Supplementary Information for

"Predicting Cancer Drug Response by Proteomic Profiling"

Yan Ma, Zhenyu Ding, Yong Qian, Xianglin Shi, Vince Castranova, E. James Harner, and Lan Guo

Table of Contents

Table of Figures

1 Data Sets Description

In this study, we used two data files from the NCI DISCOVER database (http://discover.nci.nih.gov/). Specifically, we predicted drug response of 60 human cancer cell lines (NCI-60) to 118 anti-cancer drugs by proteomic profiling. Following are the two datasets used in the study.

Data file 1: Proteomic profiling of the NCI-60 cancer cell lines.

Link: http://discover.nci.nih.gov/host/2003_profilingtable7.xls

Description: 52 proteins expression profiles were analyzed across 60 human cancer cell lines (NCI-60).
Column A: HUGO name
Column B: Antibody name
Column C: Vendor
Columns D-BK: Protein expression values on 60 human cancer cell lines

The data file was generated from a previous publication by Nishizuka et al. (1) They developed a protocol for making reverse-phase protein lysate microarrays with a larger number of spots than previously feasible. They analyzed the data points for 52 antibodies by using P-SCAN and a quantitative dose interpolation method for 60 human cancer cell lines (NCI-60). The clustered images of the protein expression profiles for the 60 cancer cell lines revealed biologically meaningful patterns. The protein expression patterns also conformed to the mRNA expression patterns for the same genes related to cell structure.

Data file 2: Drug activity data of 118 “Mechanism of Action” drugs.

Link:
http://discover.nci.nih.gov/nature2000/data/selected_data/dataviewer.jsp?baseFileName=a_matrix118&nsc=2&dataStart=3

Description: It is a database of 118 anti-cancer drug activities across the NCI-60 cancer cell lines, whose mechanisms of action are putatively understood. Some of these drugs are currently in routine clinical use for cancer treatment, while others are either in clinical trials or in late stages of drug development.

Column A: Mechanism of action
Column B: Drug name
Column C: NSC number
Columns D-BK: Drug activities ($-\log_{10} GI_{50}$) across 60 human cancer cell lines. GI_{50} is the concentration required to inhibit cell growth by 50% compared with untreated controls. The activity profile of a compound consists of 60 such activity values, one for each cell line.

The data file was from a previous publication of Scherf et al. (2) They used cDNA microarrays to assess gene expression profiles in the NCI-60 lines, and correlated gene expression and drug activity patterns in the NCI-60. They concluded that clustering the cell lines based on gene expression entailed relationships very different from those obtained by clustering the cell lines based on their response to drugs. In addition, gene-drug relationships for the chemotherapeuticagents 5-fluororacil (5-FU) and *L*-asparaginase exemplified how variations in the transcriptional levels of particular genes relate to mechanisms of drug sensitivity and resistance.

2 Defining Cutoffs for Drug Sensitivity and Resistance

We investigated the chemosensitvity profiles of these 118 anti-cancer agents using different cutoffs for defining drug responses. In the literature, there is no formal definition regarding what standard deviation should be used as cutoffs in determining drug responses.

The data file containing drug activity data of the 118 anti-cancer agents were processed to define drug resistance and sensitivity for the NCI-60 lines. Specifically, for each drug, $\log_{10}(GI_{50})$ values were normalized across the 60 cell lines. Cell lines with $\log_{10}(GI_{50})$ at least 0.5 SDs above the mean were defined as *resistant* to this drug. Those with $\log_{10}(GI_{50})$ at least 0.5 SDs below the mean were defined as *sensitive* to the drug. The remaining cell lines with $\log_{10}(GI_{50})$ within 0.5 SDs were defined as *intermediate* in the range of drug responses. We summarized the chemosensitivity profiles of each cancer type using this cutoff (0.5 SDs) by averaging the profiles on the cell lines for each cancer type in the NCI-60 lines (Figure 1).

