PRIORITIZATION OF POTENTIAL TARGETS IN A MASSIVE MULTIPLEX CRISPR SCREEN

Thesis

Presented in Partial Fulfillment of the Requirements for the Honors Distinction in Research in the field of Biomedical Engineering within the Undergraduate School of

The Ohio State University

By

Mukul Govande

B.S. in Biomedical Engineering

The Ohio State University

2021

Thesis Committee:

James Blachly, M.D., Advisor

Rosa Lapalombella, PhD.

Keith Gooch, PhD.

Abstract

Hematologic cancers, broadly categorized as leukemias, lymphomas, and myelomas, are diagnosed in 3.3 million people globally, and account for 10% of new cancer cases in the U.S. every year. In the last decade, newer treatments that directly target aberrant molecular processes, rather than the entire cell, have provided a less toxic alternative to traditional chemotherapy, however, identification of targets for molecular inhibitors is a major rate-limiting step. Recent studies have demonstrated the efficacy of utilizing Clustered Regularly Interspaced Short Palindromic Repeats (CRISPR) as a mechanism to conduct forward genetic screens and combining the results with existing drugs to significantly improve identification of probable therapeutics.

The following study explores an algorithm to process data from pooled CRISPR forward genetic screens to rapidly prioritize potential targets. Currently, targets of interest are manually curated by comparing pooled CRISPR data to existing gene, protein, and drug databases. This is a Sisyphean task as individual gene products vary among individuals, while gene pathway products are consistently similar. Thus, our algorithm will rapidly aggregate genes of interest for a pre-determined synergistic value (p-value, false discovery rate (FDR-value)), while simultaneously performing an efficient gene pathway level analysis by automatically comparing against known databases including pathway analysis, drug/protein interaction databases, and biomedical literature. The algorithm will quickly display the common gene pathways, the genes involved in respective pathways, their synergistic values in a prioritized fashion, and will include database results in an easy-to-read table to focus on potential targets for future research. The significantly reduced human labor in data processing due to the development of this algorithm will be critical in advancing research of novel drugs and improving the therapeutic outcomes for patients.

Table of Contents

List of Figures

List of Tables

1 Introduction

Hematologic cancers are broadly categorized as cancers that begin in blood-forming tissue such as the bone marrow, or in the cells of the immune system and are classified into three types: leukemias, lymphomas, and myelomas. Leukemia is identified by an uncharacteristically high production of leukocytes, a type of white blood cells, which suppresses the function of normal blood cells leading to anemia and other symptoms. Lymphoma is found primarily within the lymph nodes; these nodes are part of the lymphatic network that allow free movement of the immune system. Myeloma is defined by cancerous plasma cells, a variant of white blood cell, found within the bone marrow. After identification, the prognosis varies for every individual, however, these cancers are diagnosed in 3.3 million individuals, and account for 10% of all new cancer cases in the United States every year[1]. The prevalence of this disease does not guarantee a solution, or a cure. Most cancer treatment therapies are expensive and non-curative. As recently as 2014, the average yearly cost of orally administered cancer drugs was $135,000[2], and in 2017, a potential treatment to cure childhood leukemia cost $475,000 per patient[2]. The cost of cancer therapies has been monitored for a decade, and the continuous but steady rise is approaching an unprecedented precipice primarily due to targeted therapies supplanting traditional chemotherapeutics.

1.1 Therapeutics

Traditional chemotherapy was novel in mitigating the growth of cancer, but it is inherently flawed as it lacks specificity. This broad-spectrum approach creates a toxic environment within the body which limits proliferation of cancerous tissue, but causes numerous side effects such as hair loss, nausea, and weakness which have been extensively studied. To counteract the negative effects, significant research is being conducted into targeted therapies.

Targeted therapies are the foundation of precision medicine as the treatment regime is focused to interfere with specific molecules, and molecular processes[3] within cancer cells. The highly localized treatment ensures fewer side effects for the patient, however, physicians are limited in their selection of therapies, some of which include introducing monoclonal antibodies[3], targeting small molecule inhibitors[3], or targeting nuclear export inhibitors[4]. The identification of newer therapies is a major rate-limiting step; fortunately, the technological marvel of Clustered Regularly Interspaced Short Palindromic Repeats (CRISPR) can improve the identification of molecular targets advancing research on synthetic lethality, i.e. synergy, which arises through the cooperative effects of gene knockout and targeted drug delivery[5–7].

1.2 Identifying targets

CRISPR is the next step in advancing precision medicine for cancer treatment[8]. The rapid gene knockdown ability of CRISPR is unparalleled in specificity and allows deeper insight in identifying genes responsible for a particular phenotype of the organism[9]. The application of forward genetic screen is useful in predicting synthetic lethality[10], and is fundamentally different from reverse genetic screens where the phenotype of the organism is analyzed after the disruption of a known gene.

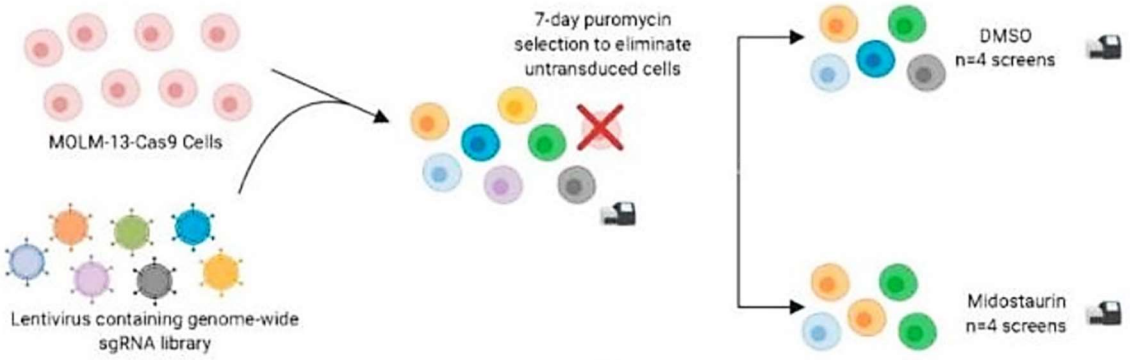

Figure 1: CRISPR knockout screen design. In this example midostaurin, a known chemotherapeutic, is used to perform the screen.[4]

