Published in final edited form as:

Nat Protoc. 2019 February ; 14(2): 415–440. doi:10.1038/s41596-018-0099-1.

Using BEAN-counter to quantify genetic interactions from multiplexed barcode sequencing experiments

Scott W. Simpkins¹, Raamesh Deshpande², Justin Nelson¹, Sheena C. Li³, Jeff S. Piotrowski^{3,4}, Henry Neil Ward¹, Yoko Yashiroda³, Hiroyuki Osada³, Minoru Yoshida³, Charles Boone^{3,5}, Chad L. Myers^{1,2,†}

¹University of Minnesota-Twin Cities, Bioinformatics and Computational Biology Graduate Program, Minneapolis, Minnesota, USA

²University of Minnesota-Twin Cities, Department of Computer Science and Engineering, Minneapolis, Minnesota, USA

³RIKEN Center for Sustainable Resource Science, Wako, Saitama, Japan

⁴Yumanity Therapeutics, Cambridge, MA, USA

⁵University of Toronto, Donnelly Centre, Toronto, Ontario, Canada

Abstract

The construction of genome-wide mutant collections has enabled high-throughput, high-dimensional quantitative characterization of gene and chemical function, particularly via genetic and chemical-genetic interaction experiments. As the throughput of these experiments increases with improvements in sequencing technology and sample multiplexing, appropriate tools must be developed that can handle the large volume of data produced. Here we describe how to apply our approach to high-throughput, fitness-based profiling of pooled mutant yeast collections using the BEAN-counter software pipeline (Barcoded Experiment Analysis for Next-generation sequencing) for analysis. The software has also successfully processed data from *S. pombe*, *E. coli*, and *Z. mobilis* mutant collections. We also provide general recommendations for the design of large-

[†]Correspondence to chadm@umn.edu.

AUTHOR CONTRIBUTIONS

CB, MY, CLM, JSP, and SCL conceived the project. RD and CLM designed and RD wrote the original analysis software, which was extended and re-implemented as a full analysis pipeline by SWS with assistance from JN and HNW. SCL, JSP, and YY generated chemical-genetic interaction data for software development and testing. JN, SCL, and JSP performed extensive user-testing of the software. HO provided the RIKEN Natural Product Depository compounds used in the example dataset. SWS wrote the manuscript with input and editing from all authors.

COMPETING FINANCIAL INTERESTS

A license is required to use the BEAN-counter software (<http://z.umn.edu/beancr>). It is free for academic use and must be purchased on a per-project basis for commercial use.

DATA AVAILABILITY

All data needed to process the example dataset into chemical-genetic interaction scores are available at http://csbio.cs.umn.edu/BEAN-counter/example_dataset/. These data are a subset of the complete large-scale chemical-genetic interaction dataset, which is available from <http://mosaic.cs.umn.edu>, the supplementary material of the associated article¹⁰, or the corresponding author upon reasonable request. A license is required to use the BEAN-counter software and can be obtained at <http://z.umn.edu/beancr>. It is free for academic use and must be purchased on a per-project basis for commercial use.

CODE AVAILABILITY

The source code for BEAN-counter is available from <https://github.com/csbio/BEAN-counter>. It requires a license for use (<http://z.umn.edu>). It is free for academic use and can be purchased on a per-project basis for commercial use.

scale, multiplexed barcode sequencing experiments. The procedure outlined in this protocol was used to score interactions for approximately 4 million chemical-by-mutant combinations in our recently-published chemical-genetic interaction screen of nearly 14,000 chemical compounds across seven diverse compound collections. Here we selected a representative subset of the data on which to demonstrate the analysis in this protocol. BEAN-counter is open-source, written in Python, and freely available for academic use at <https://github.com/csbio/BEAN-counter>. Users should be proficient at the command line, while advanced users who wish to analyze larger datasets with hundreds or more conditions should also be familiar with concepts in analysis of high-throughput biological data. BEAN-counter encapsulates the knowledge we have accumulated from, and successfully applied to, our multiplexed, pooled barcode sequencing experiments. This protocol will be useful to those in the community interested in generating their own high-dimensional, quantitative characterizations of gene or chemical function in a high-throughput manner.

Keywords

genome-wide mutant collection; genetic barcode; multiplexed screen; interaction z-score; batch effect removal; massively parallel sequencing; high throughput screen; yeast; *S. cerevisiae*; drug discovery; chemical-genetic interactions; chemical genomics

INTRODUCTION

Profiling the fitness of a genome-wide collection of mutants against different experimental perturbations provides rich information on the functional effects of these perturbations inside the cell^{1–10}. Specifically, these screens allow the inference of a functional interaction between a mutant and a perturbation when the fitness of the mutant deviates from its expected fitness in the presence of the perturbation. Each functional interaction is classified as either a positive or negative interaction, meaning that the mutant was more or less fit than expected, respectively.

Fitness-based interaction screening has been applied across a variety of organisms as a means to systematically infer functional relationships between pairs of genes or between genes and environmental conditions (e.g. chemical compounds, heat shock). For example, when the perturbation introduced to the mutant collection is another genetic perturbation, the observed genetic interactions represent functional connections within cellular pathways as well as relationships between pathways that function together within a larger biological process^{5,6}. This strategy was used in *Saccharomyces cerevisiae* to construct the first complete genetic interaction network for an organism⁶, and screening of genetic interaction networks is underway for many other model organisms and in humans^{11–16}. Another application of this approach is to perturb a mutant collection with a chemical compound, which reveals the mutants that confer resistance (positive interaction) or sensitivity (negative interaction) to the compound and provides information on the compound's mode of action^{1–5,7,8,17,18}. Thus, fitness-based interaction screening provides a systematic framework for discovering gene function, inferring the functional organization of the cell, and identifying promising new therapeutic candidates.

The ability to perform fitness-based interaction screening in a pooled format provides substantial gains in efficiency and throughput over screens performed against arrays of individual mutants^{4,19}. By inserting unique DNA barcodes with common primer sites for PCR-based amplification into the individual mutants, entire mutant collections can be pooled and grown in competition, with the abundance of each genetic barcode used as a proxy for fitness¹⁹. For chemical genomic screens, this pooled format enables screening of rare and/or expensive compounds due to substantial reductions in the amount of required compound.

The introduction of massively parallel sequencing technology enabled another dramatic increase in the throughput of these screens^{10,20}. Originally, the genetic barcodes were quantified using microarrays specific to a collection of mutants, and one microarray was required for each condition profiled against the collection. However, current sequencing technology provides for the multiplexed quantification of barcode abundance across multiple conditions, vastly increasing throughput and decreasing costs for the barcode quantification step²⁰. We leveraged this new technology to develop a high-throughput chemical genomics screening platform in *S. cerevisiae*, enabling the multiplexed quantification – within one lane of Illumina HiSeq sequencing – of genetic barcodes from a diagnostic collection of ~300 mutants (selected to represent the entire collection of ~4000 deletion mutants) screened in 768 independent experiments¹⁰ (Figure 1a).

DEVELOPMENT OF THE PROTOCOL

We developed the BEAN-counter software pipeline (Barcoded Experiment Analysis for Next-generation sequencing) to address our need to process the data from high-throughput, sequencing-based interaction screens in a systematic, reproducible manner (Figure 1b). This software evolved out of a previous set of scripts developed in our lab collectively called “barseq-counter.” The primary components of this software were a sequencing data parser designed specifically for our PCR amplicons and a LOWESS-based interaction scoring procedure, the latter of which was inspired by the use of LOWESS for microarray-based gene expression analysis^{21–24}. We used this software for small-scale screens^{25,26} and for preliminary data processing for our screen of 10,000 compounds from the RIKEN Natural Product Depository¹⁰. During this initial round of processing, we became aware of signatures associated with specific technical attributes of the experiments (batch effects), so we introduced a batch correction procedure based on the knowledge our lab gained from the analysis of genome-wide genetic interaction screens in *S. cerevisiae*²⁷. Subsequent analysis revealed that the signal explaining the largest amount of variance in the dataset did not possess functionally-specific information, so we also introduced a procedure to successively remove the strongest signals from the dataset in order of decreasing magnitude. The transition from barseq-counter to BEAN-counter, in addition to the extra analysis features, also involved a substantial effort to make the code usable outside of our group and applicable to both small- and large-scale screens of mutant collections in many species.

The core of BEAN-counter is a simple pipeline to generate fitness-based interaction scores from the raw data (Figure 1b, “Processing”). This is accomplished by parsing the raw sequencing data to determine barcode (mutant) and index tag (condition) abundance,

removing automatically-determined and user-defined mutants and conditions that do not meet specific quality thresholds, and computing an interaction z-score for each mutant-condition pair. BEAN-counter possesses additional features to perform more complex processing tasks on the resulting interaction scores (Figure 1b, “Post-processing”). These procedures are typically only applicable to large-scale screens (hundreds or more conditions) and include the visualization and removal of systematic effects and otherwise unexplained yet uninformative variance in the data. BEAN-counter has been used to process dozens of screens containing between 1 and 10,000 compounds across multiple model organism mutant collections (e.g. *S. cerevisiae*, *S. pombe*, *E. coli*, *Z. mobilis*)²⁸ and thus encapsulates the knowledge we accrued throughout our experience developing a high-throughput, sequencing-based interaction screening platform.

APPLICATIONS

To comprehensively convey the potential applications of BEAN-counter, we describe here some potentially relevant details about the core interaction scoring pipeline, the basic format of our experiments, and finally specific recommendations for the types of experiments that can be analyzed using BEAN-counter. For further details on the interaction scoring algorithm, please see the supplementary note in the manuscript that describes our application of this pipeline to a chemical-genetic interaction screen of 14,000 compounds¹⁰.

To convert the raw sequencing data into abundances of all mutant-condition pairs, each amplicon is matched to user-provided lists of expected gene barcode and index tag sequences after first checking for the presence of the expected common primer sequence. Examples of the construction of mutant collections that are compatible with BEAN-counter are provided as references for multiple *S. cerevisiae* collections^{29–31} (including a protocol for mutant pooling²⁵), the *S. pombe* deletion collection³², and the *E. coli* Keio deletion collection³³. For the gene barcodes and index tags, BEAN-counter selects the closest-matching sequence within a user-specified edit distance (Levenshtein). BEAN-counter can match gene barcodes (but not index tags) of varying length, and in this scenario, it will select the longest sequence from the list of expected barcodes that matches the observed barcode within the specified edit distance. Amplicons are excluded if they contain an observed sequence that matches equally well to multiple reference sequences (these will account for a very small proportion of reads as long as each barcode differs from all other barcodes by more than two bases). BEAN-counter can only generate interaction scores for sequences it expects to find in the data, but it does export information on observed but unmatched sequences for troubleshooting purposes.