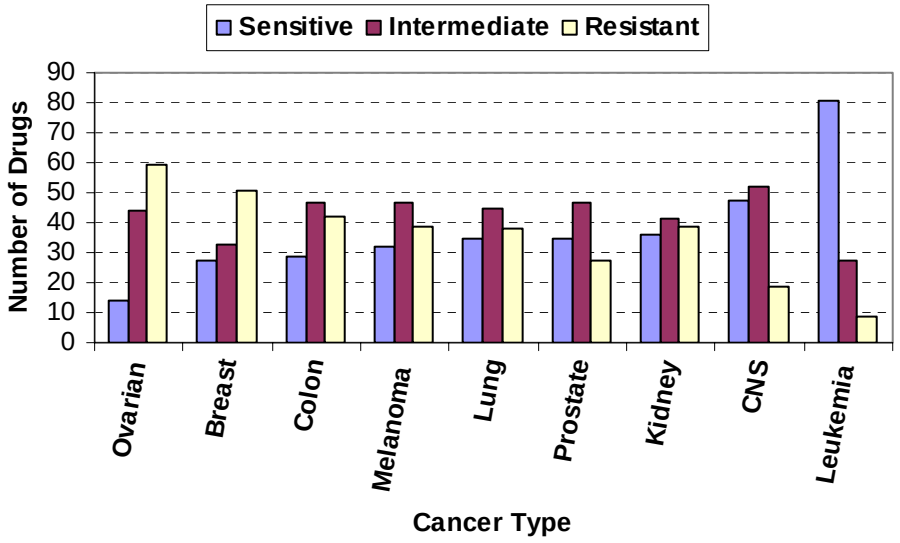

Figure 1. Chemosensitivity profiles of different cancer types (using 0.5 SDs)

We also used 0.6 SDs to define drug responses of the 60 cell lines (Figure 2).

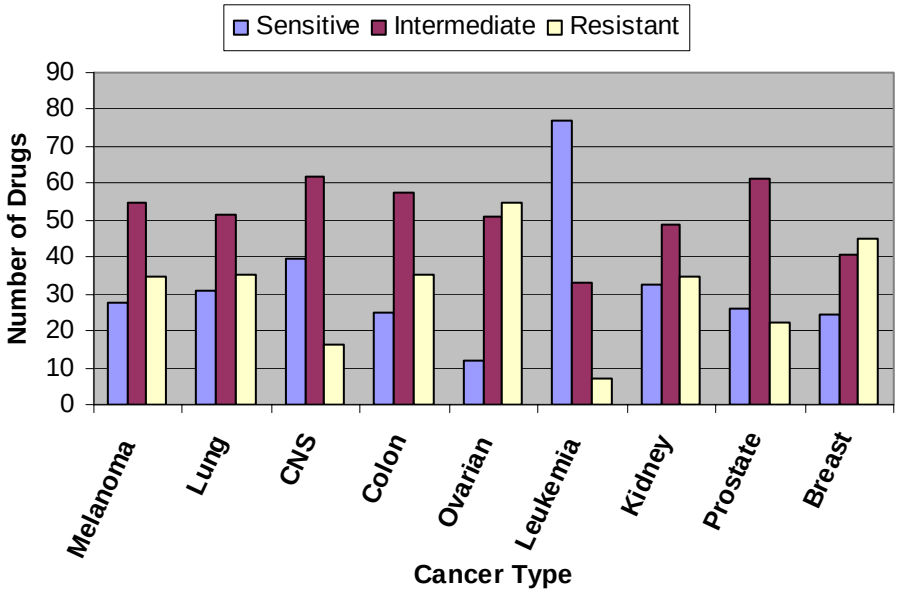

Figure 2. Chemosensitivity profiles of different cancer types (using 0.6 SDs)

In the previous study of Stuanton et al.(3), 0.8 SDs were used to define the cutoff thresholds for drug responses. The reason is that Stuanton et al.(3) screened thousands of chemical compounds, many of which have not yet been approved as anti-cancer drugs. Using 0.8 SDs is appropriate to reflect the wide range of drug activities of these chemical compounds. On the other hand, the 118 drugs analyzed in our study are either approved or potential anti-cancer drugs, which have a narrower range of drug activities in the NCI-60 lines. Therefore, using 0.8 SDs as cutoff thresholds results in unbalanced datasets, the majority of which are cases with the label *intermediate* (Figure 3). As a result, the machine learning algorithms tend to maximize the accuracy of the *intermediate* cases during the learning in order to achieve higher overall accuracy in generating prediction models for chemosensitivity. As a tradeoff, the prediction accuracy for the cases with the label *sensitive* or *resistant* is therefore low, which might pose serious problems in the clinical applications. For instance, the response of most patients who are actually sensitive or resistant to certain chemotherapeutic agents will be predicted as intermediate. Consequently, these patients might not be given the appropriate chemotherapy regimens. It is thus necessary to construct the classifiers that give highly accurate prediction results on balanced datasets. In general, statistical tests are employed to evaluate the overall accuracy of the constructed classifiers. However, the dilemma caused by heavily unbalanced datasets described above cannot be revealed using statistical tests such as a non-parametric test. We sought to explore different cutoff thresholds to define drug responses to construct chemosensitivity prediction models that achieve the optimal prediction accuracy in the complete range of drug responses.

