNDPSFEIPAKRHLGLVVFRLKGPNCLT
ENVLKEIAKAGRLFLIPATIQDKLIIRFTVTSQFTTRDDILRDWNLIRDAATLILSQHCT
SQPSPRVGNLISQIRGARAWACGTSLQSVSGAGDDPVQARKIIKQPQRVGAGPMKRENGL
HLETLLDPVDDCFSEEAPDATKHKLSSFLFSYLSVQTKKKTVRSLSCNSVPVSAQKPLPT
EASVKNGGSSRVRIFSRFPEDMMMLKKSAFKKLIKFYSVPSFPECSSQCGLQLPCCPLQA
MV
```

Copyright © 2016 Shanghai Key Lab of Intelligent Information Processing and School of Computer Science, Fudan University, Shanghai 200433, China. All Rights Reserved.

Fig. 5 Input example for new target



Input Target/Protein Profile

UniProt ID (e.g. P19113):

P19113

or Amino Acid Sequence (can be in FASTA format):

Predicting Result (Top 20 predictions for each method)

#	DrugE-Rank	kNN	BLM-svc	BLM-svr	LapRLS	NetLapRLS	WNN-GIP
1	DB00142	DB00117	DB05266	DB00117	DB00157	DB00157	DB01235
2	DB00157	DB00968	DB05260	DB00619	DB00142	DB00142	DB00765
3	DB00898	DB00190	DB00830	DB00193	DB00898	DB00898	DB00667
4	DB00201	DB00142	DB00843	DB01034	DB00334	DB00334	DB01085
5	DB00117	DB00151	DB00849	DB00201	DB00201	DB00201	DB00782
6	DB00334	DB00780	DB00847	DB01219	DB01049	DB01049	DB00997
7	DB01049	DB00149	DB06795	DB01043	DB00171	DB00171	DB00376
8	DB00171	DB00160	DB06335	DB00202	DB00143	DB00143	DB00416
9	DB00143	DB00242	DB06589	DB00606	DB00543	DB00543	DB06262
10	DB00543	DB00157	DB00411	DB00869	DB00909	DB00909	DB00221
11	DB00421	DB00116	DB00653	DB00880	DB00321	DB00321	DB00174
12	DB00909	DB00653	DB00412	DB01194	DB00139	DB00139	DB00181
13	DB05266	DB00661	DB00651	DB00626	DB00421	DB00421	DB01253
14	DB00321	DB00421	DB00894	DB00875	DB00786	DB00786	DB00723
15	DB01235	DB00622	DB00656	DB05246	DB00408	DB00408	DB00988
16	DB05260	DB00401	DB00419	DB00347	DB01159	DB01159	DB00191
17	DB00653	DB04855	DB00418	DB00593	DB00145	DB00145	DB00286
18	DB00619	DB01115	DB00661	DB01196	DB00514	DB00514	DB00157
19	DB00968	DB00270	DB00422	DB00562	DB00312	DB00312	DB01337
20	DB00765	DB00381	DB00668	DB01624	DB00128	DB00128	DB01043

Fig. 6 Output example for new target

5 Notes

1. http://www.rdkit.org/.

Acknowledgments

This work has been partially supported by National Natural Science Foundation of China (Grant Nos: 61572139), MEXT KAKENHI #16H02868, and FiDiPro by Tekes.

References

1. Keiser MJ, Setola V, Irwin JJ et al (2009) Predicting new molecular targets for known drugs. Nature 462(7270):175–181
2. Lounkine E, Keiser MJ, Whitebread S et al (2012) Large-scale prediction and testing of drug activity on side-effect targets. Nature 486(7403):361–367
3. Nunez S, Venhorst J, Kruse CG (2012) Target-drug interactions: first principles and their application to drug discovery. Drug Discov Today 17:10–22
4. Ding H, Takigawa I, Mamitsuka H, Zhu S (2014) Similarity-based machine learning methods for predicting drug–target interactions: a brief review. Brief Bioinform 15(5):734–747
5. Zheng X, Ding H, Mamitsuka H, Zhu S (2013) Collaborative matrix factorization with multiple similarities for predicting drug-target interactions. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 1025–1033
6. Takigawa I, Mamitsuka H (2013) Graph mining: procedure, application to drug discovery and recent advance. Drug Discov Today 18(1–2):50–57
7. Liu T (2009) Learning to rank for information retrieval. Found Trends Inf Retr 3(3): 225–331
8. Li H (2011) A short introduction to learning to rank. IEICE Transactions 94-D(10):1854–1862
9. Yuan Q, Gao J, Wu D et al (2016) DrugE-Rank: improving drug-target interaction prediction of new candidate drugs or targets by ensemble learning to rank. Bioinformatics 32(12):i18–i27
10. Law V, Knox C, Djoumbou Y, Jewison T, Guo AC, Liu Y, Maciejewski A, Arndt D, Wilson M, Neveu V et al (2014) Drugbank 4.0: shedding new light on drug metabolism. Nucleic Acids Res 42(D1):D1091–D1097
11. Bleakley K, Yamanishi Y (2009) Supervised prediction of drug-target interactions using bipartite local models. Bioinformatics 25(18):2397–2403
12. Van LT, Marchiori E (2013) Predicting drug-target interactions for new drug compounds using a weighted nearest neighbor profile. PLoS One 8(6):e66952
13. Van LT, Nabuurs SB, Marchiori E (2011) Gaussian interaction profile kernels for predicting drug-target interaction. Bioinformatics 27(21):3036–3043
14. Xia Z, Zhou X, Sun Y, Wu L (2009) Semi-supervised drug-protein interaction prediction from heterogeneous spaces. In: The Third International Symposium on Optimization and Systems Biology, vol 11. pp 123–131
15. Rao H, Zhu F, Yang G, Li Z, Chen Y (2011) Update of profeat: a web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence. Nucleic Acids Res 39(Suppl 2):W385–W390

Chapter 15

MeSHLabeler and DeepMeSH: Recent Progress in Large-Scale MeSH Indexing

Shengwen Peng, Hiroshi Mamitsuka, and Shanfeng Zhu

Abstract

The US National Library of Medicine (NLM) uses the Medical Subject Headings (MeSH) (*see* **Note 1**) to index almost all 24 million citations in MEDLINE, which greatly facilitates the application of biomedical information retrieval and text mining. Large-scale automatic MeSH indexing has two challenging aspects: the MeSH side and citation side. For the MeSH side, each citation is annotated by only 12 (on average) out of all 28,000 MeSH terms. For the citation side, all existing methods, including Medical Text Indexer (MTI) by NLM, deal with text by bag-of-words, which cannot capture semantic and context-dependent information well. To solve these two challenges, we developed the MeSHLabeler and DeepMeSH. By utilizing "learning to rank" (LTR) framework, MeSHLabeler integrates multiple types of information to solve the challenge in the MeSH side, while DeepMeSH integrates deep semantic representation to solve the challenge in the citation side. MeSHLabeler achieved the first place in both BioASQ2 and BioASQ3, and DeepMeSH achieved the first place in both BioASQ4 and BioASQ5 challenges. DeepMeSH is available at http://datamining-iip.fudan.edu.cn/deepmesh.

Key words MeSH indexing, Text categorization, Multi-label classification, Medical subject headings, MEDLINE, Machine learning

1 Introduction

MEDLINE (*see* **Note 2**) is the largest biomedical literature database in the world, which contains more than 24 million citations. MeSH terms are used to index almost all MEDLINE citations [1], which is crucial in biomedical text mining and information retrieval [2–8]. The NLM annotators who are responsible for annotating the MeSHs need to review the full text of a citation, which costs lots of time and money. For the year 2016, there were 869,666 new citations in MEDLINE (*see* **Note 3**), and the average cost per citation was about $9.4 [9]. As of April 2016, there were 127 staff members in NLM who are responsible for annotating the most relevant MeSH terms to the MEDLINE citations [10]. As time goes on, the rapid increase of the MEDLINE citation poses great challenges for manual annotations. A fast and accurate automated MeSH indexing system is imperative to improve the indexing efficiency and reduce the cost.

NLM has developed an automated MeSH indexing system, MTI [1, 11, 12], to facilitate the annotation of MeSH. MTI mainly consists of two parts: MMI (MetaMap Indexing) [13] and PRC (PubMed Related Citations) [14]. MMI extracts the biomedical concept from the title and abstract and then maps it to the corresponding MeSH. Moreover, PRC tries to use the improved K-nearest neighbor (KNN) algorithm to find the most similar MEDLINE citations and then extracts MeSHs from these similar citations. The results of PRC and MMI were combined into a preliminary recommendation. After some processing (e.g., the application of the index rules), MTI generate the final MeSH recommended list to the NLM annotators.

Large-scale MeSH indexing mainly has two aspects of challenges from the MeSH side and the citation side, respectively. In the MeSH side, the difference between the distributions of different MeSHs is particularly large. For example, among all 28,000 MeSH terms, the most common MeSH, “Humans,” appears more than 8 million times in the MEDLINE, while a rare MeSH, such as “Pandanaceae,” only appears 31 times. In addition, the number of MeSHs annotated for each citation varies greatly, which might be less than 5 MeSHs or more than 30 MeSHs. In addition, in the side of citations, the “Bag of Words” method cannot effectively capture the complex semantics of biomedical documents because of the large number of concepts and abbreviations in biomedical literature. In many cases, similar concepts can be represented by different words, and the same words can express a completely different meaning from the context.

In order to promote the development of semantic indexing and automatic question answering systems in the biomedical field, BioASQ [15–17], a challenge on large-scale biomedical semantic indexing and question answering, held an international competition from 2013 to 2017. There have been many effective systems that have emerged through the platform, such as MetaLabeler [18] and MeSH Now [19]. To improve the performance of automatic MeSH indexing system, we developed two systems, MeSHLabeler [20] and DeepMeSH [21], which solve the challenges in the MeSH side and citation side, respectively. MeSHLabeler use “learning to rank” framework to incorporate multiple evidences to rank the MeSHs, while DeepMeSH integrates a new semantic representation to represent citations. MeSHLabeler achieved the first place in both BioASQ2 and BioASQ3, and DeepMeSH achieved the first place in both BioASQ4 and BioASQ5 challenges [22].

2 Materials

We use 2016 MeSH, containing 27,883 unique MeSH terms. Most of training data come from 2016 MEDLINE/PubMed baseline database downloaded from the NCBI website. Another part of data is downloaded from BioASQ 2015 challenge Task 3a, with 49,774 indexed. The text of all these citations only contains abstract, article title, and journal title. DeepMeSH consists of two components, MeSHRanker and MeSHNumber. Given a target citation, MeSHRanker returns a ranked list of candidate MeSH terms, while MeSHNumber predicts the number of associated MeSH terms. For the 49,774 citations from BioASQ 2015, we randomly assign them to three sets: MeSHRanker training set (with 23,774 citations), MeSHNumber training set (with 20,000 citations), and local test set (with 6000 citations).

Our system was mainly written by C++. It also used many open source tools to implement the whole flow.

1. BioTokenizer was used to tokenize and stem raw text.
2. LIBLINEAR was used to implement logistic regression and linear SVM.
3. XGBoost was used to implement learning to rank framework.

Our server has 4 Intel XEON E5–4650 2.7GHzs CPUs and 128G memory. It costs around 7 days to train 27,000 binary classifiers with logistic regression or linear svm. Predicting 10,000 citations costs around 3 h.

3 Methods

DeepMeSH is the “state of the art” of the MeSH indexing system. It improves the MeSH indexing accuracy by incorporating deep semantic representation to MeSHLabeler. The deep semantic representation, D2V-TFIDF, combines the advantages of the D2V (document vector) and TFIDF (term frequency with inverse document frequency). According to our experiments, D2V-TFIDF represents citation texts better than both D2V and TFIDF. It is more powerful to find similar citations, so we use this representation to solve the challenge on the citation side.

MeSHLabeler is the last generation of MeSH indexing system, which uses the “learning to rank” framework to incorporate multiple evidence to solve the challenge on the MeSH side. It has two components: MeSHRanker and MeSHNumber. MeSHRanker is used to rank the candidate MeSH terms for each target citation. On the other hand, MeSHNumber is used to predict the number of associated MeSH terms for the target citation. MeSHRanker incorporates five different types of evidence to rank the MeSHs, which includes global evidence, local evidence, MeSH dependency, pattern matching, and MTI.

- Global evidence: We train a binary classifier for each MeSH with the entire MEDLINE. Since each MeSH is trained independently, the scores returned by the different classifiers are theoretically incomparable. MeSHLabeler proposed a normalized method to deal with the score comparisons between different models, which significantly improves the prediction accuracy. Because each MeSH binary classifier is trained with the entire MEDLINE, we call this part of the evidence as global evidence.
- Local evidence: For a target citation, we can score the candidate MeSHs through counting the MeSHs indexed by its similar citations.
- MeSH dependency: It is a unique feature of MeSHLabeler that effectively considers the relevance of the MeSH-MeSH terms. For infrequent MeSH terms, this information can effectively improve the accuracy of labeling. Since the number of MeSH-MeSH combinations was very large, none of the previous studies considered MeSH dependency.
- Patten matching: We directly use the string matching method to find the MeSHs or their synonyms in the title or abstract.
- MTI: MTI considers not only pattern matching and local evidence but also the index rules with domain knowledge. We integrate the results from MTI.

4 Usage

The input interface is shown in Fig. 1.

1. Select a file. This is to upload the citations for MeSH indexing. Two file extensions, ".txt" and ".json," are supported. If the file extension is ".txt," the file must contain the PubMed IDs (pmid) of all target citations, and each line contains a pmid. If the file extension is ".json," the file should contain all raw texts of target citations. The text contains abstract and title. Note that each submit file must contain at least 100 instances. Sample input was shown as Fig. 2
2. Input an email address. Input your email address to receive the prediction result. The email address will be used to receive a process ID, prediction result, or some information if any error occurs.