The output of interaction scoring is a z-score reflecting the direction and number of standard deviations the observed abundance for a mutant-condition pair deviates from its expected abundance. Thus, BEAN-counter does not report mutant fitness values but rather deviations from a fitness-based expected value for each mutant-condition pair. Broadly, these scores are generated by comparing each condition’s profile of mutant abundances against a mean profile of mutant abundances computed across a set of negative control conditions. Specifically, each condition’s logged abundance profile is LOWESS-normalized against the logged mean profile (using robust LOWESS to mitigate the effects of outliers), with

deviations from the expected abundance computed against the LOWESS line^{21,22}. A standard deviation is computed on these deviations, while a continuous estimate of standard deviation in the negative control conditions is computed as a function of strain abundance (using LOWESS on the squared deviations). For each condition, the resulting interaction z-score for each mutant is computed by dividing the calculated deviation by the largest of the two standard deviations. As both treatment and negative control conditions are scored against the mean mutant abundance profile derived from the negative control conditions, the resulting dataset contains interaction profiles for both treatment and negative control conditions. This enables the identification of technical effects that influence large groups of profiles regardless of their treatment or control status and the removal of offending mutants and/or conditions from the data.

Our chemical genomic screens are performed in 200 μ L micro-cultures in 96-well plates, where each well contains the pooled, barcoded mutant library challenged with a different condition (Figure 2a). Further experimental details are provided elsewhere^{10,25,34}. The mutant libraries are grown under these conditions for a specified amount of time (48 h for *S. cerevisiae*), after which the abundances of individual mutants from each treatment condition are compared to the respective mutant abundance distributions in the negative control conditions. We use indexed primers to amplify the genetic barcodes via PCR, resulting in PCR amplicons that map uniquely to each condition and mutant and can therefore be combined into a single lane of Illumina HiSeq sequencing (Figure 2b). As is evident from Figure 2b, the PCR amplicons are designed to be sequenced on Illumina machines.

Our specific experience is limited to Illumina HiSeq and MiSeq instruments, which present unique challenges in amplicon sequencing applications. Since all amplicons share the same common primer sequence, a 10% PhiX spike-in is required to provide sequence diversity if the amplicons are not already sequenced in the presence of other diverse sequences. This requirement for sequence diversity appears to be more stringent among the newer Illumina NextSeq and NovaSeq instruments, which may require a 25% PhiX spike-in or the use of indexed primers with variable-length index tags to provide sufficient sequence diversity. Note that while the former approach is compatible with BEAN-counter, the latter is currently not. While another sequencing platform may be compatible with PCR amplicons of this format (early work in this area utilized SOLiD sequencing²⁰), we have not tested BEAN-counter with any other next-generation sequencing technology and cannot speak to the appropriateness of our interaction scoring model on the data from different sequencing platforms.

In this protocol, we describe the application of BEAN-counter to a subset of the data (2592 out of 38,400 conditions) from our recently-published screen of nearly 14,000 chemical compounds spanning natural products and derivatives, combinatorially-synthesized compounds, and approved therapeutics and chemical probes with known modes of action¹⁰. However, BEAN-counter is not limited to chemical-genetic interaction screening and should be applicable to experiments that follow the same general design principles as those described here and throughout this protocol. For successful analyses using BEAN-counter, experiments should be designed as follows:

- As the null expectation is that each mutant is present at an expected nonzero abundance in each condition (and deviations from that abundance are thus interactions with a specific condition), experiments should be designed such that each mutant is present at a sufficient, nonzero abundance in the negative control conditions. The default detection limit is 20 counts per mutant-condition pair, and we aim for an average of 100 counts per mutant-condition pair to ensure nearly all strains meet this threshold. Strains that do not consistently meet this detection limit are automatically removed during processing and cannot be analyzed using BEAN-counter.
- At an absolute minimum, four negative controls should be included in a small-scale screen. We recommend that negative controls comprise 5–15% of the screened conditions and provide further details in the EXPERIMENTAL DESIGN section.
- The PCR amplicons to be analyzed by BEAN-counter should contain, in this order, an index tag sequence derived from the indexed primers, a common priming site that is the same across all amplicons, and a genetic barcode derived from each mutant (Figure 2b). Only the genetic barcode can vary in length.
- Index tag and genetic barcode sequences should be designed to be as dissimilar to each other as possible to reduce ambiguous mappings from observed sequences to the expected reference sequences. Each expected sequence should be a Levenshtein distance of at least two from all other expected sequences of that type, with greater distances preferred if possible.

COMPARISON WITH EXISTING TOOLS

BEAN-counter addresses a need in the pooled interaction screening community for a start-to-finish pipeline consisting of raw data processing, interaction scoring, and quality control that includes approaches for correcting the effects of technical artifacts in the data. Alternative tools can address some aspects of this conceptual pipeline and may be complementary to BEAN-counter. For example, much work has been performed to solve the problem of identifying common sequences within large collections (up to millions) of sequences, and some of these approaches could be modified to count the abundance of each index tag and barcode combination^{35–42}. While most of these tools provide differing methods for clustering sequences based on similarity across the entire observed sequence, the recently-published Bartender software allows for the extraction and clustering of barcodes given an arbitrary PCR amplicon structure (including variable-length index tags and barcodes), which facilitates the calculation of barcode abundance⁴². The edgeR package for R and the Barcas software tool are also capable of processing PCR amplicon sequences of the same general structure presented in this protocol into a count matrix, and like BEAN-counter, they must be provided with expected barcodes and index tags prior to analyzing the sequencing data^{43–45}. Barcas offers additional functionality to deal with index tags and/or barcodes with varying lengths and locations.

Both Barcas and edgeR are capable of scoring interactions, but only edgeR offers the potential to remove batch effects by specifying them as covariates in its statistical modeling framework^{43–45} (which is different from our approach based on Fisher's linear discriminate analysis). Other possible methods for supervised and unsupervised batch effect correction are found in the SVA package for R^{46–48}. It is important to note that the interaction scoring model used in BEAN-counter was developed specifically for the data from our high-throughput, pooled chemical-genetic interaction screens, is unique among the methods described here through its empirical estimation of variation as a function of barcode abundance, and has been particularly useful for obtaining high-quality, reproducible, functional information for the compounds in these screens. Additionally, using edgeR or similar methods to score interactions may require different experimental design considerations than those outlined in this protocol.

LIMITATIONS

BEAN-counter was designed to process amplicons with a fixed structure and known index tag and genetic barcode sequences, as described in APPLICATIONS. The interaction z-score it reports is based on the deviation from an expected relative abundance, which requires sufficient barcode abundance in negative control conditions and depends on the fitness distribution of other mutants in the pool in the condition of interest, ecological interactions among mutants in the pool, and the number of generations over which the pool is grown. Thus, BEAN-counter is not appropriate when barcodes are below the detectable level in negative control conditions or when the composition of the pool changes drastically in treatment vs. control conditions. Additionally, BEAN-counter is not appropriate for tracking rare barcodes in a cell population, which requires tagging the individual barcodes with unique molecular identifiers (UMIs) to quantify their absolute abundances. Examples of experiments to which BEAN-counter should not be applied include lineage tracking in a population of 500,000 uniquely-barcoded cells⁴⁹ and screens in which the expected fitness of all mutants in the negative control conditions is zero.

In general, BEAN-counter does not perform any quality control pre-processing on the raw sequencing data, although, in our applications, we have not found this necessary. Additionally, methods to summarize interaction scores at the gene level for multiple mutants of the same gene (e.g. for a pooled CRISPR screen with multiple guide RNAs per gene) are beyond the scope of this software.

REQUIRED EXPERTISE

To run BEAN-counter, you should be familiar with running software from the command line (preferably the Linux terminal), which also involves navigating the filesystem, creating directories, and transferring files. The pipeline is written completely in Python, and thus knowledge of this programming language is a major advantage if errors arise. You should also be familiar with using Java TreeView to visualize clustered heat maps. We have predominantly used version 2 of Java TreeView (<http://jtreeview.sourceforge.net/>), which has a user manual on its website. Version 3 of Java TreeView (in development, but stable: <https://>

bitbucket.org/TreeView3Dev/treeview3/) is also capable of viewing the *.cdt files generated by BEAN-counter.

Additionally, it is important to make a distinction between the level of statistical and/or data analysis training we recommend you possess for analyzing a screen of tens of compounds versus one with thousands of compounds. Large-scale screens provide more opportunities for the occurrence, detection, and removal of systematic effects, and the extent to which these effects are removed depends on judgment calls you will make at processing time. In the OVERVIEW OF THE PROTOCOL, we describe our thought processes surrounding these judgment calls when processing chemical-genetic interaction data. However, if you are interested in using BEAN-counter to analyze thousands of interaction profiles, we believe you would benefit from a formal introduction to concepts in large-scale data analysis – in particular, outlier detection, singular value decomposition and principal component analysis, batch effect correction, and precision-recall (PR) and receiver operating characteristic (ROC) analysis.

OVERVIEW OF THE PROTOCOL

Analyses with BEAN-counter can be subdivided into three major parts: (1) setup (Steps 1–5), (2) interaction scoring (Steps 6–17), (3) and post-processing (Steps 18–24) – as outlined below. The setup of the working environment involves constructing multiple configuration files and tables required for interaction scoring. More precisely, the configuration files specify important interaction scoring parameters, the location of the output, and how to map the observed sequences to mutants and conditions via the gene barcode and sample information tables. Interaction scoring first involves computing mutant and condition abundances and evaluating associated quality control measures to ensure the parsing of the sequencing data was successful (steps 6–7). Converting abundances to interaction scores is then an iterative process of data processing and visualization that concludes when the user is satisfied with the quality of the mutants and conditions included in the dataset (steps 8–17) – especially the negative control conditions which are the reference against which interactions are scored (steps 11–12). Post-processing is typically restricted to large screens (hundreds or more conditions) and involves the detection and removal of systematic biases and other artifactual signals. In this protocol we first check for batch effects based on index tag (step 19) and sequencing lane (step 20) and then characterize and remove a dominant signal that obscures compound replicate correlations (steps 21–24). All post-processing scripts can be used in any order on any matrix of interaction scores created by BEAN-counter, as their parameters are specified via command line arguments instead of a common configuration file.

