Chapter 5

Identifying Genetic Dependencies in Cancer by Analyzing siRNA Screens in Tumor Cell Line Panels

James Campbell, Colm J. Ryan, and Christopher J. Lord

Abstract

Loss-of-function screening using RNA interference or CRISPR approaches can be used to identify genes that specific tumor cell lines depend upon for survival. By integrating the results from screens in multiple cell lines with molecular profiling data, it is possible to associate the dependence upon specific genes with particular molecular features (e.g., the mutation of a cancer driver gene, or transcriptional or proteomic signature). Here, using a panel of kinome-wide siRNA screens in osteosarcoma cell lines as an example, we describe a computational protocol for analyzing loss-of-function screens to identify genetic dependencies associated with particular molecular features. We describe the steps required to process the siRNA screen data, integrate the results with genotypic information to identify genetic dependencies, and finally the integration of protein-protein interaction data to interpret these dependencies.

Key words Cancer, siRNA screening, Synthetic lethality

1 Introduction

Recent large-scale sequencing projects and decades of small-scale studies have led to the identification of hundreds of "driver" genes in cancer—genes whose alteration through genetic or epigenetic means provides a growth or survival advantage for tumor cells [1, 2]. A key remaining challenge is to understand how these driver mutations alter cellular states to promote tumor progression and how this altered state may be exploited for the development of targeted therapeutics [3]. Identifying the set of genes that are required for growth in a given tumor cell line provides both an insight into the cellular state and suggests genes whose products may be targeted therapeutically. Toward this end, a number of laboratories have used loss-of-function screening to generate resources describing the genetic requirements of panels of tumor cell lines [4–11]. The majority of these resources use either

James Campbell and Colm J. Ryan contributed equally to this work.

Louise von Stechow (ed.), *Cancer Systems Biology: Methods and Protocols*, Methods in Molecular Biology, vol. 1711, https://doi.org/10.1007/978-1-4939-7493-1_5,

genome-scale shRNA screens carried out in a pooled format [6, 7, 10] or siRNA screens carried out in an arrayed format [4, 5, 11] to identify genetic dependencies. In the near future CRISPR-based approaches will likely be used for similar purposes, although to date the number of cell lines profiled by genome-wide CRISPR libraries remains small (e.g., five cell lines in [8]). Regardless of the experimental methodology used, the goal of loss-of-function screens is largely the same—the identification of genes required for growth in specific cancer cell lines. By integrating the results of these screens with genotypic data, it is possible to identify genes that appear specifically required for growth in the presence of a particular driver gene mutation. In some cases the driver gene mutation results in an increased dependency upon the gene itself, a phenomenon known as "oncogene addiction" [12]. Examples of this include an increased sensitivity of *ERBB2*-amplified breast cancer cell lines to siRNA reagents targeting *ERBB2* [4], and an increased sensitivity of *KRAS* mutant cell lines to shRNA reagents targeting *KRAS* [7]. More frequent are instances where the driver gene and the resulting dependency gene are different, often termed non-oncogene addictions or synthetic lethalities [12, 13]. Examples of non-oncogene addictions identified from loss-of-function screens include a dependence of *ARID1A* mutant cell lines upon the *ARID1A* paralog *ARID1B* [14], an increased sensitivity of *PTEN* mutant breast cancer cell lines to inhibition of the mitotic kinase *TTK* [4], and an increased sensitivity of *MYC* amplified breast cancer cell lines to inhibition of multiple spliceosome component coding genes [15]. Ultimately both oncogene addictions and synthetic lethalities identified in these screens may be exploited for the development of novel targeted therapeutics in cancer [13].

When these screens are analyzed, statistical approaches are used to identify significant associations between the mutation of a driver gene and an increased sensitivity to the inhibition of another gene. The interpretation of the resulting associations remains challenging—the statistical tests provide information on which genes are required in the presence of specific driver genes, but not the mechanistic explanation as to why these dependencies exist. Inspired by approaches initially developed for the interpretation of genetic interactions in yeast [16], we have recently used the integration of functional interaction networks to aid the interpretation of dependencies identified in loss-of-function screens in cancer cell lines [5]. For instance in *ERBB2*-amplified cell lines we see an increased dependency upon *ERBB2* itself and also the *ERBB2* protein-interaction partners *ERBB3* and *PIK3CA* [5]. This suggests that *ERBB2* amplified cell lines are frequently "addicted" to the functionality of *ERBB2*, the binding of *ERBB2* to its interaction partner *ERBB3*, and the function of the downstream effector *PIK3CA*.