3. Upload the file. Click the submit button, the file will be uploaded. Once uploaded successfully, you will receive a process

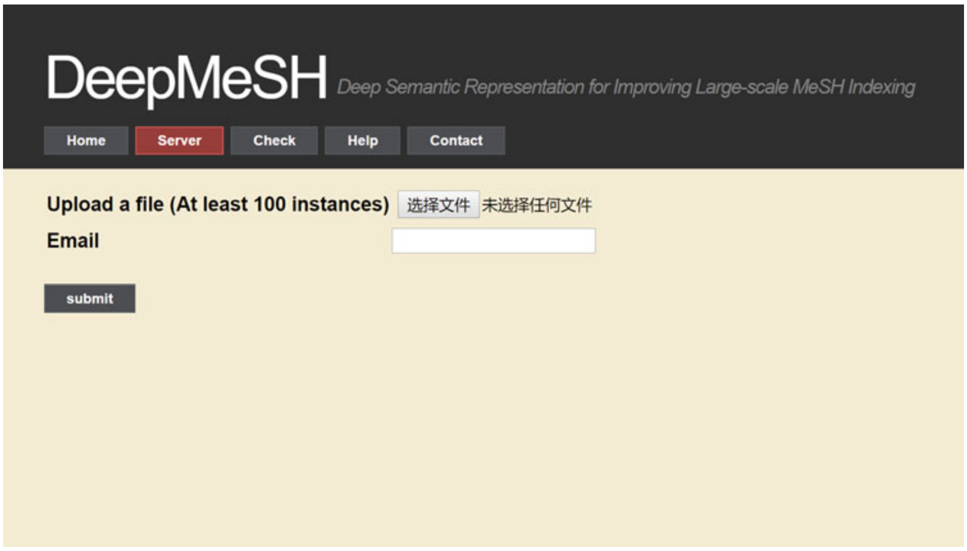
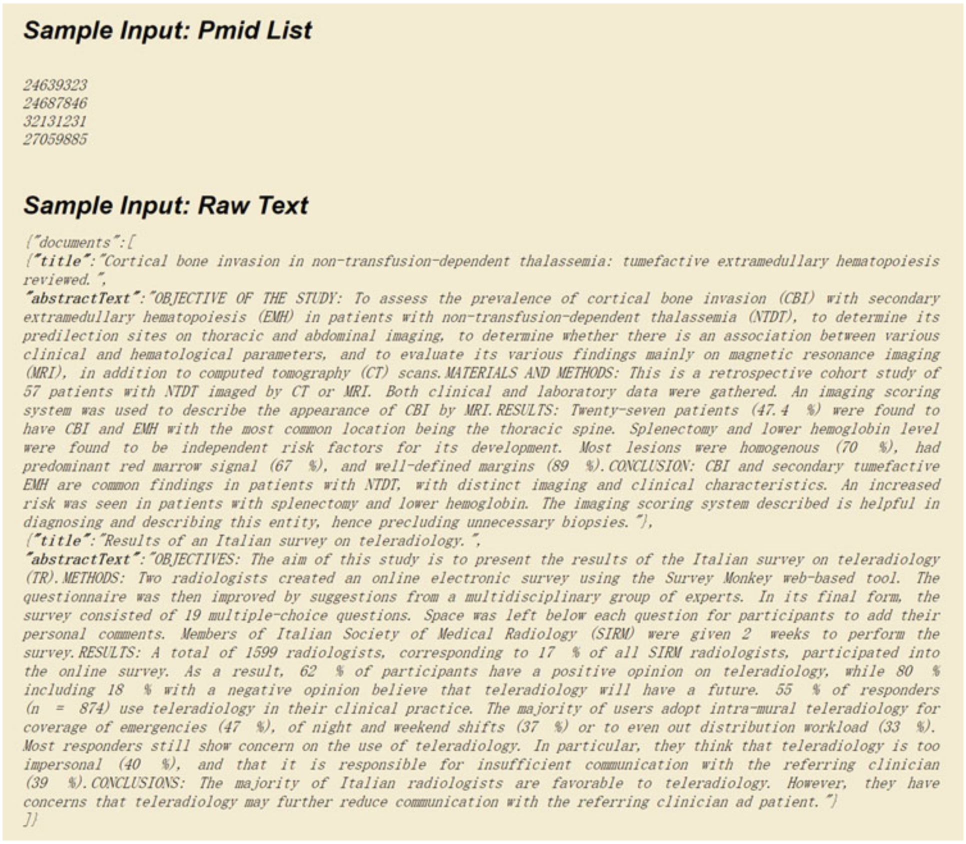

DeepMeSH
Deep Semantic Representation for Improving Large-scale MeSH Indexing
Home
Server
Check
Help
Contact
Upload a file (At least 100 instances)
选择文件
未选择任何文件
Email
submit

Fig. 1 The input interface of DeepMeSH



Sample Input: Pmid List

24639323
24687846
32131231
27059885

Sample Input: Raw Text

```
{"documents":[
{"title":"Cortical bone invasion in non-transfusion-dependent thalassemia: tumefactive extramedullary hematopoiesis reviewed.",
"abstractText":"OBJECTIVE OF THE STUDY: To assess the prevalence of cortical bone invasion (CBI) with secondary extramedullary hematopoiesis (EMH) in patients with non-transfusion-dependent thalassemia (NTDT), to determine its predilection sites on thoracic and abdominal imaging, to determine whether there is an association between various clinical and hematological parameters, and to evaluate its various findings mainly on magnetic resonance imaging (MRI), in addition to computed tomography (CT) scans.MATERIALS AND METHODS: This is a retrospective cohort study of 57 patients with NTDT imaged by CT or MRI. Both clinical and laboratory data were gathered. An imaging scoring system was used to describe the appearance of CBI by MRI.RESULTS: Twenty-seven patients (47.4 %) were found to have CBI and EMH with the most common location being the thoracic spine. Splenectomy and lower hemoglobin level were found to be independent risk factors for its development. Most lesions were homogenous (70 %), had predominant red marrow signal (67 %), and well-defined margins (89 %).CONCLUSION: CBI and secondary tumefactive EMH are common findings in patients with NTDT, with distinct imaging and clinical characteristics. An increased risk was seen in patients with splenectomy and lower hemoglobin. The imaging scoring system described is helpful in diagnosing and describing this entity, hence precluding unnecessary biopsies."},
{"title":"Results of an Italian survey on teleradiology.",
"abstractText":"OBJECTIVES: The aim of this study is to present the results of the Italian survey on teleradiology (TR).METHODS: Two radiologists created an online electronic survey using the Survey Monkey web-based tool. The questionnaire was then improved by suggestions from a multidisciplinary group of experts. In its final form, the survey consisted of 19 multiple-choice questions. Space was left below each question for participants to add their personal comments. Members of Italian Society of Medical Radiology (SIRM) were given 2 weeks to perform the survey.RESULTS: A total of 1599 radiologists, corresponding to 17 % of all SIRM radiologists, participated into the online survey. As a result, 62 % of participants have a positive opinion on teleradiology, while 80 % including 18 % with a negative opinion believe that teleradiology will have a future. 55 % of responders (n = 874) use teleradiology in their clinical practice. The majority of users adopt intra-mural teleradiology for coverage of emergencies (47 %), of night and weekend shifts (37 %) or to even out distribution workload (33 %). Most responders still show concern on the use of teleradiology. In particular, they think that teleradiology is too impersonal (40 %), and that it is responsible for insufficient communication with the referring clinician (39 %).CONCLUSIONS: The majority of Italian radiologists are favorable to teleradiology. However, they have concerns that teleradiology may further reduce communication with the referring clinician ad patient."}
]}
```

Fig. 2 A sample input of DeepMeSH

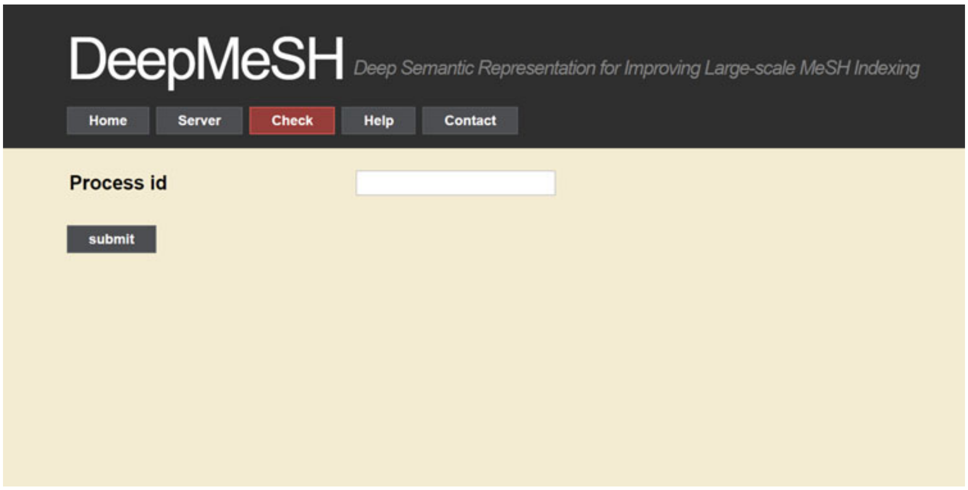

DeepMeSH
Deep Semantic Representation for Improving Large-scale MeSH Indexing
Home
Server
Check
Help
Contact
Process id
submit

Fig. 3 The process ID check interface of DeepMeSH

ID in your email. As shown in Fig. 3, you can use the process ID to check the prediction status.

For each successful submission, we will output a json file. For an unindexed pmid, we output the MeSHs recommended by DeepMeSH, and the answer_type has a value of "predicted." For an indexed pmid, we output the MeSHs indexed by PubMed, and the answer_type has a value of "annotated." If a pmid cannot be found in PubMed, the result will be empty, and the answer_type has a value of "not_found." Note that if the input file is a ".json" file which contains only texts, the output pmid is the index of the text in the file (starts from 0).

A sample output is shown as follows:

```
{"documents": [
{"labels":["D006801","D055815"],"pmid":24639323,"answer_type":"predicted
"},
{"labels":["D000293"],"pmid":24687846,"answer_type":"predicted"},
{"labels":["D005260","D058006"],"pmid":27059885,"answer_type":"annotate}
d"}, {"labels":[],"pmid":32131231, "answer_type":"not_found"}]}
```

5 Notes

1. https://www.nlm.nih.gov/pubs/factsheets/mesh.html
2. https://www.nlm.nih.gov/pubs/factsheets/medline.html
3. http://www.nlm.nih.gov/bsd/bsd_key.html

Acknowledgments

This work has been partially supported by National Natural Science Foundation of China (Grant Nos: 61572139), MEXT KAKENHI #16H02868 and FiDiPro by Tekes.

References

1. Aronson AR, Mork JG, Gay CW, Humphrey SM, Rogers WJ (2004) The NLM indexing initiativeś medical text indexer. Stud Health Technol Inform 107(Pt 1):268–272
2. Stokes N, Li Y, Cavedon L, Zobel J (2010) Exploring criteria for successful query expansion in the genomic domain. Inf Retr 12:17–50
3. Lu Z, Kim W, Wilbur WJ (2010) Evaluation of query expansion using MeSH in PubMed. Inf Retr 12:69–80
4. Zhu S, Takigawa I, Zeng J, Mamitsuka H (2009) Field independent probabilistic model for clustering multi-field documents. Inf Process Manage 45(5):555–570
5. Zhu S, Zeng J, Mamitsuka H (2009) Enhancing MEDLINE document clustering by incorporating MeSH semantic similarity. Bioinformatics 25(15):1944–1951
6. Gu J, Feng W, Zeng J, Mamitsuka H, Zhu S (2013) Efficient semisupervised MEDLINE document clustering with MeSH-semantic and global-content constraints. IEEE Trans Cybernetics 43(4):1265–1276
7. Zhou J, Shui Y, Peng S, Li X, Mamitsuka H, Zhu S (2015) MeSHSim: An R/Bioconductor package for measuring semantic similarity over MeSH headings and MEDLINE documents. J Bioinform Comput Biol 13(6):1542002
8. Huang X, Zheng X, Yuan W, Wang F, Zhu S (2011) Enhanced clustering of biomedical documents using ensemble non-negative matrix factorization. Inform Sci 181(11):2293–2302
9. Mork JG, Jimeno-Yepes A, Aronson AR (2013) The NLM medical text indexer system for indexing biomedical literature. BioASQ@ CLEF
10. Demner-Fushman D, Mork JG (2016) A report to the board of Scientific Counselors, April 2016
11. Mork JG, Demner-Fushman D, Schmidt S, Aronson AR (2014) Recent Enhancements to the NLM Medical Text Indexer. CLEF (Working Notes), pp 1328–1336
12. Nelson SJ, Schopen M, Savage AG, Schulman JL, Arluk N (2004) The MeSH translation maintenance system: structure, interface design, and implementation. Medinfo 11:67–69
13. Aronson AR, Lang FM (2004) An overview of MetaMap: historical perspective and recent advances. J Am Med Inform Assoc 17:229–236
14. Lin J, Wilbur WJ (2007) PubMed related articles: a probabilistic topic-based model for content similarity. BMC Bioinformatics 8:423
15. Partalas I, Gaussier É, Ngomo ACN et al. (2013) Results of the first BioASQ Workshop. BioASQ@ CLEF
16. Tsatsaronis G et al (2015) An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. BMC Bioinformatics 16:138
17. Balikas G, Partalas I, Ngomo AN, Krithara A, Paliouras G (2014) Results of the BioASQ track of the question answering lab at CLEF 2014. CLEF (Working Notes), pp 1181–1193
18. Tsoumakas G, Laliotis M, Markantonatos N, Vlahavas IP (2013) Large-scale semantic indexing of biomedical publications. BioASQ@ CLEF
19. Mao Y, Lu Z (2013) NCBI at the 2013 BioASQ challenge task: learning to rank for automatic MeSH indexing. BioASQ@ CLEF
20. Liu K, Peng S, Wu J, Zhai C, Mamitsuka H, Zhu S (2015) MeSHLabeler: improving the accuracy of large-scale MeSH indexing by integrating diverse evidence. Bioinformatics 12:i339–i347
21. Peng S, You R, Wang H, Zhai C, Mamitsuka H, Zhu S (2016) DeepMeSH: deep semantic representation for improving large-scale MeSH indexing. Bioinformatics 32(12):i70–i79
22. Peng S, You R, Xie Z, Wang B, Zhang Y, Zhu S (2015) The Fudan participation in the 2015 BioASQ challenge: large-scale biomedical semantic indexing and question answering. CLEF (Working Notes)

Chapter 16

Disease Gene Classification with Metagraph Representations

Sezin Kircali Ata, Yuan Fang, Min Wu, Xiao-Li Li, and Xiaokui Xiao

Abstract

This chapter is based on exploiting the network-based representations of proteins, *metagraphs*, in protein-protein interaction network to identify candidate disease-causing proteins. Protein-protein interaction (PPI) networks are effective tools in studying the functional roles of proteins in the development of various diseases. However, they are insufficient without the support of additional biological knowledge for proteins such as their molecular functions and biological processes. To enhance PPI networks, we utilize biological properties of individual proteins as well. More specifically, we integrate keywords from UniProt database describing protein properties into the PPI network and construct a novel heterogeneous PPI-Keyword (PPIK) network consisting of both proteins and keywords. As proteins with similar functional duties or involving in the same metabolic pathway tend to have similar topological characteristics, we propose to represent them with metagraphs. Compared to the traditional network motif or subgraph, a metagraph can capture the topological arrangements through not only the protein-protein interactions but also protein-keyword associations. We feed those novel metagraph representations into classifiers for disease protein prediction and conduct our experiments on three different PPI databases. They show that the proposed method consistently increases disease protein prediction performance across various classifiers, by 15.3% in AUC on average. It outperforms the diffusion-based (e.g., RWR) and the module-based baselines by 13.8–32.9% in overall disease protein prediction. Breast cancer protein prediction outperforms RWR, PRINCE, and the module-based baselines by 6.6–14.2%. Finally, our predictions also exhibit better correlations with literature findings from PubMed database.