EXPERIMENTAL DESIGN

Generating screen data

We previously demonstrated that we can combine the PCR products from at least 768 conditions into one lane of HiSeq sequencing (768-plex) when using our diagnostic pool of ~300 *S. cerevisiae* deletion mutants¹⁰, and in unpublished experiments we have achieved at least 96-plex sample multiplexing across the collection of all *S. cerevisiae* nonessential gene

deletion mutants (~4000 mutants). Assuming a minimum of 80 M reads per HiSeq lane, this results in an average of ~350 reads/mutant/condition in the former case and ~200 reads/mutant/condition in the latter. The extent to which conditions can be multiplexed has nearly doubled since we performed our original large-scale screens due to the continual increases in Illumina sequencing depth. We recommend a sequencing depth that ensures an average of 100 reads per mutant per condition, which has been sufficient for our screens across tens of thousands of conditions.

For large scale screens involving hundreds of compounds, experimental design considerations must include more than just the sequencing depth. We recommend including in each 96-well plate 4 negative controls (for our chemical genomics experiments, this is the solvent control, DMSO) and 4 positive control conditions (compounds with distinct and well-characterized chemical-genetic profiles; in our case, we use benomyl, tunicamycin, bortezomib, and methyl methanesulfonate) (Figure 2c). This enables the identification of any “flipped” (i.e. physically rotated) plates and ensures that the experiments in each plate were successful.

Plates containing only negative control conditions (with 4 positive controls) should also be screened to provide sufficient information regarding mutant fitness variability in the absence of treatment conditions. Ideally, at least one of these plates would be screened on each day for which a large number of other plates are screened. Additionally, signatures associated with specific index tags can be detected if the screen is designed such that multiple negative control conditions are tagged using the same indexed primer. If such signatures are strong enough (determined via a profile correlation threshold), the conditions associated with these offending index tags are automatically removed. Examples of screening formats that pair each index tag with at least one negative control condition are given in Figure 2d.

Basic statistical intuition should guide the design of screens to both minimize batch effects and maximize the probability of removing them. Specifically, it is important to avoid confounding the effects of compounds with those that may be attributable to other technical aspects of the experiments. In the development of our high-throughput chemical genomics assays in *S. cerevisiae* and other species, we have observed signatures associated with conditions’ screening dates, screening plates, indexed primer plates, indexed primer sequences, and sequencing lanes. We recommend performing three biological replicates for each condition, with the replicates ideally screened in different plates, amplified using different indexed primers, and sequenced in different lanes. Conditions should be grouped randomly to avoid instances where conditions with similar biological effects are screened in the same batch, resulting in what appears to be a batch effect but is actually biological similarity. For each day on which the conditions are screened, negative controls should comprise ~5–15% of the screened conditions.

We specify here a few conventions that standardize how we refer to files and folders, configuration parameters, and software commands. In this protocol, paths to files and folders are italicized, and we assume that the user’s working directory is *<dir>*/. A directory within *<dir>*/ with the name “output” is therefore notated as *<dir>/output/*. The names of the scripts that comprise BEAN-counter (and their arguments) are written in Courier New font (e.g.

`process_screen.py`). Parameters within configuration files are underlined and column names in gene barcode and sample information tables are 'single-quoted.'

Setup: preparing files for processing.

The primary BEAN-counter configuration file provides the pipeline with all of the information it needs to process raw fastq files into interaction scores (Supplementary Figure 1). The configuration file (`<dir>/config_files/config_file.yaml`) specifies the locations of the raw data (via the lane_location_file parameter), the tables that map index tags to the screened conditions (sample_table_file) and genetic barcodes to the mutant strains (gene_barcode_file), and the file that determines how to read the index tags and barcodes from the raw PCR amplicon sequences (amplicon_struct_file). It is also used to specify the thresholds for various quality control measures that determine which conditions and/or mutants are automatically removed from the dataset. Setting up the configuration file, other required files, and the recommended directory structure can be performed manually or with the help of the `setup_screen.py` script. Supplementary Table 1 describes all parameters available to the configuration file and the `setup_screen.py` script.

Interaction scoring: iteratively scoring and filtering the data.

Generating interaction z-scores from the raw sequencing data is accomplished using the `process_screen.py` script in Step 6, which contains six stages that can be run individually if necessary (Figure 3a). Stage 1 parses the fastq files to generate per-lane matrices containing the number of reads observed for each combination of index tag (condition) and barcode (mutant). Stage 2 performs an initial round of interaction scoring for each lane individually, the results of which are used to perform index tag quality control (stage 3). In stages 4 and 5, the per-lane count matrices are combined and filtered based on the automatically- and user-determined filtering information (which mutants and conditions should be retained/removed). Stage 6 generates the final interaction matrix by scoring interactions using the combined, filtered count matrix.

The ability to perform the stages in `process_screen.py` individually is particularly useful due to the iterative nature of the interaction scoring process. Viewing the final interaction matrix is usually what allows the user to identify conditions and/or mutants that should be manually removed from the dataset. However, it is not necessary to re-parse the raw data just to modify parameters that are incorporated at later stages. Using the `--start 2` flag, for example, allows the user to proceed with the first round of individual-lane interaction scoring (i.e. start at stage 2) as long as stage 1 (fastq parsing) completed successfully. Similarly, the user can perform much of the iteration between filtering and interaction scoring using only stages 5 and 6, as most of the user-specified condition/mutant filtering information (except the selection of negative controls for interaction scoring) is only incorporated when the combined count matrix is filtered in stage 5. You can therefore iterate rapidly on the stages of matrix filtering and interaction scoring by invoking `process_screen.py` with the `--start 5` flag after each modification of manual or automatic filtering parameters.

BEAN-counter provides both automatic and manual methods for removing mutants and conditions that do not meet specific quality control criteria. By default, mutants that do not possess 20 or more counts for at least 25% of the conditions are removed, as are conditions that do not possess 20 or more counts for at least 25% of the mutants (these are advanced parameters in *config_file.yaml*). Strong interaction z-scores in the profiles of negative control conditions should be dealt with by removing either the offending mutants or negative controls; this choice is arbitrary and depends on which information in the dataset is more valuable to retain. Mutants that are sensitive or resistant to most conditions should also be removed, as their signal is almost certainly not biologically relevant. In our experiments, many highly growth-inhibitory conditions (< 50% growth compared to negative controls) reveal a common subset of resistant mutants that are ultimately assigned very large positive interaction scores (e.g. > +15). We remove from the dataset either these broadly-resistant mutants or the conditions that elicit the resistance, as these mutants are not useful in interpreting compound mode of action and their disproportionately-high scores present substantial challenges in post-processing steps. For chemical genomics experiments, appropriate selection of compound screening concentration mitigates this effect.

Post-processing: removing systematic biases and uninformative signal.

Once a final matrix of interaction scores has been generated, BEAN-counter provides “post-processing” tools to detect and remove systematic biases and common yet uninformative signal (Figure 3b). These tools are most appropriate for large-scale screens (hundreds of compounds) and require more data analysis expertise than simply scoring the interactions. To the latter point, post-processing is more iterative and reliant on user judgement calls, as the individual post-processing scripts can be run in any order so that the largest effects are removed first. The core tools in BEAN-counter post-processing are Fisher’s linear discriminant analysis (LDA) and singular value decomposition (SVD), both of which learn signals (referred to as components) that can then be removed from the data. The general procedure for removing unwanted signal from the interaction data is to determine if the signal exists, if it can be removed, and how much should be removed based on metrics such as replicate correlation.

In these post-processing steps, the user must determine the number of LDA or SVD components to remove within each type of correction and also the order in which to apply the different types of correction (i.e. SVD before LDA batch correction, or vice versa). BEAN-counter assists in making these decisions by providing appropriate evaluation metrics in the form of histograms, PR curves, and ROC curves. With each subsequent removal of an LDA or SVD component, the observed separation between the distributions of within-batch and between-batch correlations should decrease in the histograms, the area under the ROC curve should approach 0.5 (the $y = x$ line), and PR curves should show a decrease toward the final value in the curve (the background precision). The success of batch effect correction can also be measured with respect to replicate correlations (treating each set of replicates as a “batch”), in which case the separation between replicate and non-replicate correlation distributions, the area under the ROC curve, and the area under the PR curve should all increase. Because these corrections are not foolproof, it is important to view a clustered heat map of each corrected dataset. If an examination of the heat map suggests that substantial

signal was removed from the dataset despite minimal improvements in the evaluation plots, it is likely that the batch effects were not correctable and that the batch correction procedure was confounded by a strong signal correlated with many conditions in the same batch.

For nearly every large-scale dataset we have generated, including screens against deletion mutants of nonessential genes and hypomorphic mutants of essential genes, we observe a common signal in 15–25% of mutants and more than 15% of conditions. Growth inhibition appears necessary but not sufficient to induce this signal, and it appears to reflect a common general effect on mutant fitness (Figure 4a). This signal is ultimately responsible for spurious correlations that obscure more interesting and specific relationships between different conditions. BEAN-counter provides the ability to remove signals such as these via singular value decomposition (SVD), an unsupervised technique that identifies the main axes of variation in a dataset (Figure 4b). Starting with the dimension of highest variation, the user can specify how many of these “SVD components” to remove from the dataset in a successive manner (i.e. it is not possible within BEAN-counter to only remove the second SVD component). Removing this signature from the dataset (in most cases, the first SVD component) typically changes the observed bimodal distribution of replicate correlations and non-zero-centered distribution of non-replicate correlations (Figure 4c) into a unimodal distribution of replicate correlations and a zero-centered distribution of non-replicate correlations (Figure 4d). Precision-recall and receiver operating characteristic (ROC) curves provide a more objective and quantitative evaluation of this change, where increases in the area under both curves reflect improvements in the ability of profile correlations to predict if two conditions are replicates of each other (Figure 4e) and the area under the ROC curve also directly reflects the separation of the replicate and non-replicate correlation distributions.