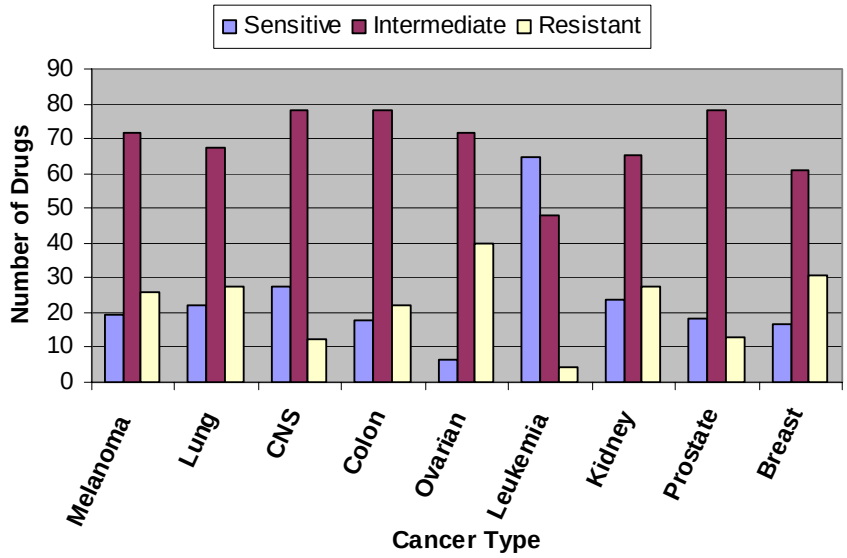

Figure 3. Chemosensitivity profiles of different cancer types (using 0.8 SDs)

We observed that using 0.6 SDs and 0.8 SDs resulted in unbalanced datasets with the majority of cases defined as *intermediate* in the range of drug responses. Figure 4 shows the percentage of the *intermediate* labels for each cancer type in the NCI-60 lines. As mentioned above, heavily unbalanced datasets generally entail unbalanced prediction error rates in supervised classification. The machine learning algorithms tend to minimize the overall prediction error rate by "sacrificing" the error rates of minority class labels. In our case, if *sensitive* and *resistant* are minority labels, these cases (cell lines) are more possible to be predicted as *intermediate* as a result. Such prediction models will cause serious problems in clinical context, although they might have favorable overall prediction accuracy. The results indicate that using 0.5 SDs is feasible for defining drug responses of the NCI-60 lines to the 118 anti-cancer drugs.

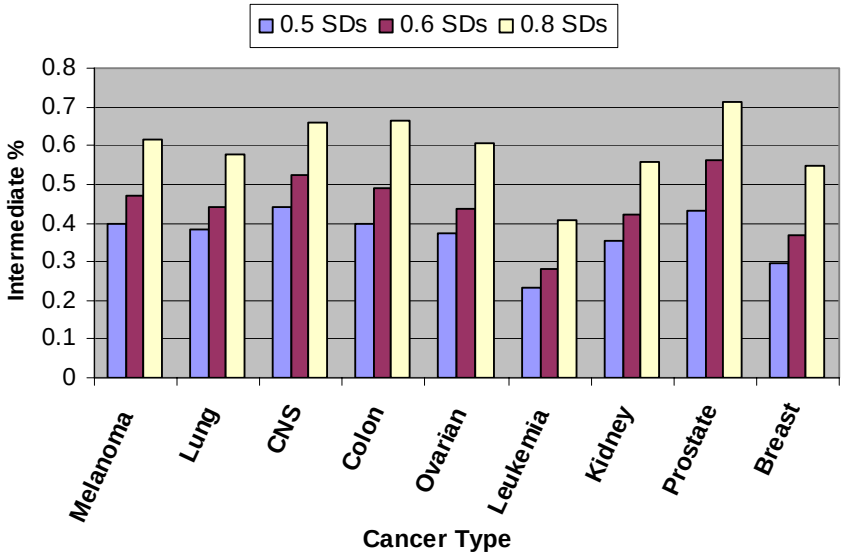

Figure 4. Comparing intermediate range of drug responses in different cancer types

3 Constructing Optimal Classifiers

3.1 Experiments Using Random Forests

For each drug, we can form a dataset with 53 variables including 52 protein variables and 1 drug response variable. The 52 protein expression variables are predictors, while the drug response is the predicted variable. The goal of this study is to predict the response of each cell line to the 118 anti-cancer drugs. For each drug, the number of the cell lines is at most 60 (because of the missing values, some drugs were tested with fewer cell lines). Compare to the sample size, the number of features (i.e. proteins) is relatively large. Some of these proteins might not be very informative in discriminating between classes. Thus, our first task is to remove the non-informative proteins. The goal is to filter out proteins step by step and ultimately, to obtain a subset of features with the smallest number of proteins and the optimal prediction accuracy of drug responses.