Here, we describe a protocol for the analysis of loss-of-function screens in a panel of cancer cell lines. We use as example data a

recent kinome-wide siRNA screen performed in a panel of osteosarcoma cell lines [5]. Our analysis protocol involves three main steps:

1. The conversion of siRNA screening results into gene-sensitivity scores.
2. The integration of these sensitivity scores with genotypic data to identify statistical associations between driver genes and sensitivity to the inhibition of particular genes.
3. The integration of additional data such as protein-protein interactions to interpret these associations.

Only the first step is specific to arrayed siRNA screens—we have successfully applied the latter analysis scripts to data resulting from additional screen types (e.g., pooled shRNA screens) (Fig. 1).

2 Materials

2.1 Software (See Notes 1 and 2)

1. R (available from https://www.r-project.org/).
2. R-packages:
   (a) *Gplots* (*see* **Note 3**).
   (b) cellHTS2 (*see* **Note 4**).
3. Python programming language (available from https://www.python.org/).
4. git repository containing the statistical analyses, code, and data resources discussed in the text (*see* **Note 5**) https://github.com/GeneFunctionTeam/identifying-genetic-dependencies

2.2 Input Files

1. Plate files (txt) contain the output from a loss of function screen. These each comprise three tab-separated columns of data containing the plate number (numeric), well position (e.g., B07), and the response value for the cell (e.g., luminosity readout). See the CellHTS2 documentation for further information.
2. Plate file list. This file contains three tab-separated columns with a header row listing "Filename," "Plate," and "Replicate." Filenames correspond to each plate file. The plate column defines which plate in the plate configuration file the data correspond to. The replicates column defines, which replicate a plate represents. See the CellHTS2 documentation for further information.
3. Plate configuration file. The first line defines the number of wells in each plate (e.g., "Wells: 384"). The second line defines the number of plates in the library (e.g., "Plates: 3"). The third line is a header associated with the subsequent columns (e.g.,