The current process of identifying a lethal target is labor and time intensive with the following example building on the work conducted by our research lab in identifying XPO1 as a lethal target[4]. A cell culture must be infected with a virus containing genome-wide sgRNA library. The target result is a 1:1 ratio with one gene knocked down in every cell. In theory, this would create a cell culture where one gene is absent in every cell throughout the genome. Now, a simple drug is injected to screen-out any cells that did not get infected by the virus. The experiment then continues by splitting the cell culture into a control, and a drug condition. Both of these experiments yield significant results as shown in **Figure 1**. The DMSO, control condition, reveals which gene knockdowns are vital for survival, while the midostaurin, drug condition, reveals two results: which gene knockdown confers resistance to the drug for the proliferating cells, i.e. positively select for survival, and which gene knockdown confers synthetic lethality to the non-proliferating cells, i.e. negatively select for survival. The data from this experiment is vast and compiled into **Figure 2** for better representation. This large dataset is then manually parsed to

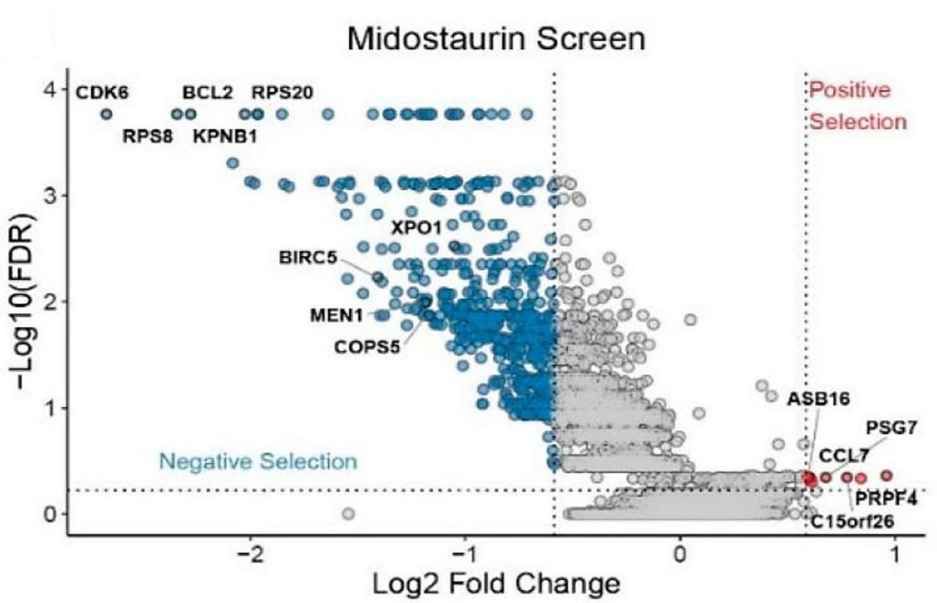

Figure 2: A Volcano Plot aggregating the results from the CRISPR knockout screen (Figure 1). Here, the significant positive and negative selection results are organized by their false discovery rate (FDR) and plotted on a log-log scale.[4]

identify genes with existing inhibitors, and the chosen targets are chosen for further experimentation to ensure consistency and significance of results.

While the utilization of CRISPR in screening for drug target discovery[11] is evident, it is a Sisyphean task to manually parse individual gene products as they could vary among individuals. It is much more important to compare gene pathway products as they are consistently similar among individuals in the same species. A sample gene pathway analysis is shown as **Figure 3**, and it is important to note the lack of prioritization of genetic pathways (y-axis), or the genes themselves (x-axis). This lack of prioritization places greater burden on the scientist and increases the time required to identify a viable therapeutic.

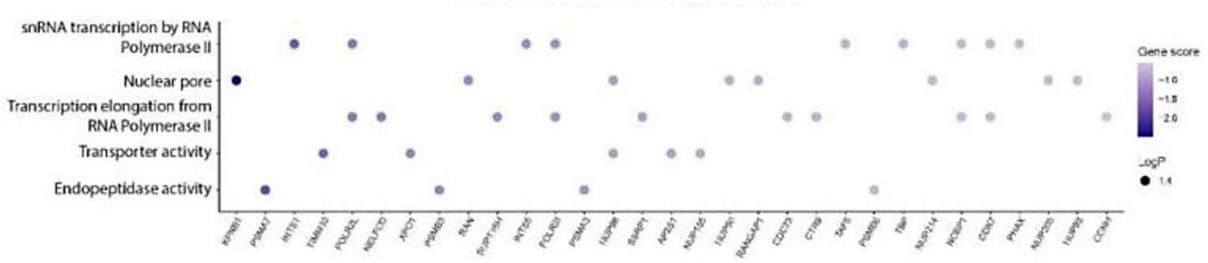

Figure 3: Enriched gene pathway analysis of the CRISPR knockout screen[4] (Figure 1). The pathways are on the y-axis, while individual genes are on the x-axis. The color of the dot represents he significance of the gene in the pathway (based on false discovery rates, FDR).

1.3 Goal

As **Figure 3** demonstrates a snippet of all the data aggregated for one drug condition, when research is conducted for multiple drug conditions, the gene pathway analysis results increase, and manually parsing the data becomes unfeasible. As such, it is important to construct a versatile algorithm to parse large datasets to find similarities among gene pathway analyses. These results must be prioritized to guide and improve research into synthetically lethal conditions using existing drugs, to mitigate cancer, and advance patient therapeutics.

2 Methods

In an effort to automate data analysis from the massive multiplex CRISPR screens, a MATLAB-based algorithm was constructed. The primary objective of the algorithm is to parse through the multiple gene pathway level analyses to identify common gene pathways and prioritize the results with respect to the false discovery rate (FDR). The FDR value is the expected proportion of type I errors, which succinctly indicates the probability of identifying a false positive. As such, a small FDR value would be better than a large FDR value, as the smaller FDR value would indicate a lower probability of identifying a false positive whereas the larger FDR value would indicate a higher probability of identifying a false positive. Thus, by prioritizing the output of the algorithm with respect to the FDR-value of the gene pathways and organizing the results in an efficient format would ensure the scientist is adequately prepared to continue their investigations.