We used Random Forests (4) in software package *R* (http://www.r-project.org) as a classification technique, which can also rank the importance of the features in prediction. Random forests involve the generalization of the classification tree algorithm. Instead of growing a single classification tree, the random forest algorithm constructs an ensemble of hundreds or thousands of trees. Each tree is built upon a bootstrap sample from the original learning set. The variables used for splitting the tree nodes are a random subset of the whole variables set. All trees are grown to full length without pruning. The classification decision of a new instance is obtained by majority voting (unless the cutoff is user-defined) over all trees.

The variable importance is defined in terms of the contribution to prediction accuracy (5). The Random Forest algorithm implemented in software package *R* provides two importance measures: "*mean decrease in accuracy*" and "*mean decrease in gini*". In random forests, about one-third of the cases in the bootstrap sample are not used in growing the tree. These cases are called "out-of-bag" cases, which play an important role in algorithm performance evaluation.

- *Mean decrease in accuracy* is defined as follows. For each tree, randomly rearrange the values of the m^{th} variable for the "out-of-bag" set, put this permuted set down the tree, and get new classifications for the forest. The importance of the m^{th} variable can be defined in "*mean decrease in accuracy*" as the difference between the "out-of-bag" error rate for randomly permuted m^{th} variable and the original "out-of-bag" error rate.
- In "*mean decrease in gini*", the importance of the m^{th} variable is the sum of all decreases in impurity (measure by *gini* index) in the forest due to this variable, normalized by the number of trees. Usually, these two measures produce consistent results.

In this study, we ranked the proteins by using the "*mean decrease in accuracy*" method of random forests. An initial run was made on the full list of proteins and a ranking of the importance of all 52 proteins was obtained. In order to construct the optimal classifiers, the following procedure was performed: for each drug, the lowest ranking

proteins were sequentially removed. The bottom 2 proteins were first removed and a subset of top 50 proteins was used for the prediction. Then, the bottom 5 proteins were removed from the prediction model each time. When the subset contained 10 proteins, the bottom 1 protein was removed each time. In this study, the smallest subset consisted of 3 proteins. Together, the random forest algorithm was run 16 times based on these different subsets of variables. The prediction accuracy along with the protein list was recorded. The optimal classifier is the one with the smallest number of proteins and the highest prediction accuracy.

The random forest algorithm uses the out-of-bag method to evaluate the prediction accuracy. The out-of-bad method is unbiased, and therefore there is no need to perform cross-validation or use a separate dataset to evaluate the results (4).

3.2 Experiments Using WEKA Learners

Several drugs had relatively low prediction accuracies by using random forests. To identify the optimal classifiers, we used several methods implemented in WEKA (http://www.cs.waikato.ac.nz/ml/weka/) (6). We focused on the drugs with the prediction accuracy below 50% by using random forests.

For these drugs, we used the *Relief* method as a filter to rank the proteins. *Relief* evaluates the importance of a variable by repeatedly sampling an instance and checking the value of the given variable for the nearest instance from the same and different classes. The values of the attributes of the nearest neighbors are compared to the sampled instance and used to update the relevance scores for each attribute. The rationale is that an informative attribute should have the same value for instances from the same class and differentiate between instances from different classes (6, 7).

Several WEKA classifiers were explored for each drug, and the one generating the highest prediction accuracy was selected to construct the optimal classifiers. Nearest neighbor methods (*IB1* and *NNge*) performed best among the WEKA classifiers in this study (data not known). *IB1* is a basic instance-based learner. It uses normalized Euclidean distance to find the training instance closest to the given test instance, and predicts the same class as this training instance. *NNge* is a nearest-neighbor method for generating rules using nonnested generalized exemplars, which are rectangular regions of instance space used for calculating a distance function to classify new instances (6).

The prediction accuracy by the WEKA classifiers was evaluated using 10-fold cross validation. In the 10-fold cross validation, the data set was randomly partitioned into 10 folds of equal size with possible exception of the last fold (the last fold contains the remaining samples). The prediction models were trained and tested 10 times. Each time, 9 folds were picked to build the prediction model, while the remaining fold was validated on the prediction model. We used 10-fold cross validation to evaluate the prediction models in this study, because the estimation accuracy by this validation method has been proven to have the lowest bias and variance among all validation methods, including the

leave-one-out method (8). It thus provides an objective evaluation of the performance of our prediction models in general.