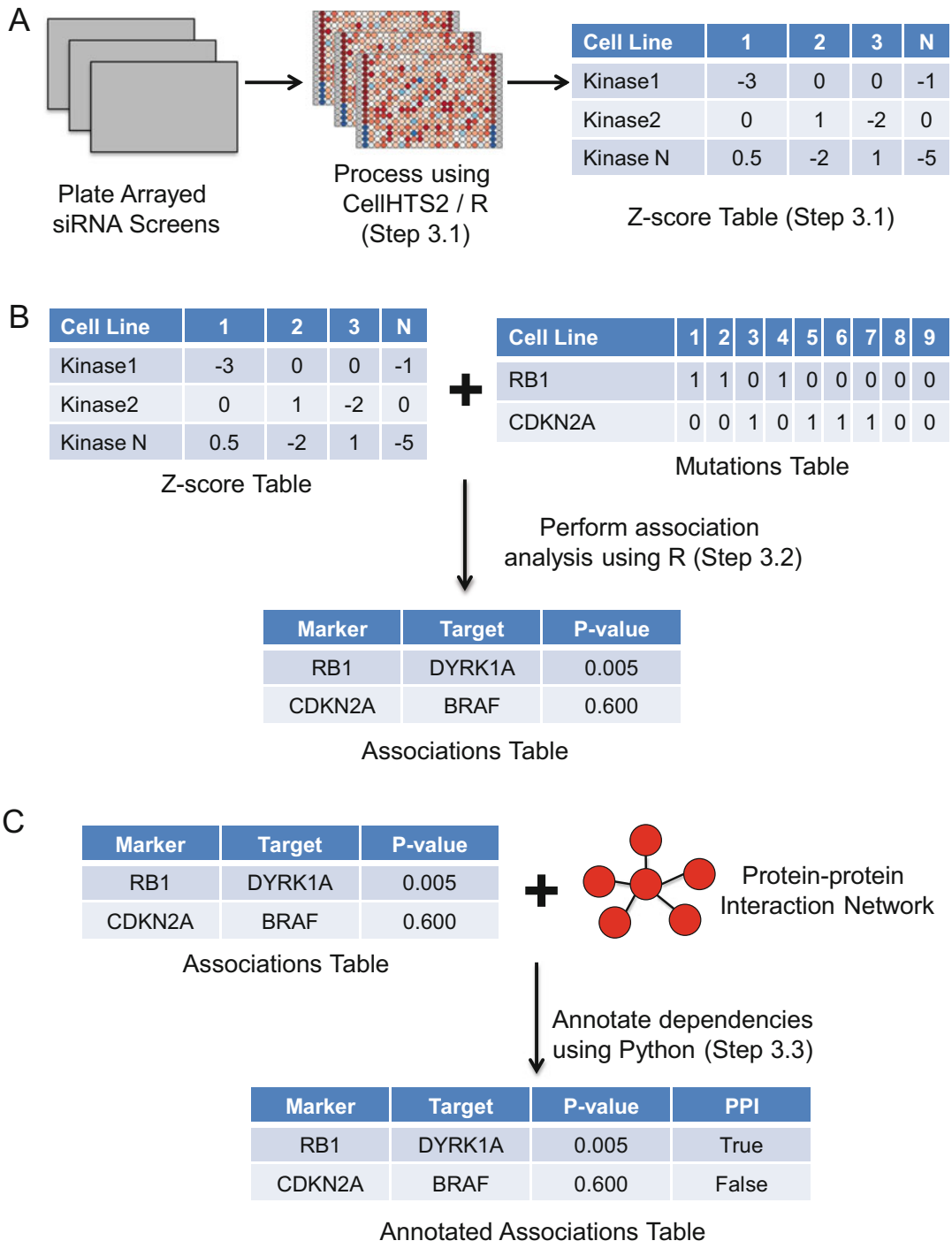

Fig. 1 Analyzing siRNA screens in Tumor Cell Line Panels. (**a**) Luminescence values derived from pooled siRNA screens are converted into *Z*-scores using CellHTS2 and custom R scripts. (**b**) *Z*-score profiles for each cell line are integrated with mutational profiles for the same set of cell lines using R. Custom R scripts are used to identify associations between the presence of particular mutations (e.g., in the RB1 gene) with increased

"Plate," "Well," "content"). The remaining lines define the wells containing samples and controls. An asterisk character (*) can be used to mean "all plates or wells." E.g., "* * sample" indicates that the all plates and all wells are "sample" unless otherwise stated. Subsequent more specific lines update the contents of other wells. E.g. "* [A-P]01 empty" indicates that on all plates (*), every row ([A-P]) of the first column (01) is marked as "empty." When defining wells as containing controls, ensure the case of the text used matches that used elsewhere. For further details on the plate configuration file, see the cellHTS2 documentation.

4. Plate annotation file. Contains at least three columns with a header. The first two columns list the plate and well IDs used in the library. The third and subsequent columns list annotations (such as the ID of the gene targeted by an siRNA). For further details on the plate configuration file, see the cellHTS2 documentation.
5. File containing functional relationships between genes (*see* **Note 6**).

3 Methods

3.1 Processing siRNA Screen Data Using CellHTS2

Typically, siRNA screens are conducted in multiwell tissue culture plates. The process of transfecting a cancer cell line with siRNAs is optimized prior to screening and once optimal conditions have been selected (described in [17]), cells are dispensed into multiwell plates containing growth media, transfection reagents, and siRNAs. The data in the example provided represent a screen of a single osteosarcoma tumor cell line using an siRNA library targeting 714 kinase and kinase-related genes. Positive and negative controls are included on each plate—typically non-targeting siRNA as a negative control and an siRNA pool targeting *PLK1* as a positive control. The full experimental protocol for this screen has been described elsewhere [4, 5]. Briefly, following siRNA transfection, the cells were cultured for 5 days, after which a luminescence assay measuring cellular ATP was used to estimate cell viability. A Victor X5 platereader was used to read luminescence values, resulting in data files in Microsoft Excel format. Prior to the analysis in R, these data files were converted to plain text *plate files*. Each plate file contains the luminescence reading from each well in one 96 or

Fig. 1 (continued) sensitivity to siRNAs targeting specific genes (e.g., DYRK1A). (**c**) The associations table is integrated with a data file describing known protein-protein interactions using Python. This results in a table of annotated dependencies—indicating whether a given association occurs between a pair of genes whose protein products are known to physically interact

384 multiwell plate. Where an siRNA library is larger than the plate format used in the screen, several plates are required for a single screen. Additionally, multiple replicate screens are typically conducted for a given cell line and siRNA library. The organization of plates into segments of an siRNA library and replicate screens is described in a *plate list file*. A plate list file contains the file names of the plate files, the replicate numbers, and plate numbers in a multiplate screen. Annotations indicating the genes targeted by siRNAs in the library across multiple plates as well as the positions of control wells are provided in separate plain text files. The analysis protocol set out below uses the cellHTS2 [18] R package developed by Huber and Boutros to combine data from the plate files, the plate list file, the plate configuration file, and the annotation file. The luminescence data are normalized to produce *Z*-scores by first $\log_2$ transforming the values and subtracting the median log luminescence value on a plate-by-plate basis. The plate-centered data are then scaled to the median absolute deviation (MAD) value calculated across the entire siRNA library to produce *Z*-scores.