2.1 Algorithm Pseudocode

```
Take user input for genetic pathway data of both conditions
Take user input for statistical genetic pathway threshold value
Loop through each row in genetic pathway data set
      Loop through each row in the other genetic pathway data set
            If row1.pathway = row2.pathway
                  Store pathway, statistical values, and gene number ids
            End if
      End loop
End loop
Loop through stored rows
      If either statistical value in row more than statistical threshold value
            Delete row
      End if
```

```
End loop
Take user input for data of number-to-name of both conditions
Loop through stored rows
      Loop through number-to-name data of condition 1
            If row.gene_number_id_1 = number-to-name.gene_number_id
                  Replace row.gene_number_id_1 with number-to-name.gene_name
            End if
      End loop
      Loop through number-to-name data of condition 2
            If row.gene_number_id_2 = number-to-name.gene_number_id
                  Replace row.gene_number_id_2 with number-to-name.gene_name
            End if
      End loop
End loop
Take user input for CRISPR-screened-gene data of both conditions
Take user input for statistical gene threshold value
Loop through stored rows
      Loop through CRISPR-screened-gene data set of condition 1
            If row.gene_name_1 = CRISPR-screened-gene.gene_name
                  Add CRISPR-screened-gene.statistical_value to stored row
            End if
      End loop
      Loop through CRISPR-screened-gene data set of condition 2
            If row.gene_name_2 = CRISPR-screened-gene.gene_name
                  Add CRISPR-screened-gene.statistical_value to stored row
            End if
      End loop
End loop
Loop through stored rows
```

```
      If gene statistical value in stored rows more than gene threshold value
            Delete gene and respective gene statistical value
      End if
End loop
Output stored rows as Excel Spreadsheet
```

2.2 Capabilities & User Interface

The MATLAB based algorithm is constructed in a modular format allowing for easy application and potential expansion. The user-friendly interface is supportive in guiding the user in determining the files necessary to operate the program. Additionally, there is inherent robustness integrated within the algorithm to ensure an output is always delivered. This simple capability is beneficial in determining the thresholding level and performing effective troubleshooting.

target	score	p-value	fdr
Gene 1	-1.3	0	0
Gene 2	-1.0	0	0.00067
Gene 3	-0.99	0	0.00067
Gene 4	-0.99	0	0.00075
Gene 5	-0.98	0	0.00075
Gene 6	-0.98	0	0.00075

Figure 4: The sample results from a massive multiplex CRISPR screen are shown in a text file format. The important columns include: "target" which refers to genes that exhibit negative-selection, and "FDR" (false discovery rate) which refers to a p-value score that is adjusted for Type I errors.

In the current version, the algorithm requires eight inputs which are summarized in **Table 1**. Two files from the CRISPR screen (**Figure 4**), and two files from the DAVID pathway analysis tool are mandatory to use the program. Following these required inputs, the algorithm asks for numeric values to threshold the significant results. The user can choose different values of

	A	B	C	D	E
1	Pathway	FDR of List 1	FDR of List 2	Genes from List 1	Genes from List 2
2	GO:xxxxxxx~Name of GOTerm	0.002	0.003	Gene 1, Gene 2, Gene 3, Gene , Gene	Gene 1, Gene 2, Gene 3, Gene 4
3	GO:xxxxxxx~Name of GOTerm	0.006	0.009	No CRISPR screened Genes available	Gene 1, Gene 2

Figure 5: A sample representation of the Non-Extended output format. The genetic pathways and individual genes are prioritized by their FDR-values, however, only the FDR-value of the genetic pathway is displayed. Additionally, the clustered format allows for a quicker representation of significant genes in a genetic pathway.

	A	B	C	D	E	F	G
1	**Pathway**	**FDR of List 1**	**FDR of List 2**	**Genes from List 1**	**FDR of Genes from List 1**	**Genes from List 2**	**FDR of Genes from List 2**
2	GO:xxxxxxx~Name of GOTerm	0.002	0.003	Gene 1	0	Gene 1	0
3				Gene 2	0.00067	Gene 2	0
4				Gene 3	0.00067	Gene 3	0
5				Gene 4	0.00067	Gene 4	0.00087
6				Gene 5	0.00075		
7				Gene 6	0.00075		
8							
9	GO:xxxxxxx~Name of GOTerm	0.006	0.009	No CRISPR screened	N/A	Gene 1	0.0098
10						Gene 2	0.012
11							

Figure 6: A sample representation of the Extended output format. The genetic pathways and individual genes are prioritized by their FDR-values. Additionally, cell D9 represents the condition when the thresholding value was too stringent.

thresholding for the pathways, and the genes. Finally, the user chooses the name of their output Excel spreadsheet, and their preferred formatting condition. The two conditions, extended and non-extended, change the amount of information displayed as the output. Both conditions prioritize the algorithm output based on the FDR-values but have slightly different benefits. The non-extended format is useful for quickly parsing output, while the extended format is beneficial in determining the next course of experimentation. **Figure 5** and **Figure 6** are examples of non-extended and extended output documents, respectively.

After ensuring valid inputs, the algorithm rapidly compares the DAVID pathway analysis files to determine overlapping pathways. The DAVID Bioinformatics Resource[12,13] provides a comprehensive set of functional annotation tools to identify enriched biological themes, discover enriched functional-related gene groups, and perform greater analysis of enriched pathways on large sets of genes. This tool is an open resource to obtain greater knowledge about enriched gene pathways, and it utilizes the CRISPR gene list to identify gene pathways and places the genes into their respective pathways. As such, when the algorithm parses the DAVID files for similarities, the pathway and corresponding genes within the pathways are extracted. Now, the results are prioritized with gene pathways and genes organized in ascending order with respect to their FDR-values. If the user indicated a threshold, it is applied at this step and if the threshold is too stringent,

the algorithm writes “No CRISPR screened Genes available at this threshold”. After performing the thresholding, the truncated results are shown in the desired output format.

Table 1: Algorithm Inputs

Name	Type	Description
CRISPR Gene List (x2)	File type - .txt	This file is the result of the multiplex CRISPR screen; must contain “target”, “FDR” values.
DAVID Pathway analysis (x2)	File type - .txt	This file contains the Pathway Analysis results from DAVID. It must contain “GO-Terms”, “FDR” values, and “Numeric IDs” for genes.
Renamed-Gene IDs (x2)	File type - .txt	This file contains the conversion of Gene IDs to Entrez-format Number IDs. It must contain “To” and “From” columns.
Pathway analysis threshold	Numeric - 0 to 1	Every pathway from DAVID has an FDR score which allows the user to compare significance.
Gene analysis threshold	Numeric - 0 to 1	Every gene from the CRISPR screen has an FDR score which allows the user to compare significance.
Output format type	Numeric - 0 or 1	Determines format of Excel spreadsheet. 0 corresponds to non-extended format, while 1 corresponds to extended format.
Output document name	String	This allows the user to name their Excel output file.

3 Results & Verification

A tool that is capable of rapid prioritization of potential targets after analyzing multiple genetic pathways is presented through this work. Prior to this tool, the multiplex CRISPR screen results from **Figure 2** were succinctly created into **Figure 3**, a pathway analysis diagram, through a laborious and an extensively manual process of comparing genes to known databases. Now, the algorithm can quickly eliminate erroneous genes (i.e. genes that do not qualify within the selected FDR-threshold value) by effectively, and efficiently parsing through the multiplex CRISPR screen data. The algorithm is also capable of comparing multiple genetic pathway analyses and prioritizing the common pathways in an easy-to-read Excel spreadsheet to holistically inform the scientist on the best potential targets to researcher further. Additionally this rapid pathway-level analysis will yield a greater number of potential targets for synthetic lethality, allowing scientists to focus their experiments to discover novel therapeutics and enhance precision medicine by targeting genetic pathways to improve patient outcomes.