For each drug in the experiment, the bottom one protein was removed at a time, and a WEKA classifier was run on the new feature set to get the prediction accuracy. The process was repeated until only one protein was left. We then identified the optimal classifier with the highest prediction accuracy along with the protein predictors for each drug. Using the WEKA techniques, the results were significantly improved for four drugs: Clomesone (NSC: 338947), Camptothecin,10-OH (NSC: 107124), Camptothecin,20-ester (S) (NSC: 606985), and Floxuridine (FUdR) (NSC: 27640).

The performance, protein predictors, algorithms, and software used in the identified optimal classifiers are listed in file Chemo.2table2.xls. The distribution of the prediction accuracy of the constructed optimal classifiers is shown in Figure 5.

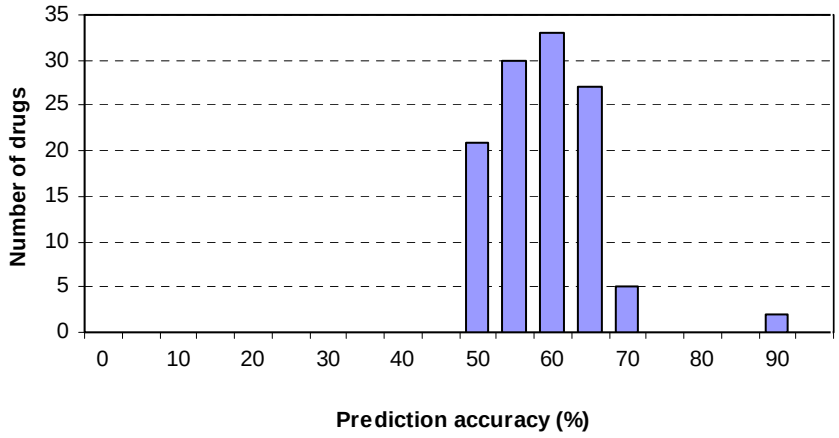

Figure 5. Prediction accuracy of the constructed optimal classifiers

4 Assessing the Significance of Our Prediction Results

In order to assess the significance of our prediction results (Figure 5), it is necessary to demonstrate that our prediction results are significantly better than random prediction. Two methods were used for the purpose. In the first method, for each drug we maintained the original class distributions and randomly permuted the class labels. For instance, a drug is examined on 60 cell lines, and the first 18 are labeled as "intermediate", the next 23 as "resistant", and the last 19 as "sensitive". Random permutation produces 60 class labels, while keeping the class distribution fixed (18 intermediate, 23 resistant, and 19 sensitive). Using this method, the matches between the rearranged class labels and the original ones were recorded. The percentage of matches was calculated as the accuracy measure for random prediction. Repeating this procedure 1000 times generated 1000 accuracies. The *P* value was calculated as the upper percentile of our prediction accuracy in the profile of 1000 random prediction results. If

the prediction accuracy produced by our classifier exceeds the 95th percentile of those 1000 random prediction accuracies, it is concluded that our prediction is significantly better than random prediction ($P < 0.05$). The results of method 1 are listed as follows (Figure 6).