A batch effect is defined here as the occurrence of a higher than expected correlation between conditions that are associated with the same technical attribute in an experiment. We have observed higher than expected profile correlations for conditions screened on the same day or in the same plate, amplified with the same indexed primer or primers from the same 96-well plate, or sequenced in the same lane. BEAN-counter provides batch effect correction via a multiclass implementation of Fisher’s linear discriminant analysis (LDA), which has been used previously to correct batch effects in genetic interaction data⁴¹. Because this is a supervised correction, the user must specify the batch labels as well as other batches that could confound the detection and removal of the effect in question. The batch correction script removes duplicate instances of the confounding batches occurring within each batch of the interrogated effect so that common signals between conditions that are expected to correlate with each other are not mistaken for batch effects. As with the SVD correction, batch-related signals are removed successively starting with those of the largest magnitude.

In some cases, we have observed batch effects severe enough to enable a different correction approach (Figure 5). Specifically, when the batches (typically the day on which the conditions were screened) were large enough and contained a sufficient number of negative experimental control conditions, it became possible to score interactions within each group

of conditions, instead of removing the batch effects with LDA after the calculation of z-scores. This option for BEAN-counter is described in detail in Box 1.

MATERIALS

EQUIPMENT

- Hardware: 64-bit computer running Linux (or Mac OS), with at least 4 GB of RAM (8 GB recommended); the software may run on Windows with few or no modifications, but we have not tested it in a Windows environment.
- Python 2.7, with the following libraries (required version number in parentheses): numpy (1.12.1), scipy (0.19.0), pandas (0.20.1), matplotlib (2.0.2), scikit-learn (0.18.1), biopython (1.68), fastcluster (1.1.20), pyyaml (3.11), networkx (1.11). The Conda environment manager for Python works well for installing Python and the above packages (<https://conda.io/docs/user-guide/install/index.html>). If using Conda, libraries can be obtained from the Anaconda Cloud (<https://anaconda.org>).
- BEAN-counter software (available at <https://github.com/csbio/BEAN-counter>, see EQUIPMENT SETUP)
- Java TreeView (<http://jtreeview.sourceforge.net/>); this is the recommended application for viewing clustered heat maps of interaction scores
- Data and configuration files (available from: <http://csbio.cs.umn.edu/BEAN-counter/>, see equipment setup for download details)

EQUIPMENT SETUP

Downloading and installing the software—Python version 2.7 and the previously listed libraries (see EQUIPMENT) must be installed in order to run the BEAN-counter software. If you are not familiar with the specifics of installing Python and the required libraries on your system, contact your system administrator for help.

BEAN-counter is hosted on GitHub at <https://github.com/csbio/BEAN-counter>. The most up-to-date stable release can be found at <https://github.com/csbio/BEAN-counter/releases/>. At the time of this protocol's submission, the latest tagged release is version 2.6.0 (please be sure to obtain the most up-to-date release). The following commands will download this version of the software to the user's home directory and unzip it (lines beginning with '\$' indicate commands run from the terminal):

```
$ cd ~
$ wget https://github.com/csbio/BEAN-counter/tags/2.6.0.tar.gz
$ tar -xzf BEAN-counter-2.6.0.tar.gz
```

More advanced users can directly fork the GitHub repository to keep up with the latest updates.

BEAN-counter requires an environment variable named `BARSEQ_PATH` that provides the path to the folder containing the BEAN-counter code. In the bash shell in Linux and using the previous example, this is performed with the following command:

```
$ export BARSEQ_PATH=$HOME/BEAN-counter-2.6.0
```

Add the BEAN-counter *master_scripts/* directory to your `PATH` environment variable. This enables you to use the commands in this directory while working in another directory (specifically, the directory specific to your screen), and is performed with the following command:

```
$ export PATH=$PATH:$HOME/BEAN-counter-2.6.0/master_scripts
```

The above two commands can be added to your `~/.bashrc` file (or the relevant script that is executed each time a new terminal is opened) so that you do not have to execute them before each analysis session. For those not familiar with setting environment variables, the following website provides a good starting point for Linux (also applicable to Mac OS) and Windows: <https://scotch.io/tutorials/how-to-use-environment-variables>.

Once you know the technical format of your next screen (specifically, the pool of barcoded mutants and the structure of the PCR amplicons), create the appropriate mutant barcode table and amplicon structure files and add them to their respective folders inside the BEAN-counter directory (*data/gene_barcode_files/* and *data/amplicon_struct_files/*). We have made these files available at <http://csbio.cs.umn.edu/BEAN-counter/> for the different types of screens we have performed. You may find these helpful if performing a screen in a similar format to ours or attempting to reproduce one of our analyses.

PROCEDURE

CRITICAL This procedure is a walk-through on a subset of the recently-published dataset containing chemical-genetic interaction profiles for nearly 14,000 compounds across a diagnostic collection of 310 deletion mutants. Where necessary, we differentiate between instructions specific to processing the example dataset and those applicable to dataset processing in general. A shell script containing all commands in the following procedure is provided as the Supplementary Manual. For the timings, “active” time is time when the user is required to actively use the software, while during “passive” time the user can step away while the software runs.

Setup of the working environment (5 min active, 1–2 h passive)

1. After installing BEAN-counter and its prerequisites as specified in the EQUIPMENT SETUP, download the relevant gene barcode table and amplicon structure file into the appropriate directory inside the BEAN-counter *data/* directory. Using the commands below, we first change the working directory to the BEAN-counter software directory, then download the gene barcode table for

the diagnostic collection of 310 deletion mutants, and finally download the relevant amplicon structure file (provided as Supplementary Data 1 and 2, respectively)

```
$ cd $HOME/BEAN-counter-2.6.0
$ wget http://csbio.cs.umn.edu/BEAN-counter/gene_barcode_files/
Sc_hap_del_mp_hs_v1.txt -P data/gene_barcode_files
$ wget http://csbio.cs.umn.edu/BEAN-counter/amplicon_struct_files/
Sc_hap_MP-WG-HET-ix10.yaml -P data/amplicon_struct_files
```

2. While designing your screen and/or while the PCR amplicons are sequenced, begin the process of setting up a directory that will contain all of the raw and processed data. In this protocol, all subsequent commands assume that the user's working directory is *<dir>*. To create and enter this directory, use the following commands:

```
$ mkdir -p <dir>
$ cd <dir>
```

3. Organize the working directory into different subfolders and generate the required configuration, sample information, and gene barcode files. In this protocol, the interactive `setup_screen.py` script will assist you with this process. Enter the following command and responses to the prompts (displayed as “>>>prompt: **user_response**”) to set up your working directory (if **ENTER**, do not specify any response – the default option will be used):

```
$ setup_screen.py -i
>>>config_file: ENTER
>>>output_folder: ENTER
>>>lane_location_file: ENTER
>>>sample_table_file: ENTER
>>>gene_barcode_file: Sc_hap_del_mp_hs_v1.txt
>>>amplicon_struct_file: Sc_hap_MP-WG-HET-MoBY.yaml
>>>num_lanes: 9
>>>new_sample_table: False
>>>verbosity: ENTER
>>>sub_screen_column: ENTER
>>>Would you like to specify advanced parameters? (y/n): n
>>>clobber: ENTER
```

CRITICAL STEP If you make a mistake and must run `setup_screen.py` again, you may need to set the “clobber” parameter to **True** in order to overwrite the original output.

4. In step 3, the location of the sample information table was set to be `<dir>/sample_table/sample_table.txt`, but the table itself was not created by the script because we have provided it for you. Copy the sample information table to `<dir>/sample_table/` using this command:

```
$ wget http://csbio.cs.umn.edu/BEAN-counter/example_dataset/sample_table/sample_table.txt -P sample_table
```

The sample information table is also provided as Supplementary Data 3.

CRITICAL Gene-barcode and sample information tables must be in plain text format – not rich text or any word processor-derived formats (*.doc, *.docx, etc.). To create and modify these files, be sure to use a plain text editor or Microsoft Excel (for the latter, be sure to save the output as tab-delimited text). The formats of these two tables, including required columns, are described in Supplementary Figure 1.

5. Transfer the raw data for each lane into their respective directories in `<dir>/raw/`. For this protocol, the following commands will download the raw data for the first two sequencing lanes into its destination directory (there are 9 lanes in total: lane1–lane9):

```
$ wget http://csbio.cs.umn.edu/BEAN-counter/example_dataset/raw/lane1/lane1_R1.fastq.gz -P raw/lane1
$ wget http://csbio.cs.umn.edu/BEAN-counter/example_dataset/raw/lane2/lane2_R1.fastq.gz -P raw/lane2
```

CRITICAL STEP BEAN-counter will read uncompressed fastq files as well as gzip, bzip2, and zip-compressed files. These files must possess the “.fastq” extension (and the additional relevant extension if compressed).

Interaction scoring and quality control determination (Timing variable, < 40 min active for the example dataset; ~4 h passive)

6. Perform the first round of interaction scoring from start to finish using this command:

```
$ process_screen.py config_files/config_file.yaml
```

7. After the fastq parsing stage has completed, review the parsing reports for each sequencing lane contained in `<dir>/output/reports/`. In the typical use case, most reads (> 80%) will possess the common primer sequence, and most of these reads (> 80%) will map to both a mutant and a condition. However, if the library of PCR amplicons from your experiment was mixed 50/50 with a completely different library before sequencing, then nearly half (> 40%) of the reads should

possess the common primer sequence. Typical distributions of reads mapped to each index tag and barcode are shown in Figure 6.

CRITICAL STEP Be sure to review the sequencing parsing reports before proceeding. If most reads do not map to index tags and barcodes, this may be a sign that the configuration files or sample/gene tables do not match the format of the experiment.

TROUBLESHOOTING

8. Once `process_screen.py` has completed running (as evidenced by the terminal being usable again), generate clustered heat maps of the interaction scores. For this command, the parameters after `config_file.yaml` specify the columns of the barcode table (e.g. 'Gene_name') and sample information table (e.g. 'name'), respectively, to be included in the row and column names of the heat map. Note that only commas (not spaces) are allowed between the column names.

```
$ visualize_zscore_matrices.py config_files/config_file.yaml
Gene_name,Strain_ID
screen_name,expt_id,name,lane,index_tag_plate,index_tag_well,index_
tag
```

9. **CRITICAL** Steps 9–15 are performed in Java TreeView to visualize the entire interaction dataset as a clustered heat map. In this representation of the data, conditions cluster more closely to each other if their patterns of interactions across the mutants are more similar, and the same applies for mutant clustering based on the similarity of interaction patterns across the conditions. The groups of conditions and mutants that emerge from the clustering process are used to visually evaluate the quality of the dataset. Java TreeView also allows users to mouse over the values and view the actual interaction scores.