The current version of the algorithm has also been verified to ensure results are consistent. The code was initially verified by comparing the number of overlapping pathways identified by the algorithm with the manual approach. The results demonstrated that the algorithm was able to identify a greater number of overlapping pathways much quicker, and further inspection revealed novel pathways previously missed by the manual approach. After this success, the algorithm performed an analysis of an experiment where a cell line infected with leukemia was treated with two drugs and was analyzed by a multiplex CRISPR screen. The algorithm identified a significant genetic pathway in “mRNA splicing via spliceosome”. This unique result was recently published and identified the same drug-pathway combination to treat acute myeloid leukemia[14]. This external

verification sheds light on the direct impact of this tool, and its efficacy in aiding scientists to develop unique therapies and improve patient outcomes.

4 Future Research

The algorithm is a powerful tool demonstrating an easy-to-use functionality, and a comprehensive output capability through an Excel spreadsheet prioritizing results from a massive multiplex CRISPR screen. This tool has already impacted the decision-making of scientists within the Experimental Hematology Research Lab, however, there are still avenues for improvement.

4.1 DAVID API

The current version of the algorithm relies on the online version of the DAVID Bioinformatics Resource[12,13] to perform the genetic pathway analysis for each massive multiplex CRISPR gene list. While this online resource is useful, it is time-consuming and error-prone to copy-paste the numerous gene lists into DAVID and save the output files in the specified format. In a future iteration, it will be beneficial to either download and locally host the DAVID Knowledgebase or utilize the DAVID API script files to link the algorithm to the DAVID Knowledgebase. It would be preferred to locally host the knowledgebase as it provides unlimited access to the entire database, whereas the API is restricted to a certain number of interactions. Additionally, as the entire database is clustered and centralized by a single index DAVID Gene identifier, locally hosting the database will enable the algorithm to perform extremely efficient and rapid data processing. The algorithm can be streamlined to require only CRISPR gene lists which would tremendously assist the scientist as the pathway analysis, and prioritization of potential targets will be automated and conducted rapidly from a single input file.

4.2 Screen-out Essential Gene

Essential genes are indispensable genes critical for an organism to grow and survive in a given environment. In some instances, targeting essential genes could be important for discovering novel treatments, while on the contrary excluding essential genes could give insight on determining novel cancer-specific gene targets. Currently, the results from the multiplex CRISPR screen (i.e. a gene list) are passed through the DAVID database to perform a comprehensive pathway-level analysis, as every pathway contains a minimum of one gene from the CRISPR screen. The current algorithm does not parse-out essential genes, and to improve the focus on cancer-specific targets, several database connections can be incorporated. The Database of Essential Genes (DEG)[15], and the Online Gene Essentiality database (OGEE)[16] are two databases which have previously been utilized to screen-out human essential genes. These databases have a script file that can be used to connect the algorithm to the online database which would drastically improve functionality of the algorithm. After incorporation, the algorithm should feature a toggle option to allow the scientist to choose whether to parse-out the essential genes.

4.3 Database Connections

The National Center for Biotechnology Information (NCBI) supported by the National Institutes of Health is a valuable resource as it hosts databases such as PubMed, PubChem, and Gene to Protein, which have established script files making them ideal candidates when expanding the algorithm. Every gene from the multiplex CRISPR screen can be passed sequentially through the databases to acquire significant information to inform the prioritization of the algorithm. For instance, a connection with PubMed will enable existing literature to be identified and curated with a focus on specific keywords and the gene of interest. Similarly, a connection with Gene to Protein will yield significant information about the specific proteins that are translated from the genes.

This specific protein information can also guide a future literature search and provide critical information on protein-level inhibitors. Finally, a connection with PubChem will enable a greater understanding of gene-level molecular inhibitors currently available as well as inhibitors currently in research. Beyond the inhibitors found on PubChem, the algorithm can also be connected to the short interfering RNA (siRNA) mod database[17] to identify non-drug inhibitors, as siRNA are a class of non-coding RNA molecules that interfere with gene expression. All of these curated results can be displayed as part of the extended-output format, and every result from the databases will be incorporated in calculating the new prioritization score to guide the scientist in performing novel research.

4.4 Prioritization Score

The improved prioritization score would follow an entirely novel format. In the current algorithm, pathways, and genes, are prioritized in ascending order based on the false discovery rate (FDR) value. As such, pathways, and genes, with a significance close to 0 are ranked higher on the output Excel spreadsheet. In the new and improved algorithm, prior to performing any comparisons, every gene from the multiplex CRISPR screen will be assigned a prioritization score as computed by Eq 1 below, and every pathway from the DAVID analysis will also be assigned a prioritization score as computed by Eq 2 below. Additionally, every result from the database (i.e. PubMed, PubChem, Gene to Protein, and siRNA mod) will be given a prioritization score of one. The database results have a non-changing score of one because existing literature could be sparse for certain genes and a score of one would prevent the database results from overweighing the prioritization score for a certain pathway. All of these prioritization scores can then be added and a threshold can be set to allow for greater specificity of results. Finally, a percentage-based scoring system can be displayed to highlight how much of the prioritization score stems from the individual

components (i.e. gene score, pathway score, and database score). In theory, a scientist can use the prioritization score percentage breakdown to identify if the gene of interest has significance, if the pathway of interest has significance, and determine if there is existing literature on the gene/pathway of interest. The prioritization score will aid the scientist in pursuing novel genes and ascertain improved therapeutics for patients.

Eq 1. $$\text{Gene Score} = 1 - \frac{\text{FDR of individual pathway}}{\text{highest value of FDR in pathway list}}$$

Eq 2. $$\text{Pathway Score} = 100 - \frac{\text{FDR of individual pathway}}{\text{highest value of FDR in pathway list}}$$

5 Conclusion

The time undertaken to successfully identify a potential target for synthetic lethality adds to the rising cost of novel targeted therapies. The laborious and extensive task of comparing gene lists manually is simplified and automated by the algorithm. Although additional functionalities can improve this tool as mentioned in the prior sections, however, it is important to note that this algorithm still lays the foundation by rapidly comparing genetic pathways, eliminating erroneous genes, highlighting a greater number of potential gene targets for synthetic lethality, and prioritizing the results in an easy-to-use Excel spreadsheet. The verified algorithm will reduce data processing time allowing scientists to focus on significantly important synthetically lethal targets which will advance research on novel drugs and improve therapeutic outcomes for patients.

