Internship Report: Predicting Drug Activity on Cancer Cell-lines Based on Gene Expression

By: Aaron Mohammed

MS Bioinformatics & Computational Biology
Department of Molecular Medicine
College of Medicine

Supervisor: Dr. Feng Cheng
Department of Pharmaceutical Science
College of Pharmacy

University of South Florida

Summer 2022

Table of Contents

I. Summary....................................Pg.1

II. Background.................................Pg.1

III. Tasks & Results

1. Correlation Analysis....................Pg.2
2. Biological Significance.................Pg.4
3. WGCNA...................................Pg.7
4. Machine Learning........................Pg.12

IV. Closing Remarks...........................Pg.15

Citations.....................................Pg.15

Key terms: Drug activity, cell-line, adjacency matrix, co-expression network, module eigengene, predictor, support vector, response vector

I. Summary

During the summer of 2022, I worked with Dr. Feng Cheng at the Department of Pharmaceutical Science at the University of South Florida. The end goal of our research project is to determine, for a given drug, which genes in the cancer cell-lines of the NCI-60 panel make the best predictors for that drug's activity. My goal for this summer was to find a good candidate to be the drug we should focus on. First, I calculated correlation coefficients between drug activity and gene expression in the cell-lines of the NCI-60 panel. Three types of gene expression data were used, RNAseq, Affy HuEx, and Agilent mRNA. The RNAseq data set was used later on to carry out machine learning since it had the highest correlations with drug activity. After getting the correlation results, I used the DAVID database and hierarchal clustering to gauge if the correlation results had any biological origin or relevance. To prepare for machine learning, I used the 'WGCNA' R package to cluster gene expression data from the RNAseq dataset into module eigengenes. Finally, I used the 'e1071' R package to construct SVMs that used the gene expression data as predictors to predict drug activity on the cell-lines, using the drug activity from the drugs with the highest correlations. In this report, I give detailed explanations of what my tasks were, the computational tools I used, the results I got, and what I learned.

II. Background

During the late 1980's, the Developmental Therapeutics Program (DTP) of the National Cancer Institute (NCI) began the development of an anticancer drug screening called the NCI-60 cell line panel. The NCI-60 is made up of 60 cancer cell lines from various types of cancer like skin, colon, and breast cancer. So far, over 80,000 compounds have been screened *in vitro*. The NCI-60 panel plays a very important role in drug discovery, in fact, most FDA approved drugs were screened in the NCI-60 panel at least twice[1]. The genomes of the cell-lines in this panel have been extensively tracked and recorded for decades. This plethora of genomic data has made it possible to gain a better understanding of anti-cancer drug activity without having to conduct wet lab experiments. For instance, the gene expression data can be used to predict drug activity[2]. This was the motivation behind our project. Since it is possible to predict drug activity using gene expression data, it should be possible to determine which genes make the best predictors for a given drug and use them to find other cancer cell-lines that drug would likely be sensitive towards.

Gene expression data for each of these cell lines were downloaded from CellMiner, a database created by the DTP[3]. The NCI-60 drug activity data that was used for this project was also downloaded from CellMiner. The raw “Compound activity: DTP NCI-60” dataset was used. It contained over 50,000 different compounds, of which 158 were FDA approved drugs and 79 were in clinical trials. In this dataset, drug activity is expressed as the negative log of GI50[4]. GI50 is the concentration of the drug at which there is a 50% reduction in growth of the cancer cells[3]. This means that a high value indicates only a small concentration of the drug was needed to inhibit the proliferation of the cells by 50%. The compound activity value of 13 is the max value in this dataset, while -4 is the lowest. 3 gene expression datasets were downloaded from CellMiner and the dataset that had the best correlations with drug activity was used for machine learning. The RNAseq, Affy HuEx 1.0, and Agilent mRNA datasets were downloaded.

III. Tasks & Results

1. Correlation Analysis

After downloading the drug activity and expression data, R was used to calculate correlation coefficients. The drug activity dataset contained over 80,000 rows since it contained multiple entries/experiments for most of the compounds. In order to reduce runtime, the size of the drug activity dataset was reduced by removing the drug activity data from past experiments and keeping the data from only the most recent experiments for each drug. This was accomplished by utilizing the rev() and duplicate() functions, and the not, !, operator to subset the data frame based on the column that contained the NSC IDs for each drug. This reduced the number of drugs from 83,680 to 56,461. To reduce the size further, the data frame was filtered and only the compounds that were FDA approved, had a max activity value >= 8, and a standard deviation >= 1 were extracted. This reduced the number of drugs to 21.