Key words Protein-protein interaction, UniProt keywords, Metagraph, Protein representations, Disease protein prediction

1 Introduction

Disease-causing genes and their protein products play a crucial role in the diagnosis and treatment of serious diseases such as cancer and diabetes. In addition to various efforts in identifying the functions of genes and proteins [1–4], protein-protein interaction (PPI) networks [5–8] have been widely exploited to reveal the proteins with similar functional duties or involving in the same metabolic pathway. Many studies validate that the position of a protein in a PPI network is not random [9–11]. Instead, proteins with the same phenotype or function tend to have common topological characteristics (e.g., degree, coreness, and closeness) and tend to be involved in the same complexes [12, 13]. By means of such characteristics, predicting the associations between proteins and diseases is possible. Network-based approaches in disease protein prediction (*see* **Note 1**) can be grouped into three categories: linkage-, module-, and diffusion-based methods [14]. Linkage methods [15–17] work on genomic linkage intervals. If a gene is in a disease linkage interval, then its protein products are disease proteins. Interacting protein with a known disease protein is considered as a disease candidate protein. Disease protein in a PPI network tends to be associated with the same disease. The linkage methods could not preserve their popularity in disease protein prediction because now it is known that the most of the diseases are results of complex interactions and it is inadequate to study solely direct interactions with a disease protein [14, 18]. Module-based methods are based on the hypothesis that proteins within the same topological or functional module on a network are more likely to be associated with the same disease [9, 18–20]. They are designed to detect a subnetwork that contains most of the disease-associated molecules (i.e., proteins/genes) in the interactome [14]. For this aim, mostly protein interaction networks are exploited based on the functional and/or topological similarities of the proteins. In disease module discovery, solely neighborhood considerations such as direct neighbors, shared neighborhood, and local shortest paths have the risk of sticking to the local assessment, so they perform better with the support of other biological information such as phenotype similarities and functional evidence. One advantage of direct neighbor-based methods is they rarely require parameter tuning phase. Diffusion-based methods are designed to discover the pathways that are closest to the known disease genes. They count known disease proteins as seeds and diffuse along interactome through random walks [14]. Since they consider the full network topology besides the placement of the known disease genes, they are dominant over module-based methods [14, 18, 21, 22]. Simply, their procedure is based on scoring the proteins as farther ones from the known disease proteins getting a lower score of being associated with the disease. Thus, diffusion-based methods are very suitable for prioritizing candidate disease proteins and widely used for this purpose.

However, it is known that PPI networks are noisy and incomplete [23, 24]. Apart from leveraging these networks, we should also consider the features of proteins, such as their Gene Ontology (GO) annotations and their subcellular compartments. Thus, we propose to integrate *keywords* acquired from the Universal Protein Resource (UniProt) database [25] into the PPI network. The *keywords* consist of various biological aspects of the proteins such as

Table 1
A summary of keywords from the UniProt database

Keyword category	Examples
Biological process	Apoptosis, cell cycle, cAMP biosynthesis
Cellular component	Golgi apparatus, vacuole, cytoplasm
Coding sequence diversity	Polymorphisms, RNA-editing, alternative splicing
Domain	SH2 domain, Kelch repeat, transmembrane
Ligand	cAMP, S-adenosyl-L-methionine, cGMP
Molecular function	RNA-binding, protein kinase inhibitor, chromatin regulator
Posttranslational modification	Phosphorylation, ubiquitination, acetylation
Technical term	Allosteric enzyme, transposable element

domain, posttranslational modification, molecular function, coding sequence diversity, etc. We sum them up in Table 1.

By means of this integration, we acquire the network characteristics between proteins and associated keywords as well. In the last decade, integration of additional biological concepts into a PPI network has been widely studied [26–30]. However, to the best of our knowledge, the keywords in UniProt database have not been used for the integration before. Thus, we construct a heterogeneous network to identify candidate disease proteins. We name the heterogeneous network as a *PPI-Keyword* (PPIK) network, which consists of knowledge about not only protein interactions with one another but also their functional and structural similarities. We propose to identify disease proteins by means of *metagraph* structures on PPIK network [31]. Apart from a traditional network motif or subgraph, a *metagraph* is a graph structure which can attain a topology of both proteins and their associated *keywords* on the network. Each metagraph keeps information of heteronomous biological layout between one or more proteins and keywords. We can represent each protein in terms of metagraphs so that we are able to identify its interactions with other proteins and associations with keywords. Our key motive is that proteins with similar functional duties or involving in the same metabolic pathway tend to have similar metagraph representations. In other words, they tend to share similar interaction patterns with other proteins and be associated with the same keywords in a similar layout on the PPIK network.

Consequently, we perform various supervised learning techniques for the prediction of disease proteins based on their metagraph representations on three PPI databases, namely, IntAct [32], STRING [33], and NCBI [34]. Finally, we observe the dominant performance of our proposed metagraph-based prediction model over the diffusion-based and module-based baselines.

2 Materials

2.1 PPI-Keyword (PPIK) Network

1. Keywords: To utilize the biological properties of individual proteins, we extract the *keywords* associated with each protein from the Universal Protein Resource (UniProt) database [25]. These keywords are terms which describe the various biological aspects of the proteins, as summarized in Table 1.
2. PPI network: Protein molecules are the products of genes. They work in collaboration to perform duties and rarely act alone. They have varying tasks within organisms, for example, enzymes catalyze reactions or proteins such as insulin and transmit a signal to other cells. To complete their tasks, they interact with each other in a transient way or stable way. Transient interactions carry out short-term actions such as binding of transcription factors. Stable interactions include the physically docking and forming a complex structure of proteins. Moreover, the success of the interactions in a cell depends on several circumstances such as cell type, cell cycle phase and state, developmental stage, environmental conditions, protein modifications (e.g., phosphorylation), etc. In a protein-protein interaction (PPI) network, proteins are represented with nodes and interactions are represented with edges. Since these interactions are essential biological processes, PPI networks are thus crucial for understanding human interactome and metabolism [35]. Recently, PPI networks have been widely explored for predicting protein functions [36], drug targets [37], essential genes [5], function modules/protein complexes [38, 39], etc.
3. PPIK network: We enrich three PPI network databases, IntAct [32], STRING [33], and NCBI Entrez Gene [34] with *keywords*, to construct a PPI-Keyword (PPIK) network. While integrating the keywords acquired from UniProt database into PPI network databases, protein Id conversions take place, and corresponding UniProt entries is tried to find. If there is a match, then the keyword is associated with the protein in the PPI network (*see* **Note 2**).

Formally, a PPIK network is an undirected graph $G = (V, E, l)$, such that V represents the set of nodes, E represents the set of edges, and l is the label function on V. Note that each node can be either a protein or a keyword. Let the label function $l : V \to \{\text{protein}, \text{keyword}\}$ discriminate the node types. Moreover, an edge can represent either the link between two proteins or a protein and a keyword. The former represents the mapped interactions between the two proteins, and the latter

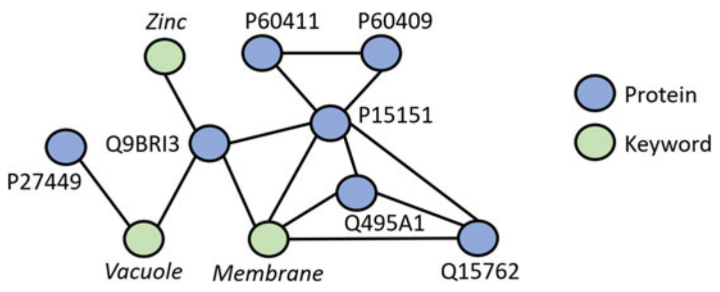


Fig. 1 Part of the PPIK network based on UniProt and IntAct databases

Table 2
Summary of the three PPIK networks

		Disease proteins				
	Proteins	**All**	**Breast cancer**	**Keywords**	**PPI edges**	**PPIK edges**
IntAct	13,063	2947	29	554	97,652	246,092
NCBI	15,951	3476	31	567	227,004	405,632
STRING	17,668	3539	29	567	3,912,853	4,107,335

represents the association between the protein and keyword. This integration improves the reliability of the original noisy and incomplete network. Protein-keyword associations may strengthen useful protein-protein interactions. Furthermore, proteins with no direct interactions can now become related through keywords.

Figure 1 illustrates a part of the constructed PPIK network grounded on the UniProt and IntAct databases. Note that the color scheme of the nodes essentially serves as the label function.

2.2 Disease Genes

Disease labels for proteins are obtained using the UniProt and OMIM databases. We first obtain disease genes from OMIM [40] and further map these genes to their product proteins based on UniProt.

Table 2 summarizes the three PPIK networks, and as we can realize, the three PPIK networks are very different in terms of number of proteins, number of PPI edges, as well as number of PPIK edges.

3 Methods

The overall framework of the study is shown in Fig. 2.

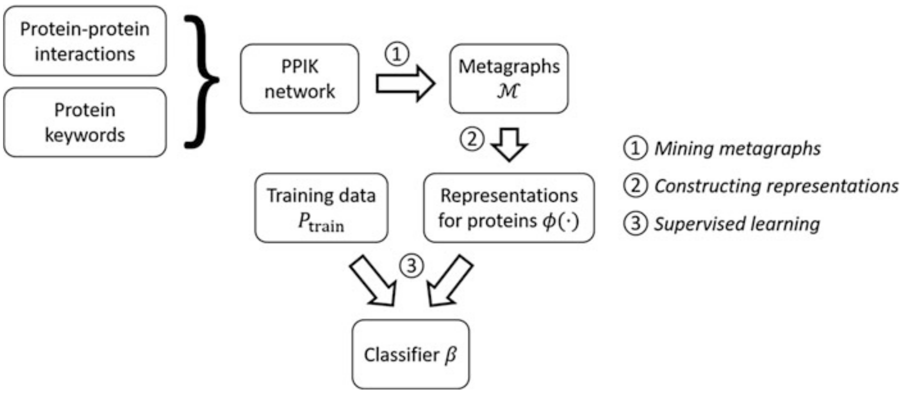

Protein-protein interactions
Protein keywords
PPIK network
①
Metagraphs $\mathcal{M}$
②
Training data P_{train}
Representations for proteins $\phi(\cdot)$
③
Classifier β
① Mining metagraphs
② Constructing representations
③ Supervised learning

Fig. 2 General framework of the proposed method

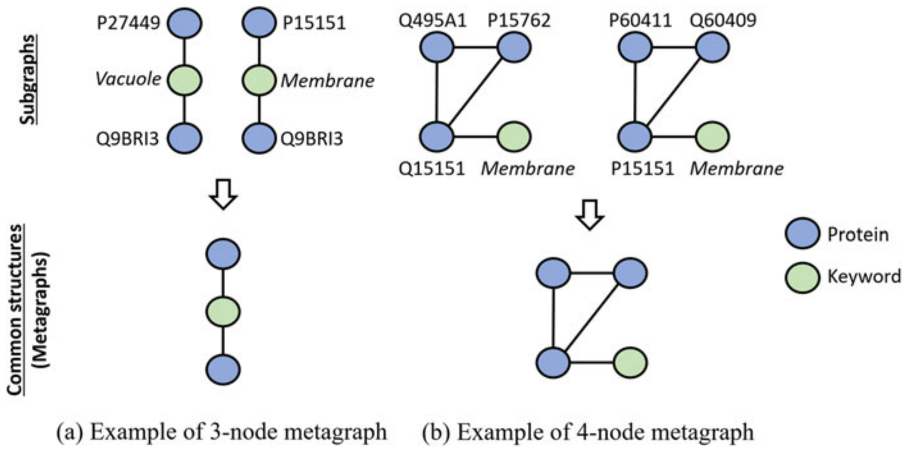


Fig. 3 Example metagraphs: common structures of subgraphs on the PPIK network – (**a**) Example of 3-node metagraph and (**b**) Example of 4-node metagraph

3.1 Metagraph and Instances of a Metagraph

In the PPIK network (Fig. 1), we notice multiple subgraphs with a common structure, which are demonstrated in Fig. 3. More precisely, Fig. 3a shows two three-node subgraphs of the PPIK network, both with a common arrangement "protein-keyword-protein." Similarly, Fig. 3b illustrates two four-node subgraphs with a common arrangement consisting of a triangle of three proteins and one keyword. We call such common structures *metagraphs*, and the corresponding subgraphs are their *instances*, i.e., *metagraph instances*. Formally, a graph $S = (V_S, E_S, l)$ is a subgraph of graph $G = (V, E, l)$ iff $V_S \subseteq V$, $E_S \subseteq E$. A graph $M = (V_M, E_M, l_M)$ is a metagraph for some label function l_M, where each node is defined by its label and its value is immaterial. We say that S is an *instance* of M iff; there exists a bijection w between the nodes of S and M such that

- $\forall v \in V_S, l(v) = l_M(w(v)$, and
- $\forall v, u \in V_S, (u, v) \in E_S$ holds iff $(w(v), w(u)) \in E_M$holds

3.2 Mining the Collection of Metagraphs

From the constructed PPIK network, we extract the collection of metagraphs $\boldsymbol{M}$ using the tool GRAMI [41] (*see* **Note 3**). $\boldsymbol{M} \triangleq \{M_1, M_2, \ldots, M_{|\boldsymbol{M}|}\}$ denotes the set of metagraphs in the PPIK network obtained by GRAMI.