6 References

1. Facts and Statistics | Leukemia and Lymphoma Society. https://llsorg.prod.acquia-sites.com/facts-and-statistics/facts-and-statistics-overview/facts-and-statistics.
2. Dolgin, E. Cancer's cost conundrum: The price trajectory of oncology drugs is unsustainable - But fixes are in the works. *Nature* **555**, S26–S29 (2018).
3. Padma, V. V. An overview of targeted cancer therapy. *BioMedicine (Netherlands)* vol. 5 1–6 (2015).
4. Brinton, L. T. *et al.* Cotargeting of XPO1 enhances the antileukemic activity of midostaurin and gilteritinib in acute myeloid leukemia. *Cancers (Basel).* **12**, 1–15 (2020).
5. O'Neil, N. J., Bailey, M. L. & Hieter, P. Synthetic lethality and cancer. *Nat. Rev. Genet.* **18**, 613–623 (2017).
6. Huang, A., Garraway, L. A., Ashworth, A. & Weber, B. Synthetic lethality as an engine for cancer drug target discovery. *Nat. Rev. Drug Discov.* **19**, 23–38 (2020).
7. Jost, M. & Weissman, J. S. CRISPR Approaches to Small Molecule Target Identification. *ACS Chemical Biology* vol. 13 366–375 (2018).
8. Xing, H. & Meng, L. hua. CRISPR-cas9: a powerful tool towards precision medicine in cancer treatment. *Acta Pharmacol. Sin.* **41**, 583–587 (2020).
9. Hanna, R. E. & Doench, J. G. Design and analysis of CRISPR–Cas experiments. *Nat. Biotechnol.* **38**, 813–823 (2020).
10. Sreedurgalakshmi, K., Srikar, R. & Rajkumari, R. CRISPR-Cas deployment in non-small cell lung cancer for target screening, validations, and discoveries. *Cancer Gene Ther.* (2020) doi:10.1038/s41417-020-00256-7.
11. Kurata, M., Yamamoto, K., Moriarity, B. S., Kitagawa, M. & Largaespada, D. A.

CRISPR/Cas9 library screening for drug target discovery. *J. Hum. Genet.* **63**, 179–186 (2018).

12. Huang, D. W., Sherman, B. T. & Lempicki, R. A. Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nat. Protoc.* **4**, 44–57 (2009).
13. Huang, D. W., Sherman, B. T. & Lempicki, R. A. Bioinformatics enrichment tools: Paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Res.* **37**, 1–13 (2009).
14. Radzisheuskaya, A. *et al.* PRMT5 methylome profiling uncovers a direct link to splicing regulation in acute myeloid leukemia. *Nat. Struct. Mol. Biol.* **26**, 999–1012 (2019).
15. Zhang, R., Ou, H. Y. & Zhang, C. T. DEG: A database of essential genes. *Nucleic Acids Res.* **32**, D271 (2004).
16. Gurumayum, S. *et al.* OGEE v3: Online GEne Essentiality database with increased coverage of organisms and human cell lines. *Nucleic Acids Res.* **49**, D998–D1003 (2021).
17. Dar, S. A., Thakur, A., Qureshi, A. & Kumar, M. SiRNAmod: A database of experimentally validated chemically modified siRNAs. *Sci. Rep.* **6**, (2016).

7 Appendix

7.1 Gene Pathway Analysis Code

```
%% Gene Pathway Analysis Code
% Author: Mukul Govande
% Acknowledgement: Andrew Lubinger.1 for assisting in very
early development
% Last modified: 03/28/2021

% This program takes in multiple pathway analyses, and
negative-selected
% genes from a massive multiplex CRISPR screen, and
provides a
% prioritization scheme to inform the user on the best path
forward.
%
% Inputs:
%       2x CRISPR Gene lists
%       2x Renamed GENE IDs (this is an output of DAVID
pathway analysis)
%       2x DAVID pathway analysis of gene lists
%       Threshold value for CRISPR Gene list
%       Threshold value for DAVID pathway analysis
%       Output Format type (extended vs. non-extended)
%
% Outputs:
%       Excel Data file with prioritized pathways and genes

% Functions used in this script file:
%   pathwaycomp

clc;
clear all;
close all;

%% INSTRUCTIONS
% Make sure the data files are present in the same folder
as the MATLAB files.
% If the data files are not in the same folder, copy the
entire pathway as
% the input when prompted to enter the file name in the
command window.
```

```
% You will need 2 files from your massive multiplex CRISPR
screen, 2 files
% from the DAVID pathway analysis, 2 files from the
renamed-gene IDs. Please
% ensure the files from CRISPR has "targets","fdr-values"
as column
% headers; ensure the files from DAVID has "go-terms","fdr-
values" as
% column headers; ensure the files from RENAMED_GENES has
"FROM","TO" as
% column headers.

% MAKE SURE ALL DATA FILES ARE IN TEXT FORMAT (i.e. they
have the .txt
% extension)

% CRISPR Screen results: A txt file created after analyzing
the massive
%multiplex screen.

% RENAMED Gene IDs: DAVID is a bioinformatics tool.
Currently, it is not
% integrated into this program. So refer to the online
% website, "david.ncifcrf.gov/home.jsp", copy the CRISPR
gene list and
% convert to ENTREZ-GENE-IDs. Store the result as this is
your renamed-gene
% IDs. This list will enable conversion from number-IDs to
name-IDs.

% DAVID Pathway Analysis: Proceed with pathway analysis of
the converted
% list. The output is a massive txt file containing GO-
Terms. This is the
% file used for pathway analysis.

%% Ask for inputs

%prompts
prompt_c1 = 'Please write the first CRISPR file pathway (or
file name, include extension!): \n';
prompt_c2 = 'Please write the second CRISPR file pathway
(or file name, include extension!): \n';
```

```
prompt_rn1 = 'Please write the first renamed-gene-number ID
file pathway (or file name, include extension!): \n';
prompt_rn2 = 'Please write the second renamed-gene-number
ID file pathway (or file name, include extension!): \n';
prompt_d1 = 'Please write the first DAVID file pathway (or
file name, include extension!): \n';
prompt_d2 = 'Please write the second DAVID file pathway (or
file name, include extension!): \n';
prompt_th_c = 'Determine the threshold for DAVID pathway
analysis (numeric): \n';
prompt_th_d = 'Determine the threshold for CRISPR analysis
(numeric): \n';
prompt_output = 'Which output format is preferred?
(extended = 1, non-extended = 0): \n';

%everything related to condition 1
crisp_1 = input(prompt_c1,'s');
david_1 = input(prompt_d1,'s');
rename_1 = input(prompt_rn1,'s');

%everything related to condition 2
crisp_2 = input(prompt_c2,'s');
david_2 = input(prompt_d2,'s');
rename_2 = input(prompt_rn2,'s');

%threshold values
threshold_c = input(prompt_th_c);
%make sure the values are between 0 and 1
while(threshold_c <= 0 || threshold_c >= 1)
    disp('ERROR. Please pick a value between 0 and 1.');
    threshold_c = input(prompt_th_c);
end

threshold_d = input(prompt_th_d);
%make sure the values are between 0 and 1
while(threshold_d <= 0 || threshold_d >= 1)
    disp('ERROR. Please pick a value between 0 and 1.');
    threshold_d = input(prompt_th_d);
end

%output format
output_type = input(prompt_output);
%make sure the values are either 0 or 1
while(true)
```

```
    if(output_type == 0)
    break;
    elseif(output_type == 1)
    break;
    else %i.e. it is not 1 or 0
        disp('ERROR. Please pick a value of 0 or 1.');
        output_type = input(prompt_output);
    end
end

%% Load in data files

% DAVID gene pathways analysis (non-thresholded, original
list)
david_cond_1 =
readtable(david_1,'Delimiter',{'\t'},'MultipleDelimsAsOne',
1);
david_cond_2 =
readtable(david_2,'Delimiter',{'\t'},'MultipleDelimsAsOne',
1);

% CRISPR - original files which contain raw data - i.e.
gene lists + score + p-value + fdr of individual genes
crisp_cond_1 =
readtable(crisp_1,'Delimiter',{'\t'},'MultipleDelimsAsOne',
1);
crisp_cond_2 =
readtable(crisp_2,'Delimiter',{'\t'},'MultipleDelimsAsOne',
1);

% RENAMED files (to convert from DAVID gene (numeric) IDs
to named genes (strings))
rename_cond_1 =
readtable(rename_1,'Delimiter',{'\t'},'MultipleDelimsAsOne'
,1);
rename_cond_2 =
readtable(rename_2,'Delimiter',{'\t'},'MultipleDelimsAsOne'
,1);

%clear some variables from workspace
clear p*; clear crisp_1; clear crisp_2; clear david_1;
clear david_2; clear rename_1; clear rename_2;

%% comparing gene pathways
```

```
% this performs the entire comparison and prioritization
% order matters: david_file_1, rename_file_1, david_file_2,
rename_file_2,
% david_pathway_threshold, crispr_file_1, crispr_file_2,
% crispr_screen_threshold

[comparison] =
pathwaycomp(david_cond_1,rename_cond_1,david_cond_2,rename_
cond_2,threshold_d,crisp_cond_1,crisp_cond_2,threshold_c,ou
tput_type);

%% write to excel file
%ask user for output file name
prompt_file = 'Name the output file (do not reuse names):
\n';
output_name = input(prompt_file,'s');
output_name = [output_name,'.xls'];

%write to the specified file name
writetable(comparison,output_name,'Sheet',1,'Range','A1:G65
536');

%% END
```