The gene expression data was filtered so that only highly expressed genes were kept. This was determined by calculating average expression for each gene across the cell lines and keeping the ones with average expression greater than or equal to a certain value. For the RNAseq dataset, genes that had average expression of at least 1 were kept, reducing the number of genes from 23,808 to 10,131. For Affy HuEx, the average expression cutoff was 6 and for Agilent mRNA, the cutoff was 5. The number of genes for Affy HuEx were reduced from 1,048,575 to 268 and 41,090 to 845 for Agilent mRNA.

Once the datasets were filtered, it was then time to get the correlation coefficients between drug activity and gene expression. At first, a nested for loop containing the cor.test() function was used, but Dr. Cheng recommended that I get in the habit of using the "apply" functions since those functions are more efficient than for loops for very large datasets. It was difficult at first, but I accomplished this by creating a functional, which is a function of one or more functions. First, the datasets were all transposed so that the drug IDs and gene IDs became columns. The names of drug and gene IDs were stored in string vectors, called drug_name and G respectively, to be used as column indexes. Two lapply() functions and a cor.test() function were used, one a function of the other. That functional is shown below,

```
Corr <- lapply(drug_name, function(drug_name, drugs_df){
        lapply(gene_name, function(gene_name, genes_df){
               cor.test(as.numeric(drugs_df[[drug_name]]),
                        as.numeric(genes_df[[gene_name]])) %>%
               tidy()}, genes_df) %>%
        bind_rows() %>%
        mutate(Gene = gene_name) %>%
        select(Gene, estimate, p.value) %>%
        as.data.frame()}, drugs_df)
```

where Corr is the data frame the output is stored in, drugs_df is the drug activity dataset, and genes_df is the gene expression dataset. In order to save the output from this functional, the 'broom' and 'dplyr' packages were used. The pipe operator, %>%, and the tidy() function from the 'broom' package were used to store the statistical output of cor.test() into a tibble, which is essentially a simple version of a data frame that minimizes memory. The bind_rows(), mutate(), and select() functions from the 'dplyr' package were used to manipulate the tibbles and prepare them for conversion to data frames. The output of this lapply() function is a list of data frames, so the ldply() function from the 'plyr' package was used to take values from each data frame stored in the list and place them into a single data frame.

The correlation coefficients and p values from the list were extracted and stored in separate data frames using for loops. Another data frame containing the max correlation values, standard deviation of activity, max activity, and names for each drug was created to be exported to excel. This process was repeated for each gene expression dataset, using both the Pearson and Spearman methods. In total, there were 6 resulting datasets. The RNAseq dataset was chosen to be the predictor data for the machine learning models since it was more correlated with drug activity than the other gene expression datasets. The max correlation coefficients using the Pearson method is shown in Table 1.

ID	Drug Name	RNAseq	Affy HuEx	Agilent mRNA
NSC_764134	Dabrafenib	0.868444595	0.617700142	0.628898795
NSC_778304	Encorafenib	0.79637691	0.604124756	0.596412285
NSC_759877	Dasatinib (Salt)	0.65954711	0.477472315	0.667926976
NSC_764042	ARRY-162	0.613473369	0.395597683	0.49680274
NSC_768068	Cobimetinib (XL-518)	0.609095692	0.454204284	0.460534312
NSC_732517	Dasatinib	0.601511664	0.443693083	0.514676278
NSC_287459	Cytarabine	0.600924975	0.405522595	0.550401927
NSC_741078	Selumetinib	0.588182718	0.382604012	0.473002774
NSC_758246	Trametinib	0.565996502	0.431944273	0.447800518
NSC_606698	Rapamycin	0.553280038	0.363508897	0.381078898
NSC_733504	Everolimus	0.547620796	0.519850501	0.490325446
NSC_778590	Cobimetinib (GDC-0623)	0.546014427	0.363009064	0.444349829
NSC_226080	Rapamycin (Salt_2)	0.53671643	0.515807417	0.487754655
NSC_683864	Temsirolimus	0.534450908	0.359875414	0.412855524
NSC_758664	Rapamycin (Salt_1)	0.514088407	0.454547183	0.381471794
NSC_628503	Docetaxel	0.493749954	0.399446112	0.491897444
NSC_728073	Irinotecan	0.48115399	0.428024279	0.474562362
NSC_698037	Pemetrexed	0.465206531	0.358077451	0.48335166
NSC_90636	Vinblastine	0.461302367	0.496862542	0.550255982
NSC_726630	Belinostat	0.447092739	0.316985451	0.426815852
NSC_760087	Vinorelbine	0.322881507	0.369471162	0.45003084

Table 1 – Max correlation coefficient values between drug activity and gene expression.