3.3 Metagraph Representations for Proteins

As a baseline, we extract keyword representation vector $\phi(p) = \boldsymbol{k}_{\mathrm{p}}$. Given a set of keywords $\boldsymbol{K}$, let $\boldsymbol{k}_{\mathrm{p}}$ be a vector of length $|\boldsymbol{K}|$, for each protein. It simply represents associated keywords with the protein p where the i-th element is 1 iff the i-th keyword is associated with protein p. We derive two different metagraph representations for proteins from the mined metagraph set $\boldsymbol{M}$ by using SymISO algorithm [31]:

1. *Metagraph*: Let $I(M_{\mathrm{i}})$ be the set of instances of $M_{\mathrm{i}} \in \boldsymbol{M}$. A protein p can be represented by a vector $\boldsymbol{m}_{\mathrm{p}}$ of length $|\boldsymbol{M}|$, where the i-th element is of M_{i} containing the protein p. That is,

$$\boldsymbol{m}_{\mathrm{p}}[i] \triangleq |\{S \in I(M_{\mathrm{i}}) : p \in V_{\mathrm{S}}\}| \qquad (1)$$

   The vector $\boldsymbol{m}_{\mathrm{p}}$ captures the topological arrangement on the PPIK network for interactions between both proteins and keywords. We boost the baseline keyword-based representation with $\boldsymbol{m}_{\mathrm{p}}[i]$ and metagraph vector representation of protein p is $\phi(p) = [\boldsymbol{k}_{\mathrm{p}}, \boldsymbol{m}_{\mathrm{p}}]$.

2. *Metagraph+*: Furthermore, the same metagraph can have multiple subgraph instances with different "utility" levels. For example, a protein taking place in a subgraph together with a disease protein is more likely to be a disease protein, which implies that such subgraph instances have a higher utility toward identifying disease proteins. As shown in Fig. 4, some subgraphs

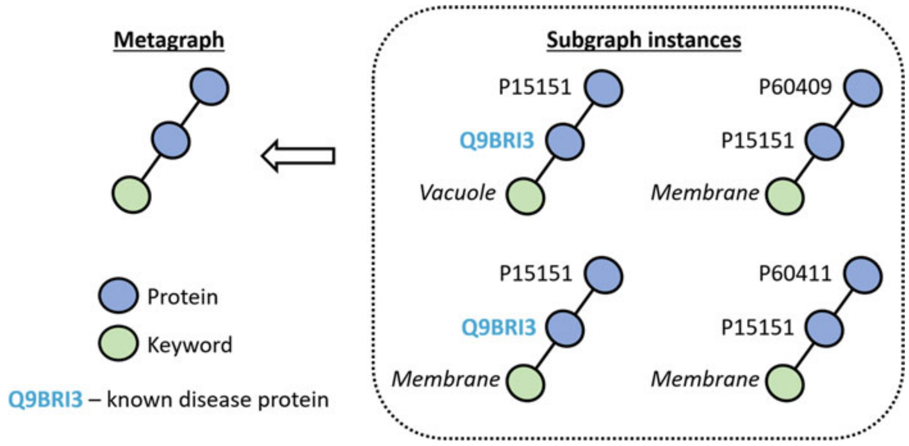


Fig. 4 Example subgraph instances of a metagraph, where some contain disease proteins, and some do not (Q9BRI3 is a known disease protein)

contain disease proteins and some do not, based on biological information from a disease database. To identify such utilities, for each metagraph, we compute the fraction of its subgraph instances containing any of the known disease proteins. The label function $\varphi : P \rightarrow \{\text{disease}, \text{non - disease}\}$ discriminates known disease proteins from other proteins. Let $\boldsymbol{d}_\text{p}$ represent a vector of length $|\boldsymbol{M}|$. We define the i-th element of $\boldsymbol{d}_\text{p}$ as follows:

$$d_\text{p}[i] \triangleq \frac{|\{S \in I(M_\text{i}) : p \in V_\text{S} \wedge (\exists v \in V_\text{S} : v \neq p \wedge \varphi(v) = \text{disease})\}|}{\boldsymbol{m}_\text{p}[i]} \quad (2)$$

Consequently, we construct Metagraph+ vector representation of protein p as, $\phi(p) = [\boldsymbol{m}_\text{p}, \boldsymbol{d}_\text{p}, \boldsymbol{k}_\text{p}]$ with $(2|\boldsymbol{M}| + \boldsymbol{K})$ dimensions.

3.4 Supervised Learning

Based on the mined protein representations and training data, we build a classifier through supervised learning. In our experiments, we adopted three well-known classification models, namely, random forest (RF), support vector machine (SVM), and generalized linear model (GLM) (*see* **Note 4**). For each technique, we divide the datasets into training and testing sets, containing 80% and 20% proteins, respectively. The same set up is distinctly repeated 5 times. Finally, for each dataset, we average over the five splits. To evaluate the effectiveness of the proposed method, we prefer the standard metric of area under the ROC curve (AUC), which is a robust measure of the classifiers' predictive power on imbalanced data (e.g., breast cancer). AUC performance result of the three representations in each of the classification method, on each of the three datasets, for *all diseases* is presented in Fig. 5. Averaging over all classifiers and datasets, Metagraph improves AUC over Keyword by 12.6%. The results denote that interactions/associations on the PPIK network are powerful for

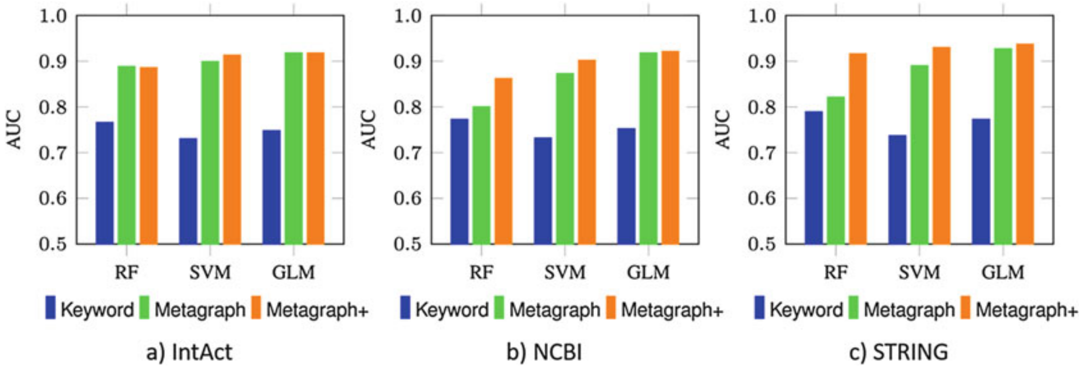


Fig. 5 Performance of Metagraph and Metagraph+ compared to keywords for all disease on (**a**) IntAct, (**b**) NCBI and (**c**) STRING

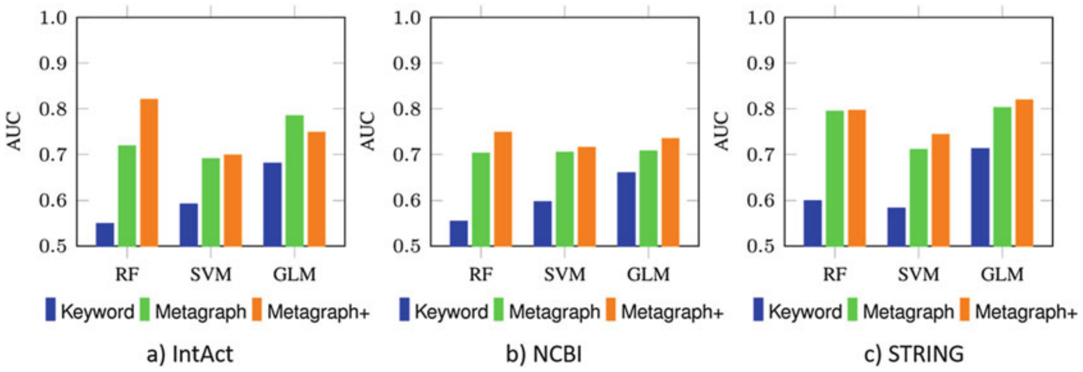


Fig. 6 Performance of Metagraph and Metagraph+ compared to keywords for breast cancer on (**a**) IntAct, (**b**) NCBI and (**c**) STRING

disease prediction, and proteins with similar functional roles tend to exhibit similar topological arrangements. On the contrary, it is inadequate to only consider keywords for individual proteins. Second, the utility-based metagraph representation Metagraph+ further improves the performance.

The performance differences of Keyword, Metagraph, and Metagraph+ for *breast cancer* are similar to those for all diseases, as shown in Fig. 6.

3.5 Comparison to Baselines

We conduct disease protein prediction for two different cases: first, whether a protein is associated with all disease, i.e., all phenotypes in OMIM, and second, whether a protein is particularly associated with breast cancer, i.e., phenotype breast cancer. We compare our proposed work to diffusion-based and module-based approaches for protein disease prediction.

1. *RWR or random walk with restart [42]*: Consider a particle on a PPI network which is initially at one of the known disease proteins. Note that the initial position of the particle has a uniform distribution over the disease proteins in the training data. Next, in each step, the particle makes a move on the network: either hopping to a randomly selected neighbor with $(1 - \alpha)$ probability or returning to one of the disease proteins in the training data with α probability which restarts the random walk. The process is reiterated until it converges to a stationary distribution over all the proteins. In the end, candidate proteins in the test data are ordered according to the stationary distribution. We used RANKS package in *R* for the implementation. The α parameter is tuned over $\{0.1, 0.2, \ldots, 0.9\}$ based on AUC performances.
2. *RWRK*: RWR performed on the PPIK network.
3. *PRINCE [43]*: Since we work on an unweighted PPI network, this method basically performs random walk with restart with only one major difference that is prior probabilities. In

RWR, prior probabilities are uniformly distributed between known disease-associated proteins. On the other hand, in PRINCE prior probabilities are allotted to each disease-associated protein based on a logistic function: $L(x) = \frac{1}{1+e^{(cx+d)}}$ and $x = S(q, p)$ where $S(q, p)$ is the similarity score between query disease q and the associated disease p with the protein. If the protein is associated with more than one disease then, p is chosen to be the one with the closest score to q. We use the suggested values as in paper [43] and set c to -15 and d to *log (9999)*. The similarity scores between phenotypes are obtained from the study [44]. The α parameter is tuned over $\{0.1, 0.2, \ldots, 0.9\}$ based on AUC performances as in RWR.

4. *PRINCEK*: PRINCE performed on the PPIK network.
5. *Subgraph+*: In this baseline, unlike metagraphs, we consider subgraphs with only protein nodes, and their statistics and utilities are formulated as protein representations similar to the case of metagraphs (*see* Eqs. 1 and 2), which are then concatenated with keyword representations for individual proteins. We call this method Subgraph+, analogous to Metagraph+. We leveraged Subgraph+ by different classifiers with the same tuning routine as Metagraph and Metagraph+.

We compare the AUC performances of the baseline methods with proposed Metagraph+, for *all diseases* and *breast cancer* cases in Table 3. The utility-based metagraph representation, Metagraph+, outperforms the baselines including the diffusion-based methods (e.g., RWR) and the module-based methods by 13.8–32.9% for overall disease protein prediction. For predicting breast cancer genes, it outperforms RWR, PRINCE, and the module-based baselines by 6.6–14.2%.

3.6 Further Analysis of the Predicted Disease Proteins

We further examine the disease proteins predicted by our proposed methods, Metagraph and Metagraph+. Since GLM generally has the best performance among the three classifiers, we only analyze the results from this classifier. A test protein is considered as a predicted disease protein if its prediction score is higher than 0.5. Firstly, for each proposed method, we ran disease classification on all three datasets (IntAct, NCBI, and STRING). Then, we merge the predictions for all three datasets to construct a predicted disease gene set for each method. We map our predicted disease proteins to their producer gene Id's using UniProt database. To enable the analysis, DisGeNET database [45] is used to search the PubMed Ids of the up-to-date publications reporting the gene-disease associations. We demonstrate the average number of existing publications per prediction that support our disease gene predictions based on the proposed methods in Fig. 7. The results are consistent with the AUC performance reported earlier, where methods with higher average publications achieve higher AUC scores.