***P*-value**	0	0.001	0.002	0.003	0.004	0.005	0.006	0.007	0.019
Frequency	97	9	4	1	1	3	1	1	1

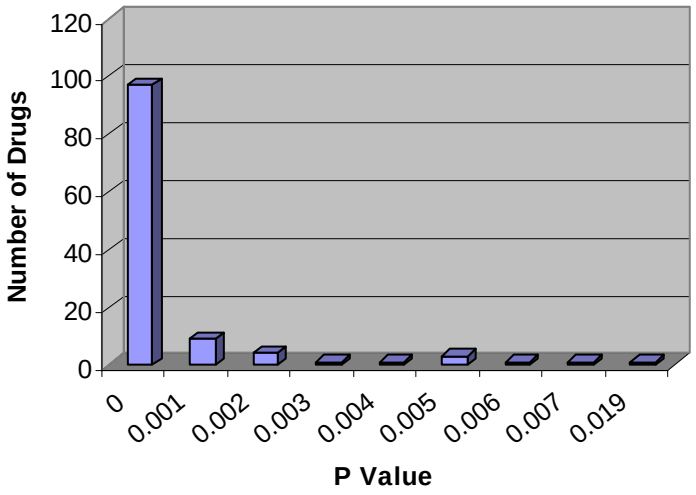

Figure 6. Histogram of *P*-values generated from method 1

The second method used in this study differs from the first in that the original class distributions were not fixed. For each cell line, we randomly assigned a label to it. It is analogous to drawing a ball from a box containing three balls labeled with "I" (Intermediate), "R" (Resistant) and "S" (Sensitive), and then assigning the label to this cell line. Using this method, the random prediction was compared with the true label and the overall accuracy across the 60 cell lines was calculated. This process was repeated for 1000 times and the *P* value was calculated the same way as described above. The results using method 2 are listed as follows (Figure 7). Using the second method, our prediction results were more significant than using the first one. The reason is that some drugs had very unbalanced chemosensitvity profiles. Therefore, keeping the class distributions as in the first method can take it into account. Thus, we chose to use the first method to evaluate our prediction results.

***P*-value**	0	0.001	0.002	0.005	0.006	0.008
Frequency	105	7	2	1	2	1

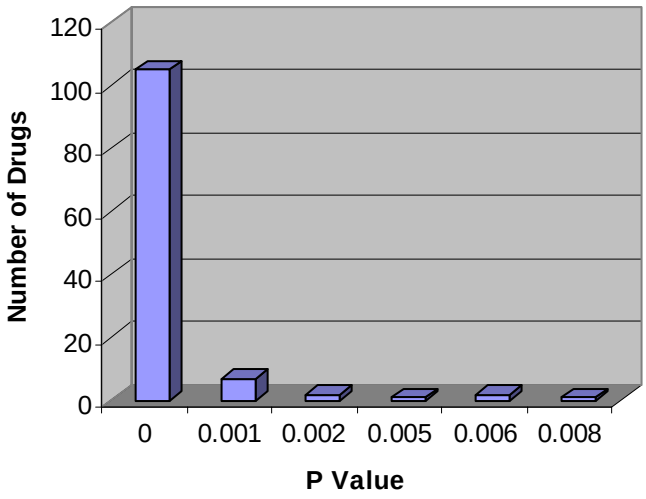

Figure 7. Histogram of *P*-values generated from method 2

5 Programming Source Code

5.1 *R* Code for Random Forests Experiments

In our study, we ranked the protein by use of "*mean decrease in accuracy*" (please refer to manuscript for details). The following is the major part of the source codes used in software package *R*.

```
#########################################################
# Identify the top 10 important proteins using random forests      #
#########################################################

# initialize a matrix which will store the prediction (result) for each drug (row)
pick10.pred <- matrix(0, ncol = 3, nrow = 118)

# initialize a matrix which will store the information on "which protein is picked" (TRUE)
pick10 <- matrix(0, ncol = 52, nrow = 118)

# initialize a matrix which will store the out-of-bag error rates
drug.error <- c()

for(i in 1:118)  # 118 drugs
{
        resp <- label[i,]  # the ith row of label contains the class labels of cell line for drug i

        resptest <- resp != "missing" # check if any cell line has missing value

        # remove cell lines which have missing value(s)
        newresp <- resp[resptest]
```

```
        drugi <- single.drug[resptest, ]

        drugi <- data.frame(drugi, newresp) # form the data frame

        # run RF on the whole list of proteins with importance measurements calculated
        drugi.rf <- randomForest(x = drugi[,1:52], y = drugi[,53], ntree = 600, importance = TRUE)

        # obtain the mean decrease in accuracy
        pick <- drugi.rf$importance[,4]

        # pick the 10 most important (measured by importance scores) proteins
        q <- quantile(pick, 0.81)

        pick <- pick >= q

        pick10[i,] <- pick

        drugi2 <- drugi[,pick]

        drugi <- data.frame(drugi2[,1:10], newresp)

        drugi.rf <- randomForest(newresp ~ ., data = drugi, ntree = 200) #run RF on the 10 proteins

        # check the dimension of confusion matrix since a few drugs do not have any sensitive samples
        tmp <- dim(drugi.rf$confusion)[2]

        if(tmp==4){
                pick10.pred[i, ] <- drugi.rf$confusion[,4]}
        else{
                pick10.pred[i, ] <- c(drugi.rf$confusion[,3],"NA")}

        drug.error[i] <- drugi.rf$err.rate[600,1]
}
```