2. Biological Significance

Before moving forward, it was necessary to find out if the results of the correlation analysis could be explained by biological phenomena or just random chance. If the answer is the latter, then there would not be much point in moving forward with using this gene expression dataset for machine learning. If the results of this correlation analysis did make sense within the context of biology, then it would be likely that the highly expressed genes that are positively or negatively correlated with drug activity are involved with pathways related to cancer. It would also be likely that positively or negatively correlated genes are associated with metabolic pathways, since many of these drugs target proteins that are involved with cell signaling. A webserver called DAVID

ID	NAME	POSITVE CORRELATION	NEGATIVE CORRELATION
287459	Cytarabine	Metabolic pathway	Pathways in cancer
698037	Pemetrexed	Metabolic pathway	Pathways in cancer
764134	Dabrafenib	Metabolic pathway	Pathways in cancer
778304	Encorafenib	Metabolic pathway	Pathways in cancer
764042	ARRY-162	Metabolic pathway	Pathways in cancer
741078	Selumetinib	Metabolic pathway	Pathways in cancer
778590	Cobimetinib (GDC-062)	Metabolic pathway	Pathways in cancer
768068	Cobimetinib (XL-518)	Metabolic pathway	Pathways in cancer
758246	Trametinib	Metabolic pathway	Neither
759877	Dasatinib (Salt)	Pathways in cancer	Metabolic pathway
732517	Dasatinib	Pathways in cancer	Metabolic pathway
758664	Rapamycin (Salt_1)	Pathways in cancer	Neither
683864	Temsirolimus	Pathways in cancer	Metabolic pathway
606698	Rapamycin	Pathways in cancer	Metabolic pathway
760087	Vinorelbine	Metabolic pathway	Pathways in cancer
90636	Vinblastine	Metabolic pathway	Pathways in cancer
728073	Irinotecan	Metabolic pathway	Pathways in cancer
726630	Belinostat	Metabolic pathway	Pathways in cancer
733504	Everolimus	Pathways in cancer	Metabolic pathway
628503	Docetaxel	Metabolic pathway	Pathways in cancer
226080	Rapamycin (Salt_2)	Pathways in cancer	Metabolic pathway

Table 2 – This table displays results of the pathway analysis and shows whether any of the positively or negatively correlated genes are related to pathways in cancer or metabolic pathways.

was used to investigate the pathways associated with genes that were positively and negatively correlated with drug activity. DAVID is an abbreviation for Database for Annotation, Visualization, and Integrated Discovery, it allows a user to access tools and databases for functional annotation analysis[5]. Gene IDs for the top 500 positively correlated and lowest 500 negatively correlated genes for each drug were inputted to DAVID. Each search checked to see if the gene IDs are included in any of the databases contained in DAVID. If they are, a data tables displaying lists of pathways associated with the genes can be accessed. The KEGG pathway database was selected for this analysis. KEGG stands for Kyoto Encyclopedia of Genes and Genomes, and it contains many annotations of gene functions[6]. A table was created to keep track of whether the genes were associated with either metabolic pathways or pathways in cancer. If both pathways appeared in the results, the one that had the most gene counts were chosen. The results of this analysis are shown in Table 2 and includes the genes involved with the mechanism of action for each drug that were included in the NCI-60 drug activity dataset. It was found that 14 drugs had positively correlated genes that were primarily associated with metabolic pathways and 7 had positively correlated genes that were primarily associated with pathways in cancer. 13 drugs had negatively correlated genes that were primarily associated with pathways in cancer and 6 that had negatively correlated genes that were primarily associated with metabolic pathways. There were two drugs who's lowest 500 correlated genes were not associated with metabolic pathways or pathways in cancer.