7.2 Function: pathwaycomp.m

```
function [storer] =
pathwaycomp(inputArg1,checker1,inputArg2,checker2,fdrthresh
old1,originalList1,originalList2,fdrthreshold2,value)
%PATHWAYCOMP: This function is the primary tool in
performing pathway
%analysis. The user parses several input arguments to
automatically compare
%multiple genetic pathway analyses. The function
prioritizes the results according to
%the FDR-values of the genetic pathways, and the individual
genes in those
%pathways. To operate correctly, and provide an effective
output, the functions reversename.m, changed_list_comp.m,
%and printer.m are utilized.
%  % Input Arguments:
   %    - inputArg1: a table containing the pathway
analysis from DAVID
```

```
   %    - inputArg2: a table containing the pathway
analysis from DAVID
   %    - checker1: a table containing the renamed gene
IDs, used for
   %    conversation between number IDs and name IDs
   %    - checker2: a table containing the renamed gene
IDs, used for
   %    conversation between number IDs and name IDs
   %    - fdrthreshold1: a value between 0 and 1 to ensure
pathway fdr
   %    value is significant
   %    - originalList1: a table containing the CRISPR
screen data (gene
   %    list)
   %    - originalList2: a table containing the CRISPR
screen data (gene
   %    list)
   %    - fdrthreshold2: a value between 0 and 1 to ensure
gene fdr value
   %    is significant
   %    - value: output format (value of 0 or 1 - binary)
   %
   % Output:
   %    - storer: a table containing the name of
overlapping Pathways, FDR
   %    values of Pathways, Overlapping Genes from List
1/List 2, FDR
   %    values of Gene List 1/ Gene List 2

%% Pathwaycomp Function

%isolate the pathways from the loaded table & compare them
term1 = inputArg1.Term;
term2 = inputArg2.Term;

%isolate the FDR-values and compare against threshold
(comparison happens later)
fdr1 = inputArg1.FDR;
fdr2 = inputArg2.FDR;

%isolate the gene IDs - this actually gives you a string,
not gene numbers!
gene1 = inputArg1.Genes;
gene2 = inputArg2.Genes;
```

```
%convert the string gene IDs into numeric gene IDs
for g = 1:length(gene1)
    holder = str2num(gene1{g}(gene1{g}~= ' '));
    gene1{g} = holder';
end
for g = 1:length(gene2)
    holder = str2num(gene2{g}(gene2{g}~= ' '));
    gene2{g} = holder';
end

%cell-counting variable
ind = 1;

%the for-loop to perform checking
for i = 1:length(term1)
    s1 = string(term1(i)); %assign ONE gene-pathway from
list ONE to a variable
    if(s1 == '') %this is the blank space checker -
prevents forever looping
        break;
    end
    for j = 1:length(term2)
        s2 = string(term2(j)); %assign ONE gene-pathway
from list TWO to a variable
        if(s2 == '') %this is the blank space checker -
prevents forever looping
            break;
        end
        if((s1 == s2) && (fdr1(i) <= fdrthreshold1) &&
(fdr2(j) <= fdrthreshold1)) %the comparison point of
pathways & fdr value
            %extended version
            temp_store{ind,1} = term1(i); %store the term
(gene pathway)
            temp_store{ind,2} = fdr1(i); %store the fdr-
value from list 1
            temp_store{ind,3} = fdr2(j); %store the fdr-
value from list 2
            temp_store_gene{ind,1} = gene1{i}; %temporary
storage for genes from overlapped pathways - this is for
list 1
```

```
            temp_store_gene{ind,2} = gene2{j}; %temporary
storage for gene from overlapped pathways - this is for
list 2

            %non-extended Version
            temp_store_nonExtended{ind,1} = term1(i);
%store the term (gene pathway)
            temp_store_nonExtended{ind,2} = fdr1(i); %store
the fdr-value from list 1
            temp_store_nonExtended{ind,3} = fdr2(j); %store
the fdr-value from list 2

            ind = 1 + ind; %increment the counting variable
        end
    end
end

clear ind; %clear the independent counting variable

%reversename function
% - inputs:
%           temp_store [variable that contains common
pathways]
%           checker1 [renamed gene IDs for list 1]
%           checker2 [renamed gene IDs for list 2]
% - outputs:
%           cell variable that is well-organized and shows
prioritized results

%changes the names from the number IDs to actual names
changed_name=
reversename(temp_store_gene,checker1,checker2);

%changed_list_comp function
% - inputs:
%           changed_name [output from reversename function]
%           originalList1 [CRISPR gene list for list 1]
%           originalList2 [CRISPR gene list for list 2]
%           fdrthreshold2 [fdr-thresholding value for
CRISPR gene lists]
% - outputs:
%           cell variable that is well-organized and shows
thresholded, and
%           prioritized results
```

```
%threshold the genes for specific values
result =
changed_list_comp(changed_name,originalList1,originalList2,
fdrthreshold2);

%add the named gene lists to temp_store
for i = 1:length(temp_store)
   %extended version
   temp_store{i,4} = result{i,1}; %gene list 1
   temp_store{i,5} = result{i,3}; %fdr list 1
   temp_store{i,6} = result{i,2}; %gene list 2
   temp_store{i,7} = result{i,4}; %fdr list 2

   %non-extended version
   temp_store_nonExtended{i,4} = result{i,1};
   temp_store_nonExtended{i,5} = result{i,2};
end

%printer function
% - inputs:
%           result [output from changed_list_comp function]
%   NOTE: column 1 & 3 are related, 2 & 4 are related.
Column 1, 2 are Gene
%   IDs, Column 3, 4 are fdr-values of Genes from Column 1,
2 respectively.
% - outputs:
%           cell-variable that is well-organized and
prioritized - ready to
%           be printed to Excel.

%print results in a NEAT format - default formats are not
appropriate
output = printer(temp_store);

%determine output format type
if(value == 1) %extended
    %returns a table with pathways, fdr of pathways, gene
ID names, fdr of gene lists
    storer = cell2table(output,'VariableNames',{'Term' 'FDR
w/ Condition 1' 'FDR w/ Condition 2' 'Gene List 1' 'FDR of
Gene List 1' 'Gene List 2' 'FDR of Gene List 2'});
else %non-extended
```

```
    %returns a table with pathways, fdr of pathways, and
gene ID names
    storer =
cell2table(temp_store_nonExtended,'VariableNames',{'Term'
'FDR w/ Condition 1' 'FDR w/ Condition 2' 'Gene List 1'
'Gene List 2'});
end

end %end of function file
```