Table 3
Performance of Metagraph+ compared to random walk and subgraph baselines

	All disease			Breast cancer		
	IntAct	**NCBI**	**STRING**	**IntAct**	**NCBI**	**STRING**
RWR	0.551	0.567	0.622	0.578	0.665	0.605
RWRK	0.590	0.587	0.629	0.587	0.664	0.612
PRINCE	–	–	–	0.506	0.716	0.632
PRINCEK	–	–	–	0.634	0.717	0.596
Classifier: RF						
Subgraph+	0.745	0.756	0.826	0.611	0.696	0.713
Metagraph+	**0.886**	**0.862**	**0.916**	**0.820**	**0.748**	**0.796**
Classifier: SVM						
Subgraph+	0.739	0.741	0.808	0.687	0.682	0.597
Metagraph+	**0.913**	**0.902**	**0.930**	**0.698**	0.715	**0.743**
Classifier: GLM						
Subgraph+	0.751	0.758	0.818	0.733	0.715	0.796
Metagraph+	**0.918**	**0.921**	**0.937**	**0.748**	**0.734**	**0.819**

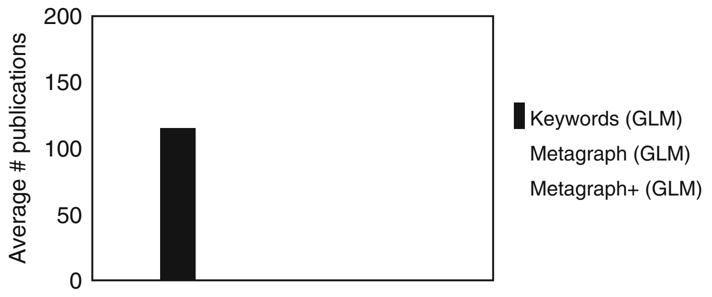


Fig. 7 Average number of PubMed publications per prediction based on DisGeNET

4 Notes

1. In this chapter, we focus on disease gene prediction. Since the proteins are products of genes and we are using the protein-protein interaction networks, disease genes and disease proteins are used interchangeably.
2. We only use the exact keywords, without evaluating the semantic similarity or overlap between them. To verify the use of exact keywords, we investigated the semantic similarity

between pairs of keywords in UniProt database. There are 768 keywords associated with a Gene Ontology (GO) term, and 36 of them are associated with more than one GO terms. We performed GO semantic analysis with GOSemSim package in R to compare the similarity between these 768 GO terms. The results show us only 326 pairs out of 294,528 (0.1%) pairs have semantic similarity score greater than 0.5. Based on the statistics, using exact match for the keywords is adequate and reasonable.

3. Mining the metagraphs is an active research topic and various solutions exist. We apply an existing state-of-the-art approach GRAMI for this step. And only consider metagraphs up to five nodes, which provide a good balance between efficiency and accuracy (i.e., the number of instances grows exponentially).
4. Note that the main purpose of our study is to propose the metagraph representations for the PPIK network, which aims to improve the performance of the disease-causing protein prediction across various supervised learning techniques including random forest, SVM, and generalized linear models. For RF, we employed *random forest* package in R and tuned the mtry parameter between 1 and the cardinality of protein representation with *tune RF* function based on OOB error. For SVM, we employed *e1071* package in R and tuned the gamma parameter over $\{10^{-5}, 10^{-4}, 0.001, 0.01, 0.1\}$ and the cost parameter over $\{0.1, 1, 10\}$ with grid-based tune.svm function (based on classification error). For GLM, we employed *stats* package in R and adopted default Gaussian distribution. Furthermore, in the case of *breast cancer*, the classes are highly unbalanced. Therefore, we used the *ROSE* package in R to oversample the breast cancer class (minority class) with probability 0.2.

References

1. Nelson MR, Tipney H, Painter JL, Shen J, Nicoletti P, Shen Y, Floratos A, Sham PC, Li MJ, Wang J, Cardon LR, Whittaker JC, Sanseau P (2015) The support of human genetic evidence for approved drug indications. Nat Genet 47(8):856–860
2. Sekar A, Bialas AR, de Rivera H, Davis A, Hammond TR, Kamitaki N, Tooley K, Presumey J, Baum M, Van Doren V, Genovese G, Rose SA, Handsaker RE, Consortium SWGotPG, Daly MJ, Carroll MC, Stevens B, McCarroll SA (2016) Schizophrenia risk from complex variation of complement component 4. Nature 530(7589):177–183
3. Yang P, Li X, Chua H-N, Kwoh C-K, Ng S-K (2014) Ensemble positive unlabeled learning for disease gene identification. PLoS One 9(5):1–11
4. Yang P, Li X-L, Mei J-P, Kwoh C-K, Ng S-K (2012) Positive-unlabeled learning for disease gene identification. Bioinformatics 28(20):2640
5. Li M, Lu Y, Wang J, Wu F-X, Pan Y (2015) A topology potential-based method for identifying essential proteins from PPI networks. IEEE/ACM Trans Comput Biol Bioinform 12(2):372–383
6. Fu L, Zhang S, Zhang L, Tong X, Zhang J, Zhang Y, Ouyang L, Liu B, Huang J (2015)

Systems biology network-based discovery of a small molecule activator BL-AD008 targeting AMPK/ZIPK and inducing apoptosis in cervical cancer. Oncotarget 6(10):8071–8088
7. Gui T, Dong X, Li R, Li Y, Wang Z (2015) Identification of hepatocellular carcinoma-related genes with a machine learning and network analysis. J Comput Biol 22(1):63–71
8. Li X-L, Ng S-K (2009) Biological data mining in protein interaction networks. IGI Global, Hershey, PA
9. Chuang H-Y, Lee E, Liu Y-T, Lee D, Ideker T (2007) Network-based classification of breast cancer metastasis. Mol Syst Biol 3(1):140–n/a
10. Ideker T, Sharan R (2008) Protein networks in disease. Genome Res 18(4):644–652
11. Xu J, Li Y (2006) Discovering disease-genes by topological features in human protein–protein interaction network. Bioinformatics 22(22):2800–2805
12. Yang P, Li X, Wu M, Kwoh C-K, Ng S-K (2011) Inferring gene-phenotype associations via global protein complex network propagation. PLoS One 6(7):1–11
13. Lage K, Karlberg EO, Storling ZM, Olason PI, Pedersen AG, Rigina O, Hinsby AM, Tumer Z, Pociot F, Tommerup N, Moreau Y, Brunak S (2007) A human phenome-interactome network of protein complexes implicated in genetic disorders. Nat Biotech 25(3):309–316
14. Barabási A-L, Gulbahce N, Loscalzo J (2011) Network medicine: a network-based approach to human disease. Nat Rev Genet 12(1):56–68
15. Krauthammer M, Kaufmann CA, Gilliam TC, Rzhetsky A (2004) Molecular triangulation: bridging linkage and molecular-network information for identifying candidate genes in Alzheimer's disease. Proc Natl Acad Sci U S A 101(42):15148–15153
16. Iossifov I, Zheng T, Baron M, Gilliam TC, Rzhetsky A (2008) Genetic-linkage mapping of complex hereditary disorders to a whole-genome molecular-interaction network. Genome Res 18(7):1150–1162
17. Oti M, Snel B, Huynen MA, Brunner HG (2006) Predicting disease genes using protein-protein interactions. J Med Genet 43(8):691–698
18. Navlakha S, Kingsford C (2010) The power of protein interaction networks for associating genes with diseases. Bioinformatics 26(8):1057
19. Suthram S, Dudley JT, Chiang AP, Chen R, Hastie TJ, Butte AJ (2010) Network-based elucidation of human disease similarities reveals common functional modules enriched for pluripotent drug targets. PLoS Comput Biol 6(2):1–10
20. Wu G, Stein L (2012) A network module-based method for identifying cancer prognostic signatures. Genome Biol 13(12):R112
21. Zhu J, Qin Y, Liu T, Wang J, Zheng X (2013) Prioritization of candidate disease genes by topological similarity between disease and protein diffusion profiles. BMC Bioinformatics 14(5):S5
22. Shim JE, Hwang S, Lee I (2015) Pathway-dependent effectiveness of network algorithms for gene prioritization. PLoS One 10(6):1–10
23. Zhu L, Deng S-P, Huang D-S (2015) A two-stage geometric method for pruning unreliable links in protein-protein networks. IEEE Trans Nanobioscience 14(5):528–534
24. Marcatili P, Tramontano A (2009) Network cleansing: reliable interaction networks. In: Biological data mining in protein interaction networks. IGI Global, Hershey, PA, pp 80–97
25. Consortium U et al (2015) UniProt: a hub for protein information. Nucleic Acids Res 43(Database issue):D204–D212
26. Liu W, Wu A, Pellegrini M, Wang X (2015) Integrative analysis of human protein, function and disease networks. Sci Rep 5:14344 EP
27. Singh-Blom UM, Natarajan N, Tewari A, Woods JO, Dhillon IS, Marcotte EM (2013) Prediction and validation of gene-disease associations using methods inspired by social network analyses. PLoS One 8(5):1–17
28. Peng W, Wang J, Cai J, Chen L, Li M, Wu F-X (2014) Improving protein function prediction using domain and protein complexes in PPI networks. BMC Syst Biol 8:35–35
29. Yang ZH, Yu FY, Lin HF, Wang J (2014) Integrating PPI datasets with the PPI data from biomedical literature for protein complex detection. BMC Med Genet 7(Suppl 2):S3–S3
30. Sun K, Gonçalves JP, Larminie C, Pržulj N (2014) Predicting disease associations via biological network analysis. BMC Bioinformatics 15(1):304
31. Fang Y, Lin W, Zheng VW, Wu M, Chang KC-C, Li X (2016) Semantic proximity search on graphs with metagraph-based learning. In: 32nd {IEEE} International Conference on Data Engineering, {ICDE} 2016, Helsinki, Finland, May 16–20, 2016. pp 277–288
32. Orchard S, Ammari M, Aranda B, Breuza L, Briganti L, Broackes-Carter F, Campbell NH, Chavali G, Chen C, del Toro N, Duesbury M,

Dumousseau M, Galeota E, Hinz U, Iannuccelli M, Jagannathan S, Jimenez R, Khadake J, Lagreid A, Licata L, Lovering RC, Meldal B, Melidoni AN, Milagros M, Peluso D, Perfetto L, Porras P, Raghunath A, Ricard-Blum S, Roechert B, Stutz A, Tognolli M, van Roey K, Cesareni G, Hermjakob H (2014) The MIntAct project IntAct as a common curation platform for 11 molecular interaction databases. Nucleic Acids Res 42(Database issue):D358–D363

33. Szklarczyk D, Franceschini A, Wyder S, Forslund K, Heller D, Huerta-Cepas J, Simonovic M, Roth A, Santos A, Tsafou KP, Kuhn M, Bork P, Jensen LJ, vonÂ Mering C (2015) STRING v10: protein-protein interaction networks, integrated over the tree of life. Nucleic Acids Res 43(D1):D447
34. Maglott D, Ostell J, Pruitt KD, Tatusova T (2007) Entrez Gene: gene-centered information at NCBI. Nucleic Acids Res 35(Suppl 1):D26
35. De Las Rivas J, Fontanillo C (2010) Protein-protein interactions essentials: key concepts to building and analyzing interactome networks. PLoS Comput Biol 6(6):e1000807
36. Chua HN, Sung W-K, Wong L (2006) Exploiting indirect neighbours and topological weight to predict protein function from protein-protein interactions. Bioinformatics 22(13):1623–1630. https://doi.org/10.1093/bioinformatics/btl145
37. Wu M, Yu Q, Li X-L, Zheng J, Huang J-F, Kwoh C-K (2013) Benchmarking human protein complexes to investigate drug-related systems and evaluate predicted protein complexes. PLoS One 8(2):e53197
38. Li X-L, Wu M, Kwoh C-K, Ng S-K (2010) Computational approaches for detecting protein complexes from protein interaction networks: a survey. BMC Genomics 11(Suppl 1):S3
39. Wu M, Li X-L, Kwoh C-K, Ng S-K (2009) A core-attachment based method to detect protein complexes in PPI networks. BMC Bioinformatics 10:169
40. Hamosh A, Scott AF, Amberger J, Bocchini C, Valle D, McKusick VA (2002) Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. Nucleic Acids Res 30(1):52
41. Elseidy M, Abdelhamid E, Skiadopoulos S, Kalnis P (2014) GraMi: frequent subgraph and pattern mining in a single large graph. Proc VLDB Endow 7(7):517–528
42. Köhler S, Bauer S, Horn D, Robinson PN (2008) Walking the interactome for prioritization of candidate disease genes. Am J Hum Genet 82(4):949–958
43. Vanunu O, Magger O, Ruppin E, Shlomi T, Sharan R (2010) Associating genes and protein complexes with disease via network propagation. PLoS Comput Biol 6(1):1–9
44. van Driel MA, Bruggeman J, Vriend G, Brunner HG, Leunissen JAM (2006) A text-mining analysis of the human phenome. Eur J Hum Genet 14(5):535–542
45. Piñero J, Bravo À, Queralt-Rosinach N, Gutiérrez-Sacristán A, Deu-Pons J, Centeno E, García-García J, Sanz F, Furlong LI (2017) DisGeNET: a comprehensive platform integrating information on human disease-associated genes and variants. Nucleic Acids Res 45(D1):D833

Chapter 17

Inferring Antimicrobial Resistance from Pathogen Genomes in KEGG

Minoru Kanehisa

Abstract

The KEGG database is widely used as a reference knowledge base for biological interpretation of genome sequences and other high-throughput data. It contains, among others, KEGG pathway maps and BRITE hierarchies (ontologies) representing high-level systemic functions of the cell and the organism. By the processes called pathway mapping and BRITE mapping, information encoded in the genome, especially the repertoire of genes, is converted to such high-level functional information. This general methodology can be applied to microbial genomes to infer antimicrobial resistance (AMR), which is becoming an increasingly serious threat to the global public health. Here we present how knowledge on AMR is accumulated in the KEGG Pathogen resource and how such knowledge can be utilized by BlastKOALA and other web tools.