An R script named "run_cellHTS.R" in the R-scripts directory contains the following commands. The first command loads the cellHTS2 R package that provides the functions required for the analysis.

```
require(cellHTS2)
```

With cellHTS2 loaded, we then use the readPlateList() function to read the plate list file which in turn creates a cellHTS object containing the luminescence data from the plate files (*see* **Note 7**).

```
x <- readPlateList(
    filename=" platelist_p3r3.txt",
    name="CGDsExample"
    path="./"
    )
```

We next use the configure() function to add information from the plate configuration file and (optionally) the screen log and description file to the cellHTS object. The plate configuration defines the locations of samples, controls and empty wells.

```
x <- configure(
    x,
    descripFile="screen_description.txt",
    confFile="plateconf_384.txt",
    logFile="Screenlog.txt",
    path="./"
    )
```

We use the annotate() function to define the genes targeted by siRNAs in each well of the plate. This information is located in the "kinome_library.txt" file.

```
x <- annotate(
    x,
    geneIDFile="kinome_library.txt",
    path="./"
    )
```

We now process the luminescence data in the cellHTS object to normalize data values across the plates in the screen. This is done by $\log_2$ transforming the luminescence values and subtracting the median value within a plate from all the values of wells in that plate. The parameters passed to the normalizePlates() function are described in **Note 8**. The original cellHTS object "x" is passed to the normalizePlates() function and the result is saved into a new cellHTS object called "xn."

```
xn <- normalizePlates(
    x,
    scale="multiplicative",
    log=TRUE,
    method="median",
    varianceAdjust = "none",
    negControls="neg",
    posControls="pos"
    )
```

The normalized data stored in "xn" are then scaled by dividing each well's value by the median absolute deviation (MAD) calculated from the normalized values across the whole siRNA library. Control wells are excluded from the estimation of the MAD. Scaling the plate median centered normalized data by the MAD produces the robust equivalent of Studentized values or *Z*-scores (*see* **Note 9**).

```
xsc <- scoreReplicates(
    xn,
    method="zscore",
    sign="+"
    )
```

For later statistical analyses, it may be preferable to summarize the values of replicate wells targeting a specific gene as a median or some other summary statistic. This can be performed using the summarizeReplicates() function in cellHTS2.

```
xsc <- summarizeReplicates(
    xsc,
    summary="median"
    )
```

CellHTS2 also provides a function called getTopTable() that writes a plain text file containing the well annotation data as well as the luminescence data at each stage of processing. Here, we write this information to a file called "TopTable.txt."

```
summary_info <- getTopTable(
    list(
        "raw"=x,
        "normalized"=xn,
        "scored"=xsc
        ),
    file="TopTable.txt"
    )
```

An HTML formatted report can also be generated describing the screen and the processing steps applied to it using the commands below. This HTML report provides information on the positive and negative controls included, the distribution of the resulting scores, and details of the quality of the screen (Z' scores, see below).

The contents of the HTML report can be modified using the setSettings() function. Here, we turn on the reproducibility and intensities reports (producing heatmap visualizations of well values) and set the range of heatmap colors for the screen summary scores report.

```
setSettings(
    list(
        plateList=list(
            reproducibility=list(
                include=TRUE,
                map=TRUE
                ),
            intensities=list(
                include=TRUE,
                map=TRUE)
                ),
        screenSummary=list(
            scores=list(
                range=c(-20, 10),
                map=TRUE
                )
            )
        )
    )
```

We then use the writeReport() function to generate the HTML report.

```
writeReport(
    raw=x,
    normalized=xn,
    scored=xsc,
    outdir=./report,
    force=TRUE,
    posControls="pos",
    negControls="neg",
    mainScriptFile="../R-scripts/run_cellHTS.R"
    )
```

The outputs from this cellHTS2 analysis so far have been a TopTable plain text file and a folder containing an HTML report. It is possible to extract any data in the cellHTS objects using accessor methods in order to produce customized outputs. Here, we extract information on the targeted genes, the plate numbers, well numbers, and median *Z*-scores and combine this into a data frame ("combinedz") containing four columns (compound, plate, well, and zscore).

```
genes <- geneAnno(xsc)
plates <-plate(xsc)
wells <- well(xsc)
scores <- Data(xsc)[,1,1]
combinedz <- data.frame(
    compound=compounds,
    plate=plates,
    well=wells,
    zscore=scores
    )
```