Using Java TreeView, open the clustered heat map containing the final interaction scores located at `<dir>/output/interactions/all_lanes_filtered/all_lanes_filtered_scaled_dev.cdt` and browse the interaction data to become familiar with it (Figure 7a, main heat map). **Steps 10–15 will guide you through manual quality control determinations.**

10. First, evaluate the quality of the positive control conditions (Figure 7a, “positive control conditions”). This can be done in Java TreeView by searching for “arrays” with the names of the positive control conditions. If four positive controls were screened in every plate as recommended, each of these four conditions should possess a large, easily-identifiable cluster (based on a shared pattern of interaction scores) in the heat map. For this dataset, the four positive controls are Benomyl, Bortezomib, Micafungin, and MMS.

TROUBLESHOOTING

11. Check the quality of the negative control conditions and mutant behavior across these conditions (Figure 7b). This can be done within Java TreeView by searching for “arrays” with containing the string “DMSO” and choosing the “Summary pop-up” option. If a mutant shows strong interactions (± 5) in more than a few of the negative control profiles, it is likely that many of this mutant’s interactions in the complete dataset are false positives. Keep track of the ‘Strain_ID’s of all such mutants so they can be removed later. In this dataset, we flag 16 mutants for removal based on their behavior in the negative control conditions. Their identities are given in Figure 7a (“Mutants flagged for removal”) and Supplementary Data 4 (where the value in the ‘include?’ column is “False”).

CRITICAL If the negative control conditions from different sample groups or batches (such as those screened on the same day, as illustrated in Figure 5) cluster strongly into distinct groups, it may be prudent to repeat the interaction scoring procedure after specifying these groups as “sub-screens” as described in Box 1. Please reference Box 1 to determine if this special procedure is appropriate for your dataset.

TROUBLESHOOTING

12. Likewise, examine all negative control conditions in this entire dataset and flag any for removal (keep track of the ‘expt_id’ value) if they possess signal that is uncharacteristic of the other negative control profiles and is not observed in the complete dataset (these would be considered outlier control conditions). In this dataset, we flag 24 negative control conditions for removal, and their identities are given in Supplementary Data 5 (where the values in the ‘include?’ and ‘control?’ columns are “False” and “True”, respectively).

TROUBLESHOOTING

13. Now, examine the entire matrix of interactions in Java TreeView to identify mutants that should be removed based on their behavior in non-control conditions. Flag mutants that appear to interact with most conditions. If post-processing steps will be performed, it may also be useful to flag mutants that demonstrate extremely large positive interaction scores ($> +15$) if the conditions that induce these scores are important to retain (the large scores can confound post-processing procedures). In the context of the diagnostic 310 mutant collection, we often remove the *gtr1* and *avt5* mutants for this reason. For the example dataset, the strains with unacceptable behavior across all conditions were already flagged due to their behavior in negative control conditions.
14. Additionally, flag conditions that do not appear to have valid interaction profiles. This instruction is intentionally subjective, as the definition of a “valid interaction profile” will depend on the mutant collection and experimental setup. One commonly observed pattern among profiles that we deem as invalid consists primarily of zero scores that are mixed with a seemingly random set of large positive interaction scores. From this dataset, we remove such profiles (Figure 7a, “Conditions flagged for removal”, left-most columns) as well as treatment

conditions with profiles similar to the outlier negative control profiles (Figure 7a, “Conditions flagged for removal”, right-most columns). In all, we flag 12 non-control conditions for removal.

PAUSEPOINT

Before continuing with the removal of manually-flagged conditions and mutants from the dataset, be sure to document these signals and reflect on what might have caused them. If you did not perform the experiments yourself, consult with your experimentalist colleague(s) to identify possible links between the signals and variations in experimental conditions. Determine if future experiments can be performed in a way that reduces the presence of these signals. Additionally, take a closer look at conditions for which either the interaction profiles or the experimental conditions that generated them arouse suspicion in your experimentalist colleague.

15. Once you have identified all of the mutants and conditions to be removed from the dataset, proceed to modify the gene barcode table (*<dir>/barcodes/barcodes.txt*) and the sample information table (*<dir>/sample_table/sample_table.txt*) such that the entries in the ‘include?’ column are changed to “False” for the mutants and conditions to be removed, respectively. For this protocol, the modified gene barcode and sample information tables are given as Supplementary Data 4 and 5, respectively. They can also be downloaded to the appropriate locations using the following commands:

```
$ wget http://csbio.cs.umn.edu/BEAN-counter/example_dataset/
barcodes/barcodes_filtered.txt -P barcodes
$ wget http://csbio.cs.umn.edu/BEAN-counter/example_dataset/
sample_table/sample_table_filtered.txt -P sample_table
```

Be sure to modify *<dir>/config_files/config_file.yaml* so that it references the filtered gene barcode and sample information tables.

CRITICAL Gene-barcode and sample information tables must be in plain text format – not rich text or any word processor-derived formats (*.doc, *.docx, etc.). To modify these files, be sure to use a plain text editor or Microsoft Excel (for the latter, be sure to save the output as tab-delimited text).

16. After removing the flagged mutants and conditions, re-run the *process_screen.py* command using the *--start 5* flag and the modified *config_file.yaml* (see step 6). Proceed from step 8 of this procedure (visualizing the dataset) and identify any further mutants or conditions that should be removed from the dataset. For the example dataset, no further iterations are needed.

```
$ process_screen.py --start 5 config_files/config_file.yaml
```

17. Re-run `process_screen.py` using the `--start 2` flag to generate a final dataset guaranteed to have been processed with exactly the same parameters through all stages (Figure 7c).

```
$ process_screen.py --start 2 config_files/config_file.yaml
```

PAUSEPOINT

For a small-scale screen (tens of compounds), the analysis is likely complete at this stage. A dataset of that size is not large enough to check for batch effects and other systematic variation. One possible exception is the very next step, which checks profile correlations among replicate conditions.

Post-processing (10 min active)

18. Check the correlations of replicate conditions (Figure 8a). Here, the `“-incl not_pos_neg_control”` argument tells the script to include only treatment conditions (via the `‘not_pos_neg_control’` column in the sample information table, which is `“True”` if the condition is not a positive or negative control), the `“name”` argument indicates that each batch is defined by the `‘name’` column in the sample information table, and `“none”` indicates the absence of any potential confounding batches.

```
$ check_batch_effects.py -incl not_pos_neg_control output/
interactions/all_lanes_filtered/
all_lanes_filtered_scaled_dev.dump.gz sample_table/
sample_table_filtered.txt name none
```

The output of the script is written to the directory: `<dir>/output/interactions/all_lanes_filtered/all_lanes_filtered_scaled_dev/name_effect_eval`.

19. Check for index tag-related batch effects (Figure 8b). Only positive control conditions are excluded from the analysis (`“-incl not_pos_control”`), as this allows us to assess if negative control conditions tagged with the same index tag are correlated with each other and contributing to the overall batch effect. However, estimates of same-index tag correlations can be artificially inflated if independent screens of the same compound were tagged with the same index tag. This is best avoided in advance during the design of the experiment, but the software can also mitigate this effect. Here, we retain only one instance of each compound for each index tag by specifying `‘name’` as a potentially confounding batch.

```
$ check_batch_effects.py -incl not_pos_control output/interactions/
```



```
all_lanes_filtered/all_lanes_filtered_scaled_dev.dump.gz
sample_table/sample_table_filtered.txt index_tag name
```

The output is written to: *output/interactions/all_lanes_filtered/all_lanes_filtered_scaled_dev/index_tag_effect_eval/*.

20. Check for lane-related batch effects (Figure 8c). As with the index tag effect correction, negative control conditions are included in the analysis but positive control conditions are not. Also, only one instance of each condition name replicate is retained for each lane.

```
$ check_batch_effects.py -incl not_pos_control output/interactions/
all_lanes_filtered/all_lanes_filtered_scaled_dev.dump.gz
sample_table/sample_table_filtered.txt lane name
```

The output is written to: *output/interactions/all_lanes_filtered/all_lanes_filtered_scaled_dev/lane_effect_eval/*.

21. Next, check the average correlation between non-replicate conditions (Figure 8a). For the example dataset, this correlation is 0.13, but it should theoretically be close to zero unless, for example, the data are from a library of functionally similar compounds. These correlations are explained by the signature that is present in 25% of mutants and ~20% of conditions and can be attenuated by removing the first SVD component. For general dataset processing, you will have to decide if this signature should be removed before or after any observed batch effects. Here, no batch effects are observed, so we proceed with SVD component removal. The following command creates a new “stacked matrix” dataset that contains 5 matrices, each possessing a version of the dataset with 0 to 4 SVD components removed:

```
$ svd_correction.py -incl not_pos_control output/interactions/
all_lanes_filtered/all_lanes_filtered_scaled_dev.dump.gz
sample_table/sample_table_filtered.txt 4 output/svd_correction
```

22. Generate clustered heat maps of each of the 0 to 4 SVD component-removed datasets (Figure 9, a–b).

```
$ visualize_stacked_zscore_matrices.py output/svd_correction/
svd_corrected_datasets.dump.gz svd_correction barcodes/
barcodes_filtered.txt sample_table/sample_table_filtered.txt
Gene_name,Strain_ID
screen_name,expt_id,name,lane,index_tag_plate,index_tag_well,index_
tag
```

The resulting clustered heat maps are in this directory: *output/svd_correction/CDTs*. Browse them using Java TreeView to become familiar with the outcome of SVD component removal.

23. Now, re-examine the replicate condition correlations on the data from which 0 to 4 SVD components have been removed (Figure 9, c–e).

```
$ check_batch_effects.py -incl not_pos_neg_control output/
svd_correction/svd_corrected_datasets.dump.gz sample_table/
sample_table_filtered.txt name none
```

The output is written to: *output/svd_correction/name_effect_eval/*. Notice that the largest increase in separation of the same-name and different-name condition correlations comes from removing the first SVD component and that the distribution of non-replicate correlations is now centered very close to zero (mean = 0.02).