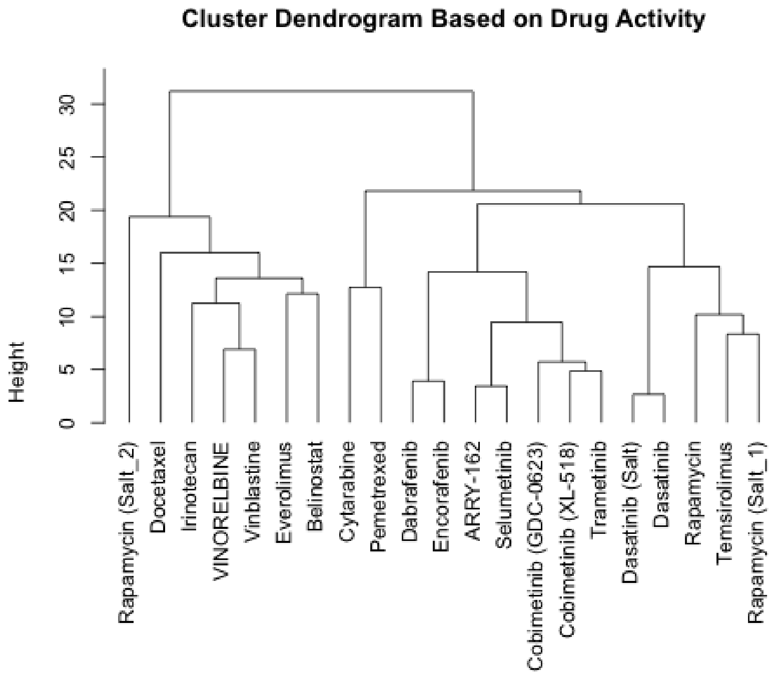

Figure 1 – This map displays how similar each drug's activity is with each other. Similarity increases as the height gets smaller.

Hierarchal clustering of drug activity was used to cluster the drugs based on drug activity. This was done to find out if the drugs cluster together based on the results of the pathway analysis. Would the drugs that were positively correlated with genes associated with metabolic pathways and negatively with cancer pathways cluster together and vice versa? A cluster dendrogram of drug activity was created in R and is shown in Figure 1. The first step in making this plot was calculating the distances between all possible pairs of drug activity values between all drugs and placing them in a distance matrix. This was done by using the dist() function. The distance matrix was then inputted into the hclust() function which uses complete linkage clustering to form the dendrogram. Although the drugs were not completely separated based on pathways in metabolism and cancer, most did come together that way in clusters that have a small height. The colored sets shown in Table 3 represent clusters with a height less than 10. There are some exceptions, but almost 70% of them clustered this way. This means that the correlations between gene expression and drug activity appear to make sense biologically. This becomes clearer once the genes associated with the mechanisms of action for each drug are included in Table 3. They were obtained from the original NCI-60 drug activity dataset.

ID	Name	Mechanism	Positively Correlated Genes	Negatively Correlated Genes
287459	Cytarabine	Ds	Metabolic pathway	Pathways in cancer
698037	Pemetrexed	Df|AM|GARTF|DHFR	Metabolic pathway	Pathways in cancer
764134	Dabrafenib	PK:BRAF	Metabolic pathway	Pathways in cancer
778304	Encorafenib	PK:BRAF	Metabolic pathway	Pathways in cancer
764042	ARRY-162	PK:STK,MAP2K,MAP2K1,MAP2K2	Metabolic pathway	Pathways in cancer
741078	Selumetinib	PK:STK,YK,MAP2K,MAP2K1,MAP2K2	Metabolic pathway	Pathways in cancer
778590	Cobimetinib (GDC-0623)	PK:MAP2K,MAP2K1	Metabolic pathway	Pathways in cancer
768068	Cobimetinib (XL-518)	PK:MAP2K,MAP2K1	Metabolic pathway	Pathways in cancer
758246	Trametinib	PK:STK,MAP2K,MAP2K1,MAP2K2	Metabolic pathway	Neither
759877	Dasatinib (Salt)	BCR-ABL|PK:YK,PDGFR,KIT	Pathways in cancer	Metabolic pathway
732517	Dasatinib	BCR-ABL|PK:YK,PDGFR,KIT	Pathways in cancer	Metabolic pathway
758664	Rapamycin (Salt_1)	PK:STK,MTOR	Pathways in cancer	Neither
683864	Temsirolimus	PK:STK,MTOR	Pathways in cancer	Metabolic pathway
606698	Rapamycin	PK:STK,MTOR	Pathways in cancer	Metabolic pathway
760087	Vinorelbine	TUBB|Tu-frag	Metabolic pathway	Pathways in cancer
90636	Vinblastine	TUBB|Tu-frag	Metabolic pathway	Pathways in cancer
728073	Irinotecan	TOP1	Metabolic pathway	Pathways in cancer
726630	Belinostat	HDAC	Metabolic pathway	Pathways in cancer
733504	Everolimus	PK:STK,MTOR	Pathways in cancer	Metabolic pathway
628503	Docetaxel	TUBB|Tu-stab	Metabolic pathway	Pathways in cancer
226080	Rapamycin (Salt_2)	PK:STK,MTOR	Pathways in cancer	Metabolic pathway

Table 3 – The blue and red boxes represent the 2 clusters with the largest distance. The colored sets are drugs inside clusters that have a height less than 10.