We can then write out the "combined" data frame to a text file. A use case for this is to enable joining data from multiple screens into a single file for analysis.

```
write.table(
    combinedz,
    "zscore.txt",
    sep="\t",
    quote=FALSE,
    row.names=FALSE
    )
```

This analysis needs to be performed for each screen in the experiment. Typically, multiple distinct screens would represent

multiple tumor cell lines screened with a specific library of siRNAs. Quality control steps need to be applied on a screen-by-screen basis. We expect siRNA screen replicates to be strongly correlated and reject screens where no pairs of replicates have a correlation coefficient greater than 0.7 (*see* **Note 10**).

In an earlier step, we saved the output from the getTopTable() function to a data frame called "summary_info." We can extract the replicate normalized luminescence values from this data frame and calculate the Pearson correlation coefficients for each pair of replicates using the following command.

```
cor(
    summary_info[,c(
        "normalized_r1_ch1",
        "normalized_r2_ch1",
        "normalized_r3_ch1"
        )],
    use="pairwise.complete.obs"
    )
```

A further quality control step that is recommended is to examine the *Z*-prime (Z) values for each screen [19]. Z scores provide a measure of the separation of the positive and negative control siRNAs included in a screen and so can be considered an estimate of how much it is possible for the individual "sample" wells to vary in *Z*-scores. Larger values of Z indicate better screens. Screens with Z values ≥ 0.5 are considered excellent. Those with Z values ≤ 0 are considered unusable and should be rejected and the experiments should be repeated. CellHTS2 calculates Z scores for each replicate and these can be found in the HTML report under the "plate summaries" section.

3.2 Identification of Kinase Dependencies Associated with Driver Gene Mutation or Copy Number Alteration

We next integrate the processed results from multiple siRNA screens with data describing the genetic alterations present in each sample. For this tutorial we use the siRNA data from 18 osteosarcoma tumor cell lines and a mutations file that describes the presence or absence of genetic alterations in different members of the Retinoblastoma (RB1) pathway. In the git repository downloaded, there is a set of directories containing pre-formatted siRNA and mutation datasets as well as R scripts to process the data. Open the script R-scripts/identifying_CGDs_RB1_osteosarcoma.R and examine its contents. The first command sets the working directory to the top level of the git repository we cloned/downloaded earlier. Modification of the path given to the setwd() function is required to point to the appropriate location on your local system.

```
setwd("~/software/identifying-genetic-dependencies")
```

The next command runs R code contained in a second file in the R-scripts directory. The dot at the beginning of the path indicates that the path is relative to the current working directory. The file "identifying_CGDs_library.R" contains a set of functions that abstract the process of loading mutation and siRNA datasets as well as running a set of statistical tests. Readers familiar with R can examine the code in this file to understand the individual analysis steps in more detail.

```
source("./R-scripts/identifying_CGDs_library.R")
```

We next define the paths to the siRNA and mutation data files used in the analysis. It is a helpful to define this kind of information near the top of scripts so that in the future the files can be changed without having to find the commands where these values are used.

```
sirna_screens_file <- "./siRNA-data/Osteosarcoma_kinome_sc-
reens.txt"
rb_pathway_func_muts_file <- "./mutation-data/combined_exo-
me_cnv_func_muts_RBpathway_160418.txt"
rb_pathway_all_muts_file <- "./mutation-data/combined_exo-
me_cnv_all_muts_RBpathway_160418.txt"
```

The next command reads the siRNA and mutation datasets, identifies cell lines in common between each dataset, and returns an R list object containing analysis-ready tables. The input files comprise tab-separated data where the first row and first column represent column and row names respectively. Aside from the first row (column headings), each row contains data for a single-cell line. Each column represents a property measured across each cell line. In the "sirna_screens_file," these properties are the *Z*-scores representing the relative viability of cells treated with siRNAs targeting specific genes. In the case of the mutation datasets (rb_pathway_-func_muts_file and rb_pathway_all_muts_file), these properties represent the presence or absence of a driver gene alteration. The file rb_pathway_func_muts_file contains a "1" where a cell line is considered to contain a likely functional cancer driver gene alteration (mutation or copy number alteration) and a "0" where such a change is absent. Similarly, the file rb_pathway_all_muts_file contains a "1" or "0" to indicate the presence of any driver gene alteration found in a cell line irrespective of presumed functional impact. These two files are used to identify sets of cell lines where a driver gene is considered to be functionally altered (the mutant group) or where alterations to the driver gene are entirely absent (the wild-type group) (*see* **Note 11**).

```
kinome_rb_muts <- read_rnai_mutations(
    rnai_file=sirna_screens_file,
```

```
    func_muts_file=rb_pathway_func_muts_file,
    all_muts_file=rb_pathway_all_muts_file
    )
```