24. Extract the dataset with 1 SVD component removed, generate a clustered heat map, and export it to text for further analyses outside the scope of this protocol.

```
$ extract_dataset.py output/svd_correction/
svd_corrected_datasets.dump.gz 1
$ visualize_dataset.py output/svd_correction/
svd_corrected_datasets/1_components_removed.dump.gz final_dataset
barcodes/barcodes_filtered.txt sample_table/
sample_table_filtered.txt Gene_name,Strain_ID
screen_name,expt_id,name,lane,index_tag_plate,index_tag_well,index_
tag
$ dataset_to_text.py output/svd_correction/svd_corrected_datasets/
1_components_removed.dump.gz
```

The final clustered heat map is located here: *output/svd_correction/svd_corrected_datasets/1_components_removed/final_dataset.cdt*. The final, text-formatted dataset is exported in the directory *output/svd_correction/svd_corrected_datasets/1_components_removed/* and is formatted as a matrix (*matrix.txt*) of interaction scores, a file (*strains.txt*) of row identifiers ('Strain_ID') matching the rows of the matrix, and a file (*conditions.txt*) of column identifiers ('screen_name' and 'expt_id') matching the columns of the matrix. Alternatively, you can use the `--table` flag for *dataset_to_text.py* to export the data to a single file in table format, with the option of using the `--value_name` flag to specify the name of the column that contains the interaction scores.

TIMING

Steps 1–4, setting up the working directory: < 5 min

Step 5, transferring the raw sequencing data: variable, allow 1–2 h

Step 6, initial round of interaction scoring: ~4 h (mostly parsing the fastq files, ~20 min per lane)

Step 7, reviewing fastq parsing reports: <10 min

Step 8, generating clustered heat maps: <5 min

Steps 9–14, visualizing the data and flagging mutants/conditions for removal: variable

Step 15, modifying the gene barcode and sample information tables: <5 min

Steps 16–17, reprocessing the dataset after removing mutants/conditions: <20 min

Steps 18–20, checking replicate correlations and batch effects: <1 min computational time for each step, variable time for browsing output

Steps 21–22, generating and visualizing SVD-corrected matrices: <5 min

Step 23, checking replicate correlations on SVD-corrected output: <2 min computational time, variable time for browsing output

Step 24, generating final clustered heat map and text-formatted output: <1 min

All timings were generated using an Intel Xeon E5–1620 CPU at 3.6 GHz, using 4 of 8 cores. The test dataset consists of 2592 conditions from 9 sequencing lanes (288 conditions per lane) subsampled from the full dataset of 38,400 conditions from 50 HiSeq sequencing lanes (768 conditions per lane), screened against a pool of 310 mutants¹⁰. Datasets with more conditions, more mutants in the pool, and/or higher sequencing depth will require more time to process. Processing time for the interaction scoring component (stages 2 and 6 of `process_screen.py`) does decrease with an increasing number of cores, although the speedup is not linear.

TROUBLESHOOTING

If BEAN-counter encounters an error, it will most often print out an informative message that will point you to the offending input. However, knowledge of the Python language, and the numpy and pandas libraries in particular, will be useful in situations where this is not the case. All scripts also accept verbosity arguments (either through the main configuration file for `process_screen.py` or as a command line flag for the other post-processing scripts) that specify the level of detail printed to the terminal while the code is executing. You are welcome to post questions, errors, and bug reports to the BEAN-counter Google Group (<http://groups.google.com/d/forum/BEAN-counter>) as well as post issues and/or submit pull

requests to the repository at <https://github.com/csbio/BEAN-counter>. Troubleshooting advice for individual steps can be found in Table 1.

ANTICIPATED RESULTS

Processing multiplexed, pooled interaction screening data with BEAN-counter involves, in addition to index tag and barcode counting with subsequent interaction scoring, various quality control procedures designed to maximize the quality of the data. The first of these quality control procedures is to ensure that the sequencing data were parsed successfully into counts of mutants by conditions. This would be shown by high percentages of index tags and barcodes matching expected sequences in the sample information and gene barcode tables, respectively, and by count distributions similar to those shown in Figure 6. Unsuccessful parsing may indicate an issue with the sequencing library or a mismatch between the BEAN-counter configuration files and the format of the screen.

Quality control continues to be a focus during interaction scoring, where the user must determine mutants and/or conditions to remove from the dataset based on the interaction scores they show. While some mutants and conditions are automatically removed if they do not pass pre-specified quality control criteria, others must be removed manually. For this, we visualize the data as a clustered heat map using Java TreeView to help identify the problem mutants and conditions (Figure 7). The most important interaction scores to focus on are those derived from negative control conditions, as 1) mutants that show many interactions in these conditions should not be trusted and 2) negative control conditions that possess many interactions may reflect experimental error (contamination, mislabeling, 180°-rotated plate) or batch effects. One of the important quality control steps to perform after interaction scoring is evaluating the correlations of biological replicate conditions (Figure 8a) to ensure that same-compound correlations can be differentiated from different-compound correlations.

The purpose of BEAN-counter's post-processing steps is the correction of systematic biases and other artifactual signals in the data, all of which obscure functional information present in the computed interactions. While we do not detect batch effects in all of our large datasets, we always check for their presence - especially for index tag and sequencing lane-related effects (Figure 8b–c). For the example dataset provided here, these effects do not exist at a detectable level, as evidenced by precision-recall curves that do not rise above the background expectation and receiver operating characteristic curves that do not deviate from the $y = x$ line. Importantly, even if batch effects are detected, removing them is only justified if it induces an improvement in either the PR and ROC curves. In addition to supervised batch effect removal, BEAN-counter also provides the ability to remove the largest sources of variation in an interaction dataset. We have found this useful for removing the functionally uninformative signal that is the predominant source of variation in most of our large datasets (Figures 4 and 9). The primary method for detecting the presence and evaluating the removal of this large, uninformative source of variation is to examine replicate profile correlations using histograms and PR/ROC curves, evaluating the ability of profile correlations to predict if two profiles were generated using the same compound.

BEAN-counter computes interaction z-scores that represent the difference between mutants' observed and expected abundances in units of standard deviations. For analysis of individual interactions (e.g. to identify mutants that confer resistance to a drug), we recommend a strict cutoff of ± 5 to focus on the mutants that have clearly deviated from their expected fitness values. We have found negative interactions to be the primary drivers of the strength and accuracy of our mode-of-action predictions and concluded that they contain the vast majority of the functional information from these interaction screens^{10,50}.

Importantly, the entire set of interactions for a perturbation across all mutants comprises an interaction profile, which is a high-dimensional, quantitative representation of gene (or chemical) function. Similarity between two interaction profiles implies that similar functions have been perturbed in the cell, a property you can use to identify, for example, novel compounds that have similar modes of action to previously-characterized compounds. To gain further insights into the cellular processes affected by your perturbation(s), you can perform a Gene Ontology enrichment analysis on the set of interacting mutants or use a tool such as CG-TARGET to predict the perturbed cellular bioprocesses using genetic interaction profiles as a functional reference (if the genetic interaction profiles are available for your species of interest)⁵⁰. Using CG-TARGET, we identified over 1500 chemical compounds with high-confidence mode-of-action predictions from our large-scale screen of nearly 14,000 compounds, validated subsets of these predictions in orthogonal assays, and used these predictions to make broader inferences about the cellular bioprocesses that are more or less easily perturbed by chemical compounds^{10,50}.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

ACKNOWLEDGEMENTS

SWS would like to thank B. VanderSluis for proofreading the manuscript and testing the software and also A. Becker at the University of Minnesota Genomics Center for discussions regarding amplicon sequencing issues. This work was supported by RIKEN (<http://www.riken.jp/en/>) Strategic Programs for R&D, the National Institutes of Health (<https://www.nih.gov/>) (R01HG005084, R01GM104975), and the National Science Foundation (<https://www.nsf.gov/>) (DBI 0953881). SWS was supported by an NSF Graduate Research Fellowship (00039202), an NIH Biotechnology training grant (T32GM008347), and a one-year fellowship from the University of Minnesota Bioinformatics and Computational Biology (BICB) Graduate Program (<https://r.umn.edu/academics-research/graduate-programs/bicb>). SCL and JSP were supported by a RIKEN Foreign Postdoctoral Research Fellowship. SCL was supported by a RIKEN CSRS (<http://www.csrs.riken.jp/en/>) Research Topics for Cooperative Projects Award (201601100228), and a RIKEN FY2017 Incentive Research Projects Grant. HNW was supported by a one-year BICB fellowship from the University of Minnesota. CB was supported by JSPS KAKENHI (<https://www.jspis.go.jp/english/e-grants/>) grant number 15H04483. CLM and CB are fellows in the Canadian Institute for Advanced Research (CIFAR, <https://www.cifar.ca/>) Genetic Networks Program. Computing resources and data storage services were partially provided by the Minnesota Supercomputing Institute and the UMN Office of Information Technology, respectively. Software licensing services were provided by the UMN Office for Technology Commercialization. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

REFERENCES

1. Giaever G et al. Genomic profiling of drug sensitivities via induced haploinsufficiency. *Nat. Genet* 21, 278–283 (1999). [PubMed: 10080179]
2. Parsons AB et al. Integration of chemical-genetic and genetic interaction data links bioactive compounds to cellular target pathways. *Nat. Biotechnol* 22, 62–69 (2004). [PubMed: 14661025]