5.2 C++ Code for Assessing the Significance of Our Prediction Results

The following shows the C++ codes for assessing the significance of our prediction results as compared with random prediction using two methods.

```
//////////////////////////////////////////////////////////////////////////
// Method 1: keep original class distribution ///
//////////////////////////////////////////////////////////////////////////

// compute p-value
#include<iostream.h>
#include <conio.h>
#include <time.h>
#include<fstream.h>
#include <stdlib.h>

void main() {
        //step 1: origin file
        ifstream in("c:\\file.txt"); //open input file,which is the Drug activity
                        //data of 118 "Mechanism of Action" drugs with the avg and dev of 60 cell lines

        if (in.fail())
           cout<<"cannot open input file"<<endl;
```

```
        float text[119][63]; //put the original data from the input file into an array.
        int text2[119][61];//change origin data to labels according 0.5 SD
        int i=0;int j=1;
        float text4[119][15];//origin predication
        float text5[119]={0};//store the predications of "random forests"

   while(!in.eof())
   {
        in>>text[j][i];
           i++;
                if (in.get()=='\n') {
                        j++;
                        i=0;
                }
   }

        in.close();
        ifstream in2("c:\\file2.txt"); //open input file, which is the predictions by "random forests"

        if (in2.fail())
           cout<<"cannot open input file"<<endl;
        i=0;
        j=1;

        while(!in2.eof())
   {

      in2>>text4[j][i];
           i++;
                if (in2.get()=='\n') {
                        j++;
                        i=0;
                }
   }

        in2.close();

        for (i=1;i<=118;i++){
                for(j=1;j<=14;j++){
                        if(text4[i][j]>text5[i])
                                text5[i]=text4[i][j];
                }
        }


//step 2: we assign the labels: sensitive 0; intermediate 1; resistant 2;


        ofstream out("c:\\pvalue2_3.txt"); //open output file
        if (out.fail())
           cout<<"cannot open output file"<<endl;
        for(i=1;i<=118;i++) { //118 drugs
                for(j=1;j<=60;j++) { //60 cell lines for each drug
                        if(text[i][j]<(text[i][61]-text[i][62]*0.5))
                                text2[i][j]=0;
                        else if(text[i][j]>(text[i][61]+text[i][62]*0.5) && text[i][j]<500)
                                text2[i][j]=2;
                        else                    if(text[i][j]>text[i][61]-text[i][62]*0.5                &&
text[i][j]<text[i][61]+text[i][62]*0.5)
                                text2[i][j]=1;
                        else if(text[i][j]==999)
                                text2[i][j]=4;
                }
        }
```

```
//step 3: get the random predictions
        int k=0;
        int l;
        float result=0.0;
        float avg=0.0;
        float p_value[119];     //array to store numbers of random predictions that are bigger than
                                //prediction by "random forests"
        float p_number[119];  //array to store pvalue
        float p=0.0;
        int neworder[61];
        i=0;
        int flag;
        int temp;

   srand( (unsigned)time( NULL ) ); //change random seed
   for(i=1;i<=118;i++){//loop of 118 drugs
                out<<"  "<<"rowno"<<i<<"--------------------------------------------------"<<"\n";
               for(k=1;k<=1000;k++) {// loop of 1000 times for each drug
                        for(j=1;j<=60;j++){
                                flag=0;
                                 while(flag==0){
                                        flag=1;
                                        temp=rand()%60+1;
                                        for (int n=1;n<j;n++){
                                                if (temp==neworder[n]){
                                                        flag=0;

                                                        break;
                                                }
                                        }
                                 }

                                neworder[j]=temp;// get random order of the //original labels
                        }

                        for(l=1;l<=60;l++){// if random labels is the same with
                                           //original file, we add 1 to result.

                                        if(text2[i][l]==text2[i][neworder[l]]&&text2[i][l]!=4)
                                        result+=1;
                        }

                        avg=result/60;// random prediction accuracy for 1 drug //at a time
                   out<<avg<<"\n";
                        if(avg>=text5[i])
                                p++;
                        result=0;
               }
               p_number[i]=p; // store numbers of random predictions that are
                              //bigger than prediction by "random forests"
               p_value[i]=p/1000;//store pvalue
               p=0.0;

          out<<i<<"    "<<text[i][0]<<""<<p_value[i]<<"\n";//output //pvalue
        }
        out.close();
}



////////////////////////////////////////////////////////////////////////////////////