With the siRNA and mutation data tables organized within kinome_rb_muts, we now run association tests between mutations or copy number alterations in RB1 pathway genes and test dependency on each gene targeted in the kinome siRNA library. The function run_univariate_tests() performs Wilcoxon Rank Sum tests between siRNA *Z*-scores of cell lines in the mutant and wild-type groups and returns a table of these test results as well as other information such as descriptive statistics (including the median *Z*-score of the mutant and wild-type group and the difference between those two values).

```
kinome_rb_mut_associations <- run_univariate_tests(
    zscores=kinome_rb_muts$rnai,
    mutations=kinome_rb_muts$func_muts,
    all_variants=kinome_rb_muts$all_muts,
    alt="less"
    )
```

We write out the results of the association tests to a text file that can be opened in a spreadsheet application or used as input for other programs such as the annotate_dependencies.py python program described in Subheading 3.3.

```
write.table(
    kinome_rb_mut_associations,
    "./results/kinome_rb_mut_associations.txt",
    sep="\t",
    col.names=TRUE,
    row.names=FALSE,
    quote=FALSE
    )
```

3.3 Annotating Molecular Dependencies According to Known Functional Relationships

In the absence of additional information, interpreting an association between the mutation of a driver gene and sensitivity to RNAi reagents targeting another gene can be difficult. One approach to aiding the interpretation of these associations is the integration of orthogonal data, including known functional relationships between genes or their protein products. We provide a simple Python script (annotate_dependencies.py) that can be used to integrate known functional relationships (e.g., protein-protein, kinase-substrate, or gene-regulatory interactions) with the associations generated by the R scripts described in Subheading 3.2. This script adds an additional column to the associations file indicating whether or

not the marker-target gene pair has a known functional relationship according to a user-supplied source of interactions.

1. Create a file containing functional relationships between genes (*see* **Note 6** for potential sources of these relationships). Each line of this file should contain two gene symbols (HUGO gene names) separated by a tab. Alternatively, files in the BioGRID Tab 2.0 Format, such as those downloaded from the BioGRID database [20], can be used as input.
2. Open a command prompt/terminal and run the script as follows:

```
python annotate_dependencies.py -a <associations> -o <output>
-i <interactions> -n <column_name>
```

where <associations> is the name of the associations file created using the R scripts above, <output> is the name of the file where the annotated associations will be output to, <interactions> is the name of the file containing known functional relationships, and <column_name> is an optional name for the column where the functional annotation will be stored. If the interactions file is in the BioGRID Tab 2.0 format then add the optional "–b" argument to this command. *See* **Note 12** for additional parameters of this file.

3. View the resulting output in a text editor or spread sheet application. There should be an additional column in the file named using the <column_name> argument, with True or False values indicating whether each marker-target association involves a gene pair with a known functional relationships according to the <interactions> file
4. Additional columns can be added (e.g., to annotate the associations according to a different source of interactions) by running the script again using the output file (<output>) of the first run as input to a subsequent run. For this step it is necessary to set the <column_name> parameter to avoid overwriting previous results.

The end result of this analysis is a file containing an annotated list of associations between a particular genomic feature (indicated in the "marker" column) and increased sensitivity to siRNA reagents targeting a particular gene (indicated in the "target" column). The column titled "PPI" in this file indicates whether the marker gene and the target have a known functional relationship (e.g., protein-protein interaction) while the column "wilcox.p" gives an indication of the statistical significance of the association. These *p*-values, together with the annotation of known functional relationships, may be used to prioritize candidate genetic dependencies (synthetic lethalities) for follow-up experiments. At a minimum these follow-up experiments should involve using orthogonal

means to test the observed association (e.g., alternative siRNA reagents or a small molecule targeting the protein product of the gene of interest) [21]. Ideally, the follow-up validation would test the association in additional cell lines harboring the mutation of interest. In the example provided, we found that *RB1* mutation is associated with increased sensitivity to siRNA targeting the kinase *DYRK1A*, a known *RB1* binding partner. In Campbell et al. [5] we validated this in a larger panel of osteosarcoma cell lines using four distinct siRNA reagents targeting the *DYRK1A* gene suggesting that the initial observation represents a real dependency.