3. Parsons AB et al. Exploring the mode-of-action of bioactive compounds by chemical-genetic profiling in yeast. *Cell* 126, 611–625 (2006). [PubMed: 16901791]
4. Pierce SE, Davis RW, Nislow C & Giaever G Genome-wide analysis of barcoded *Saccharomyces cerevisiae* gene-deletion mutants in pooled cultures. *Nat. Protoc* 2, 2958–2974 (2007). [PubMed: 18007632]
5. Costanzo M et al. The genetic landscape of a cell. *Science* 327, 425–431 (2010). [PubMed: 20093466]
6. Costanzo M et al. A global genetic interaction network maps a wiring diagram of cellular function. *Science* 353, aaf1420 (2016). [PubMed: 27708008]
7. Hoepfner D et al. High-resolution chemical dissection of a model eukaryote reveals targets, pathways and gene functions. *Microbiol. Res* 169, 107–120 (2014). [PubMed: 24360837]
8. Lee AY et al. Mapping the cellular response to small molecules using chemogenomic fitness signatures. *Science* 344, 208–211 (2014). [PubMed: 24723613]
9. Estoppey D et al. Identification of a novel NAMPT inhibitor by CRISPR/Cas9 chemogenomic profiling in mammalian cells. *Sci. Rep* 7, 42728 (2017). [PubMed: 28205648]
10. Piotrowski JS et al. Functional annotation of chemical libraries across diverse biological processes. *Nat. Chem. Biol* 13, 982–993 (2017). [PubMed: 28759014]
11. Roguev A et al. Conservation and rewiring of functional modules revealed by an epistasis map in fission yeast. *Science* 322, 405–410 (2008). [PubMed: 18818364]
12. Ryan CJ et al. Hierarchical modularity and the evolution of genetic interactomes across species. *Mol. Cell* 46, 691–704 (2012). [PubMed: 22681890]
13. Frost A et al. Functional repurposing revealed by comparing *S. pombe* and *S. cerevisiae* genetic interactions. *Cell* 149, 1339–1352 (2012). [PubMed: 22682253]
14. Vizeacoumar FJ et al. A negative genetic interaction map in isogenic cancer cell lines reveals cancer cell vulnerabilities. *Mol. Syst. Biol* 9, 696 (2013). [PubMed: 24104479]
15. Babu M et al. Quantitative genome-wide genetic interaction screens reveal global epistatic relationships of protein complexes in *Escherichia coli*. *PLoS Genet* 10, e1004120 (2014). [PubMed: 24586182]
16. Hart T et al. High-Resolution CRISPR Screens Reveal Fitness Genes and Genotype-Specific Cancer Liabilities. *Cell* 163, 1515–1526 (2015). [PubMed: 26627737]
17. Hillenmeyer ME et al. The chemical genomic portrait of yeast: uncovering a phenotype for all genes. *Science* 320, 362–365 (2008). [PubMed: 18420932]
18. Wildenhain J et al. Prediction of Synergism from Chemical-Genetic Interactions by Machine Learning. *Cell Syst* 1, 383–395 (2015). [PubMed: 27136353]
19. Smith AM et al. Quantitative phenotyping via deep barcode sequencing. *Genome Res* 19, 1836–1842 (2009). [PubMed: 19622793]
20. Smith AM et al. Highly-multiplexed barcode sequencing: an efficient method for parallel analysis of pooled samples. *Nucleic Acids Res* 38, e142 (2010). [PubMed: 20460461]
21. Cleveland WS Robust Locally Weighted Regression and Smoothing Scatterplots. *J. Am. Stat. Assoc* 74, 829 (1979).
22. Cleveland WS LOWESS: A Program for Smoothing Scatterplots by Robust Locally Weighted Regression. *Am. Stat* 35, 54 (1981).
23. Paquet YHY with contributions from A. & Dudoit S marray: Exploratory analysis for two-color spotted microarray data (2009).
24. Ritchie ME et al. limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Res* 43, e47 (2015). [PubMed: 25605792]
25. Piotrowski JS et al. Chemical genomic profiling via barcode sequencing to predict compound mode of action. *Methods Mol. Biol.* Clifton NJ 1263, 299–318 (2015).
26. Piotrowski JS et al. Plant-derived antifungal agent poacic acid targets β -1,3-glucan. *Proc. Natl. Acad. Sci. U. S. A* 112, E1490–1497 (2015). [PubMed: 25775513]
27. Baryshnikova A et al. Quantitative analysis of fitness and genetic interactions in yeast on a genome scale. *Nat. Methods* 7, 1017–1024 (2010). [PubMed: 21076421]

28. Morales EH et al. Accumulation of heme biosynthetic intermediates contributes to the antibacterial action of the metalloid tellurite. *Nat. Commun* 8, 15320 (2017). [PubMed: 28492282]
29. Giaever G & Nislow C The yeast deletion collection: a decade of functional genomics. *Genetics* 197, 451–465 (2014). [PubMed: 24939991]
30. Ho CH et al. A molecular barcoded yeast ORF library enables mode-of-action analysis of bioactive compounds. *Nat. Biotechnol* 27, 369–377 (2009). [PubMed: 19349972]
31. Ben-Aroya S et al. Toward a comprehensive temperature-sensitive mutant repository of the essential genes of *Saccharomyces cerevisiae*. *Mol. Cell* 30, 248–258 (2008). [PubMed: 18439903]
32. Spirek M et al. *S. pombe* genome deletion project: an update. *Cell Cycle Georget. Tex* 9, 2399–2402 (2010).
33. Baba T et al. Construction of *Escherichia coli* K-12 in-frame, single-gene knockout mutants: the Keio collection. *Mol. Syst. Biol* 2, 2006.0008 (2006).
34. Andrusiak K Adapting *S. cerevisiae* Chemical Genomics for Identifying the Modes of Action of Natural Compounds (University of Toronto, 2012).
35. Li W & Godzik A Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinforma. Oxf. Engl* 22, 1658–1659 (2006).
36. Bao E, Jiang T, Kaloshian I & Girke T SEED: efficient clustering of next-generation sequences. *Bioinforma. Oxf. Engl* 27, 2502–2509 (2011).
37. Shimizu K & Tsuda K SlideSort: all pairs similarity search for short reads. *Bioinforma. Oxf. Engl* 27, 464–470 (2011).
38. Mahé F, Rognes T, Quince C, de Vargas C & Dunthorn M Swarm: robust and fast clustering method for amplicon-based studies. *PeerJ* 2, e593 (2014). [PubMed: 25276506]
39. Zorita E, Cuscó P & Filion GJ Starcode: sequence clustering based on all-pairs search. *Bioinforma. Oxf. Engl* 31, 1913–1919 (2015).
40. Callahan BJ et al. DADA2: High-resolution sample inference from Illumina amplicon data. *Nat. Methods* 13, 581–583 (2016). [PubMed: 27214047]
41. Vetrovský T, Baldrian P, Morais D & Berger B SEED 2: a user-friendly platform for amplicon high-throughput sequencing data analyses. *Bioinforma. Oxf. Engl* (2018). 10.1093/bioinformatics/bty071
42. Zhao L, Liu Z, Levy SF & Wu S Bartender: a fast and accurate clustering algorithm to count barcode reads. *Bioinforma. Oxf. Engl* 34, 739–747 (2018).
43. Dai Z et al. edgeR: a versatile tool for the analysis of shRNA-seq and CRISPR-Cas9 genetic screens. *F1000Research* (2014). 10.12688/f1000research.3928.2
44. Mun J, Kim D-U, Hoe K-L & Kim S-Y Genome-wide functional analysis using the barcode sequence alignment and statistical analysis (Barcas) tool. *BMC Bioinformatics* 17, 159–167 (2016). [PubMed: 27067838]
45. Robinson DG, Chen W, Storey JD & Gresham D Design and analysis of Bar-seq experiments. *G3 Bethesda Md* 4, 11–18 (2014).
46. Johnson WE, Li C & Rabinovic A Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostat. Oxf. Engl* 8, 118–127 (2007).
47. Leek JT & Storey JD Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS Genet* 3, 1724–1735 (2007). [PubMed: 17907809]
48. Leek JT, Johnson WE, Parker HS, Jaffe AE & Storey JD The sva package for removing batch effects and other unwanted variation in high-throughput experiments. *Bioinforma. Oxf. Engl* 28, 882–883 (2012).
49. Levy SF et al. Quantitative evolutionary dynamics using high-resolution lineage tracking. *Nature* 519, 181–186 (2015). [PubMed: 25731169]
50. Simpkins SW et al. Predicting bioprocess targets of chemical compounds through integration of chemical-genetic and genetic interactions. *PLOS Computational Biology* 14, e1006532 (2018). 10.1371/journal.pcbi.1006532 [PubMed: 30376562]

Box 1.**Partitioning the data prior to interaction scoring**

In certain cases, there exists a viable alternative to performing batch correction in post-processing for correcting batch effects associated with large groups of conditions. Specifically, it involves specifying a column in the sample information table that partitions the dataset into arbitrary “sub-screen” groups (sub_screen_column in *config_file.yaml*), followed by scoring these groups separately for interactions and then recombining the data. This results in z-scores that more accurately reflect the variation in mutant abundances within each group’s respective negative control conditions.

We have found this special procedure useful in cases where samples can be partitioned by obvious differences in experimental conditions (e.g. inoculation date, length of incubation, number of PCR cycles, etc.) such that the negative control conditions in each group are the only appropriate reference for the remaining conditions in that group. Each group should possess at least 10, and ideally a full 96-well plate of, negative control conditions. One example is an optimization experiment in which mutant pools are grown for different lengths of time or the PCR amplicons are generated using different numbers of PCR cycles. While it is possible to process these partitions entirely separately using different BEAN-counter runs (indeed, this is what we did before implementing the partitioning procedure), we have found this procedure of specifying “sub-screens” to be much more efficient and convenient.

Specifying sub-screens has also been useful when screens performed days or weeks apart show obvious differences between their negative control profiles that propagate to all profiles within their respective groups. For example, the profiles from two experiments performed multiple weeks apart by the same person and processed in the same BEAN-counter run clustered almost entirely based on their inoculation date (Figure 5a). In contrast, defining sub-screens based on the inoculation date resulted in a dataset in which clustering between conditions inoculated on different dates was much more apparent (Figure 5b). A more quantitative evaluation based on precision-recall and receiver operating characteristic analyses also showed a clear change from near-perfect associations between profile similarity and batch to near-background levels of association (Figure 5c), demonstrating the utility of partitioning the data prior to interaction scoring in this case.

Figure 1. Overview of multiplexed barcode sequencing experiments and their processing using the BEAN-counter software.

(a) A collection of barcoded yeast mutants (denoted by color) is subjected to both treatment and negative control conditions, followed by PCR amplification of the genetic barcode sequences using indexed primers (indexing sequences indicated with black/gray specify different conditions) and ultimately, massively parallel sequencing. (b) The core of the BEAN-counter software is a pipeline to process raw sequencing reads into interaction z-scores, which are calculated by comparing each condition's mutant abundance profile against that of a mean profile derived from negative control conditions. Here, a positive chemical-genetic (CG) interaction score reflects a mutant's resistance to a compound and is depicted with yellow in all heatmaps. Negative interaction scores reflect mutant sensitivities to compounds and are depicted with blue. BEAN-counter also provides additional post-processing tools to remove systematic effects and/or common yet uninformative signal typically observed in our pooled screening datasets.