Key words Beta-lactamase, KEGG Orthology (KO), KEGG module, Genome annotation, BlastKOALA

1 Introduction

Antimicrobial resistance (AMR) is the ability of microbes to become resistant to antimicrobial drugs. Since the availability of first antibiotics in the 1940s, AMR has been a recurring problem every time a new drug is introduced. This reflects intrinsic capacity of bacteria to adapt to environmental changes such as by mutating genomic sequences and exchanging mobile genetic elements. Furthermore, since antibiotics originate from natural products, AMR may be viewed as a natural defense system in the ecosystem. Although it may never be possible to overcome the universal problem of AMR, characterizing such genomic and genetic features will help understand molecular mechanisms of AMR and better cope with this problem.

There are already several databases for AMR gene variants, such as NCBI antimicrobial resistance reference gene database [1], CARD (comprehensive antibiotic resistance database) [2] and

BLDB (beta-lactamase database) [3], as well as Lahey Clinic beta-lactamase nomenclature database [4], which is discontinued and transferred to NCBI. We have also been developing a resource for AMR in the KEGG (Kyoto Encyclopedia of Genes and Genomes) database [5, 6].

KEGG (www.kegg.jp) is an integrated database resource consisting of 18 main databases categorized into systems information (PATHWAY, BRITE, and MODULE), genomic information (KO, GENOME, GENES, and SSDB), chemical information (COMPOUND, GLYCAN, REACTION, RCLASS, and ENZYME), and health information (NETWORK, VARIANT, DISEASE, DRUG, DGROUP, and ENVIRON). In addition to data-oriented entry points to these databases, KEGG presents subject-oriented entry points helping users of specific subject domains to navigate through different databases. KEGG Pathogen is one of them, integrating pathogen genomes, infectious diseases, and anti-infective drugs. This paper describes datasets and tools available in the KEGG Pathogen resource.

2 Materials

2.1 KEGG Pathogen Resource

KEGG Pathogen (www.kegg.jp/kegg/genome/pathogen.html) is an interface to pathogen genomes and infectious diseases in KEGG for understanding molecular mechanisms of pathogenicity and antimicrobial resistance (AMR) from genomic information. As of October 2017, it consists of 25 pathway maps for infectious diseases and 3 pathway maps for AMR in the PATHWAY database, more than 300 infectious disease entries in the DISEASE database, more than 1000 anti-infective drugs in the DRUG database, and more than 1000 pathogen genomes in the GENOME and GENES databases. All these contents are classified by BRITE hierarchy or table files as shown in Table 1. As an example, Fig. 1 is the pathway map for beta-lactam resistance (map 01501), which indicates four main mechanisms for AMR: (i) altered target sites for penicillin-binding proteins, (ii) enzymatic inactivation by mutated beta-lactamases, (iii) decreased penetration by repression of porins, and (iv) increased efflux by overexpression of efflux pumps.

The GENES database is a collection of gene catalogs for completely sequenced genomes taken from the RefSeq and GenBank databases. They are given KEGG-original annotations in the form of assigning KO (KEGG Orthology) identifiers, also called K numbers, which represent gene/protein functional orthologs. The KO database accumulates knowledge on molecular-level functions in a generic way, namely, in terms of orthologs rather than individual genes from specific organisms. In a similar way, the PATHWAY, BRITE, and MODULE databases accumulate knowledge on higher-level functions represented as generic (not

Table 1
KEGG Pathogen resource

Database	Classification[a]	Content
PATHWAY	Infectious diseases: Bacterial	Disease pathways
	Infectious diseases: Viral	
	Infectious diseases: Parasitic	
	Drug resistance: Antimicrobial (all in br08901)	
	Chronology: Anti-infectives (br08901)	Drug structure maps
DISEASE	Infectious diseases (br08401)	Disease entries
	Human diseases in ICD-10 classification (br08403)	
DRUG	Antibacterials (br08350)	Drug entries
	Antivirals (br08351)	
	Antifungals (br08352)	
	Antiparasitics (br08353)	
GENOME	Human pathogens (br08601_key)	Microbes
GENES	Bacterial toxins (ko02042)	Genes
	Antimicrobial resistance genes (ko01504)	
	Beta-lactamases (br01553)	
	Aminoglycoside resistance genes (br01554)	
	Tetracycline resistance genes (br01556)	
	Macrolide resistance genes (br01555)	
	Other resistance genes (br01557)	
KO	KEGG signatures (br01600)	KOs
MODULE	Signature module: Pathogenicity	Gene sets
	Signature module: Drug resistance (both in ko00002)	

[a]BRITE hierarchy or table file identifiers are shown in parentheses

organism-specific) networks of molecular interactions, reactions, and relations, where the nodes of all these networks are identified by K numbers. Thus, once genes in the genome are annotated with K numbers, they can automatically be matched with the network nodes of KEGG pathway maps, BRITE functional hierarchies, and KEGG modules, enabling KEGG mapping for interpretation of higher-level functions. In the KEGG Pathogen resource, this general procedure is applied to the specific problem of inferring AMR from pathogen genome sequences by developing a specialized knowledge base consisting of AMR gene sequences, signature KOs, and signature modules as described below.

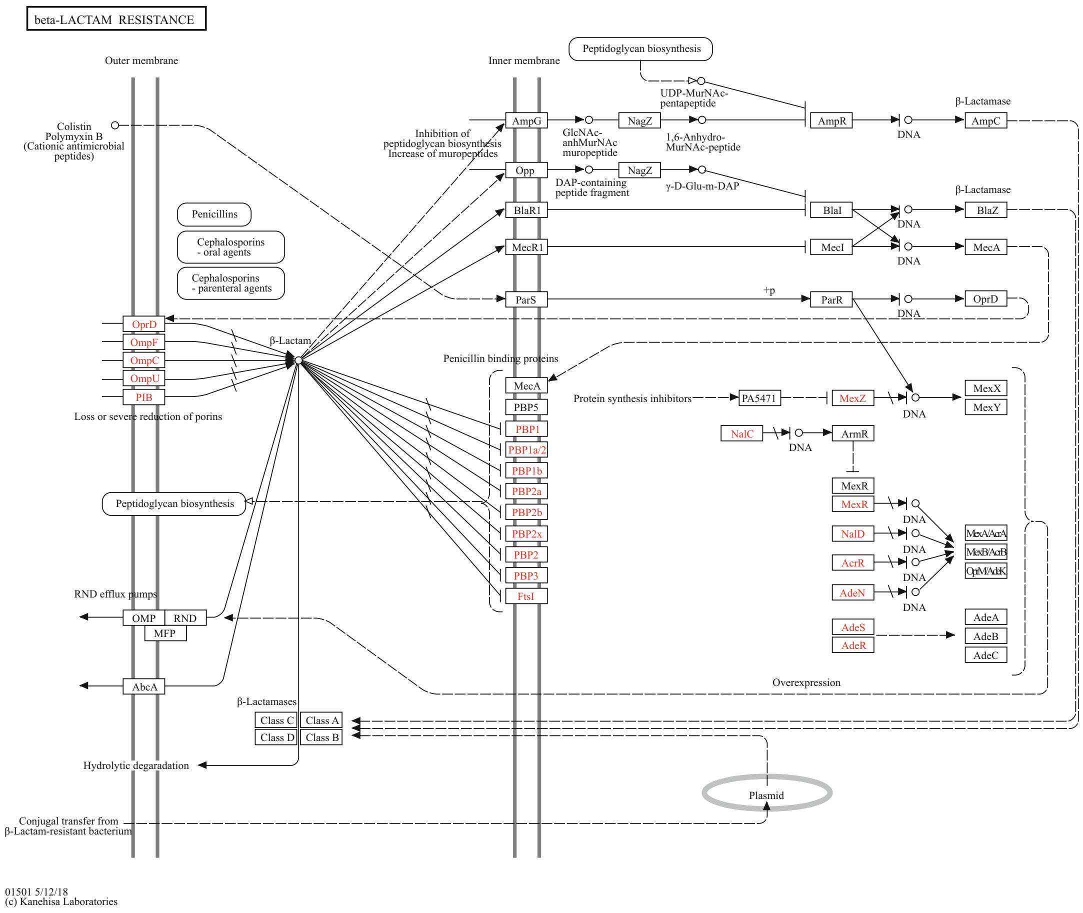

beta-LACTAM RESISTANCE
Outer membrane
Inner membrane
Peptidoglycan biosynthesis
Colistin Polymyxin B (Cationic antimicrobial peptides)
Penicillins
Cephalosporins - oral agents
Cephalosporins - parenteral agents
OprD
OmpF
OmpC
OmpU
PIB
Loss or severe reduction of porins
β-Lactam
AmpG
Opp
BlaR1
MecR1
ParS
NagZ
AmpR
AmpC
BlaI
BlaZ
MecI
MecA
ParR
OprD
β-Lactamase
DNA
UDP-MurNAc-pentapeptide
GlcNAc-anhMurNAc muropeptide
1,6-Anhydro-MurNAc-peptide
DAP-containing peptide fragment
γ-D-Glu-m-DAP
Inhibition of peptidoglycan biosynthesis Increase of muropeptides
Penicillin binding proteins
PBP5
PBP1
PBP1a/2
PBP1b
PBP2a
PBP2b
PBP2x
PBP2
PBP3
FtsI
Protein synthesis inhibitors
PA5471
MexZ
MexX
MexY
NalC
ArmR
MexR
NalD
AcrR
AdeN
MexA/AcrA
MexB/AcrB
OprM/AdeK
AdeS
AdeR
AdeA
AdeB
AdeC
Overexpression
RND efflux pumps
OMP
RND
MFP
AbcA
β-Lactamases
Class C
Class A
Class D
Class B
Hydrolytic degaradation
Plasmid
Conjugal transfer from β-Lactam-resistant bacterium
01501 5/12/18
(c) Kanehisa Laboratories

Fig. 1 The KEGG pathway map for beta-lactam resistance (map 01501)

2.2 AMR Gene Sequences

In the past, the content of the GENES database was limited to those genes present in completely sequenced genomes. Consequently, the content of functional orthologs in the KO database was also limited. In order to make the KO database more comprehensive, so that the KO-based annotation becomes more comprehensive, the addendum category has been introduced in the GENES database [6]. It is a collection of individual protein sequences of known functions based on published literature, from which new KOs are defined. This non-genome category of GENES is essential to organize knowledge on AMR gene variants, currently for the following classes of antibiotics: beta-lactam, aminoglycoside, tetracycline, macrolide, phenicol, sulfonamide, trimethoprim, quinolone, rifamycin, and fosfomycin.

Among them beta-lactam is the major class of antibiotics, which is classified by the drug groups (DG numbers) of the KEGG DGROUP database as shown in Table 2. Microorganisms that are resistant to newer generation of beta-lactam antibiotics are the main concern in global public health. They include carbapenem-resistant *Enterobacteriaceae*, extended-spectrum beta-lactamase (ESBL) producing *Enterobacteriaceae*, carbapenem-resistant *Acinetobacter*, and carbapenem-resistant *Pseudomonas aeruginosa* according to the threat levels of microorganisms defined by CDC (Centers for Disease Control and Prevention in the USA) and the priority pathogen list defined by WHO (World Health Organization).

The major mechanism of AMR for beta-lactam antibiotics involves the enzyme, beta-lactamase, that hydrolyzes the beta-lactam ring and inactivates the drug. Microorganisms, especially Gram-negative bacteria, have the ability to readily change substrate specificity of this enzyme by mutated amino acid sequences. Historically, beta-lactamases were first classified into class A and class B for serine proteases and metalloproteases, respectively, and class C and class D were later added to the serine protease group [7]. In addition, enzyme family names have been given to closely related enzyme sequences with numbering indicating instances of variant sequences [4]. The number of beta-lactamase sequences collected in KEGG (as of October 2017) is shown in Table 3.

2.3 Signature KOs

The KO group represents a functional ortholog at the molecular level, but the term function can be defined in varying degrees of granularity. KOs are usually defined in the context of molecular networks, namely, as nodes of KEGG pathway maps, BRITE hierarchies, and KEGG modules. This definition has turned out to be extremely useful to represent conserved functions among different organisms, enabling reconstruction of molecular networks from sequenced genomes. For AMR genes, meaningful functions must be defined so that substrate specificity and other molecular details can be distinguished, and the drug groups that organisms are resistant to can be distinguished.

Thus, together with the addendum category of the GENES database, we introduced the procedure to define finely classified KOs, called tight KOs, in the KO database [6]. Currently, close to 50% of about 25 million genes in the GENES database are given KO annotation, which is done by both automatic and manual modes of the KOALA (KEGG Orthology And Links Annotation) system using the computationally generated sequence similarity database SSDB. Tight KOs are manually defined using the GFIT tool, which shows for a given gene of a given organism candidates of orthologs in other organisms in the order of sequence similarity scores. Tight KOs tend to appear as a group of often over 90% sequence identity followed by less similar sequences with a wide margin.