3. WGCNA

Since the RNAseq dataset had thousands of genes, the 'WGCNA' R package was used to cluster the genes and form module eigengenes (MEs). This was necessary because SVMs are not suitable for large datasets. Instead of having to use every gene in a dataset, an ME can be used to represent a large number of genes that have similar patterns of expression. MEs can be created by carrying out weighted correlation network analysis or WGCNA. This type of analysis is based on the correlations between multiple genes and how their gene expression changes through different samples. If their gene expression is co-expressed i.e. follows similar patterns across sample, they form a connection in the network. Unlike regular gene co-expression networks (GCNs) that assign binary values to represent connected and unconnected, weighted co-expression networks assignments are weighted based on the degree of connectedness[7]. This is what is referred to as "soft-thresholding". The clusters that form in these networks are called modules and it is these modules that form eigengenes.

Before using the 'WGCNA' package, input variables needed to be prepared. The RNAseq gene expression data was imported into R along with a file that contained phenotype data for each cell-line. That phenotype data was manually created in excel and consisted of the name of each cell-line and type of cancer they originate from. The first 6 columns of data that contained identifiers and locations of each gene were extracted and stored in a .csv file. This file would be used as a gene map later on. Mitochondria and ribosomal genes were removed because they were not relevant to this study. The gene expression values were normalized and scaled according to this formula: $\log_2(\text{FPKM}+1)$. Genes with low expression were removed by calculating the average expression for each gene and only keeping the ones that had an average greater than 1. Once filtered, coefficients of variation (CV) were calculated for each gene by dividing standard deviation by average expression and the genes that had a value less than 0.2 were removed. This was done to select genes with high variance. The gene expression dataset was now left with 4,725 genes.

Once the input variables were prepared, it was time to move forward and begin using the 'WGCNA' package. The first step was to analyze the network topology resulting from the gene expression data frame and different values of soft-thresholding power, β, in what is called an adjacency function. This function is what determines how the co-expression network is constructed. This step is necessary for deciding the right thresholding power for the co-expression network to have "scale-free topology", which means the degree distribution follows a power law distribution,

$$p(k) \sim k^{-\gamma}$$

where $p(k)$ is degree distribution, k is degree, and γ is a constant. Degree refers to the number of connections a node or gene in the network has with others. The reason why the network should follow this topology is because nature appears to have many scale-free networks, at least approximate ones[8]. In a gene co-expression network that is scale free, the degree distribution decays as k increases. As mentioned earlier, weighted correlation networks use soft-thresholding to classify connections. The adjacency function is represented by,

$$a_{ij} = |s_{ij}|^{\beta}$$

where a_{ij} is an element in an adjacency matrix, and s_{ij} is an element in a similarity matrix. Similarity is a measure of how much gene expression profiles correlate with one another and is given by the following equation,

$$s_{ij} = |cor(i, j)|$$

where $cor(i, j)$ is a function for the Pearson correlation and i and j represent a pair of genes[7]. The elements of the adjacency matrix represent adjacency between a single pair of genes i.e. they represent whether a pair of genes are connected or co-expressed. In "hard-thresholding", used by GCNs, these elements are 1's and 0's. As one can see from the similarity function, similarity in soft-thresholding can take on a range of values between 0 and 1. This is how the networks are weighted.

The function pickSoftThreshold() is the function used for network topology analysis. The inputs were the gene expression dataset, vector of power values, and level of verbosity, how much a computer tells the user what it is doing in the console. The output of this function is used to make two plots shown in Figure 2. These plots are used to choose the value of soft-thresholding power. In the scale independence plot of Figure 2A, the red line represents the threshold level for a network to be scale free. In order for the networks to be approximately scale-free, the value of power chosen should be above or close to the red line and should have a low value of mean connectivity. The plots displayed in Figure 2 were created using the gene expression dataset, so the value for power chosen was 7.