7.3 Function: reversename.m

```
function [result] = reversename(temp,inputArg1,inputArg2)
%REVERSENAME: This function can convert the number IDs to
the name IDs.
%This is especially useful after the comparison of multiple
pathways yields
%a cell array that has number IDs. %This conversion is
critical as the
%output of this function is the input of the
changed_list_comp.m function.
%  % Input Arguments:
   %    - temp: a cell array containing the common pathways
& common genes
   %    on those pathways
   %    - inputArg1: a table containing the renamed gene
IDs, used for
   %    conversation between number IDs and name IDs
   %    - inputArg2: a table containing the renamed gene
IDs, used for
   %    conversation between number IDs and name IDs
   %
   % Output:
   %    - result: cell array that converts number IDs to
name IDs. Also
   %    organizes results in ascending order
(prioritization technique)

%isolate the individual gene number IDs from pathwaycomp
%isolate the 'TO' and 'FROM' column from DAVID renamed gene
IDs
to1 = inputArg1.To;
to2 = inputArg2.To;
from1 = inputArg1.From;
```

```
from2 = inputArg2.From;

%create the string array
storage1 = strings;

%perform comparison
%list1 comparison
for i = 1:length(temp)
    storage1 = strings; %initialize the string array
    ind = 1; %independent counter variable
    storage1(ind) = "Number ID is not available"; %failsafe
result
    for j = 1:length(temp{i,1})
        s1 = temp{i,1}(j);
        for k = 1:length(to1)
            s2 = to1(k);
            if(s2 == s1) %this comparison checks if the
numbers from DAVID match - i.e. checks if gene IDs match
                storage1(ind) = from1(k); %store result if
match
                ind = ind + 1;
                break; %this 'break' optimizes the code.
The comparison will only be 'TRUE' once, after finding it,
exit the if-statement.
            end
        end
    end
    result{i,1} = storage1';
    clear storage1; %clears the variable to start storing
comparison results
end %end for-loop of list1 comparison

%list 2 comparison
for i = 1:length(temp)
    storage1 = strings; %initialize the string array
    ind = 1; %independent counter variable
    storage1(ind) = "Number ID is not available"; %failsafe
result
    for j = 1:length(temp{i,2})
        s1 = temp{i,2}(j);
        for k = 1:length(to2)
            s2 = to2(k);
            if(s2 == s1) %this comparison checks if the
numbers from DAVID match - i.e. checks if gene IDs match
```

```
                storage1(ind) = from2(k); %store result if
match
                ind = ind + 1;
                break; %this 'break' optimizes the code.
The comparison will only be 'TRUE' once, after finding it,
exit the if-statement.
            end
        end
    end
    result{i,2} = storage1';
    clear storage1; %clears the variable to start storing
comparison results
end %end for-loop of list2 comparison

end %end of function file
```