We also use the term signature KOs, for those KOs that can be directly linked to phenotypes or other high-level features. Tight KOs defined for beta-lactamases are all signature KOs

Table 2
Beta-lactam antibiotics

Drug group			Drug example	Enzyme example
Beta-lactam (DG01710)	Penicillin skeleton group (DG01713)	Beta-lactamase-sensitive penicillin (DG01778)	Benzylpenicillin	Penicillinase
		Beta-lactamase-resistant penicillin (DG01779)	Meticillin	Oxacillinase
			Oxacillin	
		Extended-spectrum penicillin (DG01780)	Ampicillin	
			Carbenicillin	
		Beta-lactamase inhibitor (DG01479)	Sulbactam	
			Tazobactam	
			Clavulanic acid	
		Carbapenem (DG01458)	Imipenem	Carbapenemase Imipenemase
	Cephalosporin skeleton group (DG01714)	First-generation cephalosporin (DG01774)	Cefaloridine	Cephalosporinase
		Second-generation cephalosporin (DG01775)	Cefoxitin	Cephamycinase
			Flomoxef	
			Loracarbef	
		Third-generation cephalosporin (DG01776)	Cefotaxime	Extended-spectrum
				beta-lactamase (ESBL)
			Ceftazidime	
			Latamoxef	
		Fourth-generation cephalosporin (DG01777)	Cefepime	
	Monobactam (DG01454)	Monobactam (DG01454)	Aztreonam	

Table 3
Collection of beta-lactamase sequences

Class		Name	Number of sequences	Number of KOs
Serine	A	TEM	192	1
		SHV	156	1
		CTX-M	165	1
		PER	8	1
		VEB	15	1
		BEL	3	1
		KPC	22	1
		GES	30	1
		IMI	8	1
		SME	5	
		CARB	40	3
	D	OXA	426	23
	C	CMY	129	2
		CFE	1	
		LAT	1	
		MOX	11	
		DHA	21	1
		FOX	12	1
		ACC	5	1
		ACT	37	1
		MIR	18	1
		ADC	37	1
		PDC	10	1
Metallo	B	IMP	50	1
		VIM	43	1
		NDM	16	1
		IND	16	1
		GIM	2	1
Total			1479	48

Table 4
Signature KOs of carbapenem-hydrolyzing beta-lactamases

Class	Name	KO	Organism
A	KPC	K18768	*Enterobacteriaceae*, *Pseudomonas*, *Acinetobacter*
	GES	K18970	*Enterobacteriaceae*, *Pseudomonas*, *Acinetobacter*, *Aeromonas*
	IMI	K19316	*Enterobacteriaceae*
	SME		
D	OXA-51	K18794	*Acinetobacter*
	OXA-213	K19318	*Acinetobacter*
	OXA-24	K18971	*Acinetobacter*
	OXA-23	K18793	*Klebsiella*, *Acinetobacter*
	OXA-134	K19319	*Acinetobacter*
	OXA-211	K19320	*Acinetobacter*
	OXA-214	K19321	*Acinetobacter*
	OXA-229	K19322	*Acinetobacter*
	OXA-58	K18972	*Acinetobacter*
	OXA-62	K19211	*Pandoraea*
	OXA-48	K18976	*Enterobacteriaceae*, *Shewanella*
B	IMP	K18782	*Enterobacteriaceae*, *Pseudomonas*, *Acinetobacter*, *Aeromonas*
	VIM	K18781	*Enterobacteriaceae*, *Pseudomonas*
	NDM	K18780	*Enterobacteriaceae*, *Acinetobacter*
	IND	K19099	Enterobacter, *Pseudomonas*
	GIM	K19216	*Chryseobacterium*

characterizing enzyme groups and in many cases AMR drug groups as well. The number of signature KOs is shown in the last column of Table 3 indicating that they well correspond to enzyme families except OXA. In contrast to the other names based on sequence similarity, OXA is used based on the activity on oxacillin and related substrates and consists of a diverse set of sequence data [8, 9]. Table 4 shows carbapenem-hydrolyzing beta-lactamases (carbapenemases) including certain OXA subgroups. OXA-type carbapenemases are found mostly in *Acinetobacter*, but OXA-48 is found in *Enterobacteriaceae*.

2.4 Signature Modules

KEGG modules in the MODULE database are functional units representing gene sets of functional relevance. Each module is identified by the M number and defined by the set of K numbers or more precisely by the logical expression of K numbers. For example, a complex is defined using the AND (+) operator, and alternative genes are defined using the OR (,) operator. This allows automatic evaluation of whether the gene set for the functional unit is complete in a given genome. It is often the case that the functional units correspond to genomic units, such as operon-like structures and plasmid-encoded gene sets. Several types of KEGG modules are considered, including pathway modules representing locally well-conserved units in metabolic pathways and signature modules that can be used as markers of phenotypes.

We have been collecting signature modules for linking microbial genomes to phenotypes, especially pathogenicity and AMR. For example, the module M00363 is a signature module for identifying pathogenic strains of EHEC (enterohemorrhagic *Escherichia coli*) infection. It consists of Shiga toxin A and B subunit genes. The module M00625 is a signature module for methicillin resistance identifying MRSA (methicillin-resistant *Staphylococcus aureus*) strains. Beta-lactam resistance in this case is caused by altered target sites for penicillin-binding proteins. In addition to mecA for resistant penicillin-binding protein, this module contains mecR1for signal transducer protein and mecI for repressor protein. They may be encoded in the mobile genetic element SCCmec (staphylococcal cassette chromosome mec). Figure 2 is the ortholog table for M00625 showing KEGG organisms that possess all three genes (complete module) or mecA/mecR1 genes (one block missing module) in the genome, which are known MRSA strains.

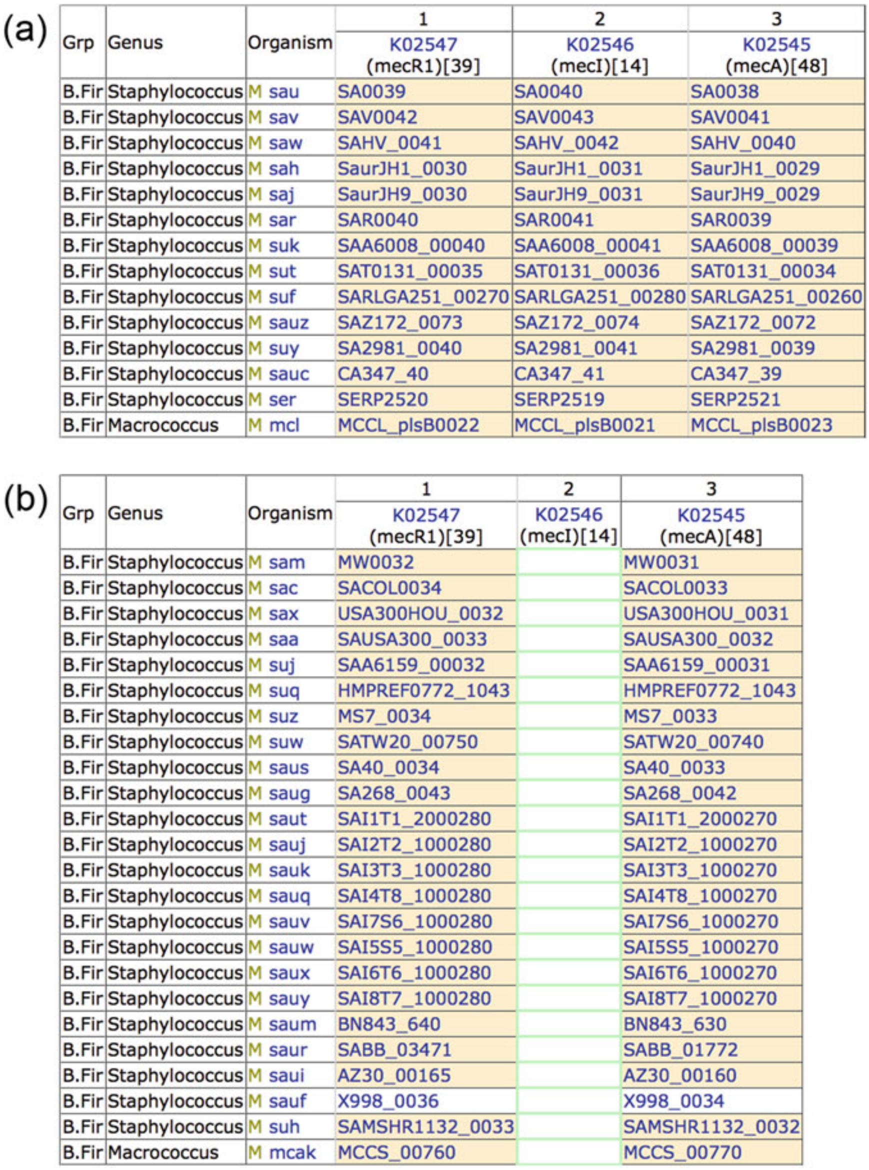
Table 5 summarizes the number of signature KOs and signature modules currently defined in KEGG Pathogen. These modules include the other types of resistance mechanisms, such as activation of efflux pumps and repression of porins.

3 Methods

3.1 Phylogenetic Tree

The KO assignment of protein-coding genes is based on comparisons of entire amino acid sequences as stored in the SSDB database, which is different from domain-based (HMM-based) assignments used in most other resources. KO grouping is more closely related to phylogenetic trees generated from multiple sequence alignments. This can be seen in the KEGG SeqData page (www.kegg.jp/kegg/seq/) where phylogenetic trees are shown for KEGG sequence data collections, including beta-lactamases (*see* **Note 1**). All the sequence data in classes A, B, and C are assigned well-defined KOs that correspond to branches of phylogenetic trees, but OXA-type sequence data in class D are not fully divided yet into KOs because of the diversity of sequence data.

KEGG Mapper (www.kegg.jp/kegg/mapper.html) is a collection of KEGG mapping tools against PATHWAY, BRITE, and



(a)

Grp	Genus	Organism	1	2	3
			K02547 (mecR1)[39]	K02546 (mecI)[14]	K02545 (mecA)[48]
B.Fir	Staphylococcus	M sau	SA0039	SA0040	SA0038
B.Fir	Staphylococcus	M sav	SAV0042	SAV0043	SAV0041
B.Fir	Staphylococcus	M saw	SAHV_0041	SAHV_0042	SAHV_0040
B.Fir	Staphylococcus	M sah	SaurJH1_0030	SaurJH1_0031	SaurJH1_0029
B.Fir	Staphylococcus	M saj	SaurJH9_0030	SaurJH9_0031	SaurJH9_0029
B.Fir	Staphylococcus	M sar	SAR0040	SAR0041	SAR0039
B.Fir	Staphylococcus	M suk	SAA6008_00040	SAA6008_00041	SAA6008_00039
B.Fir	Staphylococcus	M sut	SAT0131_00035	SAT0131_00036	SAT0131_00034
B.Fir	Staphylococcus	M suf	SARLGA251_00270	SARLGA251_00280	SARLGA251_00260
B.Fir	Staphylococcus	M sauz	SAZ172_0073	SAZ172_0074	SAZ172_0072
B.Fir	Staphylococcus	M suy	SA2981_0040	SA2981_0041	SA2981_0039
B.Fir	Staphylococcus	M sauc	CA347_40	CA347_41	CA347_39
B.Fir	Staphylococcus	M ser	SERP2520	SERP2519	SERP2521
B.Fir	Macrococcus	M mcl	MCCL_plsB0022	MCCL_plsB0021	MCCL_plsB0023

(b)

Grp	Genus	Organism	1	2	3
			K02547 (mecR1)[39]	K02546 (mecI)[14]	K02545 (mecA)[48]
B.Fir	Staphylococcus	M sam	MW0032		MW0031
B.Fir	Staphylococcus	M sac	SACOL0034		SACOL0033
B.Fir	Staphylococcus	M sax	USA300HOU_0032		USA300HOU_0031
B.Fir	Staphylococcus	M saa	SAUSA300_0033		SAUSA300_0032
B.Fir	Staphylococcus	M suj	SAA6159_00032		SAA6159_00031
B.Fir	Staphylococcus	M suq	HMPREF0772_1043		HMPREF0772_1043
B.Fir	Staphylococcus	M suz	MS7_0034		MS7_0033
B.Fir	Staphylococcus	M suw	SATW20_00750		SATW20_00740
B.Fir	Staphylococcus	M saus	SA40_0034		SA40_0033
B.Fir	Staphylococcus	M saug	SA268_0043		SA268_0042
B.Fir	Staphylococcus	M saut	SAI1T1_2000280		SAI1T1_2000270
B.Fir	Staphylococcus	M sauj	SAI2T2_1000280		SAI2T2_1000270
B.Fir	Staphylococcus	M sauk	SAI3T3_1000280		SAI3T3_1000270
B.Fir	Staphylococcus	M sauq	SAI4T8_1000280		SAI4T8_1000270
B.Fir	Staphylococcus	M sauv	SAI7S6_1000280		SAI7S6_1000270
B.Fir	Staphylococcus	M sauw	SAI5S5_1000280		SAI5S5_1000270
B.Fir	Staphylococcus	M saux	SAI6T6_1000280		SAI6T6_1000270
B.Fir	Staphylococcus	M sauy	SAI8T7_1000280		SAI8T7_1000270
B.Fir	Staphylococcus	M saum	BN843_640		BN843_630
B.Fir	Staphylococcus	M saur	SABB_03471		SABB_01772
B.Fir	Staphylococcus	M saui	AZ30_00165		AZ30_00160
B.Fir	Staphylococcus	M sauf	X998_0036		X998_0034
B.Fir	Staphylococcus	M suh	SAMSHR1132_0033		SAMSHR1132_0032
B.Fir	Macrococcus	M mcak	MCCS_00760		MCCS_00770

Fig. 2 Ortholog tables corresponding to the KEGG module for methicillin resistance (M00625), where (**a**) complete modules and (**b**) one-block-missing modules are displayed

MODULE databases. It also contains the Draw Phylogram tool to add the user's sequenced data to a given set of GENES sequence data and to generate a phylogenetic tree, from which possible KOs can be inferred for the user's data.