4 Notes

1. All the analyses can be performed on a desktop computer. A recent version of the R statistical programming environment (available from https://www.r-project.org/) and the Python programming language (available from https://www.python.org/) are required. The Python scripts presented here have been tested with Python versions 2.7 and 3.4, while R scripts have been tested with version 3.2.5.
2. Note that we provide extensive code samples throughout this document. In these samples the tilde character (~) is used as a short cut to the user's home directory on Unix-like systems. On Microsoft Windows, the forward slash characters (/) separating the file paths will need to be substituted with back slashes (\).
3. *Gplots* is provided on the Comprehensive R Archive Network (CRAN) and can be installed by starting an R session and enter the following code:

```
install.packages(
"gplots",
dependencies=TRUE,
)
```

4. CellHTS2 [18] is an R package used to process RNAi screen data and can be installed using the following command:

```
source("https://bioconductor.org/biocLite.R")
biocLite("cellHTS2")
```

5. This repository can be downloaded as a zip file by navigating to the above URL and choosing "download ZIP." Alternatively install git (software available from https://git-scm.com), open

a console window, change directory to a suitable path that must exist (e.g., cd ~/software), and enter the following command:

```
git clone https://github.com/GeneFunctionTeam/identifying-genetic-dependencies
```

This command should create a new directory (e.g., ~/software/identifying-genetic-dependencies) containing data and scripts. The data files include a file containing viability data from an siRNA screen of osteosarcoma cell lines [5] and driver gene mutation datasets compiled from publicly available compendia of mutations in tumor cell lines [22].

6. BioGRID is a database of experimentally determined molecular interactions [20]. The web interface to BioGRID allows users to download the entire database in Tab 2.0 format, and also the interactions associated with a specific gene. An alternative source is PathwayCommons [23], which integrates protein-protein, gene-regulatory, kinase-substrate, and other molecular relationships. More specialized data sources include PhosphoSitePlus [24] (kinase-substrate relationships) and HINT (high-confidence protein-protein interactions) [25].
7. Detailed instructions on how to use cellHTS2 can be found in an R vignette titled "End-to-end analysis of cell-based screens." Once the cellHTS2 package is installed, the command 'browseVignettes("cellHTS2")' can be entered into the R console to reveal links to this and other relevant vignettes.
8. In our experience, luminescence values from multiwell siRNA screens tend to be positively skewed and show a log-normal distribution. It is thus preferable to log transform values prior to normalization. Setting the "log" argument of the "normalizePlates" function to "TRUE" and the "scale" argument to "multiplicative" instructs cellHTS2 to first log transform the luminescence values and then subtract the plate median values from each value on a plate.
9. *Z*-normalization in the classical sense refers to adjusting a set of normally distributed values such that they have a mean value of zero and a standard deviation equal to one. For idealized normally distributed *Z*-scores, 95% of the values are expected to fall between $Z = -2$ and $Z = +2$ and 99.1% of the values are expected to fall between $Z = -3$ and $Z = +3$. Log-transformed and plate-centered luminescence values from siRNA screens often have negatively skewed distributions that are not well described by statistics such as the mean and standard deviation. As an alternative to standard *Z*-score normalization we use robust *Z*-normalization where the median value is subtracted from all log-transformed plate-centered values and these values

are then divided by the median absolute deviation (MAD) of the distribution. This results in approximately 95% of the values falling between $Z = -2$ and $Z = +2$. Thus, siRNAs that produce a Z-score of <-2 (or more stringently, <-3) are interpreted as causing a decrease in viability.

10. At least two replicates are required for each screen in order to assess the overall reproducibility of the screen. We typically perform screens using three replicates and take the median value for each siRNA to further minimize noise.
11. Defining functional mutations in cancer driver genes can be difficult. In some cases (e.g., amplification of a gene such as *ERBB2*) the functional relevance of an alteration is well established. In many cases however, especially those involving missense mutations, the functional relevance of an alteration is uncertain. In [5] we developed a simple pipeline to classify mutations and copy number changes as either of likely functional relevance or of uncertain relevance [5]. For tumor suppressor genes we classify homozygous deletions, mutations predicted to cause a truncation (frame shift, nonsense, or splice site alteration) or missense mutations found to occur recurrently in tumors as functionally relevant. For oncogenes, we classify amplification events or recurrent missense mutations as functionally relevant. Mutations other than these are classified as of uncertain relevance and cell lines harboring these mutations are excluded from our association tests.
12. By default the "annotate_dependencies.py" script assumes that the interactions provided in the input file are undirected (i.e., the interaction (a, b) is the same as the interaction (b, a)). Using the argument "-d" changes this default behavior such that a directed network is utilized. This may be more appropriate for directed networks—e.g., for RB1 associated dependencies it may make sense to highlight associations between RB1 and genes that it regulates, but not associations involving genes that regulate RB1.