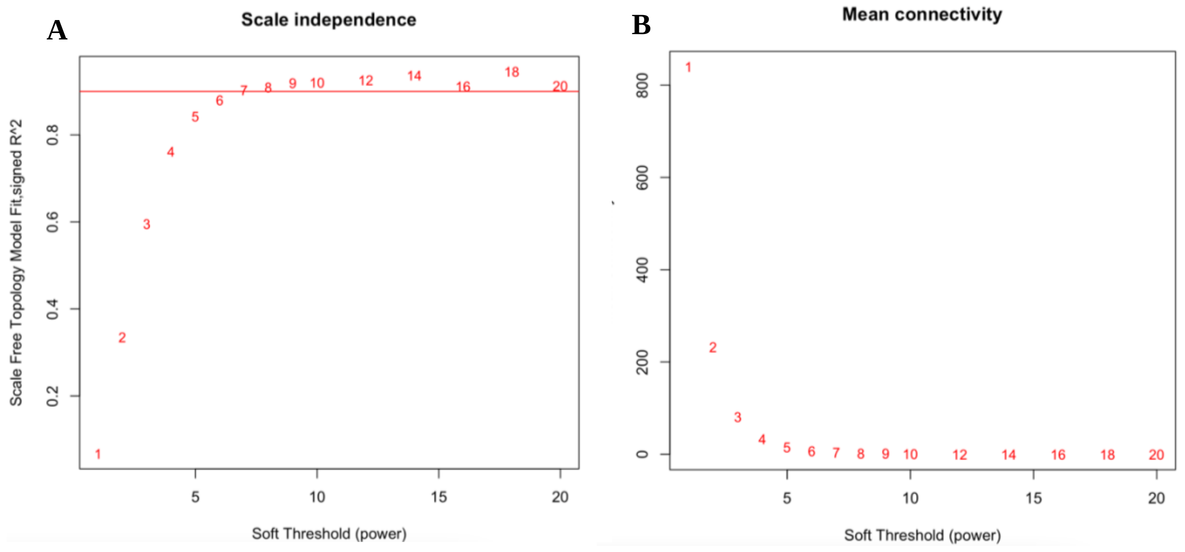

Figure 2 – A: Plot displaying how much the co-expression network with this gene expression dataset fits a scale-free topology for different values of power. B: Plot displaying the average connectivity between genes in this dataset using different values of power.

After choosing 7 as the value for soft-thresholding power, it was time for construction of the network and detection of the modules. In the ‘WGCNA’ package, this could all be done by using just one function, blockwiseModules(). It takes in the gene expression data frame and power as input along with other parameters shown here,

```
output = blockwiseModules(datExpr, power = 7,
                          TOMType = "unsigned", minModuleSize = 30,
                          reassignThreshold = 0, mergeCutHeight = 0.25,
                          numericLabels = TRUE, pamRespectsDendro = FALSE,
                          saveTOMs = TRUE,
                          saveTOMFileBase = "Celltreatment",
                          verbose = 3)
```

where TOMType stands for Topological Overlap Matrices (TOMs), minModuleSize sets the minimum size of modules, reassignThreshold is a threshold in the form of a p-value used for reassigning genes between modules, mergeCutHeight is a value for cutting the height of the dendrogram, numericalLabels is a logical where FALSE will lead to the modules being labeled with colors and TRUE with numbers, pamRespectsDendro is a logical that controls the behavior of Partitioning Around Medoids (PAM), a method for clustering genes, saveTOMs saves the TOMs to the computer, saveTOMFileBase is a character string that serves as the file name base for files containing topological overlaps, and verbose dictates how many messages are displayed while the code is running. When this function is executed, a number of actions take place. First, the adjacencies between the genes are calculated and those values are used to make a TOM. If two genes have topological overlap, it means that they share similarity based on the other genes they are connected too[9]. In other words, they are a part of the same module. The TOMs are then scaled so that the data within them are comparable across samples and then the consensus between the matrices is calculated. That result is used to create a dendrogram in order to identify the modules. The modules are then combined if their expression profiles are very similar[10]. The final dendrogram can then be plotted, the one created for our gene expression dataset is shown in Figure 3. The MEs can be extracted from the list that the output is stored in and is located in a data frame called "MEs". In total, 16 MEs were formed from this dataset.