// Method 2: random generate class label for each sample ///
////////////////////////////////////////////////////////////////////////////////////

#include<iostream.h>
#include <conio.h>
```

```
#include <time.h>
#include<fstream.h>
#include <stdlib.h>
void main() {
        //step 1: origin file
         srand( (unsigned)time( NULL ) ); //change random seed
        ifstream in("c:\\file.txt");    //open input file, which is the drug activity data of 118 "Mechanism of
                                        //Action" drugs with the avg and dev of 60 cell lines

        if (in.fail())
           cout<<"cannot open input file"<<endl;
        float text[119][63]; //put the original data from the input file into a array.

        int text2[119][61];//change origin data to labels according 0.5 SD
        int i=0;int j=1;
        float text4[119][15];//origin predication
        float text5[119]={0};//store the predications of "random forests"

   while(!in.eof())
   {
          in>>text[j][i];
           i++;
                if (in.get()=='\n') {
                        j++;
                        i=0;
                }
   }

        in.close();

        ifstream in2("c:\\file2.txt"); //open input file, which is the predictions by "random forests"
        if (in2.fail())
           cout<<"cannot open input file"<<endl;

        i=0;j=1;
        while(!in2.eof())
   {
      in2>>text4[j][i];
           i++;
                if (in2.get()=='\n') {
                        j++;
                        i=0;
                }
   }

        in2.close();
        for (i=1;i<=118;i++){
                for(j=1;j<=14;j++){
                        if(text4[i][j]>text5[i])
                                text5[i]=text4[i][j];
                }
        }

//step 2: we assign the labels: sensitive 0; intermediate 1; resistant 2;

        for(i=1;i<=118;i++) { //loop of 118 drugs
                for(j=1;j<=60;j++) { //loop of 60 cell lines for each drug
                                if(text[i][j]<(text[i][61]-text[i][62]*0.5))
                                text2[i][j]=0;
                        else if(text[i][j]>(text[i][61]+text[i][62]*0.5) && text[i][j]<500 )
                                text2[i][j]=2;
                        else if(text[i][j]>text[i][61]-text[i][62]*0.5 &&
        text[i][j]<text[i][61]+text[i][62]*0.5)

                                text2[i][j]=1;
                        else if (text[i][j]==999)
```

```
                                text2[i][j]=4;
                        }
                }


//step 3: we get the random predictions
        ofstream out("c:\\pvalue.txt"); //open output file

        if (out.fail())
           cout<<"cannot open output file"<<endl;

         int k=0;
         float result=0.0;
         float avg=0.0;
         float p_value[119];  //array to store numbers of random predictions that are bigger than
                              //prediction by "random forests"

         float p_number[119];//array to store pvalue
         int text3[61];// random labels by compputer
         float p=0.0;//numbers of random predictions that are bigger than prediction by "random forests"
         i=0;

    for(i=1;i<=118;i++){//loop of 118 drugs
                out<<"  "<<"rowno"<<i<<"----------------------------------------------------"<<"\n";
                for(k=1;k<=1000;k++) {// loop of 1000 times for each drug
                        for(j=1;j<=60;j++){
                                text3[j]=rand()%3;
                                if(text2[i][j]==text3[j])
                                result+=1;      // if random labels is the same with original file,
                                                //we add 1 to result.
                        }

                        avg=result/60;// random prediction accuracy for 1 drug 1 time

                        if(avg>=text5[i])
                                 p++;
                        result=0;
                }

                p_number[i]=p; // store numbers of random predictions that are bigger than prediction
by                              //"random forests"
                p_value[i]=p/1000;//store pvalue
                p=0.0;

           out<<i<<"    "<<text[i][0]<<""<<p_value[i]<<"\n";//output pvalue
         }
         out.close();
}
```

6 References

1. Nishizuka S, Charboneau L, Young L, et al. Proteomic profiling of the NCI-60 cancer cell lines using new high-density reverse-phase lysate microarrays. *Proc.Natl.Acad.Sci.U.S.A* 2003;100:14229-34.

2. Scherf U, Ross DT, Waltham M, et al. A gene expression database for the molecular pharmacology of cancer. *Nat.Genet.* 2000;24:236-44.

3. Staunton JE, Slonim DK, Coller HA, et al. Chemosensitivity prediction by transcriptional profiling. *Proc.Natl.Acad.Sci.U.S.A* 2001;98:10787-92.

4. Breiman L. Random Forests. *Machine Learning* 2001;45:5-32 .

5. Speed T. Statistical Analysis of Gene Expression Microarray Data. Chapman & Hall/CRC, 2003.

6. Witten IH, Frank E. Data Mining: Practical Machine Learning Tools and Techniques (2nd Edition) . Morgan Kaufmann, 2005.

7. Hall MA, Holmes G. Benchmarking Attribute Selection Techniques for Discrete Class Data Mining. *IEEE Transactions on Knowledge and Data Engineering* 2003;15.

8. Kohavi R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference on Artificial Intelligence (IJCAI)* 1995;1137-43.