References

1. Davoli T et al (2013) Cumulative haploinsufficiency and triplosensitivity drive aneuploidy patterns and shape the cancer genome. Cell 155(4):948–962
2. Lawrence MS et al (2014) Discovery and saturation analysis of cancer genes across 21 tumour types. Nature 505(7484):495–501
3. Yaffe MB (2013) The scientific drunk and the lamppost: massive sequencing efforts in cancer discovery and treatment. Sci Signal 6(269):e13
4. Brough R et al (2011) Functional viability profiles of breast cancer. Cancer Discov 1 (3):260–273
5. Campbell J et al (2016) Large-scale profiling of kinase dependencies in cancer cell lines. Cell Rep 14(10):2490–2501
6. Cheung HW et al (2011) Systematic investigation of genetic vulnerabilities across cancer cell lines reveals lineage-specific dependencies in ovarian cancer. Proc Natl Acad Sci U S A 108 (30):12372–12377

7. Cowley GS et al (2014) Parallel genome-scale loss of function screens in 216 cancer cell lines for the identification of context-specific genetic dependencies. Sci Data 1:140035
8. Hart T et al (2015) High-resolution CRISPR screens reveal fitness genes and genotype-specific cancer liabilities. Cell 163 (6):1515–1526
9. Kim HS et al (2013) Systematic identification of molecular subtype-selective vulnerabilities in non-small-cell lung cancer. Cell 155 (3):552–566
10. Marcotte R et al (2016) Functional genomic landscape of human breast cancer drivers, vulnerabilities, and resistance. Cell 164 (1–2):293–309
11. Moser R et al (2014) Functional kinomics identifies candidate therapeutic targets in head and neck cancer. Clin Cancer Res 20 (16):4274–4288
12. Luo J, Solimini NL, Elledge SJ (2009) Principles of cancer therapy: oncogene and non-oncogene addiction. Cell 136 (5):823–837
13. Lord CJ, Tutt AN, Ashworth A (2015) Synthetic lethality and cancer therapy: lessons learned from the development of PARP inhibitors. Annu Rev Med 66:455–470
14. Helming KC et al (2014) ARID1B is a specific vulnerability in ARID1A-mutant cancers. Nat Med 20(3):251–254
15. Hsu TY et al (2015) The spliceosome is a therapeutic vulnerability in MYC-driven cancer. Nature 525(7569):384–388
16. Kelley R, Ideker T (2005) Systematic interpretation of genetic interactions using protein networks. Nat Biotechnol 23(5):561–566
17. Lord CJ et al (2008) A high-throughput RNA interference screen for DNA repair determinants of PARP inhibitor sensitivity. DNA Repair (Amst) 7(12):2010–2019
18. Boutros M, Bras LP, Huber W (2006) Analysis of cell-based RNAi screens. Genome Biol 7(7): R66
19. Zhang JH, Chung TD, Oldenburg KR (1999) A simple statistical parameter for use in evaluation and validation of high throughput screening assays. J Biomol Screen 4(2):67–73
20. Chatr-Aryamontri A et al (2015) The BioGRID interaction database: 2015 update. Nucleic Acids Res 43(Database issue): D470–D478
21. Jackson AL, Linsley PS (2010) Recognizing and avoiding siRNA off-target effects for target identification and therapeutic application. Nat Rev Drug Discov 9(1):57–67
22. Forbes SA et al (2015) COSMIC: exploring the world's knowledge of somatic mutations in human cancer. Nucleic Acids Res 43(Database issue):D805–D811
23. Cerami EG et al (2011) Pathway Commons, a web resource for biological pathway data. Nucleic Acids Res 39(Database issue): D685–D690
24. Hornbeck PV et al (2015) PhosphoSitePlus, 2014: mutations, PTMs and recalibrations. Nucleic Acids Res 43(Database issue): D512–D520
25. Das J, Yu H (2012) HINT: high-quality protein interactomes and their applications in understanding human disease. BMC Syst Biol 6:92