3.2 Taxonomic Distribution

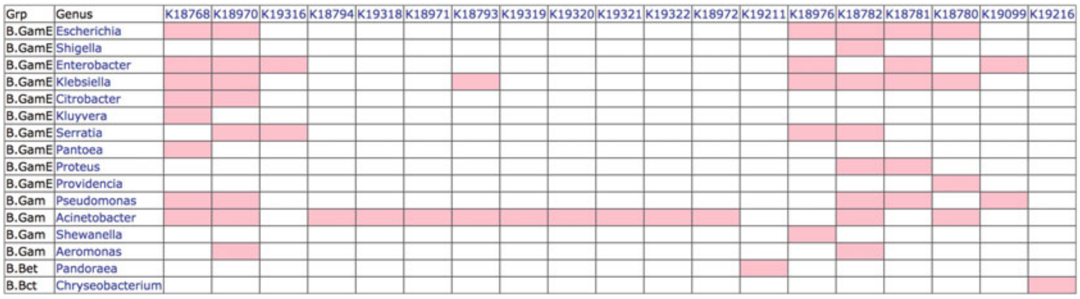
The ortholog table shown in Fig. 3 indicates whether genes for given KOs are present in KEGG organisms (complete genomes in KEGG) and also by coloring whether they are adjacent on the chromosome. There is another tool called module table. It indi-

Table 5
Signature KOs and signature modules

Type	Resistance genes/modules	Number of signatures
Gene variants	Beta-lactamase genes	48
(signature KOs)		
	Aminoglycoside resistance genes	39
	Tetracycline resistance genes	11
	Macrolide resistance genes	19
	Phenicol resistance genes	8
	Sulfonamide resistance genes	3
	Trimethoprim resistance genes	6
	Quinolone resistance genes	2
	Rifamycin resistance genes	2
	Fosfomycin resistance genes	5
Gene sets	Beta-lactam resistance modules	3
(signature modules)		
	Vancomycin resistance modules	2
	Tetracycline resistance module	1
	CAMP resistance modules	4
	Multidrug resistance modules	15

cates only the presence or absence of genes for given KOs or gene sets for given modules, but it is more suitable for characterizing taxonomic distributions of KOs or modules. Let us take the K numbers in Table 4 and see how the organism information in the last column can be obtained.

1. Access the KEGG Annotation page (www.kegg.jp/kegg/annotation.html) and go to the section of Module Table.
2. Enter the K numbers of Table 4 into the text area (*see* **Note 2**). You may copy and paste the entire text of Table 4, and click on "Filter" to clean up the text and to display only the K numbers.
3. Click on "Go" to display the module table, where pink cells indicate the existence of these K numbers in the genomes of KEGG organisms.
4. Change the display mode from "Organism" to "Species" and then to "Genus" to simplify the view where the existence of genes is displayed at the species and genus levels.



Grp	Genus	K18768	K18970	K19316	K18794	K19318	K18971	K18793	K19319	K19320	K19321	K19322	K18972	K19211	K18976	K18782	K18781	K18780	K19099	K19216
B.GamE	Escherichia	🟥	🟥												🟥	🟥	🟥	🟥		
B.GamE	Shigella															🟥				
B.GamE	Enterobacter	🟥	🟥	🟥											🟥		🟥		🟥	
B.GamE	Klebsiella	🟥	🟥					🟥							🟥	🟥	🟥	🟥		
B.GamE	Citrobacter	🟥	🟥																	
B.GamE	Kluyvera	🟥																		
B.GamE	Serratia		🟥	🟥											🟥	🟥				
B.GamE	Pantoea	🟥																		
B.GamE	Proteus															🟥	🟥			
B.GamE	Providencia																	🟥		
B.Gam	Pseudomonas	🟥	🟥													🟥	🟥		🟥	
B.Gam	Acinetobacter	🟥	🟥		🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥			🟥		🟥		
B.Gam	Shewanella														🟥					
B.Gam	Aeromonas		🟥													🟥				
B.Bet	Pandoraea													🟥						
B.Bct	Chryseobacterium																			🟥

Fig. 3 Module table for the signature KOs of carbapenem-hydrolyzing beta-lactamases shown in Table 4

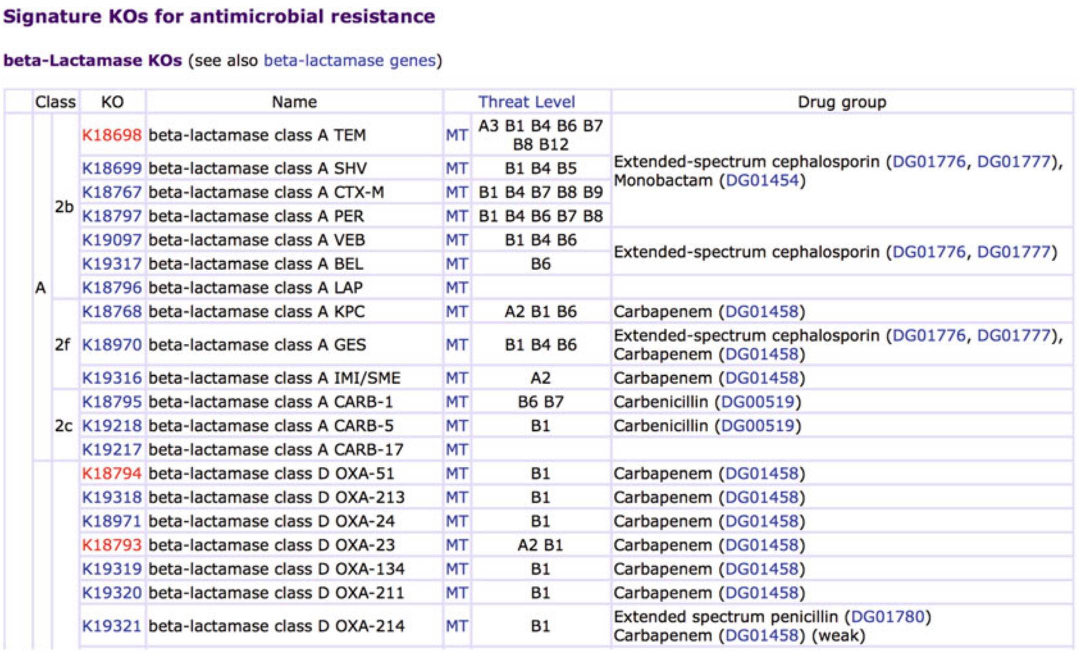
5. Click on the check box of "Include addendum" to add GENES data of addendum (non-genome) category. This is recommended because most AMR gene sequence data are collected from published literature and stored in this category.

The resulting module table is shown in Table 3. The organism group "B.GamE" stands for "Bacteria; Gammaproteobacteria - Enterobacteria" in the KEGG Organisms hierarchy (www.kegg.jp/brite/br08601), meaning *Enterobacteriaceae*.

3.3 Using BlastKOALA to Detect AMR

BlastKOALA [10] is a web server for automatic annotation of genome sequences, assigning KOs to a given set of amino acid sequences (*see* **Note 3**). This is a general server for any genome, but it can also be used to detect AMR genes and modules in pathogen genomes (*see* **Note 4**). Detailed steps of using BlastKOALA and performing KEGG Mapper analysis are described in another paper of this series [11]. Basically this server can be used as follows:

1. Access the BlastKOALA server (www.kegg.jp/blastkoala/). Enter a query set of amino acid sequences in FASTA format and select the database to be searched, such as species_prokaryotes.
2. Enter your email address and click on the "Request for email confirmation" button to upload query sequences and other data.
3. Check your email and click on the link in the email from the BlastKOALA server to submit your job.
4. You will receive another notification email when the job is completed. Click on the link to access the result page.
5. The result page contains links for viewing and downloading a list of KO assignments, as well as for performing KEGG Mapper analysis.
6. In order to understand AMR, click on the KEGG Mapper "Reconstruct Brite" link and look for relevant files in the list, especially "ko01504 Antimicrobial resistance genes" and "br01600 (table) Antimicrobial resistance: KEGG signatures."



Signature KOs for antimicrobial resistance

beta-Lactamase KOs (see also beta-lactamase genes)

Class		KO	Name	Threat Level		Drug group
		K18698	beta-lactamase class A TEM	MT	A3 B1 B4 B6 B7 B8 B12	Extended-spectrum cephalosporin (DG01776, DG01777), Monobactam (DG01454)
		K18699	beta-lactamase class A SHV	MT	B1 B4 B5	
		K18767	beta-lactamase class A CTX-M	MT	B1 B4 B7 B8 B9	
	2b	K18797	beta-lactamase class A PER	MT	B1 B4 B6 B7 B8	
		K19097	beta-lactamase class A VEB	MT	B1 B4 B6	Extended-spectrum cephalosporin (DG01776, DG01777)
		K19317	beta-lactamase class A BEL	MT	B6	
A		K18796	beta-lactamase class A LAP	MT		
		K18768	beta-lactamase class A KPC	MT	A2 B1 B6	Carbapenem (DG01458)
	2f	K18970	beta-lactamase class A GES	MT	B1 B4 B6	Extended-spectrum cephalosporin (DG01776, DG01777), Carbapenem (DG01458)
		K19316	beta-lactamase class A IMI/SME	MT	A2	Carbapenem (DG01458)
		K18795	beta-lactamase class A CARB-1	MT	B6 B7	Carbenicillin (DG00519)
	2c	K19218	beta-lactamase class A CARB-5	MT	B1	Carbenicillin (DG00519)
		K19217	beta-lactamase class A CARB-17	MT		
		K18794	beta-lactamase class D OXA-51	MT	B1	Carbapenem (DG01458)
		K19318	beta-lactamase class D OXA-213	MT	B1	Carbapenem (DG01458)
		K18971	beta-lactamase class D OXA-24	MT	B1	Carbapenem (DG01458)
		K18793	beta-lactamase class D OXA-23	MT	A2 B1	Carbapenem (DG01458)
		K19319	beta-lactamase class D OXA-134	MT	B1	Carbapenem (DG01458)
		K19320	beta-lactamase class D OXA-211	MT	B1	Carbapenem (DG01458)
		K19321	beta-lactamase class D OXA-214	MT	B1	Extended spectrum penicillin (DG01780) Carbapenem (DG01458) (weak)

Fig. 4 An example of BRITE table mapping. The genome sequence annotated by BlastKOALA is likely to contain AMR genes as shown in the mapping result against BRITE table br01600 (partially shown)

7. If these files are present in the list, click on the link to examine AMR genes and modules found in the query data.

Here *Acinetobacter baumannii* AB0057 (GenBank accession: GCA_000021245.2), which is not incorporated in KEGG, is used as an example. Over 50% of 3787 amino acid sequences were annotated with KOs by the BlastKOALA server. The BRITE mapping to ko01504 identified TEM, OXA-23, OXA-51 and ADC beta-lactamases, and other AMR genes. Furthermore, as shown in Fig. 4, the result of BRITE table mapping to br01600 helped understand the CDC threat levels and resistant drug groups.

The other types of KEGG Mapper analysis, Reconstruct Pathway and Reconstruct Module, may also be used to understand AMR. For example, check the mapping result of "01501 beta-Lactam resistance" to see how resistance pathways are reconstructed. Check also both complete and incomplete modules for drug resistance listed in the module reconstruction result.

4 Notes

1. There are two types of tree viewers in KEGG. One is the phylogram viewer for phylogenetic trees generated from multiple sequence alignments. The other is the dendrogram viewer showing similarity relationships generated from precomputed scores in the SSDB database (www.kegg.jp/kegg/ssdb/).

2. Module table is so named because it was originally developed to check taxonomic distribution of KEGG modules. It can now be used for KOs (K numbers) as well as for modules (M numbers).
3. There are two variants of the BlastKOALA server. One is the Annotate Sequence (www.kegg.jp/kegg/tool/annotate_sequence.html) tool in KEGG Mapper, which runs interactively by limiting the database to be searched. The other is GhostKOALA (www.kegg.jp/ghostkoala/), which runs much faster than BlastKOALA and is suitable for a large query dataset such as metagenome sequences.
4. The KAAS [12] server available at the GenomeNet site (www.genome.jp/tools/kaas/) is not recommended for pathogen detection, because it does not incorporate newly introduced GENES addendum category.

Acknowledgments

This work was partially supported by the National Bioscience Database Center of the Japan Science and Technology Agency. The computational resource for developing and servicing KEGG is provided by the Bioinformatics Center, Institute for Chemical Research, Kyoto University.

References

1. NCBI Resource Coordinators (2017) Database resources of the National Center for Biotechnology Information. Nucleic Acids Res 45:D12–D17
2. Jia B, Raphenya AR, Alcock B, Waglechner N, Guo P, Tsang KK, Lago BA, Dave BM, Pereira S, Sharma AN, Doshi S, Courtot M, Lo R, Williams LE, Frye JG, Elsayegh T, Sardar D, Westman EL, Pawlowski AC, Johnson TA, Brinkman FS, Wright GD, McArthur AG (2017) CARD 2017: expansion and model-centric curation of the comprehensive antibiotic resistance database. Nucleic Acids Res 45:D566–D573
3. Naas T, Oueslati S, Bonnin RA, Dabos ML, Zavala A, Dortet L, Retailleau P, Iorga BI (2017) Beta-lactamase database (BLDB) – structure and function. J Enzyme Inhib Med Chem 32:917–919
4. Bush K, Jacoby GA (2010) Updated functional classification of beta-lactamases. Antimicrob Agents Chemother 54:969–976
5. Kanehisa M, Furumichi M, Tanabe M, Sato Y, Morishima K (2017) KEGG: new perspectives on genomes, pathways, diseases and drugs. Nucleic Acids Res 45:D353–D361
6. Kanehisa M, Sato Y, Kawashima M, Furumichi M, Tanabe M (2016) KEGG as a reference resource for gene and protein annotation. Nucleic Acids Res 44:D457–D462
7. Hall BG, Barlow M (2005) Revised Ambler classification of beta-lactamases. J Antimicrob Chemother 55:1050–1051
8. Evans B, Amyes SG (2014) OXA β-lactamases. Clin Microbiol Rev 27:241–263
9. Périchon B, Goussard S, Walewski V, Krizova L, Cerqueira G, Murphy C, Feldgarden M, Wortman J, Clermont D, Nemec A, Courvalin P (2014) Identification of 50 class D β-lactamases and 65 Acinetobacter-derived cephalosporinases in Acinetobacter spp. Antimicrob Agents Chemother 58:936–949
10. Kanehisa M, Sato Y, Morishima K (2016) BlastKOALA and GhostKOALA: KEGG tools

for functional characterization of genome and metagenome sequences. J Mol Biol 428:726–731

11. Kanehisa M (2017) Enzyme annotation and metabolic reconstruction using KEGG. Methods Mol Biol 1611:135–145

12. Moriya Y, Itoh M, Okuda S, Yoshizawa A, Kanehisa M (2007) KAAS: an automatic genome annotation and pathway reconstruction server. Nucleic Acids Res 35: W182–W185

Index