7.4 Function: changed_list_comp.m

```
function [result] =
changed_list_comp(geneNames,inputArg1,inputArg2,fdr_thresho
ld)
%CHANGED_LIST_COMP: This function thresholds the gene lists
to the
%user-specified value. Additionally, it prioritizes the
results in an
%ascending FDR-value format (current prioritization
technique) and creates
%the cell array output which will be the input for the
printer.m function.
%  % Input Arguments:
   %    - geneNames: a table - this is the output from
reversename function
   %    - inputArg1: a table containing the CRISPR screen
data (gene
   %    list)
   %    - inputArg2: a table containing the CRISPR screen
data (gene
   %    list)
   %    - fdrthreshold: fdr-thresholding value for CRISPR
gene lists
   %
   % Output:
```

```
   %    - result: cell array that thresholds the gene
lists, and
   %    prioritizes the output in ascending order (current
prioritization
   %    technique)

%isolate the string from the loaded table
test1 = inputArg1.target;
test2 = inputArg2.target;

%isolate the fdr values into an array
fdr1 = inputArg1.fdr;
fdr2 = inputArg2.fdr;

%comparison
%gene list 1 comparison
for i = 1:length(geneNames)
    storage1 = strings; %initialize the string array
    ind = 1; %independent counter variable
    storage1(ind) = "No CRISPR screened Genes available at
this threshold";
    fdr_organizer(ind) = 100; %initialize an fdr storage
array
    for j = 1:length(geneNames{i,1})
        s1 = geneNames{i,1}(j);
        for k = 1:length(test1)
            s2 = test1(k);
            if(strcmp(s1,s2) && (fdr1(k) <= fdr_threshold))
%this comparison checks if the numbers from DAVID match -
i.e. checks if gene IDs match
                storage1(ind) = test1(k); %if there is a
similarity (that meets the threshold), store in storage1
                fdr_organizer(ind) = fdr1(k); %store the
fdr of the gene that meets the threshold + similarity
condition
                ind = ind + 1; %increment independent
counter
                break; %this 'break' optimizes the code.
The comparison will only be 'TRUE' once, after finding it,
exit the if-statement.
            end
            if(strcmp(s2,'')) %this is the blank space
checker for inputArg1 and inputArg2 - they have too many
blank spaces at the end
```

```
                break;
            end
        end
    end

    %organize the gene list in order of descending fdr
values
    descender = table(storage1',fdr_organizer');
    holder = sortrows(descender,{'Var2'},{'ascend'});

    result{i,1} = strjoin(holder.Var1,', '); %converts the
cell array into a string variable
    result{i,3} = num2str(sort(fdr_organizer)); %organizes
the fdr-values in ascending order (like the genes) and
makes it into a string
    clear storage1; %clears the variable to start storing
comparison results
    clear fdr_organizer; %clears the variable to start
storing comparison results

end %end for-loop of list 1 comparison

%gene list 2 comparison
for i = 1:length(geneNames)
    storage1 = strings; %initialize the string array
    ind = 1; %independent counter variable
    storage1(ind) = "No CRISPR screened Genes available at
this threshold";
    fdr_organizer(ind) = 100; %initialize an fdr array
    for j = 1:length(geneNames{i,1})
        s1 = geneNames{i,1}(j);
        for k = 1:length(test2)
            s2 = test2(k);
            if(strcmp(s1,s2) && (fdr2(k) <= fdr_threshold))
%this comparison checks if the numbers from DAVID match -
i.e. checks if gene IDs match
                storage1(ind) = test2(k); %if there is a
similarity (that meets the threshold), store in storage1
                fdr_organizer(ind) = fdr2(k); %store the
fdr of the gene that meets the threshold + similarity
condition
                ind = ind + 1; %increment independent
counter
```

```
                break; %this 'break' optimizes the code.
The comparison will only be 'TRUE' once, after finding it,
exit the if-statement.
            end
            if(strcmp(s2,'')) %this is the blank space
checker for inputArg1 and inputArg2 - they have too many
blank spaces at the end
                break;
            end
        end
    end

    %organize the gene list in order of descending fdr
values
    descender = table(storage1',fdr_organizer');
    holder = sortrows(descender,{'Var2'},{'ascend'});

    result{i,2} = strjoin(holder.Var1,', '); %converts the
cell array into a string variable
    result{i,4} = num2str(sort(fdr_organizer)); %organizes
the fdr-values in ascending order (like the genes) and
makes it into a string
    clear storage2; %clears the variable to start storing
comparison results
    clear fdr_organizer; %clears the variable to start
storing comparison results

end %end for-loop of list 2 comparison

end %end of function file
```

7.5 Function: printer.m

```
function [temp] = printer(temp_store)
%PRINTER: This function allows for the clean formatting of
the
%extended-format Excel spreadsheet.
%   % Input Arguments:
    %    - temp_store: output from changed_list_comp
function]
%   NOTE: column 1 & 3 are related, 2 & 4 are related.
Column 1, 2 are Gene
%   IDs, Column 3, 4 are fdr-values of Genes from Column 1,
2 respectively.
```

```
%
    % Output Arguments:
    %    - temp: cell-variable that is well-organized and
shows prioritized
    %    results to readily print to Excel.
%

[z, zz] = size(temp_store); %counter-limit from inArg1
b = 1; %row-counter for temp variable (list 1)
h = 1; %row-counter for temp variable (list 2)

%eventually, the two row-counters will compare against each
other and the
%larger number will prevail to ensure continuity in
printing

for a = 1:z %first for-loop to parse through the original
argument

    b_keeper = b; %creates a clone of b;
    h_keeper = h; %creates a clone of h;

    temp{b,1} = temp_store{a,1}; %this stores the pathway
term
    temp{b,2} = temp_store{a,2}; %this stores the fdr value
with condition 1
    temp{b,3} = temp_store{a,3}; %this stores the fdr value
with condition 2

    %isolating the string of genes
    temp_holder_string = temp_store{a,4};
    holder_string =
textscan(temp_holder_string,'%s','Delimiter',',');
%changing string of genes back into a cell array

    temp_holder_numeric = temp_store{a,5};
    holder_numeric = str2num(temp_holder_numeric);

    for i = 1:length(holder_string)
        for j = 1:length(holder_string{i,1}) %accessing the
specific area where the genes are
            temp{b,4} = holder_string{i,1}(j);
            b = b+1; %increment the value of b
        end
```

```
    end

    for i = 1:length(holder_numeric)
        temp{b_keeper,5} = holder_numeric(i);
        b_keeper = b_keeper + 1; %this ensures we start at
the same place as 'b'
    end

    clear temp_holder_string; clear temp_holder_numeric;
    clear holder_string; clear holder_numeric;

    %isolating the string of genes
    temp_holder_string = temp_store{a,6};
    holder_string =
textscan(temp_holder_string,'%s','Delimiter',',');
%changing string of genes back into a cell array

    temp_holder_numeric = temp_store{a,7};
    holder_numeric = str2num(temp_holder_numeric);

    for i = 1:length(holder_string)
        for j = 1:length(holder_string{i,1}) %accessing the
specific area where the genes are
            temp{h,6} = holder_string{i,1}(j);
            h = h+1; %increment the value of b
        end
    end

    for i = 1:length(holder_numeric)
        temp{h_keeper,7} = holder_numeric(i);
        h_keeper = h_keeper + 1; %this ensures we start at
the same place as 'h'
    end

    if(b > h) %this logical statement ensures the data is
correctly copied
        h = b;
    else %i.e. h > b
        b = h;
    end

    clear temp_holder_string; clear temp_holder_numeric;
    clear holder_string; clear holder_numeric;
end
```

```
end %end of function file
```