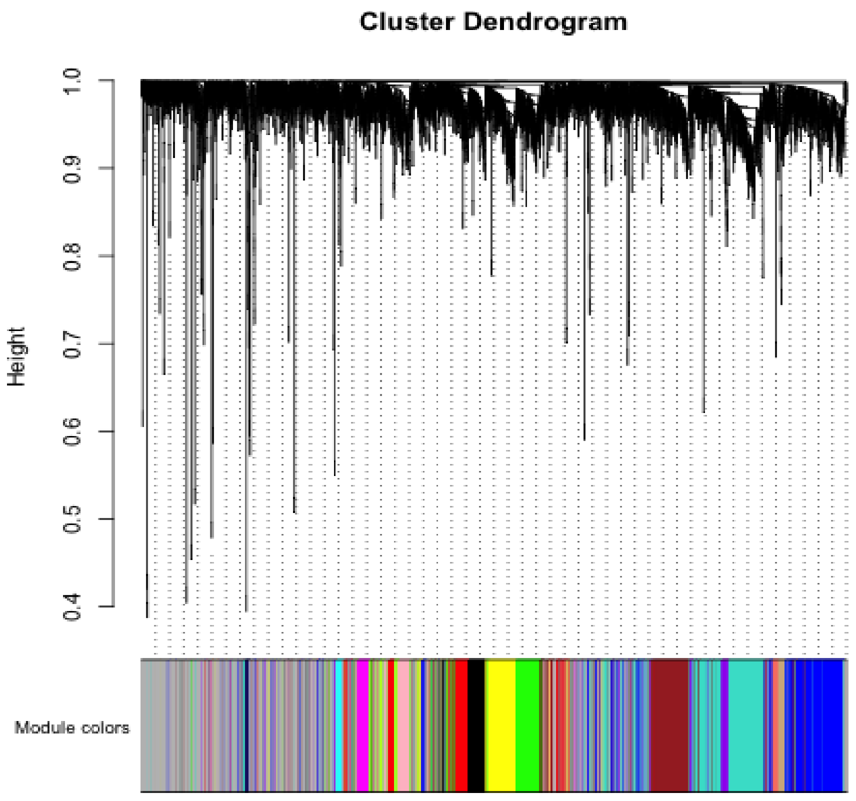

Figure 3 – A dendrogram of the genes after clustering. Each color corresponds to a module.

4. Machine Learning

After generating the MEs, it was now time to use ‘e1071’ to carry out machine learning to help us choose a drug to focus on for the rest of the project. The drugs with correlation constants >= 60 were chosen. The eigengene expression data was used as the predictor data and the activity data for each of the drugs were used as response vectors. The response vector contained 1’s and 0’s, where a 1 represented active and a 0 represented inactive. The predictor data is the data used for classification. First, the drug activity and ME datasets were imported and Na values were removed from the activity data. The rows that were removed in that data frame were also removed from the ME data frame. A response vector was created based on drug activity values. If drug activity on a cell-line was above a certain threshold, then the response vector would get a value of 1 for that row and a 0 if it were below. The value of that threshold was varied throughout this analysis to find the value that would lead to the highest accuracy.

At least 2 columns of predictor data are required for SVMs. This is why the combn() function was used to make unique pairs of MEs and place them into a list. For the 16 MEs, 120 pairs were made. A for loop was made so that a pair of the MEs would be placed into a data frame along with the response vector for a drug upon each cycle of the loop, going through each pair one by one. Also included in this loop was a function I created called SVM_(). It was in this function that I placed ‘e1071’ functions that would build, train, and test SVMs. SVM_ takes in the data frame of the ME pairs and response vector, percentage of predictor data used for training, “percent_training”, and the number of times to repeat both training and testing, “ncycles”. Within this function is a for loop that cycles the number of times specified in ncycles. The first step in this loop is a random number generator which is the sample.int() function from the ‘caTools’ package,

```
id <- sample.int(nrow(dat),
      floor(length(activity) * percent_traning/100))
```

where dat is the data frame containing the predictor data, activity is the drug activity data. Those random numbers were used as indexes to subset the predictor data and form a training set. They are also used to form the test set by way of subsetting the data and excluding those indexes. That training set is used as an input to the svm() function from ‘e1071’,

```
classifier <- svm(formula = y ~ ., data = training,
                  type = 'C-classification', kernel = 'linear')
```

where y is the response vector and the dot is the predictor data, type is the method of prediction, in this case classification, and kernel is the shape of the function being used to separate data points. It is in this step where the SVM models are constructed and trained. Points from the predictor data, which are classified as being either active or inactive, are plotted and the kernel is used to find a function that separates the points[11]. Points that are closest to the function are called "support vectors". Their distance to the function is called a "margin". The SVM continues trying out different weights for the function until the position where the maximum margins are found i.e. position that is the farthest away from all of the points as possible. The function that achieves this is the function that is used for test predictions. The function predict() from 'e10701' is used to test the model,

```
y_pred <- predict(classifier, newdata = test[,-3])
```

and takes the model created by the svm() function and the test data set as inputs. The output of this test is a predicted response vector and its is plotted with the correct response vector using the table() function to create a confusion matrix. This confusion matrix is has 2 rows and 2 columns. Each element of the matrix represents scores for the predicted classification. These scores are the number of true positives (TP), false positives (FP), true negatives (TN), and false negative (FN). TP means the model was correct in predicting active, FP means the model was wrong when predicting active, TN means the model correctly predicted inactive, and FN means the model incorrectly predicted negative. These values were used to calculate the averages of accuracy, sensitivity, specificity, positive predictive value (PPV), and negative predictive value (NPV) across all of the cycles using the following equations respectively,

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}}$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{NPV} = \frac{\text{TN}}{\text{TN} + \text{FN}}$$

The results for the 7 drugs that had correlation values >= 0.6 are shown in Table 4. A value of 7 was found to be the optimal cutoff for labeling activity as active. 80% of the predictor data was used as the training set and the number of cycles were 10,000.