Figure 2. Design of large-scale, pooled and multiplexed chemical-genetic interaction screens.

(a) A barcoded collection of mutants is pooled and then competitively grown in the presence of different chemical compounds. (b) Scheme for the generation of PCR amplicons from pooled competition experiments that enables a high degree of sample multiplexing (768-plex in our experiments). (c) Typical layout of positive and negative control conditions in each screening plate. (d). Per-flow-cell sequencing scheme to maximize coverage of index tags with negative control conditions. Each column represents the samples combined into each sequencing lane (labeled L1 – L8 for each lane in a HiSeq flow cell), and each row represents the samples amplified with a specific plate of 96 unique indexed primers (labeled P1 – P8; 768 unique indexed primers in total). The different configurations are optimal for different sizes of barcoded mutant collection. We achieved 768-plex in our large-scale chemical-genomic screen performed across ~300 diagnostic mutants¹⁰. A screen against a larger mutant collection, however, requires a decrease in sample multiplexing to ensure sufficient sequencing depth for each sample. A 384-plex scheme is preferable for collections of ~1000 mutants (such as a collection of yeast essential gene mutants), as is a 96-plex scheme for collections of ~4000 mutants (such as the entire yeast nonessential deletion collection).

Figure 3. Schematic of the steps involved in processing large-scale interaction screens using BEAN-counter.

(a) Individual stages of the `process_screen.py` script (Steps 6, 16, and 17 in PROCEDURE) for scoring interactions from raw sequencing data. Locations of important output files are shown in the right column within the `<dir>/output/` folder. (b) BEAN-counter provides post-processing tools to visualize and remove systematic biases and uninformative signal from the matrix of interaction z-scores, which is originally generated by `process_screen.py`. The user must determine the sequence of post-processing steps

based on the severity and removability of these unwanted signals. The software also includes tools to collapse profiles across replicate conditions and to export the data in text-based formats for browsing in Java TreeView or further analysis in other programming languages. We include at the bottom of the relevant boxes the steps in the PROCEDURE where each command is invoked.

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Figure 4. The large signature that we observe in and remove from most of our datasets. (a,b) Clustered heat map representations of chemical-genetic interaction profiles from our large screen of the RIKEN Natural Product Depository. Each pixel in the heat map represents an interaction z-score that reflects the deviation of the observed barcode abundance from that expected in a given condition. Rows and columns are clustered using hierarchical agglomerative clustering using average linkage and $1 - \text{Pearson's correlation coefficient}$ as the distance metric. (a) Chemical-genetic interaction profiles obtained directly after interaction scoring. (b) Chemical-genetic interaction profiles after removal of one SVD component. (c) Histogram of the data in (a) showing the mean profile correlation (Pearson's correlation coefficient) within each group of compound replicates ("same compound," mean group correlation = 0.78) or between each pair of compounds ("different compound," mean group correlation = 0.15). (d) Histogram of the data in (b) showing the mean profile correlation within each group of compound replicates (mean = 0.73) or between each pair of compounds (mean = 0.01). (e) Precision-recall and receiver operating characteristic curves for evaluating the ability of profile correlation to predict if two profiles were generated by the same compound, for 0 and 1 SVD component-removed datasets.

Figure 5. An inoculation date-related effect in one of our datasets.

(Box 1) **(a)** Chemical-genetic interaction profiles computed from data not partitioned into inoculation date-based groups. **(b)** Chemical-genetic interaction profiles computed on data partitioned into inoculation date-based groups. **(c)** Precision-recall and receiver operating characteristic analyses evaluating the ability of profile correlation to predict if two profiles were derived from inoculations performed on the same date, for interaction profiles computed from non-partitioned (“combined”) and inoculation date-partitioned (“per-date”) data.

Figure 6. Typical barcode and index tag abundance distributions.

Substantial deviations from these distributions could be the result of errors in experimental or computational procedures and should be investigated. **(a)** Distribution of reads across index tags. **(b)** Distribution of reads across genetic barcodes.

Figure 7. Manual examination of the dataset to flag mutants and conditions for removal. (a) Chemical-genetic interaction profiles before manual removal of conditions and mutants (generated in Java TreeView). Profiles for positive control conditions, conditions flagged for removal, and mutants flagged for removal are expanded for emphasis (MMS: methyl methanesulfonate). Mutants were flagged for removal from the dataset based on high variability of interaction signal (resulting in interactions with most negative control conditions) and undesired behavior in conditions of high growth inhibition (< 50% growth compared to negative control conditions). The 36 conditions flagged for manual removal from the dataset could be divided into three classes (from left to right in the heatmap inset): 1) treatment conditions that exhibit almost exclusively positive interactions; 2) negative control profiles that exhibit a common signal that is inconsistent compared to other negative control profiles; and 3) a combination of both negative control and treatment profiles that share a similar set of strong, negatively interacting strains. (b) Interaction profiles for all

negative experimental control conditions (DMSO), from both DMSO-only plates and compound-containing plates with four DMSO controls. (c) Chemical-genetic interaction profiles after manual removal of conditions and mutants.

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Figure 8. Analysis of same-compound, same-index tag, and same-lane correlations to detect the presence of batch effects and uninformative signal.

(a) Analysis of same-compound (replicate) profile correlations. The histogram shows the mean profile correlation (Pearson's correlation coefficient) within each group ("same compound," mean = 0.75) or between groups ("different compound," mean = 0.13) of compound replicates. The precision-recall and receiver operating characteristic (ROC) curves show the ability of compound replicate correlations to predict compound replicate status. (b) Analysis of same-index tag profile correlations. The histogram shows the mean profile correlation within each group ("same index tag," mean = 0.11) or between groups ("different index tag," mean = 0.06) of conditions amplified with the same indexed primer. The precision-recall and ROC curves show the ability of profile correlations to predict if two conditions were amplified with the same indexed primer. (c) Analysis of same-lane profile correlations. The histogram shows the mean profile correlation within each group ("same lane," mean = 0.11) or between groups ("different lane," mean = 0.05) of conditions sequenced in the same HiSeq lane. The precision-recall and ROC curves show the ability of profile correlations to predict if two conditions were sequenced in the same HiSeq lane. The format of the plots was modified slightly from the default BEAN-counter output.

Figure 9. Removal of large, uninformative signature via singular value decomposition (SVD).

(a) Chemical-genetic interaction profiles after the first SVD component was removed from the data. (b) Chemical-genetic interaction profiles after the first two SVD components were removed from the data. (c) Histogram showing the mean profile correlation within each group (mean = 0.71) or between groups (mean = 0.02) of compound replicates after removal of one SVD component. (d) Same as (c), but for two SVD components removed (within-group mean = 0.74, between-group mean = 0.01). (e) Precision-recall and receiver operating characteristic analyses of compound replicate correlations after removing 0 to 4 SVD components.

Table 1.

Troubleshooting table.

Step	Problem	Possible reason	Solution
7	Distributions of reads mapped to each index tag or barcode deviate substantially from those in Figure 6. Alternatively, among the reads that match the common primer sequence, less than 50% map to index tags and genetic barcodes.	Assuming the index tags and barcodes are correct, issues may have occurred with preparation of the sequencing library, particularly during the isolation of genomic DNA or the PCR reaction.	First, ensure the index tag and barcode sequences are correct. Re-check individual PCR reactions via a DNA gel and re-prepare the sequencing library if necessary.
10	In the clustered heat map, all of the positive control conditions from one plate are absent from their respective clusters and replaced by non-positive-control conditions.	The plate was physically rotated 180° at some point during the experiment or sequencing library preparation.	Change the condition names and the 'control?' status in the sample information table to reflect the flipped orientation of the plate in question and reprocess the data (<code>process_screen.py --start 2</code>).
10	In the clustered heat map, all of the positive control conditions from one plate exhibit weak interaction profiles compared to the other plates	There may have been an issue during the genomic DNA extraction, PCR amplification, or agarose gel purification for the samples in that plate. Also, the stock solutions for the positive control compounds may need to be remade.	Check the remaining conditions in the plate to see if their profiles can be trusted. If not, remove the entire plate from the dataset. Re-do the experiments or sequencing library preparation if desired.
11	When trying to view only the control conditions	This is a bug in Java TreeView. It is	Run the following commands to generate a clustered heat map containing only the control conditions:

Step	Problem	Possible reason	Solution
	using the "Summary Popup" feature, Java TreeView instead displays a clustered heat map containing all conditions.	reproducible on a per-heat map basis but not predictable.	<pre>\$ reduce_dataset.py --column 'control?' output/interactions/ all_lanes_filtered/all_lanes_filtered_scaled_dev.dump.gz sample_table/ sample_table.txt output/interactions/all_lanes_filtered/ all_lanes_filtered_scaled_dev_DMSO.dump.gz \$ visualize_dataset.py output/interactions/all_lanes_filtered/ all_lanes_filtered_scaled_dev_DMSO.dump.gz DMSO_controls barcodes/ barcodes.txt sample_table/sample_table.txt Gene_name,Strain_ID screen_name,expt_id,name,lane,index_tag_plate,index_tag_well,index_tag</pre> <p>The desired heat map is here: <i><dir>/output/interactions/all_lanes_filtered/all_lanes_filtered_scaled_dev_DMSO/</i></p>
12	Negative control conditions possess strong interaction signal.	Conditions are mislabeled, issues occurred during sequencing library preparation, or control conditions were contaminated with treatment conditions.	Check to see if the plate is flipped (see step 10 troubleshooting) and confirm condition labeling. If this does not fix the issue, remove the offending conditions or mutants from the dataset. If these conditions are part of a larger group (such as a screening plate), consider removing that entire group of conditions from the dataset. If necessary, re-prepare sequencing library or perform experiment again.
12	Negative control conditions cluster into distinct groups that are associated with sample batches.	This is a batch effect.	Score the interactions separately for each batch. First, specify the sample batch column of the sample information table as the value for the <u>sub_screen_column</u> parameter in <i>config_file.yaml</i> . Then, run <i>process_screen.py</i> with the <code>--start 2</code> flag.