NAME	ID	MEs	Accuracy	Sensitivity	Specificity	PPV	NPV
Cytarabine	287459	ME9_ME11	0.936	0.656	1.000	0.824	0.924
Dasatinib	732517	ME9_ME15	0.935	0.645	1.000	0.812	0.923
Encorafenib	778304	ME7_ME9	0.935	0.649	1.000	0.815	0.923
ARRY-162	764042	ME9_ME12	0.935	0.648	1.000	0.815	0.923
Dasatinib (Salt)	759877	ME3_ME9	0.934	0.650	1.000	0.821	0.922
Cobimetinib (XL-518)	768068	ME4_ME9	0.934	0.650	1.000	0.819	0.922
Dabrafenib	764134	ME1_ME9	0.934	0.649	1.000	0.820	0.922

Table 4 – Performance of SVMs when there were 2 support vectors, the active threshold was >7, percent data used for training was 80%, and the number of cycles was 10,000.

After using 2 eigengenes at a time for training and testing, combn() was used to create unique combinations of 3, resulting in 560 groups. The parameters used were >=7 for active labeling, 80% for percent_training, and 500 cycles. The results are shown in Table 5. This was done to see if the order of accuracy would be similar to the order obtained using 2 eigengenes at a time.

NAME	ID	MEs	Accuracy	Sensitivity	Specificity	PPV	NPV
Cobimetinib (XL-518)	768068	ME1_ME4_ME9	0.944	0.674	1.000	0.826	0.933
Encorafenib	778304	ME1_ME9_ME14	0.943	0.668	1.000	0.824	0.933
Dabrafenib	764134	ME4_ME9_ME12	0.941	0.676	1.000	0.836	0.930
Dasatinib (Salt)	759877	ME9_ME13_ME15	0.941	0.663	1.000	0.818	0.930
Cytarabine	287459	ME2_ME9_ME13	0.941	0.681	1.000	0.836	0.930
ARRY-162	764042	ME1_ME9_ME13	0.940	0.662	0.999	0.802	0.931
Dasatinib	732517	ME1_ME4_ME6	0.939	0.644	0.999	0.801	0.928

Table 5 – Performance of SVMs when there were 3 support vectors, the active threshold was >7, percent data used for training was 80%, and the number of cycles was 500.

IV. Closing Remarks

Using 3 dimensions for the predictor data ended up producing different rankings for the drugs compared to 2 dimensions based on accuracy. This could be because a lower number of cycles were used. There is still more work to be done in determining which drug would give the best evaluation measures. This entails varying the other parameters like percent_training and ncycles. Once the drug is found that gives the best evaluation measures is found, we can then move on to determining which genes are most important for predicting activity for that drug. This can be done by using Cytoscape, a software program that can be used to visualize the co-expression networks of each ME that gives the best evaluation measures for the drug that is chosen[12].

Citations

1. Holbeck, S. L., Collins, J. M. & Doroshow, J. H. Analysis of Food and Drug Administration–Approved Anticancer Agents in the NCI60 Panel of Human Tumor Cell Lines. *Mol. Cancer Ther.* **9**, 1451–1460 (2010).
2. Cortés-Ciriano, I. *et al.* Improved large-scale prediction of growth inhibition patterns using the NCI60 cancer cell line panel. *Bioinformatics* **32**, 85–95 (2016).
3. Reinhold, W. C. *et al.* CellMiner: A Web-Based Suite of Genomic and Pharmacologic Tools to Explore Transcript and Drug Patterns in the NCI-60 Cell Line Set. *Cancer Res.* **72**, 3499–3511 (2012).
4. CellMiner - Datasets. https://discover.nci.nih.gov/cellminer/datasets.do.
5. Sherman, B. T. *et al.* DAVID: a web server for functional enrichment analysis and functional annotation of gene lists (2021 update). *Nucleic Acids Res.* gkac194 (2022) doi:10.1093/nar/gkac194.
6. Kanehisa, M. & Goto, S. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.* **28**, 27–30 (2000).

7. Zhang, B. & Horvath, S. A general framework for weighted gene co-expression network analysis. *Stat. Appl. Genet. Mol. Biol.* **4**, Article17 (2005).
8. Albert, R. Scale-free networks in cell biology. *J. Cell Sci.* **118**, 4947–4957 (2005).
9. Yip, A. M. & Horvath, S. Gene network interconnectedness and the generalized topological overlap measure. *BMC Bioinformatics* **8**, 22 (2007).
10. Langfelder, P. & Horvath, S. Tutorial for the WGCNA package for R II. Consensus network analysis of liver expression data, female and male mice. (2014).
11. Mammone, A., Turchi, M. & Cristianini, N. Support vector machines. *WIREs Comput. Stat.* **1**, 283–289 (2009).
12. Shannon, P. *et al.* Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. *Genome Res.* **13**, 2498–2504 (2003).